A Classification of First-Order Progressable Action Theories in Situation Calculus

Stavros Vassos and **Fabio Patrizi**
Sapienza University of Rome
Rome, Italy
{vassos,patrizi}@dis.uniroma1.it

Abstract

Projection in the situation calculus refers to answering queries about the future evolutions of the modeled domain, while *progression* refers to updating the logical representation of the initial state so that it reflects the changes due to an executed action. In the general case projection is not decidable and progression may require second-order logic. In this paper we focus on a recent result about the decidability of projection and use it to drive results for the problem of progression. In particular we contribute with the following: (*i*) a major result showing that for a large class of intuitive action theories with *bounded unknowns* a first-order progression always exists and can be computed; (*ii*) a comprehensive classification of the known classes that can be progressed in first-order; (*iii*) a novel account of nondeterministic actions in the situation calculus.

1 Introduction

The situation calculus is a logical language that is designed for reasoning about action and change [McCarthy and Hayes, 1969]. A basic action theory (BAT) [Reiter, 2001] is a well-studied type of theory in this language that describes what holds in the initial state in a given domain, as well as how the domain evolves under the effects of the available actions. Such theories have been used to model various application domains, including the environment that a robotic assistant operates in, in order to provide deliberation capabilities as the basis for pro-active or goal-oriented behavior, e.g., [Reiter, 1993; Levesque *et al.*, 1997; De Giacomo *et al.*, 2009].

One of the main reasoning tasks that BATs facilitate is different versions of the so-called *projection problem*, that is, using entailment over this logical theory to *answer queries* about the future evolutions of the initial state. For example, one can formalize queries of the form "Is it true that condition c will be true after actions $a_1, \ldots, a_n$ have been performed?" or "Is it true that c' can never be true after action a' is performed?". As BATs are based on first-order logic, this task is quite difficult, in fact not decidable in the general case.

The so-called *progression problem* is the problem of *updating* the logical theory when actions are actually executed in the world. Essentially, a BAT is like an elaborate database that holds *static* information about the initial state and its possible evolutions. When a particular action a_1 is actually executed, the BAT can be still used to answer queries about the current state using queries of the form "Is it true that c is true *after* a_1 *is executed*?", and similarly for queries about the future of the current state. Nonetheless, as more and more actions are executed this quickly becomes impractical. For example, consider a robotic assistant that after a few hours or days of operation may have to answer queries of the form "is it true that c is true after $a_1, \ldots, a_{10,000}$ are performed?", only to talk about its current state. Progression is concerned with *updating* the description of the initial state in a way that is logically correct. This is also a very difficult task as in general second-order logic is required [Lin and Reiter, 1997].

Most of the work in this area lies within the two extremes of the trade-off between expressiveness and efficiency in the following sense. (*i*) On the one hand, work related to the theoretical framework of unrestricted action theories has been able to facilitate advanced knowledge representation features including accounts of time and concurrency, (e.g., [Reiter, 2001, Chapter 7]), epistemic states (e.g., [Scherl and Levesque, 2003; Claßen and Lakemeyer, 2006]), online action-based agent programming languages for behavior and control (e.g, [Levesque *et al.*, 1997; De Giacomo *et al.*, 2009]), and more, with the price that the reasoning methods are impractical or even undecidable in the most general case. (*ii*) On the other hand, the practical approaches for implemented deliberation systems based on the situation calculus have been very much restricted in expressiveness, where typically either complete information or a finite domain is assumed in order to provide a realistic approach to reasoning.

Nonetheless, some recent results have explored approaches that lie closer to a middle ground between these two extremes. Regarding projection, the recent work of [De Giacomo *et al.*, 2012] investigates a *boundedness* assumption that allows expressing rich action theories, while also ensuring the decidability of query answering through evaluation methods based on model checking. Similarly, wrt progression there has been a series of results about theories with actions that have a limited range of effects, such as *local-effect* and *normal* actions [Vassos *et al.*, 2008; Liu and Lakemeyer, 2009], which allows a first-order progression.

In this paper we follow the intuitions of [De Giacomo *et al.*, 2012] to introduce a new type of restriction that limits

incomplete information expressed in the theory to a bounded number of *unknowns* (acting similarly to *conditional tables* in database theory), but allows any type of action to be expressed. In particular we contribute with the following:

1. a major result showing that for a large class of action theories with *bounded unknowns* a first-order progression always exists and can be effectively computed;
2. a comprehensive classification of the known classes that can be progressed in first-order, also explaining their representational differences using an intuitive example;
3. a new approach to modeling non-deterministic actions in basic action theories, that is based on the intuitions from the account of unknowns in 1.

2 Situation calculus

The situation calculus as presented by Reiter [2001] is a three-sorted first-order language $\mathcal{L}$ with equality. The sorts are used to distinguish between actions, situations, and objects (everything else). A *situation* represents a world history as a sequence of actions. The constant S_0 is used to denote the initial situation where no actions have occurred. Sequences of actions are built using the function symbol *do*, such that $do(a, s)$ denotes the successor situation resulting from performing action a in situation s. Actions need not be executable in all situations, and the predicate $Poss(a, s)$ states that action a is executable in situation s. We will typically use a to denote a variable of sort action and α to denote a term of sort action, and similarly s and σ for situations.

A *relational fluent* is a predicate whose last argument is a situation, and thus whose value can change from situation to situation. We do not consider *functional fluents* as well as non-fluent predicates and functions, but note that they can be represented as relational fluents with some extra axioms. We also assume a finite number of fluent and action symbols, $\mathcal{F}$ and $\mathcal{A}$, and an infinite number of constants $\mathcal{C}$.

Often we need to restrict our attention to sentences in $\mathcal{L}$ that refer to a particular situation. For example, the initial knowledge base (KB) is a finite set of sentences in $\mathcal{L}$ that do not mention any situation terms except for S_0. For this purpose, for any situation term σ, we define $\mathcal{L}_\sigma$ to be the subset of $\mathcal{L}$ that does not mention any other situation terms except for σ, does not mention *Poss*, and where σ is not bound by any quantifier [Lin and Reiter, 1997]. When a formula $\phi(\sigma)$ is in $\mathcal{L}_\sigma$ we say that it is *uniform in* σ [Reiter, 2001].

Also, we will use $\mathcal{L}^2$ to denote the second-order extension of $\mathcal{L}$ that only allows predicate variables that take arguments of sort object. $\mathcal{L}^2_\sigma$ then denotes the second-order extension of $\mathcal{L}_\sigma$ by predicate variables with arguments of sort object.

2.1 Basic action theories

We will be dealing with a specific kind of $\mathcal{L}$-theory, the so-called *basic action theory* $\mathcal{D}$ which has the following form:[1]

$$\mathcal{D} = \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_0 \cup \Sigma$$

1. $\mathcal{D}_{ap}$ is a set of action precondition axioms (APs), one for each action function symbol $A_i \in \mathcal{A}$, of the form $Poss(A_i(\vec{x}), s) \equiv \Pi_i(\vec{x}, s)$, where $\Pi_i(\vec{x}, s)$ is in $\mathcal{L}_s$.
2. $\mathcal{D}_{ss}$ is a set of successor state axioms (SSAs), one per fluent symbol $F_i \in \mathcal{F}$, of the form $F_i(\vec{x}, do(a, s)) \equiv \Phi_i(\vec{x}, a, s)$, with $\Phi_i(\vec{x}, a, s) \in \mathcal{L}_s$. SSAs characterize the conditions under which F_i has a specific value at situation $do(a, s)$ as a function of situation s and action a.
3. $\mathcal{D}_{una}$ is the set of unique-names axioms for actions: $A_i(\vec{x}) \neq A_j(\vec{y})$, and $A_i(\vec{x}) = A_i(\vec{y}) \supset \vec{x} = \vec{y}$, for each pair of distinct action symbols A_i and A_j in $\mathcal{A}$.
4. $\mathcal{D}_0$ is uniform in S_0 and describes the initial situation.
5. Σ is a set of domain independent foundational axioms which formally define legal situations also using a second-order inductive axiom in $\mathcal{L}^2$.

[1] For readability we often omit the leading universal quantifiers.

2.2 The problem of progression

The progression of a basic action theory (BAT) is the problem of *updating* the initial KB so that it reflects the current state of the world after some actions have been performed, instead of the initial state of the world. In other words, in order to do a one-step progression of the BAT $\mathcal{D}$ with respect to the ground action α we need to replace $\mathcal{D}_0$ in $\mathcal{D}$ by a suitable set $\mathcal{D}_\alpha$ of sentences so that the original theory $\mathcal{D}$ and the theory $(\mathcal{D} - \mathcal{D}_0) \cup \mathcal{D}_\alpha$ are equivalent with respect to how they describe the situation $do(\alpha, S_0)$ and the situations in the future of $do(\alpha, S_0)$. In a seminal paper, Lin and Reiter [1997] gave a model-theoretic definition for the progression $\mathcal{D}_\alpha$ of $\mathcal{D}_0$ wrt α and $\mathcal{D}$ that achieves this goal. Note that we will be using S_α to denote the situation term $do(\alpha, S_0)$.

Definition 1. Let M and M' be structures with the same domains for sorts action and object. We write $M \sim_{S_\alpha} M'$ if the following two conditions hold: (*i*) M and M' have the same interpretation of all situation-independent predicate and function symbols; (*ii*) M and M' agree on all fluents at S_α, that is, for every relational fluent F, and every variable assignment μ, $M, \mu \models F(\vec{x}, S_\alpha)$ iff $M', \mu \models F(\vec{x}, S_\alpha)$.

Definition 2. Let $\mathcal{D}_\alpha$ be a set of sentences in $\mathcal{L}^2_{S_\alpha}$. $\mathcal{D}_\alpha$ is a progression of $\mathcal{D}_0$ wrt α if for any structure M, M is a model of $\mathcal{D}_\alpha$ iff there is a model M' of $\mathcal{D}$ such that $M \sim_{S_\alpha} M'$.

We now proceed to identify an extension of the relatively complete theories of [Lin and Reiter, 1997] that can be progressed using first-order logic. In Section 4 we will show that they indeed capture some natural cases that existing classes of first-order progressable theories cannot encode.

3 Bounded unknowns

First we briefly review the notion of a relatively complete initial KB $\mathcal{D}_0$ [Lin and Reiter, 1997]. The idea is that for each $F_i \in \mathcal{F}$ there is an axiom in $\mathcal{D}_0$ of the form:

$$\forall \vec{x}(F_i(\vec{x}, S_0) \equiv \phi_i(\vec{x})),$$

where $\phi_i(\vec{x})$ is situation-independent and whose free variables are in $\vec{x}$. In this way $\mathcal{D}_0$ characterizes the truth value for all atoms of F_i in S_0 relatively to formulas that do not mention situations; in our case, in terms of formulas ϕ_i built on constants and equality (and possibly quantifiers). In the typical case when we assume uniqueness of names for constants,

$\mathcal{D}_0$ is a complete theory. For example the following axiom states that there exactly two atoms true for $\mathit{In}(x_1, x_2, S_0)$:

$$\forall x \forall y (\mathit{In}(x, y, S_0) \equiv (x{=}\mathit{purse} \wedge (y{=}\mathit{keys} \vee y{=}\mathit{wallet}))).$$

There is a very simple way to progress a relatively complete $\mathcal{D}_0$ wrt a ground action α as follows. For each $F_i \in \mathcal{F}$, we start from the SSA for F_i, instantiated for α and S_0: $\forall \vec{x}(F_i(\vec{x}, S_\alpha) \equiv \Phi_i(\vec{x}, \alpha, S_0))$. Then for every occurrence of fluent atom $F_j(\vec{o}, S_0)$ in $\Phi_i(\vec{x}, \alpha, S_0)$, we replace the atom by $\phi_j(\vec{o})$. Essentially, the form of $\mathcal{D}_0$ allows us to replace fluent atoms about S_0 by their definition, and obtain a set of sentences uniform in S_α that qualifies as a progression.

We propose a simple yet powerful extension of this type of $\mathcal{D}_0$ that introduces existentially quantified variables to model incomplete information, while also preserving the intuitive progression strategy by substitution. The following states that there are exactly two atoms true for $\mathit{In}(x_1, x_2, S_0)$, namely $\mathit{In}(\mathit{purse}, \mathit{keys}, S_0)$ and $\mathit{In}(\mathit{purse}, w, S_0)$ for some $w{\neq}\mathit{keys}$.

$$\begin{aligned} \exists w.(w{\neq}\mathit{keys}) \wedge \forall x \forall y (\mathit{In}(x, y, S_0) \equiv \\ (x{=}\mathit{purse} \wedge (y{=}\mathit{keys} \vee y{=}w))). \end{aligned} \tag{1}$$

Note that (1) completely characterizes $\mathit{In}(x, y, S_0)$ relatively to the formula on the right hand side, in which case w acts almost like a "null" value of normal databases along with possible constraints. As $\mathcal{L}$ includes infinitely many constants, incomplete information over an infinite domain is implied.

We can build similar axioms about more than one fluent.

$$\begin{aligned} \exists w_1 \exists w_2 .(w_1{=}\mathit{loc}_1 \wedge w_2{=}\mathit{north} \vee w_1{=}\mathit{loc}_2 \wedge w_2{=}\mathit{east}) \\ \wedge \forall x (\mathit{RobotAt}(x, S_0) \equiv x{=}w_1) \\ \wedge \forall x (\mathit{RobotDir}(x, S_0) \equiv x{=}w_2). \end{aligned} \tag{2}$$

In this case we use w_1, w_2 to specify disjunctive information between fluent atoms of $\mathit{RobotAt}(x, S_0)$ and $\mathit{RobotDir}(x, S_0)$. For instance, (2) could be used to represent a state where a robot has only partial high-level knowledge of its location and direction due to sensor interference.

We now formalize a new class of initial KBs based on using a single axiom for all (finitely many) fluent symbols in $\mathcal{L}$. This type of $\mathcal{D}_0$ is relatively complete but also allows a rich form of incomplete information by means of existentially quantified variables that we call *unknowns*. Unknowns behave as Skolem constants but without departing from the original language $\mathcal{L}$. Also, as we will shortly see, this type of KB allows a progression by substitution in the manner discussed at the beginning of the section.

Definition 3. An initial KB $\mathcal{D}_0$ is *relatively complete with bounded unknowns* iff it is a sentence of the following form:

$$\begin{aligned} \exists w_1 \cdots \exists w_b .\, e(w_1, \ldots, w_b) \wedge \\ \textstyle\bigwedge_{i=1}^{n} \forall \vec{x_i} (F_i(\vec{x_i}, S_0) \equiv \phi_i(\vec{x_i}, w_1, \ldots, w_b)), \end{aligned}$$

where b is the *bound* on unknowns, $\vec{x_i}$ is a tuple of variables of size equal to the number of object arguments of F_i, and e, ϕ_i are first-order situation-independent formulas whose free variables are all in $\{w_1, \ldots, w_b\}$, $\{\vec{x_i}, w_1, \ldots, w_b\}$, resp.

Definition 4. A BAT $\mathcal{D}$ over a language $\mathcal{L}$ with finitely many fluent and action symbols and infinitely many constant symbols is *relatively complete with bounded unknowns* iff $\mathcal{D}_0$ is a consistent sentence of the form described in Definition 3 for some bound b, and mentions all fluent symbols in $\mathcal{L}$.

Note that no other constraint is imposed on any other part of $\mathcal{D}$ except for $\mathcal{D}_0$, in particular, there is no restriction on the actions described in the successor state axioms (SSAs) in $\mathcal{D}_{ss}$. We now proceed to show that a first-order progression can be obtained by means of a simple syntactic transformation.

Definition 5. Let $\mathcal{D}_0$ be as in Definition 3 with bound b. We define $\mathcal{T}[\psi, \vec{w}]$ as the transformation that replaces occurrences of fluent atoms of the form $F_i(\vec{o}, S_0)$ in ψ by the formula $\phi_i(\vec{o}, \vec{w})$, where $\vec{o}$ is a tuple of terms of sort object, $\vec{w}$ is tuple of variables of sort object of size b, and ϕ_i is the corresponding formula characterizing F_i in $\mathcal{D}_0$.

The intuition is that by applying $\mathcal{T}$ to the right-hand side Φ_i of the SSA for each F_i in $\mathcal{L}$, we obtain a progression similarly to the case investigated by Lin and Reiter [1997]. In our case each of the fluent atoms is replaced by a formula that also introduces new variables not mentioned in the SSA, i.e., variables $\vec{w}$ that capture the unknowns. In order then for the substitution to make sense, these unknowns need to be referenced consistently by the same variable names in all occurrences, and this is why they are given as input to $\mathcal{T}$.

Theorem 1. *Let $\mathcal{D}$ be a BAT over language $\mathcal{L}$ that is* relatively complete with bounded unknowns *as in Definition 4, and α a ground action term. Let ψ be the following sentence:*

$$\exists \vec{w}.\, e(\vec{w}) \wedge \textstyle\bigwedge_{i=1}^{n} \forall \vec{x_i} \big(F_i(\vec{x_i}, S_\alpha) \equiv \mathcal{T}[\Phi_i(\vec{x_i}, \alpha, S_0), \vec{w}]\big),$$

where $\vec{w}$ is a set of variables of sort object distinct from all $\vec{x_i}$, formulas e, ϕ_i characterize the constraints among unknowns and the fluent atoms F_i in $\mathcal{D}_0$, Φ_i is the right-hand side of the SSA for F_i, and $\mathcal{T}$ the transformation of Definition 5. Then ψ is a progression of $\mathcal{D}_0$ wrt to α and $\mathcal{D}$.

Proof. Let M be an arbitrary structure of $\mathcal{L}$. By Definition 2 it suffices to show that M is a model of ψ iff there is a model M' of $\mathcal{D}$ such that $M \sim_{S_\alpha} M'$.

($\Rightarrow$): We construct the model M' starting from M and replacing the truth value for all $F_i \in \mathcal{F}$ s.t. $M', \mu_{\vec{o}}^{\vec{x_i}\,\vec{w}}{}_{\vec{c}} \models \mathcal{F}_i(\vec{x_i}, S_0)$ iff $M', \mu_{\vec{o}}^{\vec{x_i}\,\vec{w}}{}_{\vec{c}} \models \phi_i(\vec{x_i})$ for all tuples of objects $\vec{o}$ in the domain, where $\vec{c}$ is a tuple of objects such that $M', \mu_{\vec{c}}^{\vec{w}} \models e(\vec{w})$. By the form of $\mathcal{D}_0$ and the construction of ψ it follows that M' models $\mathcal{D}_0$ and the SSAs when instantiated with α, S_0. We then use the SSAs in $\mathcal{D}_{ss}$ to replace the truth value for fluents in all situations other than S_0, S_α, and axioms in $\mathcal{D}_{ap}$ to replace the extension of *Poss*. It follows that M' models $\mathcal{D}$.

($\Leftarrow$): It suffices to show that $\mathcal{D} \models \psi$. Let M be an arbitrary model of $\mathcal{D}$. M models $\mathcal{D}_0$, therefore there exists an object vector $\vec{c}$ in the domain s.t. $M, \mu_{\vec{c}}^{\vec{w}} \models e(\vec{w})$ ***(I)*** and for all i, $M, \mu_{\vec{c}}^{\vec{w}} \models \forall \vec{x_i}(F_i(\vec{x_i}, S_0) \equiv \phi_i(\vec{x_i}, \vec{w}))$ ***(II)***. M also models $\mathcal{D}_{ss}$, therefore for all i, $M, \mu_{\vec{c}}^{\vec{w}} \models \forall \vec{x_i}(F_i(\vec{x_i}, S_\alpha) \equiv \Phi_i(\vec{x_i}, \alpha, S_0))$ ***(III)***. By ***(II),(III)***, and the definition of $\mathcal{T}$ it follows that $M, \mu_{\vec{c}}^{\vec{w}} \models \forall \vec{x_i}(F_i(\vec{x_i}, S_\alpha) \equiv \mathcal{T}[\Phi_i(\vec{x_i}, \alpha, S_0), \vec{w}])$, which along with ***(I)*** implies that $M \models \psi$. □

This form of $\mathcal{D}_0$ essentially limits all incomplete information in $\mathcal{D}$ to what can be expressed in terms of the unknowns $\vec{w}$ and the global constraints in $e(\vec{w})$. Observe that this is true not only for $\mathcal{D}_0$ and S_0 but in fact *for all future situations*. This may seem a little unintuitive, but indeed this is a property of all BATs that comes from the structure of the SSAs.

That is, SSAs being *definitions* of the form "F_i in $do(a, s)$ is true iff condition Φ_i about s holds", they cannot introduce disjunctive or existential information; any incomplete information about the truth value of fluents in any situation s has to "link back" to S_0 by means of conditions expressed in the right-hand side of SSAs. This is also true for a normal unrestricted $\mathcal{D}_0$. The difference is that this form of $\mathcal{D}_0$ makes it explicit that there is only *a fixed number of variables* that can be used to formalize incomplete information at any time. We use this observation in Section 5, to suggest a novel account for representing nondeterministic actions in normal BATs.

4 A map of first-order progressable theories

We now categorize the known classes for first-order progressable BATs into two incomparable hierarchies: one that relies on limiting the effects of actions by imposing syntactic restrictions on the SSAs, and one that relies on imposing a progression-friendly structure on the $\mathcal{D}_0$ that can be maintained. We compare two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ based on *expressive power*, by writing: $\mathcal{C}_1 \preceq \mathcal{C}_2$, if for every theory $\mathcal{D}_1 \in \mathcal{C}_1$ there exists a $\mathcal{D}_2 \in \mathcal{C}_2$ (over the same language as $\mathcal{D}_1$) s.t. the set of models of $\mathcal{D}_1$ and $\mathcal{D}_2$ coincide; $\mathcal{C}_1 \approx \mathcal{C}_2$, if $\mathcal{C}_1 \preceq \mathcal{C}_2$ and $\mathcal{C}_2 \preceq \mathcal{C}_1$; and $\mathcal{C}_1 \prec \mathcal{C}_2$, if $\mathcal{C}_1 \preceq \mathcal{C}_2$ and $\mathcal{C}_1 \not\approx \mathcal{C}_2$.

We start by introducing a motivating example.

A simple robot scenario that challenges progression

Consider an assistant robot capable of navigating and moving objects around in a grid-like world, where some objects such as bags and boxes may contain other objects. We use the fluents introduced in Section 3 along with fluents *Holding*, *Connected* and action *moveFwd* with the intuitive meaning. We assume that the SSA for any fluent F has the form: $F(\vec{x}, do(a, s)) \equiv \gamma^+_F(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg\gamma^-_F(\vec{x}, a, s)$.

For *RobotAt*(x, s), γ^+_{RobAt} and γ^-_{RobAt} are as follows:

$$\gamma^+_{RobAt}\colon\ \exists z_1 \exists z_2.\ a = \mathit{moveFwd} \wedge \mathit{RobotAt}(z_1, s) \wedge \mathit{RobotDir}(z_2, s) \wedge \mathit{Connected}(z_1, z_2, x, s),$$

$$\gamma^-_{RobAt}\colon\ a = \mathit{moveFwd}\,.$$

Similarly for *At*(x, y, s), γ^+_{At} and γ^-_{At} are as follows:

$$\gamma^+_{At}\colon\ \exists z_1 \exists z_2 \exists z_3.\ a = \mathit{moveFwd} \wedge \mathit{RobotAt}(z_1, s) \wedge \mathit{RobotDir}(z_2, s) \wedge \mathit{Connected}(z_1, z_2, y, s) \wedge (\mathit{Holding}(x, s) \vee \mathit{Holding}(z_3, s) \wedge \mathit{In}(z_3, x, s))$$

$$\gamma^-_{At}\colon\ \exists z.\ a = \mathit{moveFwd} \wedge (\mathit{Holding}(x, s) \vee \mathit{Holding}(z, s) \wedge \mathit{In}(z, x, s)).$$

Note that when the robot moves, all the objects it carries move, too. The SSAs for *RobotDir*, *Holding*, and *In* have the intuitive formalization, while *Connected* is static, i.e., its SSA copies the truth of atoms in S_0 to all situations.

4.1 Restricting the effects of actions

The main intuition behind the approaches restricting the effects of actions is that by appropriately *characterizing all fluent atoms that may be affected by an action* α, we can rely on *forgetting* [Lin and Reiter, 1994] to unset the old truth value for them, and then use $\mathcal{D}_{ss}[\alpha, S_0]$ (the instantiated SSAs) to set the new one; the rest of the fluent atoms remain unchanged. The characterization of the affected fluents is made possible by imposing a syntactic structure on SSAs.

Class BAT-SCF: BATs with strongly context-free actions

In strongly context-free actions [Lin and Reiter, 1997], each of the $\gamma^*(\vec{x}, a, s)$ formulas of SSAs (i.e., γ^+ and γ^-) is a disjunction, whose disjuncts have the form $\exists\vec{z}(a = A(\vec{y}))$, where $\vec{x}$ is included in $\vec{y}$, and $\vec{z}$ are the remaining variables of $\vec{y}$. This has the effect that a ground action $A(\vec{o})$ essentially dictates all the fluents that get affected by the execution of the action. More formally, for a strongly context-free SSA it follows by the uniqueness of names for actions that:

$$\gamma^*(\vec{x}, \alpha, S_0) \equiv \bigvee_i (\vec{x} = \vec{c_i}).$$

A method for computing a first-order progression for BAT-SCF is described in [Lin and Reiter, 1997]. Note that the SSA for *RobotAt* is clearly not strongly context-free.

Class BAT-LE: BATs with local-effect actions

Local-effect actions [Liu and Levesque, 2005] imply a finite number of affected atoms, but in a more expressive way. The trick is similar to that for strongly context-free actions: every $\gamma^*(\vec{x}, a, s)$ is a disjunction, with disjuncts of the form $\exists\vec{z}(a = A(\vec{y}) \wedge \phi(\vec{y}))$, $\vec{x}$ included in $\vec{y}$, and $\vec{z}$ the remaining variables of $\vec{y}$; however a *context* formula ϕ can be specified along with every disjunct. The arguments of fluents that may be affected by a ground action α still need to occur as arguments of α, but this form of SSA allows for instance the following:

$$\gamma^+_{RobAt}\colon\ \exists z_1 \exists z_2.\ a = \mathit{moveFwd}(z_1, z_2, x) \wedge \mathit{RobotAt}(z_1, s) \wedge \mathit{RobotDir}(z_2, s) \wedge \mathit{Connected}(z_1, z_2, x, s),$$

$$\gamma^-_{RobAt}\colon\ \exists z_1 \exists z_2.\ a = \mathit{moveFwd}(x, z_1, z_2) \wedge \mathit{RobotAt}(x, s) \wedge \mathit{RobotDir}(z_1, s) \wedge \mathit{Connected}(x, z_1, z_2, s),$$

As above, the uniqueness of action names implies:

$$\gamma^*(\vec{x}, \alpha, S_0) \equiv \bigvee_i (\vec{x} = \vec{c_i} \wedge \phi(\vec{x})).$$

Since SSAs in BAT-SCF are special cases of BAT-LE's, BAT-SCF $\preceq$ BAT-LE. To prove that BAT-SCF $\prec$ BAT-LE, it suffices to show some $\mathcal{D}$ in BAT-LE that cannot be "transformed" into a $\mathcal{D}'$ (over the same language as $\mathcal{D}$) in BAT-SCF with the same meaning in terms of models. The SSA above can be used for this, as there is no way to "compile away" the context formula for *moveFwd*(o_1, o_2, o_3):

$$\mathit{RobotAt}(o_1, s) \wedge \mathit{RobotDir}(o_2, s) \wedge \mathit{Connected}(o_1, o_2, o_3, s).$$

The context formula could be compiled into the action precondition axiom, allowing execution of valid actions only, thus resulting in a strongly-context free version of the SSA. This, however, would work only with complete information about *RobotAt* and *RobotDir*. If this is not the case, as, e.g., in (2), a conditional effect through a context formula is needed to allow each class of models to evolve appropriately.

The result that theories in BAT-LE admit a first-order progression was shown in [Vassos *et al.*, 2008], while an extension of the result wrt computing the progression can be found in [Liu and Lakemeyer, 2009]. Also note that the SSA for *RobotAt* of our running example is not local-effect either.

Class BAT-NR: Normal actions

Normal actions [Liu and Lakemeyer, 2009] explore the idea of characterizing the affected fluents from a logical perspective, taking advantage of the connection between progression and forgetting. In particular, a normal action may affect infinitely many fluent atoms but in a way that progression amounts to *forgetting a predicate* (i.e., not just a finite number of atoms). Under some conditions the result of forgetting, which in general is second-order, can be expressed in first-order logic by means of quantifier elimination.

The intuition behind normal actions is that a ground action α is allowed to have non-local effects on a fluent F as long as it has local effects on all fluents appearing in the effect formulas γ_F^*. For instance, consider the ground action *move*(*purse*, loc_1, loc_2) for moving object *purse* together with every object inside it from loc_1 to loc_2. Such action has non-local effects on *At*(x, y, s), as it affects the location of the objects inside *purse*, but these are not action arguments. Nonetheless, assuming that *In*(x, y, s) is the only fluent mentioned in γ_{At}^*, the action is normal, as it trivially has local-effects on *In*(x, y, s), i.e., no effects on it whatsoever.

It is straightforward to show that BAT-LE $\preceq$ BAT-NR as local-effect actions are a special case of normal actions by their definition. To see why BAT-LE $\prec$ BAT-NR, consider action *move*(*purse*, loc_1, loc_2) and case (1). When performing the action, also the fluent atoms corresponding to items inside *purse*, e.g., *At*(*keys*, loc_1, S_α) and *At*(*keys*, loc_2, S_α), are affected. In order for a local-effect version of the action to be able to set these atoms to true and false respectively, the object *keys* needs to be included in the action arguments. This is even more problematic for arguments that are not known, such as the other item w in the purse, implied by (1).

As a final note, observe that *moveFwd* in our motivating example is not normal. Indeed, the action has non-local effects on *RobotAt*(x, s), which is mentioned in γ^+_{RobAt}. To see this, one has to look at the technical details of Definition 4.4 in [Liu and Lakemeyer, 2009]: a ground action has local-effects on fluent F if, using $\mathcal{D}_{una}$ and the action, γ_F^+ and γ_F^- can be simplified into a disjunction that explicitly states all potentially affected ground atoms of F. Since in general the agent may not know a-priori the boundaries of the world it is situated in, the number of potential locations is not bounded, hence such a simplification is not possible by only using $\mathcal{D}_{una}$.

BAT-CF: Context-free actions

We can now look into a generalization of BAT-SCF. BATs with *context-free* effects, which we refer to as BAT-CF, allow the use of any γ^* formula in SSAs, as long as it does not mention situations, i.e., it is of the form $\gamma_F^*(\vec{x}, a)$ for a fluent $F(\vec{x}, s)$. Note that this goes beyond local-effects as it allows, e.g., making all F atoms false by setting γ_F^* to $a = A \wedge \mathit{false}$. On the other hand, actions with local-effects can express conditional effects with a γ^* that depends on s. As a result BAT-CF and BAT-LE are incomparable. It is also straightforward that BAT-SCF $\prec$ BAT-CF. The final piece of information missing is that BAT-CF $\prec$ BAT-NR. This follows from the definition of normal actions: even though a context-free axiom may have non-local effects on some fluent F, by definition it trivially has local-effects on all fluents mentioned in γ_F^* as it cannot mention situations.

We can now state the main result of this section.

Theorem 2. BAT-SCF$\prec${BAT-LE, BAT-CF}$\prec$BAT-NR.

BAT-SCF and BAT-CF were investigated in combination with restrictions on the initial KB in [Lin and Reiter, 1997]. This result also clarifies that just the restriction on the SSAs is adequate for ensuring a first-order progression.

4.2 Restricting the initial KB

The class of BATs with a relatively complete KB, referred to as BAT-RC, characterizes the truth value of all fluent atoms in S_0, in terms of formulas ϕ_i not mentioning situations. In general $\mathcal{L}$ can include non-fluent predicates and functions, and formulas ϕ_i can mention them. Similarly to the case of theories with bounded unknowns, referred to as BAT-BU, the initial KB can express incomplete information through rigid symbols. In this sense the effect of an unknown w s.t. $w = c_1 \vee w = c_2$ can be encoded in a BAT-RC by using a predicate P such that $P(c_1) \vee P(c_2)$ is included in the KB and is not mentioned elsewhere in the KB.

Nonetheless, it is typical to consider BATs that do not include functions and functional symbols in order to avoid some of the complications that arise in the treatment of terms, and also encode predicates as static relational fluents. In this setting, BAT-BU theories force incomplete information to be treated at the level of variables as unknown values. In any case, in the scope of BATs without predicates and functions (adopted in this paper) BAT-RC express complete information about all fluents, while BAT-BU allow for a bounded number of unknowns, in which sense BAT-RC $\prec$ BAT-BU.

Finally, BAT-BU is the only class (of those known to admit a first-order progression) able to capture the simple motivating example we introduced in the beginning of the section.

4.3 Combining action and KB restrictions

The restriction on bounded unknowns is inspired by a recent result about decidability of verification –which includes *projection* as a special case– over the class of the so-called bounded BATs (BAT-BD) [De Giacomo *et al.*, 2012]. A BAT-BD theory entails that in all situations the number of true fluent atoms is bounded by some fixed number. This requires appropriate restrictions both on the initial KB and the actions' specification. Concerning actions, [De Giacomo *et al.*, 2012] introduce SSAs that either ensure a balance between "adding" and "removing" constants in the extension of a fluent, or discard facts when a given bound is reached. These intuitive restrictions are fairly common in applications either because facts do not persist indefinitely or because one eventually discards some facts. The BAT-BU class, on the other hand, provides an intuitive syntactic structure for the initial KB, ensuring that a first-order *progression* always exists and can be effectively computed. The term "bounded" here is used to limit the number of "unknowns" instead of that of known facts, as in BAT-BD.

The results on BAT-BD and BAT-BU suggest that combinations of restrictions from both classes can be used to provide a solid ground for investigating practical deliberation systems based on the situation calculus. Indeed, it is not hard

to see that the intersection of BAT-BU and BAT-BD includes a wide range of theories that constitute a *well-behaved* class, for which projection is decidable and a first-order progression is always effectively computable. Moreover, these theories make a direct link between representing dynamic domains in the situation calculus and reasoning by means of methods developed in the theory of databases: (*i*) the proof of decidability for bounded theories relies on abstractions based on the notion of *active domain*; (*ii*) the form of the initial KB for theories with bounded unknowns is essentially equivalent to a *conditional table* [Imielinski and Lipski, 1984] when the formulas in the KB are also quantifier-free.

5 Nondeterministic actions using unknowns

We now show the power of BAT-BU theories in modeling nondeterministic actions. For simplicity we fix a number b of coins, and use b different fluent symbols (Coin_1–Coin_b) to represent their state and b different actions (flip_1–flip_b) to capture flipping. The idea is for each Coin_i to use one unknown w_i to characterize $\mathit{Coin}_i(x, do(\mathit{flip}_i, s))$. In order to do this we need to set w_i to have two possible values, namely *heads* or *tails*, and use an extra fluent Ndet_i as a placeholder of disjunctive information. The next sentence is a $\mathcal{D}_0$ that models this and also sets all coins to be "heads" in S_0.

$$\begin{aligned}\exists w_1 \cdots \exists w_b.\, &\textstyle\bigwedge_{i=1}^{b}(w_i{=}\mathit{heads} \vee w_i{=}\mathit{tails})\\ &\wedge \textstyle\bigwedge_{i=1}^{b} \forall x(\mathit{Ndet}_i(x, S_0) \equiv x{=}w_i)\\ &\wedge \textstyle\bigwedge_{i=1}^{b} \forall x(\mathit{Coin}_i(x, S_0) \equiv x{=}\mathit{heads}).\end{aligned}$$

The next SSAs then shows that we can "assign" Ndet_i to $\mathit{Coin}_i(x, do(\mathit{flip}_i, s))$ to make it capture disjunctive information about its state in the situation after flipping the coin:

$$\begin{aligned}\mathit{Coin}_i(x, do(a, s)) \equiv\ & a{=}\mathit{flip}_i \wedge \mathit{Ndet}_i(x, s)\\ & \vee\, a{\neq}\mathit{flip}_i \wedge \mathit{Coin}_i(x, s),\end{aligned}$$

$$\mathit{Ndet}_i(x, do(a, s)) \equiv\ \mathit{Ndet}_i(x, s).$$

Note that executing flip_i has the effect of Coin_i not having complete information any more:

$$\mathcal{D} \models \mathit{Coin}_i(\mathit{heads}, S_0),\ \ \mathcal{D} \models \neg\, \mathit{Coin}_i(\mathit{tails}, S_0),$$

$$\mathcal{D} \not\models \mathit{Coin}_i(\mathit{heads}, do(\mathit{flip}_i, S_0)),$$

$$\mathcal{D} \not\models \mathit{Coin}_i(\mathit{tails}, do(\mathit{flip}_i, S_0)),$$

$$\mathcal{D} \models \forall x(\mathit{Coin}_i(x, do(\mathit{flip}_i, S_0)) \equiv (x{=}\mathit{heads} \vee x{=}\mathit{tails})).$$

Observe also that as we discussed in Section 3, the disjunctive information about Coin_i links back to that of Ndet_i in S_0. Finally, an appropriate account for sensing (e.g., via a sensing fluent $SF(a, s)$ [Levesque, 1996]) can be used to settle the fluent in being true for exactly one of the two outcomes.

6 Related work

The notion of progression for BATs was first introduced by Lin and Reiter [1997]. They were also first to investigate restrictions that guarantee a first-order progression.

Liu and Levesque [2005] introduced the *local-effect* assumption for actions when they proposed a weaker version of progression that is logically incomplete, but remains practical. Vassos *et al.* [2008] showed that, under this assumption, a logically correct first-order progression can actually be computed by updating a finite $\mathcal{D}_0$. Liu and Lakemeyer [2009] showed how the result of [Vassos *et al.*, 2008] on local-effect progression relates to the notion of forgetting, and examined the more expressive normal actions.

A different approach to first-order progression involves identifying wide classes of queries that can be performed over a basic action theory wrt which a first-order progression is guaranteed to be correct (even though it may be incorrect in the general case). This was also first investigated by Lin and Reiter [1997] who showed that for answering projection queries that are uniform in some situation σ, a first-order progression is always adequate. Shirazi and Amir [2005] proved a similar result in the context of *logical filtering*. Vassos *et al* [2008] extended this result showing that un-nested quantification over situations may also be allowed.

Outside of the situation calculus, Thielscher [1999] defined a dual representation for basic action theories based on state update axioms that explicitly define the direct effects of each action, and investigated progression in this setting. In this work the update relies on expressing the changes using constraints which may need to be conjoined to the original database, which is similar to the idea of replacing fluent atoms by their definition in the relatively complete initial KBs.

Perhaps the most relevant work related to nondeterministic actions in situation calculus is the use of *a disjunction of SSAs*, introduced by [Lin, 1996] in the context of nonmonotonic logics. Unlike our approach though, their account operates at the level of meta-theory and cannot be used to reason about future situations by means of regular entailment.

7 Conclusions

In this paper we created a map of all known theories that admit a first-order progression and, inspired by recent results about decidability of verification for BAT-BD, extended it with the class BAT-BU of theories with *bounded unknowns* for which we proved that a first-order progression always exists and can be effectively computed. We also showed that BAT-BU theories generalize BATs with a relatively complete KB, and that they can capture nondeterministic actions. Further, we argued that the theories falling into the intersection of BAT-BU and BAT-BD form a well-behaved class of practical interest, which guarantees decidability of projection and effective computability of first-order progression. This is a major advancement, since until now such properties were guaranteed for special (less intuitive) classes, and only for the two problems separately. Finally, we pointed out that such a class builds a link between the situation calculus as a representation formalism and reasoning techniques developed in the context of database systems, a link that we intend to exploit in future work, to devise efficient methods for projection and progression, under restrictions inspired by pragmatic approaches in the database context.

Acknowledgments

The authors acknowledge the support of EU Projects FP7-ICT 257593 (ACSI) and FP7-ICT 318338 (OPTIQUE).

References

[Claßen and Lakemeyer, 2006] Jens Claßen and Gerhard Lakemeyer. Foundations for knowledge-based programs using ES. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 318–328, 2006.

[De Giacomo *et al.*, 2009] Giuseppe De Giacomo, Yves Lespérance, Hector J. Levesque, and Sebastian Sardina. *Multi-Agent Programming: Languages, Tools and Applications*, chapter IndiGolog: A High-Level Programming Language for Embedded Reasoning Agents, pages 31–72. Springer, 2009.

[De Giacomo *et al.*, 2012] Giuseppe De Giacomo, Yves Lespérance, and Fabio Patrizi. Bounded situation calculus action theories and decidable verification. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference (KR)*, 2012.

[Imielinski and Lipski, 1984] Tomasz Imielinski and Witold Lipski. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, September 1984.

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

[Levesque, 1996] Hector Levesque. What is planning in the presence of sensing? In *The Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI-96*, pages 1139–1146, Portland, Oregon, August 1996. American Association for Artificial Intelligence.

[Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. Forget it! In Russell Greiner and Devika Subramanian, editors, *Working Notes, AAAI Fall Symposium on Relevance*, pages 154–159, Menlo Park, California, 1994. American Association for Artificial Intelligence.

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

[Lin, 1996] Fangzhen Lin. Embracing causality in specifying the indeterminate effects of actions. In *AAAI/IAAI, Vol. 1*, pages 670–676, 1996.

[Liu and Lakemeyer, 2009] Yongmei Liu and Gerhard Lakemeyer. On first-order definability and computability of progression for local-effect actions and beyond. pages 860–866, 2009.

[Liu and Levesque, 2005] Y. Liu and H. J. Levesque. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proc. IJCAI-05*, Edinburgh, Scotland, August 2005.

[McCarthy and Hayes, 1969] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[Reiter, 1993] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64(2):337–351, 1993.

[Reiter, 2001] R. Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[Scherl and Levesque, 2003] R. Scherl and H. J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1–2):1–39, 2003.

[Shirazi and Amir, 2005] Afsaneh Shirazi and Eyal Amir. First-order logical filtering. In *Proc. of IJCAI-05*, pages 589–595, 2005.

[Thielscher, 1999] M. Thielscher. From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem. *Artificial Intelligence*, 111(1-2):277–299, July 1999.

[Vassos and Levesque, 2008] S. Vassos and H. J. Levesque. On the progression of situation calculus basic action theories: Resolving a 10-year-old conjecture. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1004–1009, 2008.

[Vassos *et al.*, 2008] S. Vassos, Gerhard Lakemeyer, and H. J. Levesque. First-order strong progression for local-effect basic action theories. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR-08)*, pages 662–272, 2008.