Regression With Respect to Sensing Actions and Partial States

Le-chi Tuan, Chitta Baral, and Xin Zhang
Computer Science and Engineering
Arizona State University
Tempe, AZ 85287, USA
{lctuan,baral,xin.zhang}@asu.edu

Tran Cao Son
Computer Science Department
New Mexico State University
Las Cruces, NM 88003, USA
tson@cs.nmsu.edu

Abstract

In this paper, we present a *state-based regression function* for planning domains where an agent does not have complete information and may have sensing actions. We consider binary domains [1], and employ the 0-approximation (Son & Baral 2001) to define the regression function. In binary domains, the use of 0-approximation means using 3-valued states. Although planning using this approach is incomplete, we adopt it to have a lower complexity. We prove the soundness and completeness of our regression formulation with respect to the definition of progression and develop a conditional planner that utilizes our regression function.

Introduction and Motivation

An important aspect in reasoning about actions and in characterizing the semantics of action description languages is to define a transition function encoding the transition between states due to actions. This transition function is often viewed as a *progression* function in that it denotes the progression of the world by the execution of actions. The 'opposite' or 'inverse' of progression is referred to as *regression*.

Even for the simple case where we have only non-sensing actions and the progression transition function is deterministic, there are various formulations of regression. For example, consider the following. Let Φ be the progression transition function from actions and states to states. I.e., intuitively $\Phi(a, s) = s'$ means that if the action a is executed in state s then the resulting state will be s'. One way to define a regression function Ψ_1 is to define it with respect to states. In that case $s \in \Psi_1(a, s')$ will mean that the state s' is reached if a is executed in s. Another way regression is defined is with respect to formulas. In that case $\Psi_2(a, f) = g$, where f and g are formulas, means that if a is executed in a state satisfying g then a state satisfying f will be reached.

For planning using heuristic search often a different formulation of regression is given. Since most planning research is about goals that are conjunction of literals, regression is defined with respect to a set of literals and an action. In that case the conjunction of literals (often specifying the goal) denotes a set of states, one of which needs to be reached. This regression is slightly different from Ψ_2 as the intention is to regress to another set of literals (not an arbitrary formula), denoting a sub-goal.

With respect to the planning language STRIPS, where each action a has an add list $Add(a)$, a delete list $Del(a)$, and a precondition list $Prec(a)$, the progression function is defined as $Progress(s, a) = s + Add(a) - Del(a)$; and the regression function is defined as $Regress(conj, a) = conj + Prec(a) - Add(a)$, where $conj$ is a set of atoms. The relation between these two, formally proven in (Pednault 1986), shows the correctness of regression based planners; which in recent years through use of heuristics (e.g. (Bonet & Geffner 2001; Nguyen, Kambhampati, & Nigenda 2002)) have done exceedingly well on planning competitions.

In this paper we are concerned with domains where the agent does not have complete information about the world, and may have sensing actions, which when executed do not change the world, but rather give certain information about the world to the agent. As a result, plans may now no longer be simply a sequence of (non-sensing) actions but may include sensing actions and conditionals. Various formalisms have been developed for such cases (e.g. (Lobo 1998; Son & Baral 2001)) and progression functions have been defined. Also, the complexity of planning in such cases has been analyzed in (Baral, Kreinovich, & Trejo 2000). One approach to planning in the presence of incomplete information is conformant planning where no sensing action is used, and a plan is a sequence of actions leading to the goal from every possible initial situation. However, this approach proves inadequate for many planning problems (Son & Baral 2001), i.e., there are situations where sensing actions are necessary. In that case, one approach is to use belief states or Kripke models instead of states. It is shown that the total number of belief states is double exponential while the total number of 3-valued states is exponential in the number of fluents (Baral, Kreinovich, & Trejo 2000). Here, we pursue a provably less complex formulation with sensing actions and use 3-valued states. In this approach, we will miss certain plans, but that is the price we are willing to pay for reduced complexity. This is consistent with and similar to the considerations behind conformant planning. With that trade-off in mind, *in this paper we consider the 0-approximation semantics defined in (Son & Baral 2001) and define regres-*

[1]Due to space limitation, we do not present the results for non-binary domains here. Interested readers can find such results at http://www.public.asu.edu/~lctuan/TOCL/

sion with respect to that semantics. We then formally relate our definition of regression with the earlier definition of progression in (Son & Baral 2001) and show that planning using our regression function will indeed give us correct plans. We then use our regression function in planning with sensing actions and show that, even without using any heuristics, our planner produces good results. To simplify our formulation, we only consider STRIPS like actions where no conditional effects are allowed. We also restrict domain of a fluent to a finite set of discrete values.

In summary the main contributions of our paper are:

- A state-based regression function corresponding to the 0-approximation semantics in (Son & Baral 2001);
- A formal result relating the regression function with the progression transition function in (Son & Baral 2001);
- An algorithm that uses these regression functions to construct conditional plans with sensing actions;
- Implementation of this algorithm; and
- Illustration of the performance of this algorithm with respect to several examples in the literature.

Related Work Our work in this paper is related to different approaches to regression and planning in the presence of sensing actions and incomplete information. It differs from earlier formula regression such as (Pednault 1994; Reiter 2001; Son & Baral 2001) in that it is a state-based formulation and the other are formula based. Unlike the conditional planners (Peot & Smith 1992; Cimatti, Roveri, & Traverso 1998), our planner can deal with sensing actions similar to the planners in (Etzioni *et al.* 1992; Lobo 1998; Son, Tu, & Baral 2004; Weld, Anderson, & Smith 1998). However, it does not deal with nondeterministic and probabilistic actions such as the planners in (Bonet & Geffner 2001; Pryor & Collins 1996; Rintanen 2000; 2002). It is also not a conformant planner as in (Cimatti, Roveri, & Traverso 1998; Eiter *et al.* 2000). For these reasons, we currently compare our planner with those of (Son, Tu, & Baral 2004; Weld, Anderson, & Smith 1998).

Background: 0-Approximation Semantics For A STRIPS-like Language

Action and Plan Representation

We employ a STRIPS-like action representation (Fikes & Nilson 1971) and represent a planning problem by a tuple $P = \langle A, O, I, G\rangle$ where A is a finite set of fluents, O is a finite set of actions, and I and G encode an initial state and a goal state, respectively. A fluent literal is either a positive fluent $f \in A$ or its negation (negative fluent) $\neg f$. In this paper, we are interested in the planning problem in which I and G are sets of fluent literals. An action $a \in O$ is either a *non-sensing action* or a *sensing action* and is specified as follows:

- A non-sensing action a is specified by an expression of the form

  `action` a `:Pre` Pre_a `:Add` Add_a `:Del` Del_a

  where Pre_a is a set of fluent literals representing the precondition for a's execution, Add_a and Del_a are two disjoint sets of positive fluents representing the positive and negative effects of a, respectively; and
- A sensing action a is specified by an expression of the form

  `action` a `:Pre` Pre_a `:Sense` $Sens_a$

  where Pre_a is a set of fluent literals and $Sens_a$ is a set of positive fluents that do not appear in Pre_a.

To illustrate the action representation and our search algorithm, we will use a small example, a version of the "Getting to Evanston" from (Weld, Anderson, & Smith 1998). Figure (1) shows the actions of this domain.

Non-sensing action	:Pre	:Add	:Del
goto-western-at-belmont	at-start	on-western on-belmont	at-start
take-belmont	on-belmont, traffic-bad	on-ashland	on-western
take-ashland	on-ashland	at-evanston	
take-western	¬traffic-bad, on-western	at-evanston	

Sensing action	:Pre	:Sense
check-traffic	True	traffic-bad
check-on-western	True	on-belmont

Figure 1: Actions of the "Getting to Evanston" domain

The notion of a plan in the presence of incomplete information and sensing actions has been discussed in the literature (Levesque 1996; Son & Baral 2001). In this paper, we consider *conditional plans* that are formally defined as follows.

Definition 1 (Conditional Plan) .

- *An empty sequence of actions, denoted by* [ ], *is a conditional plan.*
- *If a is a non-sensing action, then a is a conditional plan.*
- *If a is a sensing action, $\varphi_1, \ldots, \varphi_n$ are mutual exclusive conjunctions of fluent literals, and $c_1, \ldots, c_n$ are conditional plans, then so is* $a; case(\varphi_1 \rightarrow c_1, \ldots, \varphi_n \rightarrow c_n)$.
- *if c_1, c_2 are conditional plans, then so is $c_1; c_2$.*
- *Nothing else is a conditional plan.*

0-Approximation

The 0-approximation in (Son & Baral 2001) is defined by a transition function Φ that maps pairs of actions and approximate states into sets of approximate states. An *approximate state* (or a-state) is a pair $\langle T, F\rangle$ where $T \subseteq A$ and $F \subseteq A$ are two disjoint sets of fluents. Intuitively, given an a-state $\sigma = \langle T, F\rangle$, T (resp. F), denoted by $\sigma.T$ (resp. $\sigma.F$), is the set of fluents which are true (resp. false) in σ; and $f \in A \backslash (T \cup F)$ is unknown in σ. Alternatively, we can also view an a-state as the intersection of all states in a belief-state. Let $\sigma_1 = \langle T_1, F_1\rangle$ and $\sigma_2 = \langle T_2, F_2\rangle$ be two a-states. $\sigma_1 \cap \sigma_2 = \langle T_1 \cap T_2, F_1 \cap F_2\rangle$ is called the intersection of σ_1 and σ_2. We say σ_1 extends σ_2, denoted by $\sigma_2 \preceq \sigma_1$ if $T_2 \subseteq T_1$ and $F_2 \subseteq F_1$. $\sigma_1 \backslash \sigma_2$ denotes the set $(T_1 \backslash T_2) \cup (F_1 \backslash F_2)$. For a set of fluents X, we write $X \backslash \langle T, F\rangle$ to denote $X \backslash (T \cup F)$. To simplify the presentation, for a set of literals L, by L^+ and L^- we denote the set of fluents $\{f \mid f \in L,\ f \text{ is a fluent }\}$ and $\{f \mid \neg f \in L,\ f \text{ is a fluent }\}$.

Given a fluent f and an a-state $\sigma = \langle T, F\rangle$, we say that f is true (resp. false) in σ if $f \in T$ (resp. $f \in F$). f (resp. $\neg f$) holds in σ if f is true (resp. false) in σ. f is known (resp. unknown) in σ if $f \in (T \cup F)$ (resp. $f \notin (T \cup F)$). A set L of fluent literals holds in an a-state $\sigma = \langle T, F\rangle$ if every member of L holds in σ. A set X of fluents is known in σ if every fluent in X is known in σ. An action a is *executable* in σ if Pre_a holds in σ. The transition function (for progression) is defined next.

Definition 2 (Transition Function) *For an a-state $\sigma = \langle T, F\rangle$ and an action a, $\Phi(a, \sigma)$ is defined as follows:*

- *if a is not executable in σ then $\Phi(a, \sigma) = \{\bot\}$; and*
- *if a is executable in σ*
  - *if a is a non-sensing action: $\Phi(a, \sigma) = \{\langle T \setminus Del_a \cup Add_a, F \setminus Add_a \cup Del_a\rangle\}$;*
  - *if a is a sensing action: $\Phi(a, \sigma) = \{\sigma' | \sigma \preceq \sigma'$ and $Sens_a \setminus \sigma = \sigma' \setminus \sigma\}$.*

The function Φ can be extended to define the function Φ^* that maps each pair of a conditional plan p and a-states σ into a set of a-states, denoted by $\Phi^*(p, \sigma)$. Intuitively, $\Phi^*(p, \sigma)$ is the set of final a-states resulting from the execution of p in σ. Φ^* is defined similarly to $\hat{\Phi}$ in (Son & Baral 2001).

Given a planning problem $P = \langle A, O, I, G\rangle$, the a-state representing I is defined by $\sigma_I = \langle I^+ \cap A, I^- \cap A\rangle$. $\Sigma_G = \{\sigma \mid \sigma_G \preceq \sigma\}$, where $\sigma_G = \langle G^+ \cap A, G^- \cap A\rangle$, is the set of a-states satisfying the goal G. A *progression solution* to the planning problem P is a conditional plan p such that $\bot \notin \Phi^*(p, \sigma_I)$ and $\Phi^*(p, \sigma_I) \subseteq \Sigma_G$.

Regression and Its Relation with Progression

In this section, we will present our formalization of a regression function, denoted by $Regress$, and prove its correctness. $Regress$ is a state-based regression function that maps a pair of an action and a set of a-states into an a-state. For this we introduce the notion of a *partial state* (or p-state) as a pair $[T, F]$ where T and F are two disjoint sets of fluents. Intuitively, a p-state δ=$[T, F]$ represents a collection of a-states which extends the a-state $\langle T, F\rangle$. We denote this set by $ext(\delta)$ and call it *the extension set* of δ. Formally, $ext(\delta) = \{\langle T', F'\rangle | T \subseteq T', F \subseteq F'\}$. $\sigma' \in ext(\delta)$ is called an extension of δ. Given a p-state δ=$[T, F]$, we say a p-state δ'=$[T', F']$ is a partial extension of δ if $T \subseteq T', F \subseteq F'$.

The regression function will be defined separately for non-sensing actions and sensing actions. Since the application of a non-sensing action in an a-state results in a single a-state, the regression of a non-sensing action from a p-state should result into a p-state. On the other hand, as application of a sensing action in an a-state results in a set of a-states, the regression of a sensing action should start from a set of p-states and result into a p-state. Besides the regression should be sound (i.e., plans obtained through regression must be plans based on the progression) and complete (i.e., for each plan based on progression,using regression one should obtain that plan or an equivalent one) with respect to progression. To simplify the presentation, we define a partition of a set of fluents X as a pair (P, Q) such that $P \cap Q = \emptyset$ and $P \cup Q = X$. We begin with the applicability condition of non-sensing actions and then give the definition of their function $Regress$.

Definition 3 (Applicability Condition - non-sensing action) *Given a p-state $\delta = [T, F]$ and a non-sensing action a. We say that a is* applicable *in δ if (i) $Add_a \cap T \neq \emptyset$ or $Del_a \cap F \neq \emptyset$, and (ii) $Add_a \cap F = \emptyset$, $Del_a \cap T = \emptyset$, $Pre^+_a \cap F \subseteq Del_a$, and $Pre^-_a \cap T \subseteq Add_a$.*

The regression on a non-sensing action is defined next.

Definition 4 (Regression - non-sensing action) *Given a p-state $\delta = [T, F]$ and a non-sensing action a,*

- *if a is not applicable in δ then $Regress(a, \delta) = \bot$;*
- *if a is applicable in δ then $Regress(a, \delta) = [T \setminus Add_a \cup Pre^+_a, F \setminus Del_a \cup Pre^-_a]$.*

The regression function $Regress$ for non-sensing actions with respect to a set of p-states is defined as follows. $Regress(a, \{\delta_1, \ldots, \delta_n\}) = \{Regress(a, \delta_1), \ldots, Regress(a, \delta_n)\}$ where $\delta_1, \ldots, \delta_n$ are p-states and a is a non-sensing action.

Example 1 (Getting to Evanston - con't) *The actions take_western and take_ashland are applicable in $\delta_0 = [\{$at-evanston$\}, \{\}]$. We have Regress(take_western, δ_0) = $[\{$on-western$\}, \{$traffic-bad $\}]$ and Regress(take_ashland, δ_0) = $[\{$on-ashland$\}, \{\}]$*

We will now define $Regress$ for sensing actions. Recall that the execution of a sensing action a in an a-state σ requires that a is executable in σ and results in a set of a-states $\Phi(a, \sigma)$ whose member extends σ by the set of fluents in $s_a \subseteq Sens_a$ and every $f \in Sens_a \setminus s_a$ holds in σ. This leads to the following definitions.

Definition 5 (Properness) *Let a be a sensing action, $\Delta = \{\delta_1, \ldots, \delta_n\}$ be a set of distinct p-states, and $\emptyset \neq X \subseteq Sens_a$ be a set of sensing fluents. We say that Δ is proper w.r.t X if (i) $Sens_a$ is known in Δ; (ii) $n = 2^{|X|}$; (iii) for every partition (P, Q) of X, there exists only one $\delta_i \in \Delta$ $(1 \leq i \leq n)$ s.t. $\delta_i.T \cap X = P$, $\delta_i.F \cap X = Q$; and (iv) for every $(1 \leq i \neq j \leq n)$, $\delta_i.T \setminus X = \delta_j.T \setminus X$, $\delta_i.F \setminus X = \delta_j.F \setminus X$. We call X a sensed set of Δ w.r.t a.*

Lemma 1 (Sensed Set) *Consider a sensing action a and a set of p-states Δ. If there exists a sensed set of Δ w.r.t a then it is unique.*

Given a sensing action a and a set of p-states Δ, we denote $p(a, \Delta)$ as the unique sensed set of Δ w.r.t a; if there exists no sensed set w.r.t a and Δ, we write $p(a, \Delta) = \bot$.

Definition 6 (Strong Applicability Condition - sensing action) *Let a be a sensing action and $\Delta = \{\delta_1, \ldots, \delta_n\}$ be a set of p-states. We say that a is* strongly applicable *in Δ if (i) $p(a, \Delta) \neq \bot$; and (ii) $Pre^+_a \cap \delta_i.F = \emptyset$ and $Pre^-_a \cap \delta_i.T = \emptyset$.*

In the above definition, (i) corresponds to the fact that executing a sensing action a in an a-state σ results in a set of $2^{|p(a,\Delta)|}$ a-states that are represented by $2^{|p(a,\Delta)|}$ corresponding p-states of Δ where $p(a, \Delta)$ denotes the set of fluents that are not yet known, while $Sens_a \setminus p(a, \Delta)$ is already known when a is executed; (ii) guarantees that a must be executable prior to its execution.

Although this strong applicability condition guarantees the soundness of regression over sensing actions, it does not guarantee the completeness. We now provide a weaker applicability condition that guarantees both soundness and completeness of regression.

Definition 7 (Applicability Condition - sensing action) *Let a be a sensing action and $\Delta = \{\delta_1, \ldots, \delta_n\}$ be a set of p-states. We say that a is* applicable *in Δ if (i) there exists a set Δ'=$\{\delta'_1, \ldots, \delta'_n\}$, where δ'_i is a partial extension of δ_i ($i = 1, \ldots, n$), such that a is strongly applicable in Δ'; and (ii) $Sens_a$ is known in Δ.*

Lemma 2 (Unique Sensed Set) *Consider a sensing action a and a set of p-states Δ such that a is applicable in Δ. Let Δ'=$\{\delta'_1, \ldots, \delta'_n\}$, where δ'_i is a partial extension of δ_i ($i = 1, \ldots, n$), Δ''=$\{\delta_1'', \ldots, \delta_n''\}$, where δ_i'' is a partial extension of δ_i ($i = 1, \ldots, n$). If $p(a, \Delta') \neq \bot$ and $p(a, \Delta'') \neq \bot$ then $p(a, \Delta') = p(a, \Delta'')$.*

Given a sensing action a and a set of p-states Δ. If there exists a Δ'=$\{\delta'_1, \ldots, \delta'_n\}$, where δ'_i is a partial extension of δ_i ($i = 1, \ldots, n$) such that $p(a, \Delta') \neq \bot$ then, by Lemma 2, $p(a, \Delta') = p(a, \Delta'')$ for all $\Delta'' = \{\delta_1'', \ldots, \delta_n''\}$, where δ_i'' is a partial extension of δ_i ($i = 1, \ldots, n$) and $p(a, \Delta'') \neq \bot$. We refer to the set $p(a, \Delta')$ by $S_{a,\Delta}$. If there exits no such $p(a, \Delta')$, we write $S_{a,\Delta} = \bot$. Note that, from Definition 7, if a is applicable in Δ then $S_{a,\Delta}$ is defined. In that case, we also often say that a is applicable in Δ w.r.t $S_{a,\Delta}$ to make the applicability condition clearer from the context.

Definition 8 (Regression - sensing action) *Let a be a sensing action and $\Delta = \{\delta_1, \ldots, \delta_n\}$ be a set of p-states.*

- *if a is not applicable in Δ, then $Regress(a, \Delta) = \bot$;*
- *if a is applicable in Δ, then $Regress(a, \Delta) =$ $[(\bigcup_{i=1}^{n} \delta_i.T) \setminus S_{a,\Delta} \cup Pre_a^+, (\bigcup_{i=1}^{n} \delta_i.F) \setminus S_{a,\Delta} \cup Pre_a^-]$.*

Example 2 (Getting to Evanston - con't) *Let Δ=$\{\delta_1, \delta_2\}$ where δ_1 = [{at-start, traffic-bad}, {on-western, on-belmont, on-ashland, at-evanston }] and δ_2 = [{at-start}, {traffic-bad, at-evanston}].*

We have that check-traffic is applicable in Δ w.r.t {traffic-bad} and that $Regress$(check-traffic, Δ) = [{at-start)}, {on-western, on-belmont, on-ashland, at-evanston}].

We now relate our regression function $Regress$ with the progression function Φ.

Proposition 1 (Non-sensing action) *Let δ and δ' be p-states, let a be a non-sensing action. If $Regress(a, \delta) = \delta'$ where $\delta' \neq \bot$, then for every $\sigma'' \in ext(\delta')$ we have that (i) $\bot \not\in \Phi(a, \sigma'')$, and (ii) $\Phi(a, \sigma'') \subseteq ext(\delta)$.*

Proposition 2 (Sensing action) *Let $\Delta = \{\delta_1, \ldots, \delta_n\}$ be a set of p-states, δ' be a p-state, and a be a sensing action. If $Regress(a, \Delta) = \delta'$ where $\delta' \neq \bot$, then for every $\sigma'' \in ext(\delta')$, we have that (i) $\bot \not\in \Phi(a, \sigma'')$, and (ii) $\Phi(a, \sigma'') \subseteq ext(\delta_1) \cup \ldots \cup ext(\delta_n)$.*

We next extend $Regress$ to define $Regress^*$ that allows us to perform regression with respect to conditional plans.

Definition 9 (Extended Regression Function) *Let δ and $\{\delta_1, \ldots, \delta_n\}$ be a p-state and a set of p-states, respectively. The extended transition function $Regress^*$ is defined as follows:*

- *$Regress^*([\,], \delta) = \delta$;*
- *For a non-sensing action a, $Regress^*(a, \delta) = Regress(a, \delta)$;*
- *For a conditional plan $p = a; case(\varphi_1 \rightarrow c_1, \ldots, \varphi_n \rightarrow c_n)$,*
  - *if $Regress^*(c_i, \delta)$=$\bot$ for some i, $Regress^*(p, \delta) = \bot$*
  - *if $Regress^*(c_i, \delta)$=$[T_i, F_i]$ $i = 1, \ldots, n$, then $Regress^*(p, \delta) = Regress(a, \{R(c_1, \delta), \ldots, R(c_n, \delta)\})$ where $R(c_i, \delta) = [T_i \cup \varphi_i^+, F_i \cup \varphi_i^-]$ if $\varphi_i^+ \cap F_i = \emptyset$ and $\varphi_i^- \cap T_i = \emptyset$; otherwise, $R(c_i, \delta) = \bot$. Here, φ_i^+ and φ_i^- denote the sets of fluents occurring positively and negatively in φ_i, respectively;*
- *For $p = c_1; c_2$, where c_1, c_2 are conditional plans, $Regress^*(p, \delta) = Regress^*(c_1, Regress^*(c_2, \delta))$; and*
- *$Regress^*(p, \bot)$ =$\bot$ for every plan p.*

For a planning problem $P = \langle A, O, I, G \rangle$, let δ_G be the p-state $[G^+ \cap A, G^- \cap A]$, and Δ_I be the set of p-states such that for every $\delta \in \Delta_I$, $\sigma_I \in ext(\delta)$. (Recall that σ_I is the a-state representing I and Σ_G is the set of a-states in which G holds). A *regression solution* to the planning problem P is a conditional plan c that upon applying from the p-state δ_G will result to one of the p-states in Δ_I. In other words, if $\bot \neq \delta = Regress^*(c, \delta_G)$ then δ is a p-state belonging to Δ_I.

We now formalize the soundness of the regression solution with respect to the progression transition function Φ^*.

Theorem 3 (Soundness of Regression) *For a planning problem $P = \langle A, O, I, G \rangle$ and a regression solution p of P, then $\bot \not\in \Phi^*(p, \sigma_I)$ and $\Phi^*(p, \sigma_I) \subseteq ext(\delta_G)$.*

We now proceed towards a completeness result. Intuitively our completeness result states that if a conditional plan can be found through progression we can also find a conditional plan through regression. The plan found through regression may not be the same one though. It will be one which does not have redundancies, both in terms of extra actions and extra branches. We now formalize these conditions. First we need the following notions. Given a sensing action a, a sub sensing action of a is a sensing action a' where $Pre_{a'} = Pre_a$ and $Sens_{a'} \subset Sens_a$.

Definition 10 (Sub Conditional Plan) *Let c be a conditional plan. A conditional plan c' is a sub conditional plan of c if*

- *c' can be obtained from c by (i) removing an instance of a non-sensing action from c; (ii) replacing a sensing action a with a sub sensing action $subSense(a)$ of a; or (iii) removing a branch $\varphi_i \rightarrow c_i$ from a case plan in c; or*
- *c' is a sub conditional plan of c'' where c'' is a sub conditional plan of c.*

Definition 11 (Redundancy) *Let c be a conditional plan, σ be an a-state, and δ be a p-state. We say that c contains redundancy w.r.t (σ, δ) if*

- *$\bot \not\in \Phi^*(c, \sigma)$ and $\Phi^*(c, \sigma) \subseteq ext(\delta)$; and*
- *there exists a sub conditional plan c' of c such that $\bot \not\in \Phi^*(c', \sigma)$ and $\Phi^*(c', \sigma) \subseteq ext(\delta)$.*

For a non-empty set of fluents $S = \{f_1, ..., f_k\}$, a binary representation of S is a formula of the form $l_1 \wedge \ldots \wedge l_k$ where $l_i \in \{f_i, \neg f_i\}$ for $i = 1, \ldots, k$. For a non-empty set of fluents S, let us denote $BIN(S)$ as the set of all different binary representations of S. We say a conjunction ϕ of literals is consistent if there exists no fluent f such that f and $\neg f$ appear in ϕ. A set of consistent conjunctions of literals $\chi = \{\varphi_1, \ldots, \varphi_n\}$ is said to span over S if there exists a consistent conjunction of literals $\varphi \notin \chi$, such that:

1. $S \cap (\varphi^+ \cup \varphi^-) = \emptyset$ where φ^+ (resp. φ^-) denote the sets of fluents occurring positively (resp. negatively) in φ;
2. $\varphi_i = \varphi \wedge \psi_i$ where $BIN(S) = \{\psi_1, \ldots, \psi_n\}$.

We say that a set $\chi = \{\varphi_1, \ldots, \varphi_n\}$ is factorable if it spans over some non-empty set of fluents S.

Lemma 4 *Let $\chi = \{\varphi_1, \ldots, \varphi_n\}$ be a non-empty set of consistent conjunctions of literals. If χ is factorable, then there exists a unique non-empty set of fluents S such that χ spans over S.*

Definition 12 (Possibly Regressable Case Plan Structure) *Given a case plan structure $p = a; case(\varphi_1 \rightarrow c_1, \ldots, \varphi_n \rightarrow c_n)$. We say that p is possibly regressable if (i) there exists a non-empty set $\emptyset \neq S_a \subseteq Sens_a$ and $\{\varphi_1, \ldots, \varphi_n\}$ spans over S_a, and (ii) for $1 \leq i \leq n$ $Sens_a \subseteq (\varphi_i^+ \cup \varphi_i^-)$.*

Definition 13 (Possibly Regressable Conditional Plan) *A conditional plan c is possibly regressable if every case plan structure occurring in c is possibly regressable.*

We now formulate the completeness result for regressable conditional plans that have no redundancy.

Theorem 5 (Completeness of Regression) *Let $P = \langle A, O, I, G \rangle$ be a planning problem. If c is a progression solution which is a possibly regressable conditional plan without redundancy w.r.t (σ_I, δ_G) then c is also a regression solution of P.*

We now present an algorithm that uses these regression functions to construct conditional plans with sensing actions.

Conditional Planing Using Regression

In this section, we present a regression search algorithm for constructing conditional plans with sensing actions that makes use of the $Regress$ function described in the previous section. For a conditional plan c and a p-state δ, we call the pair $\langle c, \delta \rangle$ and *r-state*. For a set of r-states X, by X_s we denote the set of all the p-states occurring in X. The main idea of the algorithm is as follows. At any step, we will maintain a set N of r-states $\langle c, \delta \rangle$ such that $\delta = Regress^*(c, \delta_G)$. We print a solution if we find a r-state $\langle c, \delta \rangle \in N$ such that $\sigma_I \in ext(\delta)$ since c would be one solution (Theorem 3). Otherwise, we regress from N_s (the set of all the p-states occurring in N). This process involves the regression using non-sensing actions and sensing actions which are applicable in N_s. The algorithm will stop with failure if (i) we cannot regress from any p-state (or a set of p-states) in N_s; or (ii) no new p-state can be added to N_s. Below, we list the main steps of the algorithm:

```
Solve(P) where $P = \langle A, O, I, G \rangle$
1. Let $N = \{\langle [], \delta_G \rangle\}$ ($N_s = \{\delta_G\}$).
2. Repeat
3. If there exists some $\langle c, \delta \rangle \in N$ s.t. $\sigma_I \in ext(\delta)$ then prints $c$ as
   a solution.
4. Do one of the following:
   4.1 Find a $\langle c, \delta \rangle \in N$, a non-sensing action $a$ s.t. $a$ is applicable
       in $\delta$ and $\delta' = Regress(a, \delta) \notin N_s$. Add $\langle a; c, \delta' \rangle$ to $N$.
   4.2 Find a set $\Delta = \{\langle c_1, \delta_1 \rangle, \ldots, \langle c_n, \delta_n \rangle\} \subseteq N$ and a sensing
       action $a$ such that $a$ is applicable in $\Delta_s$ w.r.t some $\emptyset \neq S_a \subseteq$
       $Sens_a$, i.e.: (i) $Sens_a$ is known in $\Delta_s$ and there exists a set
       $S_a = \{f_1, \ldots, f_k\} \subseteq Sens_a$, $2^k = n$; (ii) there exists an
       ordering of the set of conjunctions constructed using literals
       out of $S_a$, $\{\varphi_1, \ldots, \varphi_{2^k}\}$, such that $\varphi_i$ holds in $\delta'_i$ where
       $\delta'_i$ is a partial extension of $\delta_i$ and $a$ is strongly applicable in
       $\Delta' = \{\delta'_1, \ldots, \delta'_n\}$ w.r.t $S_a$. If $\delta' = Regress(a, \Delta_s) \notin N_s$,
       add $\langle a; case(\varphi_1 \rightarrow c_1, \ldots, \varphi_n \rightarrow c_n), \delta' \rangle$ to $N$.
5. Until $N$ does not change.
6. Return NO SOLUTION.
```

Below, we demonstrate how our algorithm works.

Example 3 (Getting to Evanston - con't) *We have that $G = \{$ at-evanston$\}$ and the initial condition $I = \{$ at-start, $\neg$ on-western, $\neg$ on-belmont, $\neg$ on-ashland, $\neg$ at-evanston, $\}$. So, $\delta_G = [\{$ at-evanston $\}, \{\}]$. The algorithm goes through the iterations (#I) in Figure (2).*

#I	*Action (a)*	*Regressed-from member of N*
0		$\langle [], \delta_G \rangle$
1	a_1 =*take-ashland*	$\langle [], \delta_G \rangle$
2	b_1 =*take-western*	$\langle [], \delta_G \rangle$
3	a_2 =*take-belmont*	$\langle a_1, \delta 1_1 \rangle$
4	b_2 =*goto-western-at-belmont*	$\langle b_1, \delta 2_1 \rangle$
5	a_3 =*goto-western-at-belmont*	$\langle a_2; a_1, \delta 1_2 \rangle$
6	*check-traffic*	$\langle a_3; a_2; a_1, \delta 1_3 \rangle, \langle b_2; b_1, \delta 2_2 \rangle$

#I	$Regress(a, \delta)$/$Regress(a, \Delta)$	*New member of N*
0		
1	$\delta 1_1 = [\{$*on-ashland*$\}, \{\}]$	$\langle a_1, \delta 1_1 \rangle$
2	$\delta 2_1 = [\{$*on-western*$\}, \{$*traffic-bad*$\}]$	$\langle b_1, \delta 2_1 \rangle$
3	$\delta 1_2 = [\{$*on-belmont, traffic-bad*$\}, \{\}]$	$\langle a_2; a_1, \delta 1_2 \rangle$
4	$\delta 2_2 = [\{$*at-start*$\}, \{$*traffic-bad*$\}]$	$\langle b_2; b_1, \delta 2_2 \rangle$
5	$\delta 1_3 = [\{$*at-start,traffic-bad*$\}, \{\}]$	$\langle a_3; a_2; a_1, \delta 1_3 \rangle$
6	$\delta 1_4 = [\{$*at-start*$\}, \{\}]$	$\langle p, \delta 1_4 \rangle$

where $\Delta = \{\delta_1, \ldots, \delta_n\}$ and $p = check\text{-}traffic; case(traffic\text{-}bad \rightarrow a_3; a_2; a_1, \neg\, traffic\text{-}bad \rightarrow b_2; b_1)$.

Figure 2: Algorithm illustration

The following theorem establishes the correctness of our algorithm.

Theorem 6 *For every $\langle c, \delta \rangle \in N$ where N represents the search space of the planning algorithm, then $Regress^*(c, \delta_G) = \delta$.*

Since the algorithm search through all possible regression path, we have the following theorem.

Theorem 7 *For a planning problem $P = \langle A, O, I, G \rangle$,*

- *if P has a regression solution then* **Solve**(P) *will return a conditional plan; and*
- *if P has no regression solution then* **Solve**(P) *will return NO SOLUTION.*

Experimentation

We have experimentally compared our system with the two systems (Weld, Anderson, & Smith 1998; Son, Tu, & Baral 2004) in domains with *sensing actions and incomplete information* but did not compare our planner with (Pryor & Collins 1996) since the planner in (Weld, Anderson, & Smith 1998) is significantly better than that of (Pryor & Collins 1996). We also did not compare our system with others that deal with nondeterministic or probabilistic actions as our action representation does not have this capability.

We run our Java-based planner with three well known *domains with sensing actions: Cassandra, Bomb in the toilet, and Sickness domain.* These domains are obtained from the SGP distribution (Weld, Anderson, & Smith 1998). All experiments are run on a Compaq laptop 1.8Ghz CPU with 512 MbRAM. The experimental result (obtained without using heuristics) is presented in Figure (4). It is necessary to note that, Figure (4) is a crude comparison as the other two use static causal laws and boolean sensing fluents (e.g. in Bomb in the toilet domain) while ours uses multi-valued sensing fluents; and the Logic Programming based planner ($\pi(P)$) uses conditional effects but ours does not.

Domains/		aSense			$\pi(P)$	SGP
Problem		preprocessing	search	total		
Cassandra	a1-prob	50	10	**60**	510	130
	a2-prob	50	10	**60**	891	60
	a3-prob	70	0	**70**	119	70
	a4-prob	60	220	**280**	1030	431
	a5-prob	30	10	**40**	130	20
	a6-prob	200	1392	**1592**	18036	NA [2]
	a7-prob	40	10	**50**	150	110
Bomb	bt-1sa	40	0	**40**	15812	751
	bt-2sa	40	10	**50**	18676	1161
	bt-3sa	40	10	**50**	18445	1512
	bt-4sa	200	10	**210**	22391	1892

Figure 4: Performance comparison (time in milliseconds).

Conclusion and Future Work

In this paper, we used the 0-approximation semantics (Son & Baral 2001) and defined regression with respect to that semantics. We considered domains where an agent does not have complete information about the world, and may have sensing actions. We first started with domains having only Boolean fluents and formally related our definition of regression with the earlier definition of progression in (Son & Baral 2001). We showed that planning using our regression function would indeed give us correct plans. We then presented a search algorithm for generating conditional plans. Lastly, we presented preliminary experimental results and discussed difficulties we faced as well as future enhancements. To simplify our formulation, we used the STRIPS-like action representation and considered fluents with finite domains.

Our planner is sound, however the use of 0-approximation leads to its incompleteness w.r.t. the full semantics [3]. This is a trade-off to counter the higher complexity thus leading to the efficiency of search for plans. Other limitations due to state space regression are difficulties in handling static causal laws and conditional effects. To further improve the search efficiency, we have derived several heuristics by extending the work of (Bonet & Geffner 2001) to handle sensing actions. However, due to the space limitation, we do not present them here. We will include them in the full paper. An issue that needs to be addressed in the future is to find sufficiency conditions that would guarantee the completeness of our approach with respect to the full semantics. We also need to directly consider actions with conditional effects, nondeterministic actions, and static causal laws and develop regression operators for these cases.

[3]Note that 0-approximation reduces the complexity of the planning problem to NP

Acknowledgement

This work was supported by NSF grant number 0070463 and NASA grant number NCC2-1232. Tran Cao Son was supported by NSF grants EIA-0130887 and EIA-0220590.

References

Baral, C.; Kreinovich, V.; and Trejo, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *AI-00* 122:241–267.

B. Bonet and H. Geffner. Planning with Incomplete Information as Heuristic Search in Belief Space AIPS-00, 52-61, 2000.

Bonet, B., and Geffner, H. 2001. Planning As Heuristic Search. *Artificial Intelligence* 129:5–33.

Cimatti, A.; Roveri, M.; and Traverso, P. 1998. Automatic OBDD-based Generation of Universal Plans in Non-Deterministic. AAAI-98, 875–891.

T. Eiter; W. Faber; N. Leone; G. Pfeifer; and A. Polleres. 2000. Planning under incomplete information. CL-2000, 807–821.

Etzioni, O.; Hanks, S.; Weld, D.; Draper, D.; Lesh, N.; and Williamson, M. 1992. An approach to planning with incomplete information. KR-92, 115–125.

Fikes, R., and Nilson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *AI-71*.

Goldman, R., and Boddy, M. 1994. Conditional linear planning. Planning AIPS-94.

Levesque, H. 1996. What is planning in the presence of sensing. AAAI-96.

Lobo, J. 1998. COPLAS: a COnditional PLAnner with Sensing actions. Technical Report FS-98-02, AAAI.

Nguyen, X.L; Kambhampati, S., and Nigenda, R. 2002. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. AIJ, 135(1-2):73–123.

Pednault, E. 1986. *Toward a Mathematical Theory of Plan Synthesis*. Ph.D. Dissertation.

Pednault, E. 1994. ADL and the state-transition model of actions. *Journal of Logic and Computation* 4(5):467–513.

Peot, M., and Smith, D. 1992. Conditional Nonlinear Planning. AIPS-92, 189–197.

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision-based approach. *JAIR-96*, 4:287–339.

Reiter, R. *KNOWLEDGE IN ACTION: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press.

Rintanen, J. 2000. Constructing conditional plans by a theorem prover. *Journal of Artificial Intelligence Research* 10:323–352.

Rintanen, J. 2002. Backward Plan Construction for Planning with Partial Observability. Planning AIPS'02.

Scherl, R., and Levesque, H. 1993. The frame problem and knowledge producing actions. Artificial AAAI-93, 689–695.

Scherl, R., and Levesque, H. 2003. Knowledge, Action, and the Frame Problem. *Artificial Intelligence* 144(1-2).

Son, T., and Baral, C. 2001. Formalizing sensing actions - a transition function based approach. AIJ, 125(1-2):19–91.

Son, T.; Tu, P.; and Baral, C. 2004. Planning with Sensing Actions and Incomplete Information using Logic Programming. Logic LPNMR'04, 261–274.

Weld, D.; Anderson, C.; and Smith, D. 1998. Extending graphplan to handle uncertainity and sensing actions. AAAI 98.