Progressing Basic Action Theories with Non-Local Effect Actions

Stavros Vassos
Department of Computer Science
University of Toronto
Toronto, Canada
stavros@cs.toronto.edu

Sebastian Sardina
School of Computer Science
RMIT University
Melbourne, Australia
ssardina@cs.rmit.edu.au

Hector Levesque
Department of Computer Science
University of Toronto
Toronto, Canada
hector@cs.toronto.edu

Abstract

In this paper we propose a practical extension to some recent work on the progression of action theories in the situation calculus. In particular, we argue that the assumption of *local-effect actions* is too restrictive for realistic settings. Based on the notion of *safe-range queries* from database theory and *just-in-time* action histories, we present a new type of action theory, called *range-restricted*, that allows actions to have non-local effects with a restricted range. These theories can represent incomplete information in the initial database in terms of *possible closures* for fluents and can be progressed by directly updating the database in an algorithmic manner. We prove the correctness of our method and argue for the applicability of range-restricted theories in realistic settings.

Introduction

One of the requirements for building agents with a pro-active behavior is the ability to reason about action and change. The ability to *predict* how the world will be after performing a sequence of actions is the basis for offline automated planning, scheduling, web-service composition, etc. In the situation calculus (McCarthy & Hayes 1969; Reiter 2001) such reasoning problems are examined in the context of the basic action theories (BATs). These are logical theories that specify the preconditions and effects of actions, and an initial database (DB) that represents the initial state of the world before any action has occurred.

A BAT can be used to solve offline problems as well as to equip a situated agent with the ability to *keep track* of the current state of the world. As a BAT is a static entity, in the sense that the axioms do not change over time, the reasoning about the current state is typically carried over using techniques based on *regression*, that transform the queries about the future into queries about the initial state (Reiter 2001). This is an effective choice for some applications, but a poor one for many settings where an agent may act autonomously for long periods of time. In those cases, it is mandatory that the BAT be (periodically) updated so that the initial DB be replaced by a new one reflecting the changes due to the actions that have already occurred. This is identified as the problem of *progression* for BATs (Lin & Reiter 1997).

In general, a DB in a BAT is an unrestricted first-order logical theory that offers great flexibility and expressiveness. The price to pay is high: for most realistic scenarios it is hard to find practical solutions. As far as progression is concerned, it was shown by Lin and Reiter (1997) that the updated DB requires second-order logic in the general case. For this reason, many restrictions on the BATs have been proposed so that the updated DB is first-order representable. It was recently shown that progression is practical provided actions are limited to have local effects only (Vassos, Gerhard, & Levesque 2008).

The restriction on so-called local-effects actions essentially means that all the properties of the world that may be affected by an action are directly specified by the arguments of the action. For example, an action that may affect two boxes box_1 and box_2 that are located next to the agent needs to explicitly mention them in the arguments of the action, e.g., $break(box_1, box_2)$. In that way, *global effects*, which are considered to be one of the reasons why progression may be second-order, are avoided all-together (e.g., the explosion of a bomb affecting all the objects in the world).

Clearly, the local-effect assumption is too restrictive for many realistic scenarios. For instance, the action of moving a container which causes all objects in it to be moved as well cannot be represented. Similarly, the effect of objects being broken when they are near an object that is exploded cannot be captured with local-effect actions. Such type of *indexical*, though not fully global-effect, information arises naturally in many real domains, e.g., consider the case of a non-player-character in a video game that needs to reason about the effects of moving a container object.

In this paper, we extend local-effect BATs to account for such kind of indexical information. To that end, we present what we call *range-restricted* BATs, that allow effects to be non-local but with a restricted range. For such theories, we describe a method for progression such that the new DB is first-order and finite, and we prove that the method is logically correct. To our knowledge, it is the first result on progression for BATs with an infinite domain, incomplete information, and sensing that goes beyond local-effect.

Formal preliminaries

The situation calculus (McCarthy & Hayes 1969) is a first-order logic language with some limited second-order features, designed for representing and reasoning about dynamically changing worlds. A *situation* represents a world history as a sequence of actions. The constant S_0 is used to denote the initial situation where no action has yet been per-

formed; sequences of actions are built using the function *do*: $do(a, s)$ denotes the situation resulting from performing action a in situation s. Relations whose truth values vary from situation to situation are called *fluents*, and are denoted by predicate symbols taking a situation term as their last argument (e.g., $Holding(x, s)$). A special predicate $Poss(a, s)$ is used to state that action a is executable in situation s; and special function $sr(a, s)$ denotes the (binary) sensing outcome of action a when executed in situation s (Scherl & Levesque 2003).

In this paper, we shall restrict our attention to a language $\mathcal{L}$ with a finite number of *relational* fluent symbols (i.e., no functional fluents) that only take arguments of sort object (apart their last situation argument), an infinite number of constant symbols of sort object, and a finite number of function symbols of sort action that take arguments of sort object. We adopt the following notation with subscripts and superscripts: α and a for terms and variables of sort action; σ and s for terms and variables of sort situation; t and x, y, z, w for terms and variables of sort object. Also, we use A for action function symbols, F, G for fluent symbols, and b, c, d, e, o for constants of sort object.

Often we will focus on sentences that refer to a particular situation. For this purpose, for any σ, we define the set of *uniform formulas in σ* to be all those (first-order or second-order) formulas in $\mathcal{L}$ that do not mention any other situation terms except for σ, do not mention *Poss*, and where σ is not used by any quantifier (Lin & Reiter 1997).

Basic action theories

Within the language, one can formulate action theories that describe how the world changes as the result of the available actions. We focus on a variant of the *basic action theories* (BAT) (Reiter 2001) of the following form:[1]

$$\mathcal{D} = \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{sr} \cup \mathcal{D}_0 \cup \mathcal{D}_{fnd} \cup \mathcal{E}, \text{ where:}$$

1. $\mathcal{D}_{ap}$ is the set of action precondition axioms (PAs), one per action symbol A, of the form $Poss(A(\vec{y}), s) \equiv \Pi_A(\vec{y}, s)$, where $\Pi_A(\vec{y}, s)$ is uniform in s.

2. $\mathcal{D}_{ss}$ is the set of successor state axioms (SSAs), one per fluent symbol F, of the form $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$, where $\Phi_F(\vec{x}, a, s)$ is uniform in s. SSAs capture the effects, and non-effects, of actions.

3. $\mathcal{D}_{sr}$ is the set of sensing-result axioms (SRAs), one for each action symbol A, of the form $sr(A(\vec{y}), s) = r \equiv \Theta_A(\vec{y}, r, s)$, where $\Theta_A(\vec{y}, r, s)$ is uniform in s. SRAs relate sensing outcomes with fluents.

4. $\mathcal{D}_{una}$ is the set of unique-names axioms for actions.

5. $\mathcal{D}_0$, the *initial database (DB)*, is a set of sentences uniform in S_0 that describe the initial situation S_0.

6. $\mathcal{D}_{fnd}$ is the set of domain independent axioms of the situation calculus, formally defining the legal situations.

7. $\mathcal{E}$ is a set of unique-names axioms for object constants.

Progression

We follow the definition of the so-called *strong progression* of (Vassos, Gerhard, & Levesque 2008); we only extend it slightly to account for sensing actions.

Let $\mathcal{D}$ be a BAT over relational fluents $F_1, \ldots, F_n$, and let $Q_1, \ldots, Q_n$ be second-order predicate variables. For any formula ϕ in $\mathcal{L}$, let $\phi\langle \vec{F} : \vec{Q} \rangle$ be the formula that results from replacing any fluent atom $F_i(t_1, \ldots, t_n, \sigma)$ in ϕ, where σ is a situation term, with atom $Q_i(t_1, \ldots, t_n)$.

Definition 1. Let $\mathcal{D}$ be a BAT over fluents $\vec{F}$, α an action of the form $A(\vec{c})$, and d a sensing result. Then, $Pro(\mathcal{D}, \alpha, d)$ is the following second-order sentence uniform in $do(\alpha, S_0)$:

$$\exists \vec{Q}. \mathcal{D}_0 \langle \vec{F} : \vec{Q} \rangle \wedge \Theta_A(\vec{c}, d, do(\alpha, S_0)) \wedge$$
$$\bigwedge_{i=1}^{n} \forall \vec{x}. F_i(\vec{x}, do(\alpha, S_0)) \equiv \big(\Phi_i(\vec{x}, \alpha, S_0)\langle \vec{F} : \vec{Q} \rangle\big).$$

We say that a set of formulas $\mathcal{D}_\alpha$ uniform in $do(\alpha, S_0)$ is a *strong progression* of $\mathcal{D}$ wrt (α, d) iff $\mathcal{D}_\alpha$ is logically equivalent to $Pro(\mathcal{D}, \alpha, d)$. ∎

The important property of strong progression is that $\mathcal{D}_\alpha \cup (\mathcal{D} - \mathcal{D}_0)$ is equivalent to the original theory $\mathcal{D}$ wrt answering *unrestricted* queries about $do(\alpha, S_0)$ and the future situations after $do(\alpha, S_0)$, even queries that quantify over situations. Although $Pro(\mathcal{D}, \alpha, e)$ is defined in second-order logic we are interested in cases where we can find a $\mathcal{D}_\alpha$ that is *first-order representable*. In the sequel, we shall present a restriction on $\mathcal{D}$ that is a sufficient condition for doing this as well as a method for computing a finite $\mathcal{D}_\alpha$.

Range-restricted basic action theories

In this section we present a new type of basic action theories such that $\mathcal{D}_0$ is a *database of possible closures* and the axioms in $\mathcal{D}_{ap}$, $\mathcal{D}_{ss}$, and $\mathcal{D}_{sr}$ are built on *range-restricted* formulas.

A database of possible closures

Intuitively, we treat each fluent as a *multi-valued function*, where the last argument of sort object is considered as the "*output*" and the rest of the arguments of sort object as the "*input*" of the function.[2] This distinction then is important as we require that $\mathcal{D}_0$ expresses incomplete information only about the output of fluents.

Definition 2. Let $V = \{e_1, \ldots, e_m\}$ be a set of constants and τ a fluent atom of the form $F(\vec{c}, w, S_0)$, where $\vec{c}$ is a vector of constants and w a variable. We say that τ has the *ground input* $\vec{c}$ and the *output* w. The *atomic closure* χ of τ on $\{e_1, \ldots, e_m\}$ is the following sentence:

$$\forall w. F(\vec{c}, w, S_0) \equiv (w = e_1 \vee \cdots \vee w = e_m).$$

The notion generalizes to the vector of atoms $\vec{\tau}$ and the vector of sets of constants $\vec{V}$, as the conjunction of each of the atomic closures of τ_i on V_i. A *possible closures axiom (PCA)* for $\vec{\tau}$ is a disjunction of closures of $\vec{\tau}$. We say that each atomic closure mentioned in the PCA is a *possible closure* wrt the PCA. ∎

The following is a straightforward property of closures.

Lemma 1. *Let ϕ be the closure of $\vec{\tau}$ and ψ be a closure of $\vec{\pi}$ on some appropriate vectors. Then $\phi \wedge \psi$ is a consistent closure iff for every i, j such that $\tau_i = \pi_j$, the atomic closure of τ_i in ϕ and the one of π_j in ψ are identical.*

---

[1]For legibility, we typically omit leading universal quantifiers.

[2]The notion of input-output arguments is similar to that of *modes* in logic programming (Apt & Pellegrini 1994). Also, the results obtained here generalize easily to multiple outputs.

A closure of $\vec{\tau}$ expresses complete information about the output of $\vec{\tau}$ while a PCA for $\vec{\tau}$ expresses disjunctive information it. For example, let $Near(x, y, s)$ represent that y is lying near the object x, and χ_1 be $\forall w.Near(bomb, w, S_0) \equiv (w = agent \lor w = box_1)$. Then, χ_1 is the atomic closure of $Near(bomb, w, S_0)$ on $\{agent, box_1\}$ which states that there are *exactly* two objects near the bomb, namely $agent$ and box_1. Similarly, let χ_2 be the closure of $Near(bomb, w, S_0)$ on $\{agent, box_2\}$. Then, $\chi_1 \lor \chi_2$ is a PCA for $Near(bomb, w, S_0)$ expressing that there are exactly two objects near the bomb, one being the agent and the other being either box_1 or box_2.

Next, let us define the form of the initial database $\mathcal{D}_0$.

Definition 3. A *database of possible closures (DBPC)* is a finite set of PCAs such that there is no fluent atom with a ground input that appears in more than one PCA. ∎

This implies that for every fluent atom τ with a ground input, either the output of τ is completely unknown in S_0 or there is a finite list of possible closures for τ that are explicitly listed in exactly one PCA.

Going back to the bomb example, let $Status(x, y, s)$ represent that the object x has the status y and let $\mathcal{D}_0$ be the following DBPC: $\{\chi_1 \lor \chi_2, \chi_3, \chi_4, \chi_5\}$, where χ_3 is the closure of $Status(agent, w, S_0)$ on $\{ready\}$, χ_4 the closure of $Status(box_1, w, S_0)$ on $\{closed\}$, χ_5 the closure of $Status(box_2, w, S_0)$ on $\{closed, broken\}$, and χ_1, χ_2 as before. Each sentence in $\mathcal{D}_0$ is a PCA: $\chi_1 \lor \chi_2$ lists two possible closures for $Near(bomb, w, S_0)$, while χ_3, χ_4, χ_5 list one possible closure and express complete information.

We now turn our attention to the so-called *possible answers* to a query $\gamma(\vec{x})$ wrt a DBPC $\mathcal{D}_0$.

Definition 4. Let $\mathcal{D}_0$ be a DBPC, and $\gamma(\vec{x})$ a first-order formula uniform in S_0 whose only free variables are in $\vec{x}$. The *possible answers* to γ wrt $\mathcal{D}_0$, denoted as $\mathsf{pans}(\gamma, \mathcal{D}_0)$, is the smallest set of pairs $(\vec{c}, \chi)$ such that:

- χ is a closure of some vector $\vec{\tau}$ s.t. $\mathcal{E} \cup \{\chi\} \models \gamma(\vec{c})$;
- χ is consistent with $\mathcal{D}_0$ and minimal in the sense that every atomic closure in χ is necessary. ∎

Intuitively, $\mathsf{pans}(\gamma, \mathcal{D}_0)$ is a way to characterize all the cases where the query formula $\gamma(\vec{x})$ is satisfied in a model of $\mathcal{D}_0$ for some instantiation of $\vec{x}$. For example, let $\gamma(x)$ be the query $Near(bomb, x, S_0)$. Then, $\mathsf{pans}(\gamma, \mathcal{D}_0)$ is the set $\{(agent, \chi_1), (box_1, \chi_1), (agent, \chi_2), (box_2, \chi_2)\}$.

It is important to observe that the possible answers to a query may be *infinite*. For instance, let $\gamma_1(x)$ be $Near(agent, x, S_0)$. Since nothing is said about the objects near the agent in $\mathcal{D}_0$, for every constant c in $\mathcal{L}$, $(c, \chi_c) \in \mathsf{pans}(\gamma_1(x), \mathcal{D}_0)$, where χ_c is the closure of $Near(agent, w, S_0)$ on $\{c\}$, i.e., there is always a model in which $Near(agent, c)$ would indeed hold. Similarly, let $\gamma_2(x)$ be $\neg Near(bomb, x, S_0)$. Then, $\mathsf{pans}(\gamma_2(x), \mathcal{D}_0)$ includes the infinite set $\{(c, \chi_1) \mid c \neq agent, c \neq box_1\}$, since everything but $agent$ or box_1 is far when χ_1 is assumed.

Formulas with finite possible answers

We distinguish two ways that the set of possible answers can be infinite. In the query γ_1 above, this happens because what is being queried is *completely* unknown in $\mathcal{D}_0$. In the second

query though, fluent atom $Near(bomb, w)$ is mentioned in some PCA and the infinite number of instantiations c for x are in fact due to the possible closure χ_4 of the PCA.

Our objective is to use $\mathcal{D}_0$ to answer queries for which the possible answers depend on the information that is explicitly expressed in the PCAs. This is captured with the following *just-in-time* assumption for formulas.

Definition 5. Let $\mathcal{D}_0$ be a DBPC and $\gamma(\vec{x})$ a first-order formula uniform in S_0 whose only free variables are in $\vec{x}$. Then $\gamma(\vec{x})$ is *just-in-time (JIT)* wrt $\mathcal{D}_0$ iff for every vector of constants $\vec{c}$, $\gamma(\vec{c})$ is consistent with $\mathcal{D}_0 \cup \mathcal{E}$ iff there exists a closure χ such that $\{\chi\} \cup \mathcal{E} \models \gamma(\vec{c})$, where χ is a conjunction of closures such that each conjunct is a possible closure wrt a PCA in $\mathcal{D}_0$. ∎

Assuming that a formula is JIT is not enough to avoid an infinite set of possible answers. We need also to ensure that it is *range-restricted* in the following sense.

Definition 6. The situation-suppressed formula γ in $\mathcal{L}$ is *safe-range* wrt a set of variables X according to the rules:

1. let $\vec{c}, \vec{c}_1, \vec{c}_2$ be a vectors of constants, c, d constants, and x, y distinct variables, then:
   - $x = c$ is safe-range wrt $\{x\}$;
   - $F(\vec{c}, d, S_0)$, $F(\vec{c}_1, x, \vec{c}_2, d, S_0)$ are safe-rage wrt $\{\}$;
   - $F(\vec{c}, y, S_0)$, $F(\vec{c}_1, x, \vec{c}_2, y, S_0)$ are safe-range wrt $\{y\}$;
2. if ϕ is safe-range wrt X_ϕ, ψ is safe-range wrt X_ψ then,
   - $\phi \lor \psi$ is safe-range wrt $X_\phi \cap X_\psi$;
   - $\phi \land \psi$ is safe-range wrt $X_\phi \cup X_\psi$;
   - $\neg \phi$ is safe-range wrt $\{\}$;
   - $\exists x \phi$ is safe-range wrt $X/\{x\}$ provided that $x \in X$;
3. no other formula is safe-range.

A formula is said to be *range-restricted* iff it is safe-range wrt the set of its free variables. ∎

For example, the formula $Near(x, y, S_0)$ is safe-range wrt $\{y\}$, but not range-restricted and not JIT wrt the $\mathcal{D}_0$ of our example. The formulas $Near(bomb, y, S_0)$ and $Near(bomb, y, S_0) \land Status(y, z, S_0)$ are range-restricted as well as JIT wrt $\mathcal{D}_0$.

We now state the main result of this section.

Theorem 1. *Let $\mathcal{D}_0$ be a DBPC and $\gamma(\vec{x})$ a first-order formula uniform in S_0 that is range-restricted and just-in-time wrt $\mathcal{D}_0$. Then, $\mathsf{pans}(\gamma, \mathcal{D}_0)$ is a finite set $\{(\vec{c}_1, \chi_1), \ldots, (\vec{c}_n, \chi_n)\}$ such that the following holds:*

$$\mathcal{D}_0 \cup \mathcal{E} \models \forall \vec{x}.\gamma(\vec{x}) \equiv \bigvee_{i=1}^{n} (\vec{x} = \vec{c}_i \land \chi_i).$$

Proof sketch. It suffices to prove a stronger lemma about the safe-range formulas as follows. Let $\gamma(\vec{x}, \vec{y})$ be a first-order formula that is just-in-time wrt $\mathcal{D}_0$, safe-range wrt the variables in $\vec{x}$, and does not mention any free variable other than $\vec{x}, \vec{y}$. Then for every constant vector $\vec{d}$ that has the same size as $\vec{y}$, $\mathsf{pans}(\gamma(\vec{x}, \vec{d}), \mathcal{D}_0)$ is a finite set $\{(\vec{e}_1, \chi_1), \ldots, (\vec{e}_n, \chi_n)\}$ such that the following holds:

$$\mathcal{D}_0 \cup \mathcal{E} \models \forall \vec{x}.\gamma(\vec{x}, \vec{d}) \equiv \bigvee_{i=1}^{n} (\vec{x} = \vec{e}_i \land \chi_i).$$

We prove this lemma by induction on the construction of the formulas γ. Since γ is safe-range wrt the variables in $\vec{x}$ we only need to consider the cases of the Definition 6. Due to space limitations we only show the case that $\gamma(x, y)$ is $F(\vec{c}_1, y, \vec{c}_2, x)$. Let d be an arbitrary constant of the language. Then $\gamma(x, d)$ is the formula $F(\vec{c}_1, d, \vec{c}_2, x)$. By the fact that $\gamma(x, y)$ is JIT wrt $\mathcal{D}_0$ it is not difficult to show that there is a PCA ϕ in $\mathcal{D}_0$ that mentions $F(\vec{c}_1, d, \vec{c}_2, w)$. Without loss of generality we assume that ϕ is a PCA for $F(\vec{c}_1, d, \vec{c}_2, w)$. We will show how to rewrite ϕ in the form that the lemma requires. The axiom ϕ has the form $\bigvee_{i=1}^{n} \chi_i$, where each χ_i is an atomic closure of $F(\vec{c}_1, d, \vec{c}_2, w)$ on some set of constants $\{e_1, \ldots, e_m\}$, i.e, a sentence of the form $\forall w.F(\vec{c}_1, d, \vec{c}_2, w) \equiv w = e_1 \vee \ldots \vee w = e_m$. For each χ_i of this form let χ_i' be the formula $\bigvee_{j=1}^{m}(x = e_j \wedge \chi_i)$, and let ϕ' be $\forall x.F(\vec{c}_1, d, \vec{c}_2, x) \equiv \bigvee_{i=1}^{n} \chi_i'$. It suffices to show that $\mathcal{D}_0 \cup \mathcal{E} \models \phi'$. Let M be an arbitrary model of $\mathcal{D}_0 \cup \mathcal{E}$. Since ϕ is a sentence in $\mathcal{D}_0$ it follows that $M \models \phi$. By the definition of a possible closures axiom and the Lemma 1 it follows that there is exactly k, $1 \leq k \leq n$, such that $M \models \chi_k$. Observe that if we simplify χ_k to *true* and all the other χ_i to *false* in ϕ' we obtain the sentence χ_k. Therefore, $M \models \phi'$ and since M was an arbitrary model of $\mathcal{D}_0 \cup \mathcal{E}$, it follows that $\mathcal{D}_0 \cup \mathcal{E} \models \phi'$. Also, by the Definition 4 and the structure of ϕ' it follows that the set $\mathsf{pans}(\gamma(x, d), \mathcal{D}_0)$ is the set that the lemma requires. $\square$

In other words, the range-restricted and the JIT assumptions on queries are sufficient conditions to guarantee *finitely* many possible answers. The idea then is to build action theories from range-restricted formulas and allow progression to take place only when the JIT assumption also holds. In this case we shall show in the next session that we are able to effectively progress $\mathcal{D}_0$ in a logically correct way.

First, we assume that the formulas $\Phi_F(\vec{x}, a, s)$ of SSAs have the usual general form (Reiter 2001):
$$\gamma_F^+(\vec{x}, a, s) \vee (F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s)),$$
where γ_F^+ and γ_F^- characterize the positive and negative effects of actions. A *range-restricted* BAT is built on formulas such that when instantiated with any action argument α and any sensing result e, they become range-restricted.

Definition 7. An SSA for F is *range-restricted* iff $\gamma_F^+(\vec{x}, a, s)$ and $\gamma_F^-(\vec{x}, a, s)$ are disjunctions of formulas of the form:
$$\exists \vec{z}(a = A(\vec{y}) \wedge \phi(\vec{y}, \vec{w}, s)),$$
where $\vec{z}$ corresponds to the variables in $\vec{y}$ but not in $\vec{x}$, $\vec{w}$ to the ones in $\vec{x}$ but not in $\vec{y}$, and $\phi(\vec{x}, \vec{w}, s)$, called a *context formula*, is such that $\phi(\vec{c}, \vec{w}, S_0)$ is range-restricted for any $\vec{c}$. Similarly, an SRA for A is *range-restricted* iff $\Theta_A(\vec{c}, d, S_0)$ is range-restricted for any $\vec{c}$ and d. A *range-restricted basic action theory (RR-BAT)* is a BAT such that all axioms in $\mathcal{D}_{ss}, \mathcal{D}_{sr}$ are range-restricted and $\mathcal{D}_0$ is a DBPC. $\blacksquare$

For example, consider an SSA for $Status(x_1, x_2, s)$. The context formula in γ_{Status}^+ that refers to the action of the bomb exploding may be as follows:
$$a = expl \wedge Near(bomb, x_1, s) \wedge x_2 = broken,$$
This has the effect of setting the "broken" status to all objects near the bomb. Note that the action *expl* has no arguments, and that the context formula is range-restricted. It is

easy to verify that the formula is JIT wrt $\mathcal{D}_0$ as well. The same holds for a context formula in γ_{Status}^+ that removes any other status for all the affected objects:
$$a = expl \wedge Near(bomb, x_1, s) \wedge Status(x_1, x_2, s) \wedge x_2 \neq broken.$$

Just-in-time progression

The RR-BATs are defined so that the axioms in $\mathcal{D}_{ss}, \mathcal{D}_{sr}$ are built on range-restricted formulas. We now show that under a *just-in-time* assumption there is a finite set of ground fluent atoms that may be affected. The intuition is that in this case we can progress $\mathcal{D}_0$ by appealing to the techniques in (Vassos, Gerhard, & Levesque 2008) that work when the set of fluents that may be affected is fixed by the action.

The progression method for the general case

The next definition captures the condition under which our method for progression is logically correct.

Definition 8. An RR-BAT $\mathcal{D}$ is *just-in-time (JIT)* wrt the ground action α and the sensing result d iff for all fluent symbols F, $\gamma_F^+(\vec{x}, \alpha, S_0)$ and $\gamma_F^-(\vec{x}, \alpha, S_0)$ are JIT wrt $\mathcal{D}_0$, and $\Theta_A(\vec{c}, d, S_0)$ is JIT wrt $\mathcal{D}_0$, where α is $A(\vec{c})$. $\blacksquare$

We introduce the following notation.

Definition 9. Let $\mathcal{D}$ be an RR-BAT that is JIT wrt the ground action α and the sensing result d. The *context set* of (α, d) wrt $\mathcal{D}$, denoted as $\mathcal{J}$, is the set of all the fluent atoms $F(\vec{e}, w, S_0)$ such that one of the following is true:[3]

1. for some b, χ, the pair $(\langle \vec{e}, b\rangle, \chi)$ is a possible answer to $\gamma_F^*(\langle \vec{x}, w\rangle, \alpha, S_0)$ wrt $\mathcal{D}_0$;

2. for some $\vec{o}, b, \chi$, the pair $(\langle \vec{o}, b\rangle, \chi)$ is a possible answer to $\gamma_F^*(\langle \vec{x}, y\rangle, \alpha, S_0)$ wrt $\mathcal{D}_0$ and $F(\vec{e}, w, S_0)$ appears in χ;

3. for some χ, the pair $(\emptyset, \chi)$ is a possible answer to $\Theta_A(\vec{c}, d, S_0)$ wrt $\mathcal{D}_0$, where α is the term $A(\vec{c})$ and $\emptyset$ the empty vector and $F(\vec{e}, w, S_0)$ appears in χ. $\blacksquare$

Intuitively, the context set $\mathcal{J}$ specifies all those atomic closures that need to be updated after the action is performed (case 1) as well as those on which the change is conditioned on (case 2), and the atomic closures for which some condition is sensed to be true (case 3). The important property of $\mathcal{J}$, which follows from Theorem 1, is that it is a *finite* set.

Lemma 2. *Let $\mathcal{D}$ be an RR-BAT that is JIT wrt the ground action α and the sensing result d. Then the context set of (α, d) wrt $\mathcal{D}$ is a finite set.*

We now define the $\mathcal{J}$-models which provide a way of separating $\mathcal{D}_0$ into two parts: one that remains unaffected after the action is performed and one that needs to be updated.

Definition 10. Let $\mathcal{J} = \{\tau_1, \ldots, \tau_n\}$ be the context set of (α, d) wrt a RR-BAT $\mathcal{D}$. A *$\mathcal{J}$-model χ* is a closure of the vector $\langle \tau_1, \ldots, \tau_n \rangle$ such that for every i, $1 \leq i \leq n$, the atomic closure of τ_i in χ is a possible closure wrt some PCA in $\mathcal{D}_0$. $\blacksquare$

Note that there are finitely many $\mathcal{J}$-models. The disjunction ϕ then of all the $\mathcal{J}$-models is a larger PCA that corresponds to the "cross-product" of the PCAs in $\mathcal{D}_0$ that capture the same information about $\vec{\tau}$. Observe that ϕ corresponds to

---

[3]Whenever the notation γ^* is used, γ^* can be either γ^+ or γ^-.

the part of $\mathcal{D}_0$ that needs updating. The intuition then is that we can progress $\mathcal{D}_0$ by progressing each of the $\mathcal{J}$-models.

Definition 11. Let $\mathcal{D}$ be an RR-BAT that is JIT wrt the ground action α and sensing result d, $\mathcal{J}$ the context set of (α, d), and χ a $\mathcal{J}$-model, where χ is the closure of $\langle F_1(\vec{c}_1, w, S_0), \dots, F_n(\vec{c}_n, w, S_0) \rangle$ on $\langle V_1, \dots, V_n \rangle$. The *progression* of χ wrt (α, d) is the closure $\psi_1 \wedge \cdots \wedge \psi_n$, where ψ_i is the closure of $F_i(\vec{c}_i, w, S_0)$ on $(V_i \cup \Gamma_i^+)/\Gamma_i^-$ and Γ_i^* is the following set of constants:

$$\{e \mid (\langle \vec{c}_i, e \rangle, \omega) \in \mathsf{pans}(\gamma_{F_i}^*(\langle \vec{x}, w \rangle, \alpha, S_0)), \omega \wedge \chi \not\models \bot\}.$$

The $\mathcal{J}$-model χ is *filtered* iff for all possible answers $(\vec{o}, \phi)$ to $\Theta_A(\vec{c}, d, S_0)$ wrt $\mathcal{D}_0$, where $\alpha = A(\vec{c})$, $\chi \wedge \phi$ is inconsistent. ■

Each of the $\mathcal{J}$-models χ is updated based on the possible answers of the formulas γ_F^* in $\mathcal{D}_{ss}$. For every possible answer $(\vec{o}, \omega)$ of the instantiated γ_F^*, the atom $F(\vec{o})$ is either removed or added to the closure provided that the condition ω for the change is consistent with the $\mathcal{J}$-model χ in question. Moreover, a $\mathcal{J}$-model may be filtered if it is not consistent with the conditions that are implied by the sensing result d.

We now state the main result of this section that illustrates how the new database is constructed from $\mathcal{D}_0$.

Theorem 2. *Let $\mathcal{D}$ be an RR-BAT that is consistent and JIT wrt the ground action α and the sensing result d, $\mathcal{J}$ the context set of (α, d) wrt $\mathcal{D}$, $\{\chi_1, \dots, \chi_n\}$ the set of all the $\mathcal{J}$-models that are not filtered, and $\{\phi_1, \dots, \phi_m\}$ the set of all PCAs in $\mathcal{D}_0$ that do not have any atoms in common with any $\mathcal{J}$-model. Let $\mathcal{D}_\alpha$ be the following set:*

$$\{\bigvee_{i=1}^{n} \psi_i, \phi_1, \dots, \phi_m\},$$

where ψ_i is the progression of χ_i wrt (α, d). Then, the set $\mathcal{D}_\alpha(S_0/do(\alpha, S_0))$ is a strong progression of $\mathcal{D}$ wrt (α, d), where $\mathcal{D}_\alpha(\sigma/\sigma')$ denotes the result of replacing every occurrence of σ in every sentence in $\mathcal{D}_\alpha$ by σ'.

Observe that the progression of $\mathcal{D}_0$ is again a DBPC.

A practical case

Our method of progression is based on the ability to compute possible answers. The time complexity of the method, as well as the size of $\mathcal{D}_\alpha$, is dominated by the size of the sentence $\bigvee_i \phi_i$ in Theorem 2. Roughly speaking, we do two things that have a high computational cost: first, we compute $\mathsf{pans}(\gamma, \mathcal{D}_0)$ for formulas γ in $\mathcal{D}_{ss}, \mathcal{D}_{sr}$, and second, we combine the answers in a way that is similar to a cross-product.

In order to give some insight on the practicality of our method, we examine the case that the formulas γ that need to be evaluated are similar to the so-called *conjunctive queries* (Abiteboul, Hull, & Vianu 1994), in particular, formulas of the form $\exists \vec{x}(\phi_1 \wedge \cdots \wedge \phi_n)$, where ϕ_i is a possibly non-ground fluent atom with variables that may not be in $\vec{x}$.

Given a conjunctive query γ as input and a DBPC $\mathcal{D}_0$, Algorithm 1 checks whether γ is range-restricted and JIT wrt $\mathcal{D}_0$, and if so, computes the set $\mathsf{pans}(\gamma, \mathcal{D}_0)$. The algorithm works by selecting a fluent atom for which a finite-range assumption can be made (line 4 & 8), simplifying γ wrt this

---

Algorithm 1 $\mathsf{pans}(\gamma, \mathcal{D}_0)$

```
 1: if γ is the empty conjunction then
 2:     return {(∅, ⊤)}        // query reduced to ⊤
 3: end if
 4: Δ = {F(c̄, t, S₀) ∈ γ | F(c̄, w, S₀) is mentioned in 𝒟₀}
 5: if Δ = ∅ then
 6:     return failure        // no fluent to continue
 7: else
 8:     Pick F(c̄, t, S₀) ∈ Δ   // arbitrary selection
 9:     X := ∅                  // init answer set
10:     for all χ_F = F(c̄, w, S₀) ≡ w = d₁ ∨ ... ∨ dₙ ∈ 𝒟₀
        do
11:         if t is a variable then
12:             Γ = {d₁, ..., dₙ}
13:         else
14:             Γ = {d₁, ..., dₙ} ∩ {t}
15:         end if
16:         for all constants e ∈ Γ do
17:             θ' := {t/e | t is variable}
18:             Y := pans(γθ' \ {F(c̄, e, S₀)}, 𝒟₀)
19:             if Y = failure then
20:                 return failure   // propagate failure
21:             else
22:                 W := {(θθ', χ ∧ χ_F) | (θ, χ) ∈ Y, 𝒟₀ ∪ {χ ∧
                    χ_F} ⊭ ⊥}              // merge results
23:                 X := X ∪ W   // update current set
24:             end if
25:         end for
26:     end for
27:     X := {(θ|ₓ̄, χ)|(θ, χ) ∈ X, x̄ are the free variables in γ}
28:     return X
29: end if
```

---

atom, and recursively finding the possible answers for the simplified formula (line 18) until all atoms in γ have been selected (line 1). Instead of working with vectors of terms, the algorithm computes bindings for all variables.

It turns out that the algorithm is a sound and complete way for computing the possible answers of range-restricted and JIT formulas, when these are conjunctive queries.

Theorem 3. *Let $\mathcal{D}_0$ be a DBPC and γ a first-order conjunctive query uniform in S_0. Then, Algorithm 1 always terminates with inputs γ and $\mathcal{D}_0$, and moreover, if γ is range-restricted and JIT wrt $\mathcal{D}_0$, it returns the set $\mathsf{pans}(\gamma, \mathcal{D}_0)$.*

The conjunctive queries are expressive enough to represent basic features of practical domains. For example, the context formula of γ_{Status}^+ that we examined earlier, namely $Near(bomb, x_1, s) \wedge x_2 = broken$, is a simple conjunctive query. As another example consider an agent living in a grid-world, typical of many video games. The agent may reason about its next location $Loc(z, do(a, s))$ after doing action a by using an SSA whose positive effect $\gamma_{Loc}^+(z, a, s)$ contains the following disjunct:

$$a = moveFwd \wedge$$
$$\exists x \exists y (Dir(y, s) \wedge Loc(x, s) \wedge Adj(x, y, z, s) \wedge Clear(z, s)).$$

That is, when moving forward, the agent is in location z if z is the adjacent cell to its current location x towards its current direction y (e.g., north, east), and z is not blocked with

an obstacle. Clearly, this positive effect relies on multiple indexical information and action *moveFwd* is not local-effect.

Algorithm 1 can easily be extended to handle equalities as well as negated atoms. The first case can be easily addressed via standard unification procedures. For negative literals the idea is to collect also the set Δ^- of *ground* literals of the form $\neg F(\vec{c}, d, S_0)$ such that $F(\vec{c}, w, S_0)$ is mentioned in $\mathcal{D}_0$. When a negative literal is selected, the algorithm works in the same way as for the ground positive literal except that it iterates over the possible closures of $F(\vec{c}, w, S_0)$ for which $F(\vec{c}, d, S_0)$ is *not* true. (Observe that this is similar to the way logic-programming implementation techniques for *negation as failure* (Apt & Pellegrini 1994).)

Finally, a comment about the complexity of Algorithm 1 and progression. Let ℓ be the size of the largest closure in $\mathcal{D}_0$ and k the maximum number of possible closures in a PCA in $\mathcal{D}_0$. Then, Algorithm 1 runs in time $O(|\gamma|^{k\ell})$: there are $k\ell$ value-closure pairs to be tested for each atom in γ. With respect to progression, this implies that, in the worst case, the size of the new database $\mathcal{D}_\alpha$ may be exponential to the size of $\mathcal{D}_0$. Nevertheless, we expect the size of $\mathcal{D}_\alpha$ to be manageable in practical scenarios like the previous example, where the expressiveness of γ and $\mathcal{D}_0$ is mostly used to answer queries that require indexical reasoning.

Related and future work

The notion of progression for BATs was first introduced by Lin and Reiter (1997). The version we use here is due to Vassos *et al* (2008) which we extended slightly to account for sensing. Lin and Reiter (1997) suggested some strong syntactic restrictions on the BATs that allow for a first-order progression, while Vassos and Levesque (2008) suggested a restriction on the queries. 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) later showed that under this assumption a correct first-order progression can be computed by updating a finite $\mathcal{D}_0$. Our restriction of Definition 7 is similar. The main difference is that we do not require that the arguments $\vec{x}$ of the fluent F are included in the arguments $\vec{y}$ of the action, thus handling cases like the *moveFwd* example. To stay practical though we had to restrict the structure of $\mathcal{D}_0$. Finally, similar to the notion of progression, Shirazi and Amir (2005) proposed *logical filtering* as a way to progress $\mathcal{D}_0$ and proved that their method is correct for answering uniform queries.

The notion of possible closures is a generalization of the *possible values* of Vassos and Levesque (2007). The notions of the safe-range and range-restricted queries come from the database theory where this form of "safe" queries has been extensively studied (Abiteboul, Hull, & Vianu 1994). The notion of just-in-time formulas was introduced for a different setting in (De Giacomo, Levesque, & Sardina 2001) and, in our case, is also related to the *active domain* of a database (Abiteboul, Hull, & Vianu 1994). Outside of the situation calculus, Thielscher (1999) defined a dual representation for BATs based on state update axioms that explicitly define the direct effects of each action, and investigated progression in this setting. Unlike our work where the sentences in $\mathcal{D}_0$ are replaced with an updated version, there, the update relies on expressing the changes using constraints.

Conclusions

In this paper, we proposed a new type of basic action theories, where the initial description is a set of *possible closures* and the effects of actions have a *restricted range*. For these theories, called *range-restricted*, we presented a method that computes a finite first-order progression by directly updating the initial database, and proved its correctness. To the best of our knowledge, it is the first result on the progression of basic action theories with an infinite domain, incomplete information, and sensing that goes beyond the local-effect assumption. We argue that the type of indexical information that our theories can handle arises naturally in real domains, e.g., when an agent needs to reason about the effects of moving a container. We considered also a practical restriction that is typical in logic-programming, and presented an algorithm for the task that our progression method relies on, namely computing possible answers. Our next step is to evaluate the approach by relying on logic-programming frameworks and recent work on inconsistent/incomplete databases (e.g., Fuxman *et al* (2005)).

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1994. *Foundations of Databases : The Logical Level.* Addison Wesley.

Apt, K., and Pellegrini, A. 1994. On the occur-check free Prolog program. *ACM Toplas* 16(3):687–726.

De Giacomo, G.; Levesque, H. J.; and Sardina, S. 2001. Incremental execution of guarded theories. *Computational Logic* 2(4):495–525.

Fuxman, A.; Fazli, E.; and Miller, R. J. 2005. Conquer: efficient management of inconsistent databases. In *Proc. of SIGMOD-05*, 155–166. ACM Press.

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

Liu, Y., and Levesque, H. J. 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proc. of IJCAI.*

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

Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dyn. Sys.* MIT Press.

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

Shirazi, A., and Amir, E. 2005. First-order logical filtering. In *Proc. of IJCAI-05*, 589–595.

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

Vassos, S., and Levesque, H. 2007. Progression of situation calculus action theories with incomplete information. In *Proc. IJCAI*, 2024–2029.

Vassos, S., and Levesque, H. J. 2008. On the progression of situation calculus basic action theories: Resolving a 10-year-old conjecture. In *Proc. of AAAI.*

Vassos, S.; Gerhard, L.; and Levesque, H. J. 2008. First-order strong progression for local-effect basic action theories. In *Proc. of KR*, 662–272.