

Progression of Situation Calculus Action Theories with Incomplete Information

Stavros Vassos and Hector Levesque

Department of Computer Science

University of Toronto

Toronto, ON, CANADA

{stavros,hector}@cs.toronto.edu

Abstract

In this paper, we propose a new progression mechanism for a restricted form of incomplete knowledge formulated as a basic action theory in the situation calculus. Specifically, we focus on functional fluents and deal directly with the *possible values* these fluents may have and how these values are affected by both physical and sensing actions. The method we propose is logically complete and can be calculated efficiently using database techniques under certain reasonable assumptions.

1 Introduction

This paper deals with the problem of getting a robot or agent to reason efficiently about action and change under the following three conditions:

1. the agent has *incomplete knowledge* about certain changing values in the world (fluents);
2. the agent has both physical (world-changing) and sensing (knowledge-producing) actions at its disposal;
3. the agent is *long-lived* and may have performed thousands or even millions of actions to date.

These conditions are typical of robots or agents that must operate autonomously over long periods of time, such as the agents found in role-playing games. For example, an agent may not know the relative locations of other agents or important objects, but may be able to move and to sense what agents and objects are nearby. The challenge is to reason efficiently about the overall state of the world in this setting, taking into account what has been done and what has been sensed, as a large number of such actions are performed.

There is a significant body of work on logical formalisms for dynamic domains that are appropriate here, such as the situation calculus [McCarthy and Hayes, 1969], the event calculus [Kowalski and Sergot, 1986], and the fluent calculus [Thielscher, 1999]. In logical terms, we seek an efficient solution to the *projection problem* [Reiter, 2001]: given an action theory that specifies the preconditions and effects of actions (including sensing), and a knowledge base about the initial state of the world, determine whether or not some condition holds after a given sequence of actions has been per-

formed. This is a fundamental reasoning problem and a necessary prerequisite to other forms of reasoning in dynamic domains such as planning and high-level program execution. While the projection problem can be solved by *regression* or by *progression* [Lin, 2007], it is generally agreed that regression alone is impractical when the number of actions is large [Lakemeyer and Levesque, 2007].

While formalisms based on logic can represent a general form of incomplete knowledge in a straightforward way, they would then require systems based on some form of theorem-proving. It is quite typical of the practical *implementations* of these formalisms to forego this generality and either insist on complete knowledge or to employ an inference mechanism that is logically incomplete. For the situation calculus, the former approach was taken in the practical implementations of the GOLOG agent programming language [Levesque *et al.*, 1997; De Giacomo and Levesque, 1999]; the latter was taken by Liu and Levesque [2005] where incomplete knowledge was dealt with using a logically incomplete form of progression. In the FLUX implementation of the fluent calculus [Thielscher, 2004] incomplete knowledge is dealt with using constraint rules and a logically incomplete constraint solver.

In this paper, we propose a new progression mechanism for a restricted form of incomplete knowledge formulated as a basic action theory in the situation calculus. Specifically, we focus on functional fluents and deal directly with the *possible values* these fluents may have and how these values are affected by both physical and sensing actions. Our approach handles a restricted form of (functional) *local-effect* action theories but generalizes the Liu and Levesque account where progression is logically complete only for action theories that are *context complete*, that is, where there is complete knowledge about the context of any context-dependent successor state axioms. Our method can be calculated efficiently using database techniques (related to those of [Antova *et al.*, 2007]) under certain reasonable assumptions. To our knowledge, no other logical formalism handles progression (and therefore the projection task) in a way that is both logically complete and practical under the three conditions listed above.

The rest of the paper is organized as follows. In Section 2, we review the situation calculus notation, as well as the definitions of projection and progression. In Section 3, we discuss the notion of a possible value and define the local-effect and bounded action theories. In Section 4, we present a

progression mechanism for local-effect and bounded theories and in Section 5, we discuss the complexity of the approach. In Section 6, we discuss related work. Finally, in Section 7, we draw conclusions and discuss future work.

2 Situation calculus theories and progression

The situation calculus [McCarthy and Hayes, 1969] is a first-order language with some limited second-order features specifically designed for representing dynamically changing worlds. The language has disjoint sorts for 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 been performed. 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 . A *fluent* is a predicate or function whose last argument is a situation, and thus whose value can change from situation to situation. Actions need not be executable in all situations, and the predicate $Poss(a, s)$ states that action a is executable in situation s . Actions have sensing results, and the function $sr(a, s)$ denotes the sensing result of action a when executed in situation s [Scherl and Levesque, 2003].

Here, we restrict our attention to a situation calculus language \mathcal{L} that includes only the function symbols do, sr, S_0 as described above, the predicate symbol $Poss$, a finite number of functional¹ fluent symbols \mathcal{F} , a finite number of action function symbols \mathcal{A} whose arguments are of sort object, and a finite number of constant symbols \mathcal{C} of sort object.

Within \mathcal{L} , we 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* [Reiter, 2001]. An action theory \mathcal{D} has the following form:²

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

1. \mathcal{D}_{ap} is a set of action precondition axioms, one per action symbol A , of the form $Poss(A(\vec{y}), s) \equiv \Pi_A(\vec{y}, s)$.
2. \mathcal{D}_{ss} is a set of successor state axioms (SSAs), one for each fluent symbol f , of the form $f(\vec{x}, do(a, s)) = v \equiv \Phi_f(\vec{x}, a, v, s)$. SSAs characterize the conditions under which the fluent has a specific value at situation $do(a, s)$ as a function of situation s .
3. \mathcal{D}_{sr} is a set of sensing-result axioms (SRAs), one for each action symbol A , of the form $sr(A(\vec{y}), s) = r \equiv \Theta_A(\vec{y}, r, s)$. SRAs relate sensing outcomes with fluents.
4. \mathcal{D}_{una} is the set of unique-names axioms for actions: $A(\vec{x}) \neq A'(\vec{y})$, and $A(\vec{x}) = A(\vec{y}) \supset \vec{x} = \vec{y}$.
5. \mathcal{D}_0 is a set of axioms describing the initial situation S_0 .
6. \mathcal{D}_{fnd} consists of the axioms for equality and a set of domain independent foundational axioms which formally define legal situations.

¹Relational fluents can be represented by functional fluents that are forced to take only values *true/false*. Non-fluent functions and predicates can be formalized as unchanging fluents [Reiter, 2001].

²We will be omitting leading universal quantifiers.

7. \mathcal{E} is a set of unique-names axioms for constants along with a domain closure axiom for sort object.

To describe actions along with their sensing results, we use terminology as in [De Giacomo *et al.*, 2001]. A *history* h is a sequence $(a_1, r_1) \cdot \dots \cdot (a_n, r_n)$, where each a_i is a ground action term and r_i is a constant that represents the action's sensing outcome. We assume that non-sensing actions always return the fixed result c . We use $end(h)$ as an abbreviation for the ground situation term that corresponds to h . This is defined as follows: $end(h)$ is S_0 for the empty history, and $end(h \cdot (a, r))$ is $do(a, end(h))$. Similarly, $\mathcal{S}(h)$ is the set of atoms that describe the sensing results in h . $\mathcal{S}(h)$ is an abbreviation for $\{sr(a_i, end(h_i)) = r_i \mid 1 \leq i \leq n\}$, where h_i is the sub-history up to action a_i , $(a_1, r_1) \cdot \dots \cdot (a_{i-1}, r_{i-1})$.

A *situation-suppressed* expression is a formula in the language without $Poss$ and sr , and with all situation arguments suppressed. $\phi[\sigma]$ denotes the situation calculus formula obtained from ϕ by restoring the situation argument σ into all fluents in ϕ . We use a, b, c, d, e to range over \mathcal{C} and f, g to range over \mathcal{F} . A *primitive* fluent term is a ground term that mentions at most one fluent symbol, e.g. $f(c, S_0)$, and a *primitive atom* is one that mentions at most one primitive fluent term, e.g. $f(c, S_0) = d$, $f(c, S_0) = x$. A *primitive formula* (sentence) is a formula (sentence) that mentions only primitive atoms. Note that quantification is allowed in primitive formulas but not when the variable appears as an argument of a fluent. Finally, we use τ, u to denote a situation-suppressed primitive fluent term.

In this notation, the *projection task* [Reiter, 2001] is the following: given a history h and a situation-suppressed sentence ϕ , determine whether or not $\mathcal{D} \cup \mathcal{S}(h) \models \phi[end(h)]$. We propose to solve it for certain restricted action theories using a form of *progression*, which we define along the lines of [Lin and Reiter, 1997] and [Liu and Levesque, 2005]. Let \mathcal{I} be a *standard model* iff \mathcal{I} satisfies \mathcal{E} .

Definition 1. Let \mathcal{I} be a standard model of \mathcal{D}_0 . \mathcal{I}' is a *successor model* of \mathcal{I} wrt a world-changing action $A(\vec{e})$ iff \mathcal{I}' is a standard model and for any primitive fluent atom $f(\vec{c}) = d[S_0]$, $\mathcal{I}' \models f(\vec{c}) = d[S_0]$ iff $\mathcal{I} \models \Phi_f(\vec{c}, A(\vec{e}), d)[S_0]$. For a sensing action $A(\vec{e})$ with result b , \mathcal{I}' is a successor model of \mathcal{I} iff \mathcal{I}' is a standard model, and $\mathcal{I}' \models f(\vec{c}) = d[S_0]$ iff $\mathcal{I} \models f(\vec{c}) = d[S_0]$ and $\mathcal{I} \models \Theta_A(\vec{e}, b)[S_0]$.

Definition 2. \mathcal{D}'_0 is a *progression* of \mathcal{D}_0 wrt $(A(\vec{e}), b)$ iff the standard models of \mathcal{D}'_0 are exactly the successor models of \mathcal{D}_0 wrt $(A(\vec{e}), b)$.

One step of progression is a solution to one step of projection in the sense that if $h = (a, r)$ and $\mathcal{D} \cup \mathcal{S}(h)$ is consistent, then $\mathcal{D} \cup \mathcal{S}(h) \models \phi[end(h)]$ iff $\mathcal{D}'_0 \models \phi[S_0]$.

3 Possible values and bounded action theories

Although we are assuming that every fluent is functional, and thus has a single value, we do not assume that this value is known. For example, to say that the value of the fluent *age* in situation S_0 might be 19, 20 or 21, \mathcal{D} may contain the following disjunction: $(age = 19 \vee age = 20 \vee age = 21)[S_0]$. Thus, we say that 19, 20 and 21 are the possible values that

the fluent *age* can take.³ As another example, suppose that we have the following sentence in \mathcal{D} :

$$(f = 1 \vee f = 0) \wedge (g = 1 \vee g = 0) \wedge (f \neq g)[S_0].$$

In this case, the fluents f and g each have two possible values. But because of the constraint between them, we should consider the *pair* $\langle f, g \rangle$ which has two (rather than four) possible values: $\langle 1, 0 \rangle$ and $\langle 0, 1 \rangle$. More generally, we have the following definition:

Definition 3. Let $\vec{\tau}$ be a tuple of distinct situation-suppressed primitive fluent terms. A finite set of constant tuples $V = \{\vec{c}_1, \dots, \vec{c}_n\}$, each of the same length as $\vec{\tau}$ is the *set of possible values* for $\vec{\tau}$ at history h iff V is the smallest set such that $\mathcal{D} \cup \mathcal{S}(h) \models (\bigvee_{i=1}^n \vec{\tau} = \vec{c}_i)[end(h)]$. Each element of V is a *possible value* of $\vec{\tau}$ at h .

Note that V is the empty set only if $\mathcal{D} \cup \mathcal{S}(h)$ is inconsistent.

When we say that a sentence is possibly true, we mean that it is entailed for some possible value of the fluents in the sentence, taking into account that some fluents may be grouped with others because of constraints.

Definition 4. Let h be a history and let ϕ be a situation-suppressed primitive sentence. Let $\mathcal{T} = \{\vec{\tau}_1, \dots, \vec{\tau}_k\}$ be a set of tuples of situation-suppressed primitive fluent terms such that every primitive fluent term in ϕ appears in exactly one $\vec{\tau}_i$. We say that ϕ is *possibly true* at history h wrt \mathcal{T} iff $\mathcal{E} \models \phi'$, where ϕ' is ϕ with each $\vec{\tau} \in \mathcal{T}$ replaced by one of its possible values at h . (A tuple $\langle \tau_1, \dots, \tau_n \rangle$ is replaced by $\langle c_1, \dots, c_n \rangle$ by replacing each term τ_i by the corresponding c_i .)

We say that ϕ is *known to be true* at h wrt \mathcal{T} iff $\neg\phi$ is not possibly true at h wrt \mathcal{T} .

It is not hard to see that a sentence ϕ is known to be true at h iff for *every* possible value of the fluents at h , the substituted ϕ' obtained is indeed entailed. But this is not the same thing as saying that $\phi[end(h)]$ itself is entailed by $\mathcal{D} \cup \mathcal{S}(h)$. This depends on how we group the fluent terms into tuples.

Example 1. Consider a theory \mathcal{D} containing the above sentence about f and g . Let h be the empty history and let ϕ be $(f \neq g)$. Then we have the following:

1. $\mathcal{D} \cup \mathcal{S}(h) \models \phi[end(h)]$;
2. ϕ is not known to be true at h wrt $\{f, g\}$;
3. ϕ is known to be true at h wrt $\{\langle f, g \rangle\}$.

What is going on here is that for $\mathcal{T} = \{f, g\}$ we look at the possible values of f and g independently, including cases where $f = g$. For $\mathcal{T} = \{\langle f, g \rangle\}$ on the other hand, we only look at the two possible values for $\langle f, g \rangle$, and so rule out the cases where $f = g$.

This idea of fluent terms taking values independently is very much related to the *weak disjunctive knowledge property* defined in [Petrick and Levesque, 2002; Petrick, 2006] for an epistemic setting of the situation calculus. Our account here generalizes this idea by allowing tuples of fluent terms.

What we will be proposing in this paper, is a method of computing progression under incomplete knowledge that uses

³Note that this notion of possibility has an *epistemic* flavor and is distinct from the *physical* possibility of action execution, as characterized by *Poss*.

the possible values of fluent terms as above. To ensure that we get a \mathcal{T} that correctly captures any constraints among the fluents, we will restrict the form of the basic action theory. The main idea behind what we call a *bounded* action theory is that \mathcal{D}_0 will explicitly list the initial possible values of the fluents according to some grouping \mathcal{T} , and that the bodies of the successor state and sensing-result axioms will be suitable for specifying a new grouping \mathcal{T}' such that what is known about the fluents after some action is performed is accurately captured by the new grouping \mathcal{T}' .

Definition 5. Let $\mathcal{T} = \{\vec{\tau}_1, \dots, \vec{\tau}_k\}$ be a set of tuples of situation-suppressed primitive fluent terms such that every primitive fluent term appears in exactly one $\vec{\tau}_i$. We say that \mathcal{D} is *local-effect and bounded* by \mathcal{T} iff the following all hold:

1. $\mathcal{D}_0 = \{\alpha_1[S_0], \dots, \alpha_k[S_0]\}$, where for each i , there is a set C of constant tuples such that α_i is $\bigvee_{\vec{c} \in C} \vec{\tau}_i = \vec{c}$.
2. Each successor state axiom in \mathcal{D}_{ss} has the form

$$f(\vec{x}, do(a, s)) = v \equiv \gamma_f(\vec{x}, a, v)[s] \vee f(\vec{x}, s) = v \wedge (\neg \exists v') \gamma_f(\vec{x}, a, v')[s],$$

where $\gamma_f(\vec{x}, a, v)$ is a disjunction of formulas $\gamma_f^i(\vec{x}, a, v)$ of the form $\exists \vec{z}(a = A(\vec{y}) \wedge \psi(\vec{y}))$, \vec{y} contains $\vec{x} \cup \{v\}$, \vec{z} is the remaining variables of \vec{y} , and ψ is a primitive formula that is called the *context* of $\gamma_f^i(\vec{x}, a, v)$.

3. Each sensing result axiom in \mathcal{D}_{sr} has the form $sr(A(\vec{y}), s) = r \equiv \Theta_A(\vec{y}, r)[s]$, where $\Theta_A(\vec{y}, r)$ is a primitive formula.
4. Each precondition axiom in \mathcal{D}_{ap} has the form $Poss(A(\vec{y}), s) \equiv \Pi_A(\vec{y})[s]$, where $\Pi_A(\vec{y})$ is a primitive formula.
5. Each action symbol A is either a sensing action or a world-changing action and appears exclusively in exactly one SRA or in some SSAs, respectively.
6. If action symbol A appears in γ_f then it cannot appear in γ_g for any fluent g that appears in γ_f .

Local-effect and bounded action theories have the property that for any pair of ground action term $A(\vec{e})$ and sensing result b , both $\Theta_A(\vec{e}, b)$ and $\Pi_A(\vec{e})$ are primitive sentences and $\gamma_f^i(\vec{x}, A(\vec{e}), v)$ is logically equivalent to $(\vec{x} = \vec{c} \wedge v = d \wedge \phi)$, where \vec{c}, d are contained in \vec{e} and ϕ is a primitive sentence. These theories are restricted in that if an action $A(\vec{e})$ changes the value of a fluent term $f(\vec{c})$, then \vec{c} is contained in the arguments of the action.

4 Progression using databases

Given an action theory \mathcal{D} that is local-effect and bounded by \mathcal{T} , we will use a database to represent the possible values for each τ in \mathcal{T} , as a way to capture the models of \mathcal{D}_0 . Then we will define a procedure *PROG* that transforms this database into one that represents a progression of \mathcal{D}_0 .

We first review some basic notions of relational database theory. Let \mathcal{L} be a first-order language with a finite number of predicate symbols and a finite number of constant symbols \mathcal{C} . A database instance for the language \mathcal{L} is a first-order structure \mathcal{I} such that the domain of \mathcal{I} is \mathcal{C} , $=$ is interpreted

$$\begin{aligned} \phi &:= DBQ(\Theta_A(\vec{e}, b)) \\ \vec{\tau} &:= \vec{\tau}_1 \cdot \vec{\tau}_2 \cdot \dots \cdot \vec{\tau}_k, \\ &\text{where } R_{\vec{\tau}_1}, \dots, R_{\vec{\tau}_k} \text{ are the relation names in } \phi \\ \mathcal{T} &:= \mathcal{T} - \{\vec{\tau}_1, \dots, \vec{\tau}_k\} + \{\vec{\tau}\} \\ \mathcal{I}_0 &:= \mathcal{I}_0 - \{R_{\vec{\tau}_1}, \dots, R_{\vec{\tau}_k}\} + \{R_{\vec{\tau}}\}, \\ &\text{where the extension of } R_{\vec{\tau}} \text{ is } \phi(\mathcal{I}_0) \end{aligned}$$

Figure 1: Progress \mathcal{I}_0 wrt sensing action $A(\vec{e})$ and b

as identity, and each constant is interpreted as itself. Let \vec{x}, \vec{y} be tuples of variables. An \vec{x} -relation over \mathcal{I} is a finite set of mappings ζ from \vec{x} into \mathcal{C} , which we write as $\{\zeta : [\vec{x} \rightarrow \mathcal{C}]\}$. Let ϕ be a formula with free variables $\vec{x} = \langle x_1, \dots, x_n \rangle$. The answer to ϕ wrt \mathcal{I} is the \vec{x} -relation

$$\phi(\mathcal{I}) = \{\zeta : [\vec{x} \rightarrow \mathcal{C}] \mid \mathcal{I} \models \phi(\zeta(x_1), \dots, \zeta(x_n))\}.$$

Definition 6. Let \mathcal{L}_0 be the language that has the constants in \mathcal{C} , and for each $\vec{\tau} \in \mathcal{T}$ the relation name $R_{\vec{\tau}}$ with arity $|\vec{\tau}|$. We define \mathcal{I}_0 , the *database that represents* \mathcal{D}_0 , as the database instance for language \mathcal{L}_0 such that the interpretation of each $R_{\vec{\tau}}$ is the set of constant tuples \mathcal{C} that appear in the corresponding $\alpha[S_0] \in \mathcal{D}_0$, in item 1 of Definition 5.

The following definition shows how to transform a primitive situation calculus formula ϕ into a query $DBQ(\phi)$ of \mathcal{L}_0 . We will need to use a set of new variables, and we assume that there is a mapping ξ from primitive fluent terms to these variables such that each term is mapped to a distinct variable.

Definition 7. Let ϕ be a situation-suppressed primitive formula. Let ϕ' be ϕ with each primitive fluent term replaced by the corresponding variable wrt the mapping ξ . Let $\mathcal{T}' \subseteq \mathcal{T}$ be the smallest set such that $\vec{\tau} \in \mathcal{T}'$ if some element of $\vec{\tau}$ appears in ϕ . We define $DBQ(\phi)$ as the formula

$$\bigwedge_{\vec{\tau} \in \mathcal{T}'} R_{\vec{\tau}}(\xi(\vec{\tau})) \wedge \phi'$$

Now, on to progression. Here is how $PROG[\mathcal{I}_0, A(\vec{e}), b]$ works when A is a sensing action: The fluents that are affected are those that appear in $\Theta_A(\vec{e}, b)$. We use the concatenation of all the $\vec{\tau}_i \in \mathcal{T}$ that mention these fluent terms to construct a new relation. The tuples in this relation will be the answers to the query $DBQ(\Theta_A(\vec{e}, b))$. The progression for a world-changing action is more complex. For each fluent term $f(\vec{c})$ that is potentially given value d by action $A(\vec{e})$ according γ_f , we collect all the $\vec{\tau}_i \in \mathcal{T}$ that contain either $f(\vec{c})$ or a fluent term in the context, and we concatenate them together to make a new $\vec{\tau}$. We then assign $R_{\vec{\tau}}$ to be the tuple of all possible values for these terms, using both the new value d for $f(\vec{c})$ when the context is true and the previous values when the context is false.

Definition 8. Let \mathcal{I}_0 be a database, $A(\vec{e})$ be a ground action term, and b be a sensing result. Then the procedure $PROG[\mathcal{I}_0, A(\vec{e}), b]$ is defined as follows:

1. if A is a sensing action, then do as in Figure 1;
2. if A is a world-changing action, then use Figure 2.

Now we are ready to present our main result:

Theorem 1 (Progression). *Let \mathcal{D} be a local-effect action theory that is bounded by \mathcal{T} and $(A(\vec{e}), b)$ be an action-sensing*

for each $f \in \mathcal{F}$ do
for each $\gamma_f^i(\vec{x}, a, v)$ in $\gamma_f(\vec{x}, a, v)$ that mentions A do
 $\vec{c} :=$ the subset of \vec{e} that corresponds to \vec{x}
 $d :=$ the element of \vec{e} that corresponds to v
 $\vec{u} :=$ the element of \mathcal{T} that mentions $f(\vec{c})$
 $\phi^+ := R_{\vec{u}}(\xi(\vec{u})) \wedge DBQ(\gamma_f^i(\vec{x}, A(\vec{e}), v))$
 $\phi^- :=$ same as ϕ^+ but with context formula negated
 $Q^+ := \phi^+(\mathcal{I}_0)$ but with the column corresponding to $\xi(f(\vec{c}))$ replaced by the value d in all rows
 $Q^- := \phi^-(\mathcal{I}_0)$
 $\vec{\tau} := \vec{\tau}_1 \cdot \vec{\tau}_2 \cdot \dots \cdot \vec{\tau}_k,$
where $R_{\vec{\tau}_1}, \dots, R_{\vec{\tau}_k}$ are the relation names in ϕ^+
 $\mathcal{T} := \mathcal{T} - \{\vec{\tau}_1, \dots, \vec{\tau}_k\} + \{\vec{\tau}\}$
 $\mathcal{I}_0 := \mathcal{I}_0 - \{R_{\vec{\tau}_1}, \dots, R_{\vec{\tau}_k}\} + \{R_{\vec{\tau}}\},$
where the extension of $R_{\vec{\tau}}$ is $Q^+ \cup Q^-$
end for
end for

Figure 2: Progress \mathcal{I}_0 wrt world-changing action $A(\vec{e})$

result pair such that $\mathcal{D} \cup \mathcal{S}((A(\vec{e}), b))$ is consistent. Let \mathcal{I}_0 be the database that represents \mathcal{D}_0 . Then $PROG[\mathcal{I}_0, A(\vec{e}), b]$ represents a progression of \mathcal{D}_0 wrt $(A(\vec{e}), b)$.

The proof of Theorem 1 follows from the following lemma and the property of bounded theories listed in item 6 of Definition 5 which implies that the processing of fluents in the main loop of Figure 2 can be done in any order.

Lemma 2. *Let ϕ be a situation-suppressed primitive sentence. Then, ϕ is possibly true at the empty history wrt to \mathcal{T} iff there exists a standard model of \mathcal{D}_0 that satisfies $\phi[S_0]$ iff $DBQ(\phi)(\mathcal{I}_0)$ is non-empty.*

We conclude this section with an example.

Example 2. Consider the following scenario where an agent lives in a fantasy world of some role playing game. The agent is trapped inside a dungeon and in order to get out she has to unlock the main door and climb out. Unfortunately, the main door can only be unlocked by casting a magical spell. In particular, the agent has to cast the spell while being in the same room with a magical orb that is hidden somewhere in the dungeon, otherwise the spell has no effect. Finally, even though the agent cannot find the magical orb, she knows that it is hidden somewhere in rooms $R1$ or $R2$ of the dungeon.

For simplicity we use three (functional) fluents to represent this scenario: *hero*, *orb*, *door* which hold the location of the agent, the location of the orb, and the state of the door respectively. We use two actions: *go*(y), the agent moves to room y ; *cast*(y), the agent casts a spell that magically instructs the door to change to state y . Also, we use the constants Cl , Op to represent the state of the door (closed, open) and $R1, R2$ for the two rooms.

Let \mathcal{D} be an action theory such that \mathcal{D}_0 consists of the sentences $door = Cl[S_0]$, $(orb = R1 \vee orb = R2)[S_0]$, and $hero = R1[S_0]$; \mathcal{D}_{ap} consists of precondition axioms such that Π_{go} and Π_{cast} are always true; and \mathcal{D}_{ss} consists of the SSAs such that $\gamma_{hero}(a, v)[s]$ is $a = go(v)$, $\gamma_{orb}(a, v)[s]$ is *false*, and $\gamma_{door}(a, v)[s]$ is $a = cast(v) \wedge \exists z (hero = z \wedge orb = z)$.

\mathcal{D} is local-effect and bounded by $\mathcal{T} = \{agent, orb, door\}$, but it is not context-complete. The following figure shows

the database that represents \mathcal{D}_0 and the transformed database after applying $PROG$ once for action $cast(Op)$, and after applying $PROG$ two more times for $go(R2)$ and $cast(Op)$.

<i>hero</i>	<i>orb</i>	<i>door</i>	<i>hero</i>	<i>orb</i>	<i>door</i>	<i>hero</i>	<i>orb</i>	<i>door</i>
<i>R1</i>	<i>R1</i>	<i>Cl</i>	<i>R1</i>	<i>R1</i>	<i>Op</i>	<i>R2</i>	<i>R1</i>	<i>Op</i>
	<i>R2</i>		<i>R1</i>	<i>R2</i>	<i>Cl</i>	<i>R2</i>	<i>R2</i>	<i>Op</i>

Note that after the three steps of progression the fluent *door* has exactly one possible value, *Op*, as intended.

Finally, to justify concerns about why this approach is better than just calculating all the possible worlds, the reader should imagine there being a large number of fluents with unknown value (e.g. other doors, objects) for which the database remains unchanged. A more detailed discussion follows.

5 Complexity analysis

We first review some results about the complexity of database queries of a certain form. The guarded fragment GF is the smallest set of formulas such that any existentially quantified sub-formula ϕ is conjoined with a guard, i.e. an atom containing all the free variables of ϕ [Andréka *et al.*, 1998]. In [Gottlob *et al.*, 2003] GF was extended to the k -guarded fragment GF_k where up to k atoms may jointly act as a guard. When \mathcal{I} is a database of size n such that the domain and all relations are sorted, and $\phi \in GF_k$ is a formula of size l , then $\phi(\mathcal{I})$ can be computed in time $O(ln^k)$ (see [Liu and Levesque, 2003]).

The complexity of $PROG(\mathcal{I}_0, A(\vec{e}), b)$ is dominated by the evaluation of database queries of the form $DBQ(\phi)$. By Definition 7, it follows that $DBQ(\phi)$ is a k -guarded formula, where k is the number of relation names that appear in $DBQ(\phi)$. For any theory \mathcal{D} , we let the *width* of \mathcal{D} be the maximum k and the *depth* of \mathcal{D} be the maximum length over all of the formulas $\gamma_f^i(\vec{x}, a, v)$ and $\Theta_A(\vec{y}, r)$ in \mathcal{D} . By Definition 8, we get the following:

Theorem 3. *Let \mathcal{D} be a local-effect action theory that is bounded by \mathcal{T} and $(A(\vec{e}), b)$ be an action-sensing result pair. Let \mathcal{I}_0 be the database that represents \mathcal{D}_0 . Then, the complexity of $PROG[\mathcal{I}_0, A(\vec{e}), b]$ is $O(ln^{k+1})$, where l is the depth and k is the width of \mathcal{D} , and n is the size of \mathcal{I}_0 .*

The tractability of the approach relies on the assumption that the width of \mathcal{D} is often a small constant K . (Note that the width of \mathcal{D} is bounded by what we are calling the depth of \mathcal{D} .) There is also one more issue: As we saw in Example 2, after a progression step is performed some of the relations in \mathcal{I}_0 may be joined together in the updated database and eventually this can lead to an exponential blow-up of the size of the database.

Nevertheless, we expect that in real-world domains there is a small upper-bound on the number of primitive fluent terms that are both *unknown* and *mutually constrained* at any history, and thus need to appear in the same relation in \mathcal{I}_0 . Also assuming that there is a constant upper-bound for the number of possible values that a single fluent may have at any history then the size of the largest relation in \mathcal{I}_0 is always bounded by a large constant N .

With the current definition of $PROG$, the size of the relations in \mathcal{I}_0 may not be optimal because fluents that are known to be true may not be stored in a separate relation,

such as *hero* in Example 2. This can be avoided by augmenting $PROG$ to check, as a final step, whether some of the updated fluents have only one possible value and should be stored separately. Finally, following the proof of the complexity result about GF_k in [Liu and Levesque, 2003], note that n is the size of the largest relation in \mathcal{I}_0 . Hence, under the stated assumptions the complexity of $PROG[\mathcal{I}_0, A(\vec{e}), b]$ is $O(ln^{K+1})$, where l is the depth of \mathcal{D} , and K, N as above.

6 Related Work

One of the main differences between our approach and most of the related work in the literature is that we work with functional fluents. Our intuition is that when it comes to restricted forms of incomplete knowledge, functional fluents with possible values are an attractive alternative. In a formalism with relational fluents (such as [Son and Baral, 2001; Liu and Levesque, 2005]) it is not clear how to handle information of the form $(age = 19 \vee age = 20 \vee age = 21)$ without using a general method for unrestricted disjunctions, which will have computational problems of its own. Our way of handling the disjunctions by using a database of possible values is similar to ideas in [Antova *et al.*, 2007] (where, however, actions and progression are not handled).

One serious limitation of our approach is that we are forced to represent all the primitive fluent terms in the language. Also, the theories we are dealing with are essentially propositional and we insist that an action does not affect both a fluent and its context. The approach of [Liu and Levesque, 2005] does not suffer from these limitations. Nevertheless, our progression mechanism is always logically correct, unlike the *weak progression* mechanism that is employed by Liu and Levesque which is logically complete only for theories that are context-complete. An example of the theories that we can handle better is illustrated at the end of Section 4.

One approach that is very similar in spirit to our own is that of the FLUX system for implementing fluent calculus theories [Thielscher, 2004]. FLUX can express incomplete knowledge in the form of finite domain arithmetic constraints that are handled by constraint handling rules [Frühwirth, 1998]. When incomplete knowledge is progressed in FLUX, new constraints that represent the progression are appended to the constraint store. When fluents remain unknown and are constantly progressed, the “chain” of stored constraints becomes larger which, apart from affecting the efficiency of reasoning, can lead to failure due to lack of memory resources. So while FLUX can be more efficient in computing the progression of a state after an action has been executed, appending constraints can be problematic for long-lived agents.

Another similar approach is that of [Funge, 1998], dealing with *real-valued* functional fluents. Unlike our approach though, Funge’s does not provide a mechanism for joining fluents together whenever they are related with some constraint. Other related approaches for doing approximate progression of action theories include (approximate) logical filtering [Amir and Russell, 2003], and the 0-approximation semantics for an extension of the action language \mathcal{A} [Son and Baral, 2001]. Finally, the idea of reasoning with possible values has been applied in an iterative planner [Levesque, 2005]

and in a variant of GOLOG [Sardina and Vassos, 2005].

7 Conclusion

In this paper, we have proposed a new progression mechanism for a restricted form of situation calculus basic action theories that feature incomplete knowledge and sensing. Our approach handles theories which are essentially the functional version of the local-effect action theories [Liu and Levesque, 2005], further restricted so that they are also bounded by a set of groupings for fluents. We use a large database in order to represent the models of a theory by dealing directly with the possible values that these groupings of fluents may have. Our progression mechanism updates the database according to how the possible values are affected by both world-changing and sensing actions. The method we propose is logically complete and can be calculated efficiently using database techniques under certain reasonable assumptions.

Our current and future work focuses on some of the limitations of our approach. We are investigating alternative formalisms for representing \mathcal{D}_0 so that we avoid listing *all* primitive fluent terms in the language as well as all the possible values for each fluent. In particular, we consider the incorporation of intervals as a way to concisely represent large lists of possible values and the use of quantified axioms in \mathcal{D}_0 . Furthermore, we believe that the local-effect requirement can be lifted with a small additional cost in the progression mechanism. Finally, we want to extend our approach so that it can handle the evaluation of unrestricted quantified formulas.

References

- [Amir and Russell, 2003] E. Amir and S. Russell. Logical filtering. In *Proc. IJCAI-03*, pages 75–82, 2003.
- [Andréka *et al.*, 1998] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.
- [Antova *et al.*, 2007] L. Antova, C. Koch, and D. Olteanu. 10^{106} worlds and beyond: Efficient representation and processing of incomplete information. In *Proc. ICDE-07*, Istanbul, Turkey, April 2007. (To appear).
- [De Giacomo and Levesque, 1999] G. De Giacomo and H. J. Levesque. An incremental interpreter for high-level programs with sensing. *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*, pages 86–102, 1999.
- [De Giacomo *et al.*, 2001] G. De Giacomo, H. J. Levesque, and S. Sardina. Incremental execution of guarded theories. *Computational Logic*, 2(4):495–525, 2001.
- [Frühwirth, 1998] T. Frühwirth. Theory and practice of constraint handling rules. *Journal of Logic Programming, Special Issue on Constraint Logic Programming*, 37(1-3):95–138, October 1998.
- [Funge, 1998] J. Funge. Cognitive modeling for computer games. In *AAAI Fall Symposium on Cognitive Robotics*, pages 44–51, Orlando, FL., October 1998.
- [Gottlob *et al.*, 2003] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: Game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, 66(4):775–808, June 2003.
- [Kowalski and Sergot, 1986] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.
- [Lakemeyer and Levesque, 2007] G. Lakemeyer and H. J. Levesque. Chapter: Cognitive robotics. In V. Lifschitz, F. van Harmelen, and F. Porter, editors, *Handbook of Knowledge Representation*. Elsevier, 2007. (Forthcoming).
- [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, 2005] H. J. Levesque. Planning with loops. In *Proc. IJCAI-05*, Edinburgh, Scotland, August 2005.
- [Lin and Reiter, 1997] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, 92(1-2):131–167, 1997.
- [Lin, 2007] F. Lin. Chapter: Situation calculus. In V. Lifschitz, F. van Harmelen, and F. Porter, editors, *Handbook of Knowledge Representation*. Elsevier, 2007. (Forthcoming).
- [Liu and Levesque, 2003] Y. Liu and H. J. Levesque. A tractability result for reasoning with incomplete first-order knowledge bases. In *Proc. IJCAI-03*, pages 83–88, Acapulco, Mexico, August 2003.
- [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.
- [Petrick and Levesque, 2002] R. Petrick and H. J. Levesque. Knowledge equivalence in combined action theories. In *Proc. KR-02*, Toulouse, France, April 2002.
- [Petrick, 2006] R. Petrick. *A knowledge-based representation for effective acting, sensing, and planning*. PhD thesis, University of Toronto, 2006.
- [Reiter, 2001] R. Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
- [Sardina and Vassos, 2005] S. Sardina and S. Vassos. The Wumpus World in IndiGolog: A Preliminary Report. In *Proc. NRAC-05*, pages 90–95, Edinburgh, Scotland, 2005.
- [Scherl and Levesque, 2003] R. Scherl and H. J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1-2):1–39, 2003.
- [Son and Baral, 2001] T. C. Son and C. Baral. Formalizing sensing actions a transition function based approach. *Artificial Intelligence*, 125(1-2):19–91, 2001.
- [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.
- [Thielscher, 2004] M. Thielscher. FLUX: A logic programming method for reasoning agents. *Theory and Practice of Logic Programming*, 5(4-5):533–565, 2004.