A STATE-BASED REGRESSION FORMULATION FOR DOMAINS WITH SENSING ACTIONS AND INCOMPLETE INFORMATION

LE-CHI TUAN [a], CHITTA BARAL [b], AND TRAN CAO SON [c]

[a] GCAS Incorporated, 1531 Grand Avenue, San Marcos, CA 92078, USA
  e-mail address: lctuan@gcas.net

[b] Computer Science and Engineering, Arizona State University, Tempe, AZ 85287, USA
  e-mail address: chitta@asu.edu

[c] Computer Science Department, New Mexico State University, Las Cruces, NM 88003, USA
  e-mail address: tson@cs.nmsu.edu

ABSTRACT. 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 and employ a three-valued characterization of domains with sensing actions to define the regression function. We prove the soundness and completeness of our regression formulation with respect to the definition of progression. More specifically, we show that (i) a plan obtained through regression for a planning problem is indeed a progression solution of that planning problem, and that (ii) for each plan found through progression, using regression one obtains that plan or an equivalent one.

1. INTRODUCTION AND MOTIVATION

An important aspect in reasoning about actions and characterizing the semantics of action description languages is to define a transition function that encodes 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 a simple case where we have only non-sensing actions and the progression transition function is deterministic, there are various formulations of regression. For example, let us 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 the 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 to define regression 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. Typically, a planning problem is specified by a set of actions, an initial state, and a goal state, which is a conjunction of literals. As such, regression is often defined with respect to a set of literals and an action. In that case the conjunction of literals (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 [9], 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. Intuitively, $Regress(conj, a)$ represents a minimal requirement on states from which the execution of a leads to states satisfying $conj$. The relation between these two, formally proven in [16], shows the correctness of regression based planners; which, through use of heuristics (e.g. [4, 14]), have done well in planning competitions. However, the focus of these papers has been the regression function in domains where agents have *complete* knowledge about the world. The following example shows that this property does not always holds.

Example 1.1. Consider the following do-or-die story[1]:

> A *wannabe prince* faces the last task in his endeavor. He stands in front of two rooms. In one room is a tiger and in the other is the princess, whom he wants to marry. Opening the room with the tiger will result in him being eaten. Otherwise, he will be able to rescue the princess and will get to marry her. He does not know exactly in which room the princess is. However, he can use a specialized[2] smell sensor that can precisely tell him where the tiger is.

The story can be formalized as follows. Let us denote the rooms by 1 and 2. $in(t, R)$ (resp. $in(p, R)$) denotes that the tiger (resp. the princess) is in room R. Initially, the agent (i.e., the want-to-be prince) is *alive*; he does not know what is behind the door of each room (i.e., the truth value of $in(t, R)$ and $in(p, R)$ is unknown to him) but he knows that the tiger and the princess are in different rooms (i.e., if $in(t, 1)$ is true then $in(p, 2)$ is true, etc.); he can execute $open(1)$ and $open(2)$. Executing the action $open(R)$, when $in(t, R)$ is true, causes him to be death ($\neg alive$); otherwise, the princess gets rescued. The agent can *determine* (by smelling) the truth value of $in(t, 1)$ and $in(t, 2)$. If he dies, he can not execute any action.

It is easy to see that the only possible way for the agent to achieve his goal is to begin by determining where the tiger is (by executing the action *smell*); after that, depending on where the tiger is, he can open the other room to rescue the princess. Observe that this plan involves the action *smell* whose execution does not change the world but changes the knowledge of the agent. Furthermore, the second action of the plan depends on the knowledge of the agent after the execution of the first action. We say that the agent needs a *conditional plan* with sensing actions to achieve his goal.

Reasoning about the effects of actions and changes in the presence of sensing actions and incomplete information has been the topic of intensive research (e.g., [10, 12, 13, 18, 20]

---

[1]This story was brought to us by a participant of a Texas Action Group (TAG) meeting at Lubbock in 2002 during a discussion on the need of sensing actions in reasoning about actions and changes and planning.

[2]Because the rooms are too close to each other, the natural smelling ability of a human is not quite accurate.

and the discussion in these papers). In general, the progression function for action theories with sensing actions and incomplete information is defined as a mapping from pairs of actions and belief states to belief states, where each belief state is a set of possible states. Intuitively, a belief state represents the set of possible states an agent thinks he might be in given his knowledge about the world. For example, the initial belief state of the want-to-be-prince in Example 1.1 consists of every possible state of the world; and, after the execution of the action *smell*, his belief state consists of a single state in which he is alive and knows the location of the tiger and the princess.

It has been also recognized that the planning problem in domains with sensing actions and incomplete information has a higher complexity than the planning problem in domains with complete information [1]. Furthermore, plans for achieving a goal in these domains will sometime require sensing actions and conditionals [10, 20]. It should be noted that there is an alternative approach to planning in the presence of incomplete information, called 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. Example 1.1 indicates that this is inadequate for many planning problems. In the past, several planners capable of generating conditional plans have been developed (e.g., [5, 11, 21]) in which some form of the progression function has been used.

In this regards, two natural questions arise:

- How to define a regression function in the presence of incomplete information and sensing actions?
- What should be the result of the regression of a state or a formula over a conditional plan? and, how can it be computed?

In the literature, we can find several proposals addressing the first question [19, 18, 6, 20], among them only the proposal in [20] partly discusses the second one. Moreover, all previous regression functions with respect to domains with incomplete information and sensing actions are about regression of formulas.

In this paper we are concerned with domains where the agent does not have complete information about the world, and may have sensing actions. For such domains, we define a regression function with respect to states. We then formally relate our definition of regression with the earlier notion of progression and show that planning using our regression function will not only give us correct plans but also will not miss plans. In summary the main contributions of our paper are:

- A state-based regression function for STRIPS domains with sensing actions and incomplete initial state;
- An extended regression function that allows for the regression from a (goal) state over a conditional plan; and
- A formal result showing the soundness and completeness of our regression function with respect to the progression function.

The rest of this paper is organized as follows. First, we review the necessary background information for understanding the technical details of the paper. We then present the regression formulation (Section 3) and prove its soundness and completeness with respect to the progression function (Section 4). We relate our work to other work in regression in Section 5 and conclude in Section 6.

2. Background

In this section, we present our action and plan representation and its semantics.

2.1. **Action and Plan Representation.** We employ a STRIPS-like action representation [9] 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 are sets of fluent literals where a fluent literal is either a fluent $f \in A$ (a.k.a. *positive fluent literal*) or its negation $\neg f$ (a.k.a. *negative fluent literal*). Intuitively, I encodes what is known about the initial state and G encodes what is desired of a goal state.

An action $a \in O$ is either a *non-sensing action* or a *sensing action* and is defined as follows:

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

$$\text{action } a \quad \text{:Pre } Pre_a \quad \text{:Add } Add_a \quad \text{:Del } Del_a$$

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 fluents representing the positive and negative effects of a, respectively; and

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

$$\text{action } a \quad \text{:Pre } Pre_a \quad \text{:Sense } Sens_a$$

where Pre_a is a set of fluent literals and $Sens_a$ is a subset of the fluents that do not appear in Pre_a, .i.e., $Pre_a \cap (\{\neg f \mid f \in Sens_a\} \cup Sens_a) = \emptyset$. As with non-sensing actions, a sensing action might only be executed under certain condition, which is represented by Pre_a. Intuitively, Pre_a is the condition under which a can be executed, and hence, needs to be known to be true before the execution of a. On the other hand, $Sens_a$ is the set of fluents that are unknown at the time of execution. For this reason, we require that none of the fluents in $Sens_a$ appear in Pre_a. As an example of a sensing action with precondition, consider the action of looking into the refrigerator to determine whether there is some beer or not. This action requires that the refrigerator door is open and can be represented by the action *look* with the condition $Pre_{look} = \{door\_open\}$ and $Sens_{look} = \{beer\_in\_fridge\}$.

The next example shows a simple domain in our representation.

Example 2.1. Figure (1) shows the actions of the "Getting to Evanston" domain from [17] in our representation.

The first four rows of the tables describe different non-sensing actions (driving actions) with their corresponding preconditions and add- and delete-effects. Each action can be executed when the agent is at certain locations (the second column) and changes the location of the agent after its completion. For instance, goto-western-at-belmont can be executed if the agent is at-start; its effect is that the agent will be on-western and on-belmont (third column) and no longer at-start (fourth column).

The last two rows represent two sensing actions, neither requires a precondition; one allows the agent to check for traffic condition (check-traffic) and the other one (check-on-western) allows for the agent to check whether it is (on-belmont) or not.

The notion of a plan in the presence of incomplete information and sensing actions has been extensively discussed in the literature [10, 12, 19, 20]. In this paper, we consider *conditional plans* that are formally defined as follows.

Action Name	: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}	

Action Name	:Pre	:Sense	
check-traffic	∅	{traffic-bad}	
check-on-western	∅	{on-belmont}	

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

Definition 2.2 (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 and $\varphi_1, \ldots, \varphi_n$ are mutually exclusive conjunctions of fluent literals and $c_1, \ldots, c_n$ are conditional plans, then

$$a; case(\varphi_1 \rightarrow c_1, \ldots, \varphi_n \rightarrow c_n)$$

is a conditional plan [3].
- If a is a non-sensing action and c is a conditional plan, then $a; c$ is a conditional plan.
- Nothing else is a conditional plan.

Intuitively, to execute a plan $a; case(\varphi_1 \rightarrow c_1, \ldots, \varphi_n \rightarrow c_n)$, first a is executed, φ_i's are then evaluated. If one of φ_i is true then c_i is executed. If none of φ_i is true then the plan fails. To execute a plan $a; c$, first a is executed then c is executed.

Example 2.3 (Getting to Evanston). The following is a conditional plan:

check-traffic;
case(
    traffic-bad →
        goto-western-at-belmont;
        take-belmont;
        take-ashland
    ¬traffic-bad →
        goto-western-at-belmont;
        take-western
)

2.2. **The Progression Function.** In the presence of incomplete information, the knowledge of an agent can be approximately captured by three disjoint sets of fluents: the set of fluents known to be true, false, and unknown to him, respectively. Thus, we represent the knowledge of an agent by a pair $\langle T, F \rangle$, called an *approximate state* (or a-state), where

---

[3]We often refer to this type of conditional plans as *case plans*.

$T{\subseteq}A$ and $F{\subseteq}A$ are two disjoint sets of fluents. Intuitively, $\langle T, F \rangle$ represents the knowledge of an agent who knows that fluents in T (resp. F) are true (resp. false) and does not have any knowledge about fluents in $A \setminus (T \cup F)$. It can also be considered as the intersection of all belief states satisfying $T \cup \{\neg f \mid f \in F\}$.

Given a fluent f, 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. Furthermore, for two a-states $\sigma_1 {=} \langle T_1, F_1 \rangle$ and $\sigma_2 {=} \langle T_2, F_2 \rangle$:

(1) We call $\sigma_1 \cap \sigma_2 {=} \langle T_1 \cap T_2, F_1 \cap F_2 \rangle$ the intersection of σ_1 and σ_2.
(2) We say σ_1 extends σ_2, denoted by $\sigma_2 {\preceq} \sigma_1$ if $T_2 {\subseteq} T_1$ and $F_2 {\subseteq} F_1$. $\sigma_1 {\setminus} \sigma_2$ denotes the set $(T_1 {\setminus} T_2) \cup (F_1 {\setminus} F_2)$.
(3) For a set of fluents X, we write $X {\setminus} \langle T, F \rangle$ to denote $X {\setminus} (T \cup F)$. To simplify the presentation, for a set of literals L, by L^+ and L^- we denote the set of fluents $\{f \mid f {\in} L, \ f$ is a fluent $\}$ and $\{f \mid \neg f {\in} L, \ f$ is a fluent $\}$.

The transition function (for progression) is defined next.

Definition 2.4 (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 $\Phi(a, \sigma) = \{\perp\}$;
- if a is executable in σ and a is a non-sensing action then

$$\Phi(a, \sigma) = \{\langle (T \setminus Del_a) \cup Add_a, (F \setminus Add_a) \cup Del_a \rangle\};$$

- if a is executable in σ and a is a sensing action then

$$\Phi(a, \sigma) = \{\sigma' \mid \sigma \preceq \sigma' \text{ and } Sens_a \setminus \sigma = \sigma' \setminus \sigma\}.$$

Here, $\perp$ denotes the "error state." $\Phi(a, \sigma) = \{\perp\}$ indicates that the action a cannot be executed in the a-state σ. The next example illustrates the above definition.

Example 2.5 (Getting to Evanston)**.** Consider the a-state

$$\sigma = \langle \{at\text{-}start\}, \{on\text{-}western, on\text{-}belmont, on\text{-}ashland, at\text{-}evanston\} \rangle.$$

We have that *check-traffic* is executable in σ and

$$\Phi(check\text{-}traffic, \sigma) = \{\sigma_1, \sigma_2\}$$

where:
$\sigma_1 = \langle \{at\text{-}start, traffic\text{-}bad\}, \{on\text{-}western, on\text{-}belmont, on\text{-}ashland, at\text{-}evanston\} \rangle,$
$\sigma_2 = \langle \{at\text{-}start\}, \{traffic\text{-}bad, on\text{-}western, on\text{-}belmont, on\text{-}ashland, at\text{-}evanston\} \rangle.$
Similarly,

$$\Phi(goto\text{-}western\text{-}at\text{-}belmont, \sigma) = \{\sigma_3\}$$

where: $\sigma_3 = \langle \{$on-western,on-belmont$\}, \{$at-start, on-ashland, at-evanston $\} \rangle.$

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 $\Phi^*(p, \sigma)$. Φ^* is defined similarly to the extended function $\hat{\Phi}$ in [20].

Definition 2.6 (Extended Transition Function)**.** For an a-state σ,

- if $c = [\ ]$, then $\Phi^*([\ ], \sigma) = \{\sigma\}$;

- if $c = a$ and a is a non-sensing action, then $\Phi^*(c, \sigma) = \Phi(a, \sigma)$;
- if $c = a; case(\varphi_1 \to p_1, \ldots, \varphi_n \to p_n)$ is a case plan, then

$$\Phi^*(c, \sigma) = \bigcup_{\sigma' \in \Phi(a, \sigma)} E(case(\varphi_1 \to p_1, \ldots, \varphi_n \to p_n), \sigma')$$

where

$$E(case(\varphi_1 \to p_1, \ldots, \varphi_n \to p_n), \gamma) = \begin{cases} \Phi^*(p_j, \gamma), & \text{if } \varphi_j \text{ holds in } \gamma \ (1 \leq j \leq n); \\ \{\bot\}, & \text{if none of } \varphi_1, \ldots, \varphi_n \text{ holds in } \gamma. \end{cases}$$

- if c is a conditional plan and a is a non-sensing action, then

$$\Phi^*(a; c, \sigma) = \bigcup_{\sigma' \in \Phi(a, \sigma)} \Phi^*(c, \sigma').$$

Furthermore, $\Phi^*(c, \bot) = \{\bot\}$ for any conditional plan c.

Intuitively, $\Phi^*(c, \sigma)$ is the set of a-states resulting from the execution of the plan c in σ.

Given a planning problem $P = \langle A, O, I, G \rangle$, the a-state representing I is defined by $\sigma_I = \langle I^+, I^- \rangle$. $\Sigma_G = \{\sigma \mid \sigma_G \preceq \sigma\}$, where $\sigma_G = \langle G^+, G^- \rangle$, is the set of a-states satisfying the goal G. We define a progression solution as follows.

Definition 2.7 (Progression Solution)**.** A *progression solution* to the planning problem P is a conditional plan c such that $\Phi^*(c, \sigma_I) \subseteq \Sigma_G$.

Note that, since $\bot$ is not a member of Σ_G, we have that $\bot \notin \Phi^*(c, \sigma_I)$ if c is a progression solution to P. In other words, the execution of c will not fail if c is a progression solution of P.

Example 2.8 (Getting to Evanston - cont'd)**.** Let $P = \langle A, O, I, G \rangle$ where A and O are given in Figure (1) and

$I = \{at\text{-}start, \neg on\text{-}western, \neg on\text{-}belmont, \neg on\text{-}ashland, \neg at\text{-}evanston\}$;
$G = \{at\text{-}evanston\}$,

respectively. We can easily check that the conditional plan in Example 2.3 is a progression solution of P.

2.3. **Some Properties of the Progression Function Φ.** There have been several proposals on defining a progression function for domains with sensing actions and incomplete information [10, 12, 13, 18, 20]. We will show next that for domains considered in this paper, the function Φ (Definition 2.4) is equivalent to the transition function defined in [20]. By virtue of the equivalent results between various formalisms, in [20], we can conclude that Φ is equivalent to the progression functions defined in several other formalisms as well. First, let us review the definition of the function in [20], which will be denoted by Φ_c. We need the following notations. For an action theory given by a set of fluents A, a set of operators O, and an initial state I, a state s is a set of fluents. A *combined state* (or *c-state*) of an agent is a pair $\langle s, \Sigma \rangle$ where s is a state and Σ is a set of states. Intuitively, the state s in a c-state $\langle s, \Sigma \rangle$ represents the real state of the world whereas Σ is the set of possible states which an agent believes it might be in. A c-state $\omega = \langle s, \Sigma \rangle$ is *grounded* if $s \in \Sigma$. A fluent f is true (resp. false) in s iff $f \in s$ (resp. $f \notin s$). f is known to be true (resp. false) in a c-state $\langle s, \Sigma \rangle$ iff f is true (resp. false) in every state $s' \in \Sigma$; and f is *known* in $\langle s, \Sigma \rangle$, if f is known to be true or known to be false in $\langle s, \Sigma \rangle$.

For an action a and a state s, a is executable in s if $Pre_a^+ \subseteq s$ and $Pre_a^- \cap s = \emptyset$. The state resulting from executing a in s, denoted by $Res(a, s)$, is defined by $Res(a, s) = (s \setminus Del_a) \cup Add_a$. The function Φ_c is a mapping from pairs of actions and c-states into c-states and is defined as follows. For a c-state $\omega = \langle s, \Sigma \rangle$ and action a,

(1) if a is not executable in s then $\Phi_c(a, \omega)$ is undefined, denoted by $\Phi_c(a, \omega) = \bot$;

(2) if a is executable in s and a is a non-sensing action, then

$$\Phi_c(a, \omega) = \langle Res(a, s), \{s' \mid s' = Res(a, s''), \ \exists s'' \in \Sigma \text{ s.t. } a \text{ is executable in } s''\} \rangle;$$

and

(3) if a is executable in s and a is a sensing action then

$$\Phi_c(a, \omega) = \langle s, \{s' \mid s' \in \Sigma \text{ s.t. } Sens_a \setminus s = Sens_a \setminus s', \text{ and } a \text{ is executable in } s'\} \rangle.$$

The set of initial states of a planning problem $P = \langle A, O, I, G \rangle$ is $\Sigma_0 = \{s \mid I^+ \subseteq s, \text{ and } I^- \cap s = \emptyset\}$; and the set of initial c-states of P, denoted by Ω_I, is given by $\Omega_I = \{\langle s_0, \Sigma_0 \rangle \mid s_0 \in \Sigma_0\}$. The function Φ_c can be extended to define an extended progression function Φ_c^* over conditional plans and c-states, similar to the extended function Φ^* in Definition 2.6. The notion of a progression solution can then be defined accordingly. The following theorem states the equivalence between Φ and Φ_c for domains representable by the action representation language given in the previous subsection.

Proposition 2.9. For a planning problem $\langle A, O, I, G \rangle$, a conditional plan c is a progression solution with respect to Φ iff it is a progression solution with respect to Φ_c.

Proof. For an a-state $\sigma = \langle T, F \rangle$, let $\Sigma_\sigma = \{s \mid T \subseteq s \subseteq A \text{ and } F \cap s = \emptyset\}$, $\Delta_\sigma = \{\langle T', F' \rangle \mid T \subseteq T' \subseteq A, \ F \subseteq F' \subseteq A, \text{ and } T' \cap F' = \emptyset\}$, and $\Omega_\sigma = \{\langle s, \Sigma_\sigma \rangle \rangle \mid s \in \Sigma_\sigma\}$. Furthermore, for an action a and a set of c-states Ω, let $\widehat{\Phi_c}(a, \Omega) = \{\Phi_c(a, \omega) \mid \omega \in \Omega\}$. From the definition of Φ and Φ_c, we can easily verify that the following properties hold:

(1) An action a is executable in σ iff a is executable in every c-state belonging to Ω_σ.

(2) If a non-sensing action a is executable in σ and $\Phi(a, \sigma) = \{\langle T', F' \rangle\}$ then

$$T' = \bigcap_{u \in \Sigma, \langle s, \Sigma \rangle \in \widehat{\Phi_c}(a, \Omega_\sigma)} u$$

and

$$F' = \bigcap_{u \in \Sigma, \langle s, \Sigma \rangle \in \widehat{\Phi_c}(a, \Omega_\sigma)} (A \setminus u)$$

(3) If a sensing action a is executable in σ then

$$\widehat{\Phi_c}(a, \Omega_\sigma) = \bigcup_{\sigma' \in \Delta_\sigma, \sigma'' \in \Phi(a, \sigma')} \Sigma_{\sigma''}$$

The conclusion of the proposition can be verified using induction on the structure of a plan and the fact that for a planning problem P, $\Omega_I = \Omega_{\sigma_I}$. $\square$

Remark 2.10. For the discussion on the complexity of planning using the progression function Φ, it will be useful to note that Φ is also equivalent to the 0-approximation Φ_0 defined in [20]. Indeed, this can be easily verified from the definitions of Φ_0 and Φ. In the notation of this paper, Φ_0 is also a mapping from pairs of actions and a-states into a-states and for an a-state $\sigma = \langle T, F \rangle$ and an action a, $\Phi_0(a, \sigma)$ is defined as follows:

- if a is not executable in σ then $\Phi_0(a, \sigma) = \{\bot\}$;
- if a is executable in σ and a is a non-sensing action then

$$\Phi_0(a, \sigma) = \{\langle (T \setminus Del_a) \cup Add_a, (F \setminus Add_a) \cup Del_a \rangle\};$$

- if a is executable in σ and a is a sensing action then

$$\Phi_0(a, \sigma) = \{\sigma' | \sigma \preceq \sigma' \text{ and } Sens_a \setminus \sigma = \sigma' \setminus \sigma\}.$$

This implies that Φ is identical to Φ_0.

3. A State-Based Regression Formulation

In this section, we present our formalization of a regression function, denoted by $\mathcal{R}$, and prove that it is both sound and complete with respect to the progression function Φ. $\mathcal{R}$ is a state-based regression function that maps each pair of an action and a set of a-states into an a-state.

Observe that our progression formulation states that a plan p achieves the goal G from an a-state σ if G holds in *all* a-states belonging to $\Phi^*(p, \sigma)$, i.e., G holds in $\cap_{\sigma' \in \Phi^*(p,\sigma)} \sigma'$. In addition, similar to [4], we will also define regression with respect to the goal. These suggest us to introduce the notion of a *partial knowledge state* (or p-state) as a pair $[T, F]$ where $T \subseteq A$ and $F \subseteq A$ are two disjoint sets of fluents. Intuitively, a p-state $\delta = [T, F]$ represents a collection of a-states which extend the a-state $\langle T, F \rangle$. We denote this set by $ext(\delta)$ and call it *the extension set* of δ. Formally, $ext(\delta) = \{\langle T', F' \rangle \mid T \subseteq T' \subseteq A, F \subseteq F' \subseteq A, T' \cap F' = \emptyset\}$. Any a-state $\sigma' \in ext(\delta)$ is called an extension of δ. Given a p-state $\delta = [T, F]$, we say a partial state $\delta' = [T', F']$ is a partial extension of δ if $T \subseteq T', F \subseteq F'$. For a set of p-states $\Delta = \{\delta_1, \ldots, \delta_n\}$, $\Delta' = \{\delta'_1, \ldots, \delta'_n\}$ is said to be an extension of Δ, written as $\Delta \sqsubseteq \Delta'$ if δ'_i is a partial extension of δ_i for every $i = 1, \ldots, n$. For a fluent f, we say that f is true (resp. false, known, unknown) in δ if $f \in T$ ($f \in F$, $f \in T \cup F$, $f \notin T \cup F$). A set of fluents S is said to be true (resp. false, known, unknown) in δ if every fluent f in S is true (resp. false, known, unknown) in δ.

The regression function $\mathcal{R}$ will be defined separately for non-sensing actions and sensing actions to take into consideration the fact that the execution of a non-sensing action (resp. sensing action) in an a-state results in a single a-state (resp. set of a-states). Thereafter, $\mathcal{R}$ is extended to define regression over conditional plans. The key requirement on $\mathcal{R}$ is that it should be sound (i.e., plans obtained through regression must be plans based on the progression function) and complete (i.e., for each plan based on progression, using regression one should obtain that plan or a simpler plan with the same effects) with respect to progression. We will also need to characterize the conditions under which an action should not be used for regression. Following [4], we refer to this condition as "the applicability condition." We begin with non-sensing actions.

3.1. **Regression Over Non-Sensing Actions.** We begin with the applicability condition of non-sensing actions and then give the definition of the function $\mathcal{R}$ for non-sensing actions.

Definition 3.1 (Regression Applicability Condition – Non-Sensing Action)**.** Given a non-sensing action a and a p-state $\delta = [T, F]$. 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 $Pre_a^- \cap T \subseteq Add_a$.

Intuitively, the aforementioned applicability condition requires that a is *relevant* (item (i)) and *consistent* (item (ii)) in δ. Item (i) is considered "relevant" as it makes sure that the effects of a will contribute to δ after its execution. Item (ii) is considered "consistent" as it makes sure that the situation obtained by progressing a, from a situation yielded by regressing from δ through a, will be consistent with δ. Observe also that this definition will exclude the conventional operator *no-op* from consideration for regression as it is never applicable.

Since the application of a non-sensing action in an a-state results in a single a-state, the regression of a p-state over a non-sensing action should result in a p-state. This is defined next.

Definition 3.2 (Regression – Non-Sensing Action)**.** Given a non-sensing action a and a p-state $\delta = [T, F]$,

- if a is not applicable in δ then $\mathcal{R}(a, \delta) = \bot$;
- if a is applicable in δ then $\mathcal{R}(a, \delta) = [(T \setminus Add_a) \cup Pre_a^+, (F \setminus Del_a) \cup Pre_a^-]$.

Like in the progression function, the symbol $\bot$ indicates a "failure." In other words, $\mathcal{R}(a, \delta) = \bot$ means that δ cannot be regressed on a (or the regression from δ over a fails). For later use, we extend the regression function $\mathcal{R}$ for non-sensing actions over a set of p-states and define

$$\mathcal{R}(a, \{\delta_1, \ldots, \delta_n\}) = \{\mathcal{R}(a, \delta_1), \ldots, \mathcal{R}(a, \delta_n)\}$$

where $\delta_1, \ldots, \delta_n$ are p-states and a is a non-sensing action.

Example 3.3 (Getting to Evanston - con't)**.** The actions *take-western* and *take-ashland* are applicable in $\delta = [\{\text{at-evanston}\}, \{\}]$.
  $\mathcal{R}(take\text{-}western, \delta) = [\{on\text{-}western\}, \{traffic\text{-}bad\}]$, and
  $\mathcal{R}(take\text{-}ashland, \delta) = [\{on\text{-}ashland\}, \{\}]$.

3.2. **Regression Over Sensing Actions.** Let a be a sensing action and σ be an a-state. The definition of the progression function Φ states that the execution of a in σ results in a set of a-states $\Phi(a, \sigma)$. Furthermore, if a is executable in σ then every member of $\Phi(a, \sigma)$ extends σ by a set of fluents $s_a \subseteq Sens_a$ and every $f \in Sens_a \setminus s_a$ is known in σ. As such, the regression over a sensing action should be with respect to a set of p-states and result in a p-state. Moreover, our definition for the applicability condition of a sensing action must account for the fact that the set of p-states, from which the regression is done, satisfies the two properties: (*i*) $Sens_a$ is known in each of its members; and (*ii*) the difference between two of its members is exactly $Sens_a$. This leads to the following definition.

Definition 3.4 (Sensed Set of Fluents)**.** Let $\Delta = \{\delta_1, \ldots, \delta_n\}$ be a set of p-states and a be a sensing action. A *sensed set of fluents* of Δ with respect to a, denoted by $p(a, \Delta)$, is a non-empty subset of $Sens_a$ satisfying the following properties:

- $Sens_a$ is known in Δ;
- $n = 2^{|p(a,\Delta)|}$;
- for every partition[4] (P, Q) of $p(a, \Delta)$, there exists only one $\delta_i \in \Delta$ $(1 \leq i \leq n)$ such that $\delta_i.T \cap p(a, \Delta) = P$, $\delta_i.F \cap p(a, \Delta) = Q$; and
- $\delta_i.T \setminus p(a, \Delta) = \delta_j.T \setminus p(a, \Delta)$ and $\delta_i.F \setminus p(a, \Delta) = \delta_j.F \setminus p(a, \Delta)$ for every pair of i and j, $1 \leq i \leq n$ and $1 \leq j \leq n$.

It can be seen from Definition 2.4 (Case 2) that when a sensing action a is executed in an a-state σ, the result is a set of a-states $\Phi(a, \sigma)$ where for each $\sigma' \in \Phi(a, \sigma)$, $\sigma' \setminus \sigma = Sens_a \setminus \sigma$. The above definition captures the inverse of the progression process. Intuitively, $p(a, \Delta)$ is the set of fluents which are unknown before the execution of a and are known after its execution; for example, if $\Delta = \Phi(a, \sigma)$, then $p(a, \Delta)$ should encode the set $Sens_a \setminus \sigma$. It is easy to see that the second condition on $p(a, \Delta)$ warrants that it is a maximal subset of $Sens_a$ satisfying the four stated conditions. Observe also that due to the second condition, Δ must be a non-empty set. The next lemma proves that the sensed set of a set of p-states with respect to an action is unique.

Lemma 3.5. For every sensing action a and set of p-states Δ, $p(a, \Delta)$ is unique if it exists.

Proof. Abusing the notation, we write $p(a, \Delta) = \bot$ whenever $p(a, \Delta)$ does not exist. Clearly, the lemma holds if $\Delta = \emptyset$ as $p(a, \Delta) = \bot$ for every a. So, we need to prove it for the case $\Delta \neq \emptyset$.

Assume that $p(a, \Delta)$ exists but it is not unique, i.e, we can find different sensed sets of Δ with respect to a, say X and X'. By Definition 3.4, we have that $X \neq \emptyset$ and $X' \neq \emptyset$.

Since $X \neq \emptyset$, let us consider a fluent $f \in X$. By Definition 3.4, for two partitions $(\{f\}, X \setminus \{f\})$ and $(X \setminus \{f\}, \{f\})$ of X, there exist $\delta \in \Delta$ and $\delta' \in \Delta$ such that $\{f\} = \delta.T \cap X$ and $\{f\} = \delta'.F \cap X$, i.e. f is true in δ and false in δ'. [*].

Suppose that $f \notin X'$. By Item 4, Definition 3.4, either $f \in \delta.T \setminus X'$ or $f \in \delta.F \setminus X'$ for $\delta \in \Delta$, i.e f is either true or false in every $\delta \in \Delta$. In either case, this contradicts with [*]. Therefore, $f \in X'$.

Symmetrically, we can argue that, if $f \in X'$ then $f \in X$. Thus, $f \in X$ iff $f \in X'$, i.e. $X = X'$. $\qquad\square$

Definition 3.6 (Properness). A set of p-states Δ is *proper* with respect to a sensing action a if $p(a, \Delta) \neq \bot$.

For convenience, we sometime write $p(a, \Delta) = \bot$ to indicate that Δ is not proper with respect to a.

Example 3.7 (Getting to Evanston – Cond't). Consider a set $\Delta_1 = \{\delta_1, \delta_2\}$ where $\delta_1 = [\{at\text{-}start, traffic\text{-}bad\}, \{on\text{-}western, on\text{-}belmont, on\text{-}ashland, at\text{-}evanston\}]$ and

$$\delta_2 = [\{at\text{-}start\}, \{traffic\text{-}bad, on\text{-}western, on\text{-}belmont, on\text{-}ashland, at\text{-}evanston\}].$$

We can easily check that

$$p(check\text{-}traffic, \Delta_1) = \{traffic\text{-}bad\}.$$

So, Δ_1 is proper with respect to *check-traffic*.

On the other hand

$$p(check\text{-}traffic, \Delta_2) = \bot.$$

---

[4]For a set of fluents X, a *partition* of X is a pair of sets of fluents (P, Q) where $P \cap Q = \emptyset$ and $P \cup Q = X$.

where $\Delta_2 = \{\delta_1, \delta_3\}$ and $\delta_3 = [\{at\text{-}start\}, \{traffic\text{-}bad, at\text{-}evanston\}]$. This is because the fourth condition (Definition 3.4) cannot be satisfied for any non-empty subset of $traffic\text{-}bad$. So, Δ_2 is not proper with respect to $check\text{-}traffic$.

We are now ready to define the applicability condition for sensing actions. The definition is given in two steps. First, we define the strong applicability condition as follows.

Definition 3.8 (Strong Regression Applicability Condition – Sensing Action). Let a be a sensing action and Δ be a set of p-states. We say that a is *strongly applicable* in Δ if

  (i) $p(a, \Delta) \neq \bot$; and
  (ii) $Pre_a^+ \cap \delta.F = \emptyset$ and $Pre_a^- \cap \delta.T = \emptyset$ for every $\delta \in \Delta$.

In the above definition, (i) and (ii) correspond to the "relevancy" and "consistency" requirement for non-sensing actions (Definition 3.1) respectively. (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, each of which extends σ by $p(a, \Delta)$ and (ii) guarantees that a must be executable prior to its execution.

It is easy to see that if a sensing action a is strongly applicable in Δ, then for every a-state σ' extending the p-state

$$\delta' = [((\bigcup_{\delta \in \Delta} \delta.T) \setminus p(a, \Delta)) \cup Pre_a^+, ((\bigcup_{\delta \in \Delta} \delta.F) \setminus p(a, \Delta)) \cup Pre_a^-]$$

it holds that every member of $\Phi(a, \sigma')$ belongs to the extension of some $\delta_i \in \Delta$. As such, δ' could be viewed as the result of the regression from Δ through a. Unfortunately, the conditions in Definition 3.8 are sometime unnecessarily strong as the following example demonstrates.

Example 3.9 (Strong Regression Applicability Condition). Let

$$P = \langle \{f, g, h\}, \{sense_f, a_1, a_2\}, \{h\}, \{g\} \rangle$$

be a planning problem, where $sense_f$ is a sensing action with

$$Pre_{sense_f} = \{h\}, \ Sens_{sense_f} = \{f\};$$

a_1 and a_2 are two non-sensing actions with

$$Pre_{a_1} = \{h, f\}, \ Add_{a_1} = \{g\}, Del_{a_1} = \emptyset,$$
$$Pre_{a_2} = \{\neg f\}, \ Add_{a_2} = \{g\}, \text{ and } Del_{a_2} = \emptyset.$$

Clearly,

$$c = sense_f; case(f \to a_1, \neg f \to a_2)$$

is a progression solution to P. Thus, it is reasonable to expect that if we regress from the goal $\delta = [\{g\}, \emptyset]$ on c — step-by-step — we will receive a p-state δ' such that $\langle \{h\}, \emptyset \rangle \in ext(\delta')$. This process begins with the regression on a_1 and a_2 from δ. Thereafter, we receive a set of p-states from which the regression on $sense_f$ can be done. It is easy to see that $\mathcal{R}(a_1, \delta) = [\{h, f\}, \emptyset] = \delta_1$ and $\mathcal{R}(a_2, \delta) = [\emptyset, \{f\}] = \delta_2$. It is also easy to see that the strong applicability condition implies that $sense_f$ is not applicable in $\{\delta_1, \delta_2\}$ because $p(sense_f, \{\delta_1, \delta_2\}) = \bot$. This means that, we cannot regress on c from the goal state.

Notice that the problem in the above example lies in the fact that $\{\delta_1, \delta_2\}$ violates the properness definition in that δ_1 and δ_2 do not have the same values on the set of fluents that do not belong to $Sens_{sense_f}$. To overcome the problem posed by the strong applicability condition, we relax this condition.

Definition 3.10 (Regression Applicability Condition – Sensing Action)**.** Let a be a sensing action and Δ be a set of p-states. a is *applicable* in Δ if

  (i) there exists a set of p-states Δ' such that $\Delta \sqsubseteq \Delta'$ and a is strongly applicable in Δ'; and

  (ii) $Sens_a$ is known in Δ.

Example 3.11 (Continuation of Example 3.9)**.** It is easy to see that $sense_f$ is applicable in $\{\delta_1, \delta_2\}$ since it is strongly applicable in $\{\delta_1, \delta_2'\}$ where $\delta_2' = [\{h\}, \{f\}]$ and $\{\delta_1, \delta_2\} \sqsubseteq \{\delta_1, \delta_2'\}$.

Definition 3.12 (Sensed Set)**.** Let Δ be a set of p-states and a be a sensing action such that a is applicable in Δ. We say that X is a sensed set of fluents of Δ with respect to a, denoted by $S_{a,\Delta}$, if there exists a set of p-states Δ' such that $\Delta \sqsubseteq \Delta'$, a is strongly applicable in Δ', and $X = p(a, \Delta')$.

Again, we write $S_{a,\Delta} = \bot$ to say that the sensed set of fluents of Δ with respect to a does not exist. The next lemma states that $S_{a,\Delta}$ is unique.

Lemma 3.13. For every sensing action a and set of p-states Δ, $S_{a,\Delta}$ is unique if it exists.

Proof. Obviously, the lemma holds for $\Delta = \emptyset$. So, we need to prove it for the case $\Delta \neq \emptyset$.

Assume the contrary, $S_{a,\Delta}$ is not unique. This implies that there exists Δ' and Δ'' such that $\Delta \sqsubseteq \Delta'$ and $\Delta \sqsubseteq \Delta''$, $p(a, \Delta') \neq \bot$, $p(a, \Delta'') \neq \bot$, and $p(a, \Delta') \neq p(a, \Delta'')$. Again, by Definition 3.4 we can conclude that $p(a, \Delta') \neq \emptyset$ and $p(a, \Delta'') \neq \emptyset$. Without loss of generality, we conclude that there exists some $f \in p(a, \Delta') \setminus p(a, \Delta'')$.

Since $p(a, \Delta'') \neq \bot$, $f \in Sens_a$, and f is known in Δ'', by Definition 3.4, we must have two cases.

(1) $f \in \delta''.T \setminus p(a, \Delta'')$ for every $\delta'' \in \Delta''$. Since $f \in p(a, \Delta')$, by Definition 3.4, for the partition $(p(a, \Delta') \setminus \{f\}, \{f\})$ of $p(a, \Delta')$, there exists $\delta' \in \Delta'$ such that $\delta'.F \cap p(a, \Delta') = \{f\}$, i.e. f is false in δ'.

Because $\Delta \sqsubseteq \Delta'$, there exists some $\delta \in \Delta$ such that δ' is a partial extension of δ. So, we have that $\delta.F \subseteq \delta'.F$. Also, as $Sens_a$ is known in δ, we must have that $f \in \delta.F$.

Since $\Delta \sqsubseteq \Delta''$, we know that there exists some $\delta'' \in \Delta''$ which is a partial extension of δ. This implies that $\delta.F \subseteq \delta''.F$, i.e., $f \in \delta''.F$. This contradicts with the fact that $f \in \delta''.T$.

(2) $f \in \delta''.F \setminus p(a, \Delta'')$ for every $\delta'' \in \Delta''$. Similarly to the first case, we can derive a contradiction.

The above two cases show that if $f \in p(a, \Delta')$ then $f \in p(a, \Delta'')$.

This shows that $p(a, \Delta') = p(a, \Delta'')$. $\qquad\square$

We illustrate the above definition in the next example.

Example 3.14 (Getting to Evanston - con't)**.** Consider the set Δ_2 and the sensing action *check-traffic* in Example 3.7. We have that

  (i) *check-traffic* is not strongly applicable in Δ_2 (because $p(\textit{check-traffic}, \Delta_2) = \bot$, Example 3.7); however,

 (ii) *check-traffic* is applicable in Δ_2. This is because Δ_1 (Example 3.7) consists of partial extensions of p-states in Δ_2, and *check-traffic* is strongly applicable in Δ_1.

We are now ready to define the regression function for sensing actions.

Definition 3.15 (Regression – Sensing Action). Let a be a sensing action and Δ be a set of p-states.
- if a is not applicable in Δ then $\mathcal{R}(a, \Delta) = \perp$; and
- if a is applicable in Δ

$$\mathcal{R}(a, \Delta) = [((\bigcup_{\delta \in \Delta} \delta.T) \setminus S_{a,\Delta}) \cup Pre_a^+, ((\bigcup_{\delta \in \Delta} \delta.F) \setminus S_{a,\Delta}) \cup Pre_a^-].$$

Example 3.16 (Getting to Evanston – Cond't). The action *check-traffic* is applicable in Δ_2 with respect to $\{traffic\text{-}bad\}$ (see Example 3.14) and we have
$$\mathcal{R}(check\text{-}traffic, \Delta_2) =$$
$$[\{at\text{-}start\}, \{on\text{-}western, on\text{-}belmont, on\text{-}ashland, at\text{-}evanston\}].$$

3.3. **Regression Over Conditional Plans.** We now extend $\mathcal{R}$ to define $\mathcal{R}^*$ that allows us to perform regression over conditional plans. For a conjunction of fluent literals, by φ^+ and φ^- we denote the sets of fluents occurring positively and negatively in φ, respectively.

Definition 3.17 (Extended Regression Function). Let δ be a p-state. The extended transition function $\mathcal{R}^*$ is defined as follows:
- $\mathcal{R}^*([\,], \delta) = \delta$.
- For a non-sensing action a, $\mathcal{R}^*(a, \delta) = \mathcal{R}(a, \delta)$.
- For a conditional plan $p = a; case(\varphi_1 \rightarrow c_1, \ldots, \varphi_n \rightarrow c_n)$ where a is a sensing action and c_i's are conditional plans,
  – if $\mathcal{R}^*(c_i, \delta) = \perp$ for some i, $\mathcal{R}^*(p, \delta) = \perp$;
  – if $\mathcal{R}^*(c_i, \delta) = [T_i, F_i]$ for $i = 1, \ldots, n$, then
  $$\mathcal{R}^*(p, \delta) = \mathcal{R}(a, \{R(\varphi_1 \rightarrow c_1, \delta), \ldots, R(\varphi_n \rightarrow c_n, \delta)\})$$
  where $R(\varphi_i \rightarrow c_i, \delta) = [T_i \cup \varphi_i^+, F_i \cup \varphi_i^-]$ if $\varphi_i^+ \cap F_i = \emptyset$ and $\varphi_i^- \cap T_i = \emptyset$; otherwise, $R(\varphi_i \rightarrow c_i, \delta) = \perp$.
- For $p = a; c$, where a is a non-sensing action and c is a conditional plan,
  $$\mathcal{R}^*(p, \delta) = \mathcal{R}(a, \mathcal{R}^*(c, \delta));$$
- $\mathcal{R}^*(p, \perp) = \perp$ for every plan p.

The notion of a regression solution is defined as follows.

Definition 3.18 (Regression Solution). A conditional plan c is a *regression solution* to the planning problem $P = \langle A, O, I, G \rangle$ if $\mathcal{R}^*(c, \delta_G) \neq \perp$ and $\sigma_I \in ext(\mathcal{R}^*(c, \delta_G))$ where $\delta_G = [G^+, G^-]$ and $\sigma_I = \langle I^+, I^- \rangle$.

The above definition is a generalization of the notion of a plan obtained by regression in domains without sensing actions and with complete information about the initial state to domains with sensing actions and incomplete information. An important property that any regression function needs to satisfy is its soundness with respect to the corresponding progression function. Here, we would like to guarantee that $\mathcal{R}$ and $\mathcal{R}^*$ are sound with respect to the progression function Φ and Φ^*, respectively. As such, we require that a regression solution c to a planning problem $P = \langle A, O, I, G \rangle$ be a plan achieving the goal G from I. This property is proved in Theorem 4.8.

As the soundness of the regression function with respect to the progression function is guaranteed, it will be interesting to investigate its completeness. In this paper, we opt for a

definition that — when used in planning — will give us optimal solutions in the sense that regression solutions do not contain redundant actions. This is evident from Definitions 3.1 and 3.10 in which we require that the action, over which the regression is done, must add new information to the regressed state. We will elaborate in more detail on this point in the next section.

4. Soundness and Completeness Results

In this section, we show that our regression function $\mathcal{R}^*$ is sound and complete with respect to the progression function Φ.

4.1. **Soundness Result.** As with its definition, the soundness of $\mathcal{R}^*$ is proved in two steps. First, we prove the soundness of $\mathcal{R}$, separately for non-sensing and sensing actions. Second, we extend this result to regression solutions. To establish the soundness of $\mathcal{R}$ on non-sensing actions, we need the following lemma.

Lemma 4.1. Let δ be a p-state. An a-state σ is an extension of δ (i.e. $\sigma \in ext(\delta)$) iff σ is an a-state of the form $\langle \delta.T \cup X, \delta.F \cup Y \rangle$ where X, Y are two disjoint sets of fluents and $X \cap \delta.F = \emptyset$, $Y \cap \delta.T = \emptyset$.

Proof.
- Case "$\Rightarrow$":

  Let $\sigma \in ext(\delta)$ be an extension of δ. By the definition of an extension, σ is an a-state where $\delta.T \subseteq \sigma.T$ and $\delta.F \subseteq \sigma.F$. Denote $X = \sigma.T \setminus \delta.T$ and $Y = \sigma.F \setminus \delta.F$. Clearly, X and Y are two set of fluents where $X \cap Y = \emptyset$ and $X \cap \delta.F = \emptyset$, $Y \cap \delta.T = \emptyset$.
- Case "$\Leftarrow$":

  Let σ be an a-state of the form $\langle \delta.T \cup X, \delta.F \cup Y \rangle$ where X, Y are two disjoint sets of fluents and $X \cap \delta.F = \emptyset$, $Y \cap \delta.T = \emptyset$.

  It's easy to see that $\sigma.T \cap \sigma.F = \emptyset$, i.e. σ is consistent. Furthermore, $\delta.T \subseteq \sigma.T$ and $\delta.F \subseteq \sigma.F$, i.e. by definition of an extension, σ is an extension of δ. $\qquad\square$

Intuitively, the soundness of $\mathcal{R}$ for a non-sensing action states that the regression over a non-sensing action from a p-state yields another p-state such that the execution of the action in any extension of the latter results in a subset of a-states belonging to the extension set of the former. This is illustrated in Figure 2.
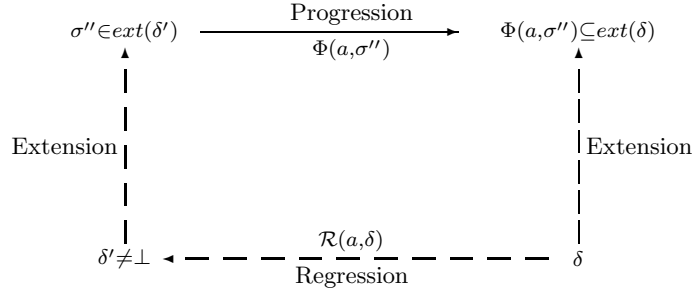

$$\sigma'' \in ext(\delta') \xrightarrow[\Phi(a,\sigma'')]{\text{Progression}} \Phi(a,\sigma'') \subseteq ext(\delta)$$

Extension $\quad$ Extension

$$\delta' \neq \perp \xleftarrow[\text{Regression}]{\mathcal{R}(a,\delta)} \delta$$

Figure 2: Illustration of Theorem 4.2.

Theorem 4.2 (Non-sensing Action). Let δ be a p-state and a be a non-sensing action. If $\mathcal{R}(a,\delta) = \delta'$ and $\delta' \neq \bot$, then for every $\sigma'' \in ext(\delta')$ we have that $\Phi(a,\sigma'') \subseteq ext(\delta)$.

Proof. Let $\delta = [T, F]$. From the fact that $\mathcal{R}(a,\delta) = \delta' \neq \bot$, we have that a is applicable in δ.

By Definition 3.2,

$$\delta' = \mathcal{R}(\delta,a) = [(T \setminus Add_a) \cup Pre_a^+, (F \setminus Del_a) \cup Pre_a^-].$$

Let $\sigma'' \in ext(\delta')$. It follows from Lemma 4.1 that

$$\sigma'' = \langle (T \setminus Add_a) \cup Pre_a^+ \cup X, (F \setminus Del_a) \cup Pre_a^- \cup Y \rangle,$$

where X and Y are two sets of fluents such that $\sigma''.T \cap \sigma''.F = \emptyset$. We now prove that (i) a is executable in σ'' and (ii) $\Phi(a,\sigma'') \subseteq ext(\delta)$.

- Proof of (i):
    Since $Pre_a^+ \subseteq \sigma''.T$ and $Pre_a^- \subseteq \sigma''.F$, we conclude that $lem1 - maintexta$ is executable in σ''.
- Proof of (ii):
    By definition of the transition function Φ, we have that

$$\Phi(a,\sigma'') = \{\langle (((T \setminus Add_a) \cup Pre_a^+ \cup X) \setminus Del_a) \cup Add_a, (((F \setminus Del_a) \cup Pre_a^- \cup Y) \setminus Add_a) \cup Del_a \rangle \}$$

   Since a is applicable in δ, we have that $T \cap Del_a = \emptyset$, $F \cap Add_a = \emptyset$. Furthermore, $Del_a \cap Add_a = \emptyset$. Therefore, we have that $(((T \setminus Add_a) \cup Pre_a^+ \cup X) \setminus Del_a) \cup Add_a = ((T \setminus Add_a) \cup ((Pre_a^+ \cup X) \setminus Del_a)) \cup Add_a \supseteq T \cup ((Pre_a^+ \cup X) \setminus Del_a) \supseteq T$. This concludes that $T \subseteq \Phi(a,\sigma'').T$. Similarly, we have that $F \subseteq \Phi(a,\sigma'').F$. This shows that $\Phi(a,\sigma'') \subseteq ext(\delta)$. $\square$

Observe that the conclusion of the theorem indicates that $\bot \notin \Phi(a,\sigma'')$, i.e., a is executable in σ''. This shows that $\mathcal{R}$ can be "reversed" for non-sensing actions.

We will next establish a result similar to Theorem 4.2 for sensing actions. Intuitively, the result should state that the regression over a sensing action from a set of p-states yields a p-state such that the execution of the action in any extension of the latter results in a set of a-states belonging to the union of the extension sets of the former, i.e., it should allow us to conclude that $\mathcal{R}$ can be "reversed" for sensing actions. Figure 3 illustrates this idea.
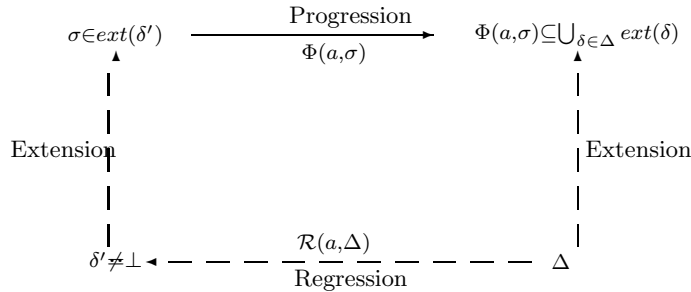

Figure 3: Illustration of Theorem 4.4.

.

We need the following lemma.

Lemma 4.3. Let σ' be an a-state and a be a sensing action executable in σ'. For any $S_a \subseteq Sens_a$ and $\sigma \in \Phi(a, \sigma')$, let $\sigma.T \cap S_a = S_\sigma^+$ and $\sigma.F \cap S_a = S_\sigma^-$, we have that $S_\sigma^+ \cup S_\sigma^- = S_a$ and $S_\sigma^+ \cap S_\sigma^- = \emptyset$.

Proof. It is easy to see that the lemma is correct for the case $S_a = \emptyset$. Let us consider the case $S_a \neq \emptyset$. Since $S_\sigma^+ \subseteq \sigma.T$ and $S_\sigma^- \subseteq \sigma.F$, we have that $S_\sigma^+ \cap S_\sigma^- = \emptyset$. . This also shows Consider $f \in S_\sigma^+ \cup S_\sigma^-$, we have that $f \in S_\sigma^+$ or $f \in S_\sigma^-$. In both cases, we have $f \in S_a$.

Consider $f \in S_a$. Since $S_a \subseteq Sens_a$, we have that $f \in Sens_a$. By the definition of Φ, we have that $f \in \sigma.T$ or $f \in \sigma.F$. From this fact, it's easy to see that $f \in S_\sigma^+$ or $f \in S_\sigma^-$. $\square$

With the help of the above lemma, we can prove the following theorem.

Theorem 4.4 (Sensing action). Let Δ be a set of p-states and a be a sensing action. If $\mathcal{R}(a, \Delta) = \delta'$ and $\delta' \neq \perp$, then for every $\sigma'' \in ext(\delta')$, we have that $\Phi(a, \sigma'') \subseteq \bigcup_{\delta \in \Delta} ext(\delta)$.

Proof. From the fact that $\mathcal{R}(a, \Delta) = \delta' \neq \perp$, we have that a is applicable in Δ with respect to some set $S_{a,\Delta} \subseteq Sens_a$ ($S_{a,\Delta} \neq \emptyset$). By Definition 3.15 we have:

$$\delta' = \mathcal{R}(a, \Delta) = [(\bigcup_{\delta \in \Delta} \delta.T \setminus S_{a,\Delta}) \cup Pre_a^+, (\bigcup_{\delta \in \Delta} \delta.F \setminus S_{a,\Delta}) \cup Pre_a^-].$$

Let $\sigma'' \in ext(\delta')$ be an arbitrary extension of δ'. We will now prove (i) a is executable in σ'' and (ii) $\Phi(a, \sigma'') \subseteq \bigcup_{\delta \in \Delta} ext(\delta)$.

- Proof of (i): It follows from Lemma 4.1 that

$$\sigma'' = \langle (\bigcup_{\delta \in \Delta} \delta.T \setminus S_{a,\Delta}) \cup Pre_a^+ \cup X, (\bigcup_{\delta \in \Delta} \delta.F \setminus S_{a,\Delta}) \cup Pre_a^- \cup Y \rangle$$

  where X and Y are two sets of fluents such that $\sigma''.T \cap \sigma''.F = \emptyset$.

  From the fact that $Pre_a^+ \subseteq \sigma''.T$ and $Pre_a^- \subseteq \sigma''.F$, we conclude that a is executable in σ''. [*]

- Proof of (ii): Consider an arbitrary $\sigma \in \Phi(a, \sigma'')$. We need to prove that there exists some $\delta \in \Delta$ such that $\sigma \in ext(\delta)$.

  Since a is applicable in Δ, by Definition 3.10, there exists Δ' such that $\Delta \sqsubseteq \Delta'$ and a is strongly applicable in Δ' and $S_{a,\Delta} = p(a, \Delta')$.

  Let $S_\sigma^+ = \sigma.T \cap S_{a,\Delta}$ and $S_\sigma^- = \sigma.F \cap S_{a,\Delta}$. By Lemma 4.3, we have that $S_\sigma^+ \cup S_\sigma^- = S_{a,\Delta}$ and $S_\sigma^+ \cap S_\sigma^- = \emptyset$.

  By Definition 3.4, there exists $\delta' \in \Delta'$ such that $\delta'.T \cap S_{a,\Delta} = S_\sigma^+$ and $\delta'.F \cap S_{a,\Delta} = S_\sigma^-$. Because $\Delta \sqsubseteq \Delta'$, there exists some $\delta \in \Delta$ such that δ' is a partial extension of δ.

  We will show that $\sigma \in ext(\delta)$, i.e., $\delta.T \subseteq \sigma.T$ and $\delta.F \subseteq \sigma.F$.

  Since $\delta.T \subseteq \delta'.T$, we have that $\delta.T \cap S_{a,\Delta} \subseteq \delta'.T \cap S_{a,\Delta} = S_\sigma^+$. Therefore:

$$\begin{aligned} \delta.T &= (\delta.T \setminus (\delta.T \cap S_{a,\Delta})) \cup (\delta.T \cap S_{a,\Delta}) \\ &= (\delta.T \setminus S_{a,\Delta}) \cup (\delta.T \cap S_{a,\Delta}) \\ &\subseteq (\delta.T \setminus S_{a,\Delta}) \cup S_\sigma^+. \end{aligned}$$

  Similarly, we can show that $\delta.F \subseteq (\delta.F \setminus S_{a,\Delta}) \cup S_\sigma^-$.

  Since $\sigma \in \Phi(a, \sigma'')$, by the definition of Φ, we have that $\sigma''.T \subseteq \sigma.T$. Let $\sigma.T \setminus \sigma''.T = \omega$, we have that

$$\sigma.T = \sigma''.T \cup \omega = ((\bigcup_{\delta \in \Delta} \delta.T) \setminus S_{a,\Delta}) \cup Pre_a^+ \cup X \cup \omega.$$

Since $\sigma.T \cap S_{a,\Delta} = S_\sigma^+$ and $(((\bigcup_{\delta \in \Delta} \delta.T) \setminus S_{a,\Delta}) \cup Pre_a^+) \cap S_{a,\Delta} = \emptyset$ (because $Sens_a \cap Pre_a^+ = \emptyset$), we must have that $(X \cup \omega) \cap S_{a,\Delta} = S_\sigma^+$, i.e. $S_\sigma^+ \subseteq X \cup \omega$. From the fact that $\delta.T \subseteq (\delta.T \setminus S_{a,\Delta}) \cup S_\sigma^+$ and $S_\sigma^+ \subseteq X \cup \omega$, it is easy to see that $\delta.T \subseteq \sigma.T$. Similarly, we can show that $\delta.F \subseteq \sigma.F$. From this fact, we conclude that $\sigma \in ext(\delta)$.   [**]

From [*] and [**] the theorem is proved.   □

To prove the final result about the correctness of $\mathcal{R}$ (and Theorem 4.27 in the next section) we need a number of additional notations and definitions.

Definition 4.5 (Branching Count). Let c be a conditional plan, we define the number of case plans of c, denoted by $count(c)$, inductively as follows:

(1) if $c = [\,]$ then $count(c) = 0$;
(2) if $c = a$, a is a non-sensing action, then $count(c) = 0$;
(3) if a is a non-sensing action and c is a conditional plan then $count(a; c) = count(c)$;
(4) if c is a case plan of the form a; $case(\varphi_1 \to c_1, \ldots, \varphi_n \to c_n)$ where a is a sensing action, then $count(c) = 1 + \sum_{i=1}^n count(c_i)$.

Lemma 4.6 (Sequence of Non-sensing Action). For p-states δ and δ', and a sequence of non-sensing actions $c = a_1; \ldots; a_n$ $(n \geq 0)$, $\mathcal{R}^*(c, \delta) = \delta' \neq \bot$ implies that $\Phi^*(c, \sigma'') \subseteq ext(\delta)$ for every $\sigma'' \in ext(\delta')$.

Proof. By induction on n.

• Base Case: $n = 1$. This means that c has only one action a. Using Theorem 4.2, and Definition 3.17 – item 2 – the base case is proved. Notice that for the case $n = 0$, i.e. $c = [\,]$, the lemma follows directly from Definitions 3.17 and 2.6.
• Inductive Step:

Assume that the lemma is shown for $1 \leq n \leq k$. We now prove the lemma for $n = k+1$. Let $c' = a_2; \ldots; a_{k+1}$ and $\mathcal{R}^*(c', \delta) = \delta^*$. By Definition 3.17

$$\mathcal{R}^*(c, \delta) = \mathcal{R}(a_1, \mathcal{R}^*(c', \delta)) = \delta'.$$

Since $\mathcal{R}(a_1, \delta^*) = \delta' \neq \bot$, we have that $\delta^* \neq \bot$.

Let $\sigma'' \in ext(\delta')$. By Theorem 4.2, we have that $\Phi(a_1, \sigma'') = \{\sigma\} \subseteq ext(\delta^*)$, i.e., $\sigma \in ext(\delta^*)$.

By the definition of Φ^*, we also have that $\Phi^*(c, \sigma'') = \Phi^*(c', \Phi^*(a_1, \sigma''))$. Using the induction hypothesis for c', where $\mathcal{R}^*(c', \delta) = \delta^*$ and $\sigma \in ext(\delta^*)$, we have:

$$\Phi^*(c', \Phi^*(a_1, \sigma'')) = \Phi^*(c', \sigma) \subseteq ext(\delta).$$

Therefore, $\Phi^*(c, \sigma'') \subseteq ext(\delta)$.   □

Lemma 4.7. Let δ be a p-state and c be a conditional plan. If $\mathcal{R}^*(c, \delta) = \delta'$ and $\delta' \neq \bot$, then for every $\sigma \in ext(\delta')$, $\Phi^*(c, \sigma) \subseteq ext(\delta)$.

Proof. By induction on $count(c)$, the number of case plans in c.

• Base Case: $count(c) = 0$. Then c is a sequence of non-sensing actions. The base case follows from Lemma 4.6.
• Inductive Step: Assume that we have proved the lemma for $count(c) \leq k$ $(k \geq 0)$. We need to prove the lemma for $count(c) = k + 1$. By the definition of a conditional plan, we have two cases:

(1) $c = a; p$ is a case plan where a is a sensing action and $p = case \ (\varphi_1 \rightarrow p_1, \ldots, \varphi_m \rightarrow p_m)$. Since $count(c) = 1 + \sum_{j=1}^{m} count(p_j) \leq k + 1$, we have that $count(p_i) \leq k$ for $i = 1, \ldots, m$. By Definition 3.17,

$$\perp \neq \delta' = \mathcal{R}^*(c, \delta) = \mathcal{R}(a, \{R(\varphi_1 \rightarrow p_1, \delta), \ldots, R(\varphi_m \rightarrow p_m, \delta)\}).$$

Let us denote $R(\varphi_i \rightarrow p_i, \delta)$ by δ_i $(1 \leq i \leq m)$ and $\Delta = \{\delta_1, \ldots, \delta_m\}$. We have that $\delta_i \models \varphi_i$ for $1 \leq i \leq m$, and a is applicable in Δ.

From Theorem 4.4, we have that

$$\Phi(a, \sigma) \subseteq \bigcup_{\delta'' \in \Delta} ext(\delta'')$$

for every $\sigma \in ext(\delta')$.

Consider an arbitrary $\sigma' \in \Phi(a, \sigma)$. Because of the above relation, we can conclude that there exists some i, $1 \leq i \leq m$, such that $\sigma' \in ext(\delta_i)$. Because $\mathcal{R}^*(p_i, \delta).T \subseteq R(\varphi_i \rightarrow p_i, \delta).T$ and $\mathcal{R}^*(p_i, \delta).F \subseteq R(\varphi_i \rightarrow p_i, \delta).F$, $\sigma' \in ext(\delta_i)$ implies $\sigma' \in ext(\mathcal{R}^*(p_i, \delta))$. Using inductive hypothesis for $count(p_i) \leq k$, we have that $\Phi^*(p_i, \sigma') \subseteq ext(\delta)$. Since this holds for every $\sigma' \in \Phi(a, \sigma)$, from Definition 2.6, we conclude that $\Phi^*(c, \sigma) \subseteq ext(\delta)$.

(2) $c = a; p$ where a is a non-sensing action and p is a conditional plan. Because $count(c) > 0$, from Definition 2.2 we conclude that there exists a sequence of non-sensing actions $b_1, \ldots, b_t$ and a case plan q such that $c = b_1; \ldots; b_t; q$. Let $c' = b_1; \ldots; b_t$ and $\mathcal{R}^*(q, \delta) = \delta^*$. Using the first case, we can show that for every $\sigma' \in ext(\delta^*)$, $\Phi^*(q, \sigma') \subseteq ext(\delta)$. Furthermore, because

$$\delta' = \mathcal{R}^*(c, \delta) = \mathcal{R}^*(c', \mathcal{R}^*(q, \delta))$$

and Lemma 4.6, we can show that for every $\sigma \in ext(\delta)$, $\Phi^*(c, \sigma) \subseteq ext(\delta)$.

From cases 1 and 2, the lemma is proved. □

We are now ready to prove the soundness of the extended regression function $\mathcal{R}^*$ with respect to the extended progression transition function Φ^*, which is illustrated in the next figure.
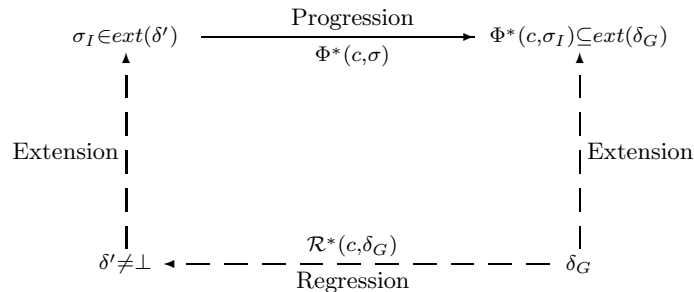

Figure 4: Soundness of $\mathcal{R}^*$.

Theorem 4.8 (Soundness of Regression)**.** Let $P = \langle A, O, I, G \rangle$ be a planning problem and c be a regression solution of P. Then, c is also a progression solution of P, i.e., $\Phi^*(c, \sigma_I) \subseteq ext(\delta_G)$.

Proof. Let $\delta' = \mathcal{R}^*(c, \delta_G)$. Since $\delta' \neq \perp$ and $\sigma_I \in ext(\delta')$ (Definition 3.18), the conclusion of the theorem follows immediately from Lemma 4.7. □

4.2. **Completeness Result.** We now proceed towards a completeness result. Ideally, one would like to have a completeness result that expresses that for a given planning problem, any progression solution can also be found by regression. In our formulation, however, the definition of the progression function allows an action a to execute in any a-state σ if a is executable in σ, regardless whether or not a would add "new" information to σ. In contrast, our definition of the regression function requires that an action a can only be applied in a p-state (or a set of p-states) if a contributes something to the applied p-state(s) [5]. Thus, given a planning problem $P = \langle A, O, I, G \rangle$, a progression solution c of P may contain redundant actions or extra branches. As a result, we may not obtain c via our regression, i.e. $\mathcal{R}^*(c, \delta_G) = \bot$. To illustrate this point, let us consider the following two examples.

Example 4.9 (Redundancy). Let $P = \langle \{f, g\}, \{b, c\}, \{f\}, \{g\} \rangle$ be a planning problem where c is a non-sensing action with $Pre_c = \{f\}$, $Add_c = \{g\}$, and $Del_c = \emptyset$; b is also a non-sensing action with $Pre_b = \{g\}$, $Add_b = \{f\}$, and $Del_b = \emptyset$. Clearly

$$p_1 = c \quad p_2 = c; b \quad p_3 = c; c$$

are three progression solutions of P. Plan p_1 indicates that b (in p_2) and a copy (a.k.a. an instance) of c (in p_3) are redundant.

It is easy to check that

$$\mathcal{R}^*(p_2, [\{g\}, \emptyset]) = \mathcal{R}^*(p_3, [\{g\}, \emptyset]) = \bot$$

whereas

$$\mathcal{R}^*(c, [\{g\}, \emptyset]) = [\{f\}, \emptyset].$$

Example 4.10 (Redundancy). Let $P = \langle \{f, g\}, \{b, c\}, \{f\}, \{g\} \rangle$ be a planning problem. Let c be a sensing action where $Pre_c = \emptyset$, $Sens_c = \{f, g\}$; b is a non-sensing action where $Pre_b = \{f, \neg g\}$, $Add_b = \{g\}$, and $Del_b = \emptyset$. A plan achieving g is:

$$p = c; case(f \wedge \neg g \rightarrow b, f \wedge g \rightarrow [\ ], \neg f \wedge \neg g \rightarrow [\ ], \neg f \wedge g \rightarrow [\ ]).$$

Notice that, the conditions $\neg f \wedge \neg g$ and $\neg f \wedge g$ are always evaluated to false after the execution of c because f is true before the execution of c. Thus, the two last branches of p are never used to achieve g.

We have that

$$\mathcal{R}^*([], [\{g\}, \emptyset]) = [\{g\}, \emptyset]$$
$$\mathcal{R}^*(b, [\{g\}, \emptyset]) = \mathcal{R}(b, [\{g\}, \emptyset]) = [\{f\}, \{g\}]$$

We can also verify that

$$R(f \wedge \neg g \rightarrow b, [\{g\}, \emptyset]) = [\{f\}, \{g\}] \qquad R(f \wedge g \rightarrow [], [\{g\}, \emptyset]) = [\{f, g\}, \emptyset]$$
$$R(\neg f \wedge \neg g \rightarrow [], [\{g\}, \emptyset]) = \bot \qquad R(\neg f \wedge g \rightarrow [], [\{g\}, \emptyset]) = [\{g\}, \{f\}]$$

This implies that

$$\mathcal{R}^*(p, [\{g\}, \emptyset]) = \mathcal{R}(c, \{[\{f\}, \{g\}], [\{f, g\}, \emptyset], \bot\}) = \bot.$$

Let p' be the conditional plan obtained from p by removing the last two branches of p, i.e.,

$$p' = c; case(f \wedge \neg g \rightarrow b, f \wedge g \rightarrow [\ ]).$$

We can easily check that $\mathcal{R}^*(p', [\{g\}, \emptyset]) = [\{f\}, \emptyset] \neq \bot$.

---

[5]Note that this condition is also applied for regression planning systems such as [4] and [14].

The above discussion suggests us the following completeness result: if a conditional plan can be found through progression we can find an equivalent conditional plan through regression. The plan found through regression does not have redundancies, both in terms of extra actions and extra branches. We refer to these notions as "redundancy" and "plan equivalence". We now formalize these notions.

Definition 4.11 (Subplan). Let c be a conditional plan. A conditional plan c' is a *subplan* of c if

- c' can be obtained from c by
    - (i) removing an instance of a non-sensing action from c; or
    - (ii) removing a case plan or a branch $\varphi_i \to c_i$ from a case plan in c; or
    - (iii) replacing a case plan $a; case(\varphi_1 \to p_1; c_n \ldots, \varphi_m \to p_m)$ in c with one of its branches p_i for some i, $1 \le i \le m$; or
- c' is a subplan of c'' where c'' is a subplan of c.

The above definition allows us to define redundant plans as follows.

Definition 4.12 (Redundancy). Let c be a conditional plan, σ be an a-state, and δ be a p-state. We say that c *contains redundancy* (or is *redundant*) with respect to (σ, δ) if

- (i) $\Phi^*(c, \sigma) \subseteq ext(\delta)$; and
- (ii) there exists a subplan c' of c with respect to σ such that $\Phi^*(c', \sigma) \subseteq ext(\delta)$.

Note that, if c' is a subplan of a conditional plan c then $c' \ne c$. The equivalence of two conditional plans is defined formally as follows.

Definition 4.13 (Equivalent Plan). Let σ be an a-state, δ be a p-state, and c be a conditional plan such that and $\Phi^*(c, \sigma) \subseteq ext(\delta)$. A conditional plan c' is equivalent to c with respect to (σ, δ) if $\Phi^*(c', \sigma) \subseteq ext(\delta)$.

Example 4.14 (Equivalence). Consider the plans in Example 4.9, we have that p_1 is a subplan of p_3 which is equivalent to p_3 with respect to $(\langle\{f\}, \emptyset\rangle, [\{g\}, \emptyset])$.

Similarly, for planning problem in Example 4.10, p' is a subplan of p and is equivalent to p with respect to $(\langle\{f\}, \emptyset\rangle, [\{g\}, \emptyset])$.

It is easy to see that if c' and c'' are equivalent to c with respect to (σ, δ) then c' and c'' are equivalent with respect to (σ, δ). To prove the completeness result of our regression formulation, we will need to introduce a few more definitions and notations. Recall that our purpose is to use regression to find an equivalent conditional plan for a given progression solution. To do that, we will provide conditions characterizing when a conditional plan is regressable, i.e. when the $\mathcal{R}^*$ function can be applied on it to produce a p-state. We refer to conditional plans satisfying such conditions as *regressable conditional plans*. We will later show that, for a given progression solution of a planning problem P there exists an equivalent, regressable conditional plan that is also a regression solution of P.

To define a regressable conditional plan, we begin with some additional notations. 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 $BIN(S)$ denote 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 both 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 some set of fluents S if there exists a consistent conjunction of literals $\varphi \notin \chi$, such that:

(1) $S \cap (\varphi^+ \cup \varphi^-) = \emptyset$ where φ^+ and φ^- denote the sets of fluents occurring positive and negative in φ, respectively;

(2) $\varphi_i = \varphi \wedge \psi_i$ where $BIN(S) = \{\psi_1, \ldots, \psi_n\}$.

Notice that for a non-empty set S, we can easily check whether the set $\chi = \{\varphi_1, \ldots, \varphi_n\}$ spans over S. We say that a set $\chi = \{\varphi_1, \ldots, \varphi_n\}$ is factorable if it spans over some non-empty set of fluents S.

Example 4.15 (Getting to Evanston – Cond't). Consider a set $S = \{traffic\text{-}bad\}$, a conjunction $\varphi = on\text{-}ashland$ and a set of conjunctions $\chi = \{on\text{-}ashland \wedge traffic\text{-}bad, on\text{-}ashland \wedge \neg traffic\text{-}bad\}$.

We have that $BIN(S) = \{traffic\text{-}bad, \neg traffic\text{-}bad\}$ and χ spans over S.

We can show that for a non-empty set of consistent conjunctions of literals $\chi = \{\varphi_1, \ldots, \varphi_n\}$ be a non-empty set if χ is factorable, then there exists a unique non-empty set of fluents S such that χ spans over S. This allows us to define the notion of regressable plans as follows.

Definition 4.16 (Potentially Regressable Case Plan). A case plan

$$p = a; case(\varphi_1 \to c_1, \ldots, \varphi_n \to c_n)$$

is potentially regressable if

(i) there exists a non-empty set $\emptyset \neq S_a \subseteq Sens_a$ such that $\{\varphi_1, \ldots, \varphi_n\}$ spans over S_a, and

(ii) for $1 \leq i \leq n$, $Sens_a \subseteq (\varphi_i^+ \cup \varphi_i^-)$.

Definition 4.17 (Regressable Conditional Plan). Let c be a conditional plan, σ be an a-state, and δ be a p-state. We say c is regressable with respect to (σ, δ) if

(i) every case plan occurring in c is potentially regressable,

(ii) $\Phi^*(c, \sigma) \subseteq ext(\delta)$, and

(iii) c is not redundant with respect to (σ, δ).

We will now prove a series of lemmae that will be used in the proof of the completeness of $\mathcal{R}^*$. Lemma 4.18 is about the uniqueness of a set of literals over which a factorable set of conjunctions spans. Lemmae 4.19-4.20 state that the regressable property of a sequence of non-sensing actions is maintained by the function $\mathcal{R}^*$. Lemma 4.21-4.23 extend this result to regressable conditional plans. Lemmae 4.24-4.26 show that for each progression solution there exists an equivalent regressable plan which can be found through regression.

Lemma 4.18. 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.

Proof. Since χ is factorable, there exists a non-empty set of fluents S such that χ spans over S, i.e. there exists φ such that $\varphi_i = \varphi \wedge \psi_i$ where $\psi_i \in BIN(S)$ for $i = 1, \ldots, n$ and $BIN(S) = \{\psi_1, \ldots, \psi_n\}$. Assume that S is not unique. This means that there exists a non-empty set $S' \neq S$ such that χ spans over S', i.e. there exists φ' such that $\varphi_i = \varphi' \wedge \psi_i'$ where $\psi_i' \in BIN(S')$ for $i = 1, \ldots, n$.

Consider $f \in S \setminus S'$. For every $1 \leq i \leq n$, we have that $\varphi_i = \varphi' \wedge \psi_i'$. Since $f \notin S'$ and φ_i is consistent $(1 \leq i \leq n)$, f must occur either positively or negatively in φ'. This means that f occurs either positively or negatively in all φ_i for $1 \leq i \leq n$.

Consider the case that f occurs positively in all φ_i for $1 \leq i \leq n$ [*]. Since $f \in S$, there exists a binary representation $\psi_j \in BIN(S)$ $(1 \leq j \leq n)$ such that f appears negatively in ψ_j i.e. f appears negatively in φ_j. This contradicts with [*]. Similarly we can show a contradiction in the case that f occurs negatively in all φ_i for $1 \leq i \leq n$. We conclude that S is unique. $\qquad\square$

Lemma 4.19. Let σ be an a-state, δ be a p-state, and $c = a_1; \ldots; a_n$ $(n \geq 1)$ be a sequence of non-sensing actions. Assume that c is regressable with respect to (σ, δ). Then, $\mathcal{R}^*(a_n, \delta) = \delta'$, $\delta' \neq \perp$, and $c' = a_1; \ldots; a_{n-1}$ is regressable with respect to (σ, δ').

Proof. By induction on n.

- Base Case: $n = 1$. Similar to the inductive step, we can show that a_1 is applicable in δ. Let $\delta' = \mathcal{R}(a_1, \delta)$ and $\Phi(a_1, \sigma) = \{\sigma'\}$. We have that, $\delta'.T = (\delta.T \setminus Add_{a_1}) \cup Pre_{a_1}^+$ and $\sigma'.T = (\sigma.T \setminus Del_{a_1}) \cup Add_{a_1}$. Using the facts $\sigma' \in ext(\delta)$, $Add_{a_1} \cap Del_{a_1} = \emptyset$, and the above equations, we can show that $\delta'.T \subseteq \sigma.T$. Similarly, $\delta'.F \subseteq \sigma.F$. Since $[\,]$ is not redundant with respect to (σ, δ'), we have that $[\,]$ is a plan that is regressable with respect to (σ, δ').

- Inductive Step: Assume that we have proved the lemma for $0 < n \leq k$. We need to prove the lemma for $n = k + 1$.

  Let $\Phi^*(a_1; \ldots; a_k, \sigma) = \{\sigma_k\}$, we have that

  $$\Phi^*(c, \sigma) = \Phi(a_{k+1}, \sigma_k) = \{\sigma'\} \subseteq ext(\delta).$$

  We will prove that (1) a_{k+1} is applicable in δ, (2) $\mathcal{R}(a_{k+1}, \delta) = \delta^* \neq \perp$ and $\sigma_k \in ext(\delta^*)$, and (3) $c' = a_1; \ldots; a_k$ is regressable with respect to (σ, δ^*).

  - Proof of (1): We first show that $Add_{a_{k+1}} \cap \delta.T \neq \emptyset$ or $Del_{a_{k+1}} \cap \delta.F \neq \emptyset$. Assume the contrary, $Add_{a_{k+1}} \cap \delta.T = \emptyset$ and $Del_{a_{k+1}} \cap \delta.F = \emptyset$. By Definition 2.4, we have that

  $$\sigma'.T = (\sigma_k.T \setminus Del_{a_{k+1}}) \cup Add_{a_{k+1}}$$

  and

  $$\sigma'.F = (\sigma_k.F \setminus Add_{a_{k+1}}) \cup Del_{a_{k+1}}.$$

  Since $\sigma' \in ext(\delta)$, we have $\delta.T \subseteq \sigma'.T$. By our assumption, $Add_{a_{k+1}} \cap \delta.T = \emptyset$, we must have that $\delta.T = \delta.T \setminus Add_{a_{k+1}} \subseteq \sigma'.T \setminus Add_{a_{k+1}}$. Because for arbitrary sets X, Y, $(X \cup Y) \setminus Y = X \setminus (X \cap Y)$, we have that

  $$\sigma'.T \setminus Add_{a_{k+1}} = ((\sigma_k.T \setminus Del_{a_{k+1}}) \cup Add_{a_{k+1}}) \setminus Add_{a_{k+1}} =$$

  $$(\sigma_k.T \setminus Del_{a_{k+1}}) \setminus ((\sigma_k.T \setminus Del_{a_{k+1}}) \cap Add_{a_{k+1}}) \subseteq \sigma_k.T,$$

  i.e. $\delta.T \subseteq \sigma_k.T \setminus Del_{a_{k+1}}$. This shows that $\delta.T \subseteq \sigma_k.T$. Similarly, we can show that $\delta.F \subseteq \sigma_k.F$. We conclude that $\sigma_k \in ext(\delta)$, i.e. c is redundant with respect to (σ, δ). This is a contradiction. Therefore, $Add_{a_{k+1}} \cap \delta.T \neq \emptyset$ or $Del_{a_{k+1}} \cap \delta.F \neq \emptyset$. $\qquad$ (i)
  Since $\sigma' \in ext(\delta)$, we have $\delta.T \subseteq \sigma'.T$ and $\delta.F \subseteq \sigma'.F$. As a_{k+1} is executable in σ_k, we have $Add_{a_{k+1}} \cap \sigma'.F = \emptyset$ and $Del_{a_{k+1}} \cap \sigma'.T = \emptyset$. This concludes that $Add_{a_{k+1}} \cap \delta.F = \emptyset$ and $Del_{a_{k+1}} \cap \delta.T = \emptyset$. $\qquad$ (ii)
  Now, assume that there exists $f \in Pre_{a_{k+1}}^+ \cap \delta.F$ and $f \notin Del_{a_{k+1}}$. By Definition 2.4, it's easy to see that $f \in \sigma'.T$ and $f \in \sigma'.F$. This is a contradiction, therefore $Pre_{a_{k+1}}^+ \cap \delta.F \subseteq Del_{a_{k+1}}$. Similarly, we can show that $Pre_{a_{k+1}}^- \cap \delta.T \subseteq Add_{a_{k+1}}$. $\quad$ (iii).
  From (i), (ii), and (iii) we conclude that a_{k+1} is applicable in δ.

– Proof of (2): Because a_{k+1} is applicable in δ, we have that $\mathcal{R}(a_{k+1}, \delta) = \delta^*$ for some partial state $\delta^* \neq \perp$. We will show that $\sigma_k \in ext(\delta^*)$. From the fact $\sigma' \in ext(\delta)$, by Definition 2.4, we have

$$\delta.T \subseteq \sigma'.T = (\sigma_k.T \setminus Del_{a_{k+1}}) \cup Add_{a_{k+1}}$$

and

$$\delta.F \subseteq \sigma'.F = (\sigma_k.F \setminus Add_{a_{k+1}}) \cup Del_{a_{k+1}}.$$

By Definition 3.2 we have

$$\delta^*.T = (\delta.T \setminus Add_{a_{k+1}}) \cup Pre^+_{a_{k+1}}$$

and

$$\delta^*.F = (\delta.F \setminus Del_{a_{k+1}}) \cup Pre^-_{a_{k+1}}.$$

Since a_{k+1} is executable in σ_k, we have that $Pre^+_{a_{k+1}} \subseteq \sigma_k.T$ and $Pre^-_{a_{k+1}} \subseteq \sigma_k.F$. Therefore, to prove that $\delta^*.T = (\delta.T \setminus Add_{a_{k+1}}) \cup Pre^+_{a_{k+1}} \subseteq \sigma_k.T$, we only need to show that $\delta.T \setminus Add_{a_{k+1}} \subseteq \sigma_k.T$. As $\delta.T \subseteq (\sigma_k.T \setminus Del_{a_{k+1}}) \cup Add_{a_{k+1}}$, we have

$$\delta.T \setminus Add_{a_{k+1}} \subseteq ((\sigma_k.T \setminus Del_{a_{k+1}}) \cup Add_{a_{k+1}}) \setminus Add_{a_{k+1}}.$$

From the proof of item (1), we have that $((\sigma_k.T \setminus Del_{a_{k+1}}) \cup Add_{a_{k+1}}) \setminus Add_{a_{k+1}} \subseteq \sigma_k.T$. This concludes that $\delta.T \setminus Add_{a_{k+1}} \subseteq \sigma_k.T$. Similarly, we can show that $\delta.F \setminus Del_{a_{k+1}} \subseteq \sigma_k.F$, i.e., $\sigma_k \in ext(\delta^*)$ or $\{\sigma_k\} \subseteq ext(\delta^*)$.

– Proof of (3): Suppose that c' is redundant with respect to (σ, δ^*). By Definition 4.12, there exists a subplan c'' of c such that $\Phi^*(c'', \sigma) = \{\sigma''\} \subseteq ext(\delta^*)$. By Theorem 4.2, we have that $\Phi(a_{k+1}, \sigma'') \subseteq ext(\delta)$. Since

$$\Phi^*(c''; a_{k+1}, \sigma) = \Phi(a_{k+1}, \sigma'') \subseteq ext(\delta),$$

we have that c is redundant with respect to (σ, δ). This contradicts with the assumption that c is not redundant with respect to (σ, δ). Since c' has no case plan, this concludes that c' is not redundant with respect to (σ, δ^*). Since $\Phi^*(a_1; \ldots; a_k, \sigma) = \{\sigma_k\} \subseteq ext(\delta^*)$ we have that c' is regressable with respect to (σ, δ^*). $\square$

Lemma 4.20. Let σ be an a-state and δ be a p-state. Let $c = a_1; \ldots; a_n$ be a sequence of non-sensing actions that is regressable with respect to (σ, δ). Then, there exists some p-state $\delta^* \neq \perp$ such that $\mathcal{R}^*(c, \delta) = \delta^*$ and $\sigma \in ext(\delta^*)$.

Proof. By induction on n.

• Base Case: $n = 0$. Then c is an empty sequence of non-sensing actions. The base case follows from Definition 3.17 (with $\delta^* = \delta$ and $[\,]$ is not redundant with respect to (σ, δ)).

• Inductive Step: Assume that we have proved the lemma for $0 \leq n \leq k$. We need to prove the lemma for $n = k + 1$. It follows from Lemma 4.19 that $\delta' = \mathcal{R}(a_{k+1}, \delta)$, $\delta' \neq \perp$, and $c' = a_1; \ldots; a_k$ is a plan that is regressable with respect to (σ, δ'). By inductive hypothesis, we have that $\mathcal{R}^*(c', \delta') = \delta^* \neq \perp$ and $\sigma \in ext(\delta^*)$. The inductive step follows from this and the fact $\mathcal{R}^*(c, \delta) = \mathcal{R}^*(c', \mathcal{R}(a_{k+1}, \delta))$. $\square$

Lemma 4.21. Let σ be an a-state, a be a sensing action which is executable in σ. Let $S_a = Sens_a \setminus \sigma$. Then, we have that

(1) $\Phi(a, \sigma) = \{\sigma_1, \ldots, \sigma_m\}$ where $m = 2^{|S_a|}$,

(2) a is strongly applicable in $\Delta = \{\delta_1, \ldots, \delta_m\}$ where $\delta_i = [\sigma_i.T, \sigma_i.F]$, $i = 1, \ldots, m$, and

(3) $\mathcal{R}(a, \Delta) = [\sigma.T, \sigma.F]$.

Proof.

(1) From Definition 2.4, we have that

$$\perp \notin \Phi(a, \sigma) = \{\sigma' | Sens_a \setminus \sigma = \sigma' \setminus \sigma\}$$

and, for every $\sigma' \in \Phi(a, \sigma)$, $\sigma' \setminus \sigma = (\sigma'.T \setminus \sigma.T) \cup (\sigma'.F \setminus \sigma.F)$. Denote $\sigma'.T \setminus \sigma.T$ by P and $\sigma'.F \setminus \sigma.F$ by Q, we have that (P, Q) is a partition of S_a. Since there are $2^{|S_a|}$ partitions of S_a, we have that $m \leq 2^{|S_a|}$. Furthermore, for a partition (P, Q) of S_a there exists an a-state $\sigma' = \langle P \cup \sigma.T, Q \cup \sigma.F \rangle \in \Phi(a, \sigma)$ because $\sigma' \setminus \sigma = P \cup Q$. Therefore $2^{|S_a|} \leq m$. We conclude that $m = 2^{|S_a|}$.

(2) We first show that Δ is proper with respect to S_a, i.e. S_a is a sensed set of Δ with respect to a. Indeed, by Definition 2.4 and the proof of (1), we have that the first three conditions of Definition 3.6 are satisfied. The fourth condition of Definition 3.6 is satisfied because we have that $\delta_i.T \setminus S_a = \sigma_i.T \setminus S_a = \sigma.T$ and $\delta_i.F \setminus S_a = \sigma_i.F \setminus S_a = \sigma.F$ $(1 \leq i \leq m)$. Therefore, we conclude that $p(a, \Delta) = S_a$.

   Since a is an action that is executable in σ we have that $(Pre_a^+ \cup Pre_a^-) \cap Sens_a = \emptyset$ and $Pre_a^+ \cap \sigma.F = \emptyset$, $Pre_a^- \cap \sigma.T = \emptyset$, therefore $Pre_a^+ \cap \delta_i.F = \emptyset$, $Pre_a^- \cap \delta_i.T = \emptyset$ $(1 \leq i \leq m)$. By Definition 3.8, we conclude that a is strongly applicable in Δ.

(3) Since a is executable in σ, we have that $Pre_a^+ \subseteq \sigma.T$ and $Pre_a^- \subseteq \sigma.F$. From the proof of (2), $\delta_i.T \setminus S_a = \sigma.T$ and $\delta_i.F \setminus S_a = \sigma.F$ $(1 \leq i \leq m)$. The proof follows from Definition 3.15. $\qquad\square$

Lemma 4.22. Let σ be an a-state, δ be a p-state, and $c = \alpha; c'$ is a conditional plan where α is a non-empty sequence of non-sensing actions and $c' = a; case(\varphi_1 \to p_1, \ldots, \varphi_m \to p_m)$. If c is regressable with respect to (σ, δ), then

(1) there exists some a-state $\sigma_1 \neq \perp$ such that $\Phi^*(\alpha, \sigma) = \{\sigma_1\}$;

(2) $m = 2^{|S_a|}$ where $S_a = Sens_a \setminus \sigma_1$;

(3) $\{\varphi_1, \ldots, \varphi_m\}$ spans over S_a;

(4) For each i, $1 \leq i \leq m$, there exists a unique a-state $\sigma' \in \Phi(a, \sigma_1)$ such that p_i is regressable with respect to (σ', δ).

Proof.

(1) By Definition 2.6, we have that

$$\Phi^*(c, \sigma) = \bigcup_{\sigma' \in \Phi^*(\alpha, \sigma)} \Phi^*(c', \sigma').$$

   Since c is regressable with respect to (σ, δ) we have that $\perp \notin \Phi^*(c, \sigma)$. This implies that $\perp \notin \Phi^*(\alpha, \sigma)$. Furthermore, because α is a sequence of non-sensing actions, we conclude that there exists some a-state $\sigma_1 \neq \perp$. such that $\Phi^*(\alpha, \sigma) = \{\sigma_1\}$.

(2) By definition of S_a we conclude that S_a is the set of fluents that belong to $Sens_a$ which are unknown in σ_1. By Definition 2.4, we conclude that $\Phi(a, \sigma_1)$ consists of $2^{|S_a|}$

elements where for each $\sigma' \in \Phi(a, \sigma_1)$, $\sigma' \setminus \sigma_1 = S_a$. Because

$$\perp \notin \Phi^*(c, \sigma) = \bigcup_{\sigma' \in \Phi(a, \sigma_1)} E(case(\varphi_1 \to p_1, \ldots, \varphi_m \to p_m), \sigma')$$

we conclude that for each $\sigma' \in \Phi(a, \sigma_1)$ there exists one j, $1 \leq j \leq m$, such that φ_j is satisfied in σ'. Since φ's are mutual exclusive we conclude that for each j, $1 \leq j \leq m$, there exists at most one $\sigma' \in \Phi(a, \sigma_1)$ such that φ_j is satisfied in σ'. This implies that $m \geq 2^{|S_a|}$. The non-redundancy property of c implies that $m \geq 2^{|S_a|}$. Thus, $m = 2^{|S_a|}$.

(3) Since c is regressable with respect to (σ, δ) we have that $a; (case(\varphi_1 \to p_1, \ldots, \varphi_m \to p_m)$ is potentially regressable. This implies that $\{\varphi_1, \ldots, \varphi_m\}$ spans over a set of fluents $S \subseteq Sens_a$ and there exists a φ such that for every i, $\varphi_i = \psi_i \wedge \varphi$ where $\psi_i \in BIN(S)$ and $S \cap (\varphi^+ \cup \varphi^-) = \emptyset$. From Lemma 4.18 we know that S is unique. We will show now that $S = S_a$. Assume the contrary, $S \neq S_a$. We consider two cases:

- $S \setminus S_a \neq \emptyset$. Consider a fluent $f \in S \setminus S_a$. Because $\{\varphi_1, \ldots, \varphi_m\}$ spans over S, there exists some i such that f occurs positively in φ_i. From the proof of the previous item and the fact that $f \notin S_a$, we conclude that f must be true in σ_1 (otherwise, we have that the subplan c' of c, obtained by removing the branch $\varphi_i \to p_i$, satisfies $\perp \notin \Phi^*(c', \sigma) \subseteq ext(\delta)$, which implies that c is redundant with respect to (σ, δ)). Similarly, there exists some j such that f occurs negatively in φ_j, and hence, f must be false in σ_1. This is a contradiction. Thus, this case cannot happen.

- $S_a \setminus S \neq \emptyset$. Consider a fluent $f \in S_a \setminus S$. Again, from the fact that c is regressable with respect to (σ, δ), we conclude that f occurs either positively or negatively in φ_i. Because $f \notin S$, we have that f occurs in φ, and hence, f occurs positively or negatively in all φ_i. In other words, f is true or false in every $\sigma' \in \Phi(a, \sigma_1)$. Thus, f is true or false in σ_1. This contradicts the fact that $f \in S_a = Sens_a \setminus \sigma_1$. Thus, this case cannot happen too.

The above two cases imply that $S_a = S$. This means that $\{\varphi_1, \ldots, \varphi_m\}$ spans over S_a.

(4) Consider an arbitrary i, $1 \leq i \leq m$. From the proof of the second item, we know that there exists a unique $\sigma' \in \Phi(a, \sigma_1)$ such that φ_i is satisfied by σ'. We will show now that p_i is regressable with respect to (σ', δ). From the fact that c is regressable, we conclude that every case plan in p_i is potentially regressable. Furthermore, because $\Phi^*(p_i, \sigma') \subseteq \Phi^*(c, \sigma)$, we have that $\perp \notin \Phi^*(p_i, \sigma') \subseteq ext(\delta)$. Thus, to complete the proof, we need to show that p_i is not redundant with respect to (σ', δ). Assume the contrary, there exists a subplan p' of p_i such that $\perp \notin \Phi^*(p', \sigma') \subseteq ext(\delta)$. This implies that the subplan c' of c, obtained by replacing p_i with p', will satisfy that $\perp \notin \Phi^*(c', \sigma) \subseteq ext(\delta)$, i.e., c is redundant with respect to (σ, δ). This contradicts the condition of the lemma, i.e., our assumption is incorrect. Thus, p_i is not redundant with respect to (σ', δ), and hence, p_i is regressable with respect to (σ', δ). $\square$

Lemma 4.23. Let σ be an a-state, δ be a p-state, and c is a conditional plan that is regressable with respect to (σ, δ). Then, there exists some p-state $\delta' \neq \perp$ such that $\mathcal{R}^*(c, \delta) = \delta'$ and $\sigma \in ext(\delta')$.

Proof. By induction on $count(c)$, the number of case plans in c.

- Base Case: $count(c) = 0$. Then c is a sequence of non-sensing actions. The base case follows from Lemma 4.20.
- Inductive Step: Assume that we have proved the lemma for $count(c) \leq k$. We need to prove the lemma for $count(c) = k + 1$. Since c is a conditional plan, we have that

$c = \alpha; c'$ where α is a sequence of non-sensing actions and $c' = a; p$ and $p = case \ (\varphi_1 \to p_1, \ldots, \varphi_m \to p_m)$. Because α is a sequence of non-sensing actions we have that $\Phi^*(\alpha, \sigma)$ is a singleton. Let $\Phi^*(\alpha, \sigma) = \{\sigma_1\}$.

Let $S_a = Sens_a \setminus \sigma_1$. Since c is not redundant with respect to (σ, δ) we conclude that $S_a \neq \emptyset$.

It follows from the fact that c is regressable with respect to (σ, δ) and Lemma 4.22 that $\{\varphi_1, \ldots, \varphi_m\}$ spans over S_a and for every i, $1 \leq i \leq m$, there exists a unique $\sigma' \in \Phi(a, \sigma_1)$ such that p_i is regressable with respect to (σ', δ). By inductive hypothesis for p_i, we conclude that $\mathcal{R}^*(p_i, \delta) = \delta_i \neq \bot$ and $\sigma' \in ext(\delta_i)$. Because φ_i is satisfied by σ' we have that $R(\varphi_i \to p_i, \delta) = [\delta_i.T \cup \varphi_i^+, \delta_i.F \cup \varphi_i^-]$ is consistent and hence $R(\varphi_i \to p_i, \delta) \neq \bot$. This also implies that $\sigma' \in ext(R(\varphi_i \to p_i, \delta))$ and $R(\varphi_i \to p_i, \delta) \neq R(\varphi_j \to p_j, \delta)$ for $i \neq j$.

Let $\Delta = \{R(\varphi_i \to p_i, \delta) \mid i = 1, \ldots, m\}$. We will show next that a is applicable in Δ. Consider $\Delta' = \Phi(a, \sigma_1)$, we have that for each i, $1 \leq i \leq m$, there exists one $\sigma' \in \Delta'$ and $\sigma' \in ext(R(\varphi_i \to p_i, \delta))$. It follows from Lemma 4.21 that a is strongly applicable in Δ'. Thus, a is applicable in Δ.

By definition of $\mathcal{R}$, we have that

$$\mathcal{R}(a, \Delta) = [((\bigcup_{i=1}^m R(\varphi_i \to p_i, \delta).T) \setminus S_a) \cup Pre_a^+,$$
$$((\bigcup_{i=1}^m R(\varphi_i \to p_i, \delta).F) \setminus S_a) \cup Pre_a^-] = \delta^* \neq \bot.$$

Since a is executable in σ_1, from Lemma 4.21, and the fact that for each $\sigma' \in \Phi(a, \sigma_1)$ there exists an i such that $\sigma' \in ext(R(\varphi_i \to p_i, \delta))$, we can conclude $\sigma_1 \in ext(\delta^*)$.

To continue our proof, we will now show that $q = \alpha$ is not redundant with respect to (σ, δ^*). Assume the contrary, there exists a subplan q' of q such that $\Phi^*(q', \sigma) \subseteq ext(\delta^*)$. This, together with the fact that $\mathcal{R}^*(c', \delta) = \delta^*$ and Theorem 4.8 implies that $\Phi^*(c'', \sigma) \subseteq ext(\delta)$ for $c'' = q'; c'$, i.e., c is redundant with respect to (σ, δ). This contradicts the assumption of the lemma, i.e., we have proved that q is not redundant with respect to (σ, δ^*).

Applying the inductive hypothesis for the plan q and (σ, δ^*), we have that $\mathcal{R}^*(q, \delta^*) = \delta' \neq \bot$ and $\sigma \in ext(\delta')$. The inductive hypothesis is proved because $\mathcal{R}^*(c, \delta) = \mathcal{R}^*(q, \delta^*)$. $\square$

Lemma 4.24. Let σ be an a-state, δ be a p-state, and c be a sequence of non-sensing actions such that $\Phi^*(c, \sigma) \subseteq ext(\delta)$. Then, there exists a subplan c' of c that is not redundant with respect to (σ, δ) and c' is equivalent to c with respect to (σ, δ).

Proof. Notice that the length of c is finite[6]. Consider two cases:
- Case (i): c is not redundant with respect to (σ, δ).
  It's easy to see that $c' = c$ satisfies the condition of the lemma.
- Case (ii): c is redundant with respect to (σ, δ).
  By definition of redundancy, there exists a subplan of c which are equivalent to c with respect to (σ, δ). Let c' be a subplan of c which is equivalent to c with respect to (σ, δ) whose length is minimal among all subplans which is equivalent to c with respect to (σ, δ). To prove the lemma, it suffices to show that c' is not redundant with respect to (σ, δ). Assume the contrary, there exists a subplan c'' of c' which is equivalent to c with respect

---

[6]By this we mean that c is given and hence its length (the number of actions in c) is finite.

to (σ, δ). Trivially, the number of actions in c'' is smaller than the number of actions in c'. By definition, we have that c'' is also a subplan of c which is equivalent to c with respect to (σ, δ). This contradicts the fact that c' has the minimal length among all subplans of c which are equivalent to c. So, we conclude that c' is not redundant with respect to (σ, δ). The lemma is proved. $\qquad\square$

Lemma 4.25. Let σ be an a-state and $c = a; case \ (\varphi_1 \to p_1, \ldots, \varphi_m \to p_m)$ be a case plan such that $\bot \notin \Phi^*(c, \sigma)$. Then, if $Sens_a \setminus \sigma \neq \emptyset$, there exists a potentially regressable plan $c' = a; case \ (\varphi_1' \to p_1', \ldots, \varphi_n' \to p_n')$ such that $\Phi^*(c, \sigma) = \Phi^*(c', \sigma)$.

Proof. We prove the lemma by constructing c'. Let $S = \{\varphi_1, \ldots, \varphi_m\}$ and $S_a = Sens_a \setminus \sigma$. Let $L = \{f \mid f \in S_a\} \cup \{\neg f \mid f \in S_a\}$. First, observe that because of $\bot \notin \Phi^*(c, \sigma)$ we have that a is executable in σ. Furthermore, for each $\sigma' \in \Phi(a, \sigma)$ there exists one $\varphi_i \in S$ such that φ_i is satisfied in σ'. Without loss of generality, we can assume that for each $\varphi_i \in S$, there exists (at least) one $\sigma' \in \Phi(a, \sigma)$ such that φ_i is satisfied in σ'.

It is easy to see that for each i, we can write $\varphi_i = \psi_i \wedge \chi_i$ where ψ_i is the conjunction of literals occurring in φ_i and belonging to L and χ_i is the conjunction of literals that do not belong to L. From the above observation, we have that χ_i is satisfied by σ. So, $\varphi = \wedge_{i=1}^m \chi_i$ holds in σ. Thus, the conditional plan $c_1 = a; case \ (\varphi_1' \to p_1, \ldots, \varphi_m' \to p_m)$ where $\varphi_i' = \psi_i \wedge \varphi$ satisfies that $\Phi^*(c, \sigma) = \Phi^*(c_1, \sigma)$.

Since ψ_i is a consistent conjunction of literals from L and ψ_i's are mutual exclusive, there exists a partition $(S_1, \ldots, S_m)$ of $BIN(S_a)$ such that for every $\eta \in S_i$, $\eta = \psi_i \wedge \eta'$. Let

$$c_2 = a; case($$
$$\gamma_1^1 \to p_1, \ldots, \gamma_1^{|S_1|} \to p_1,$$
$$\gamma_2^1 \to p_1, \ldots, \gamma_2^{|S_2|} \to p_2,$$
$$\ldots$$
$$\gamma_m^1 \to p_1, \ldots, \gamma_m^{|S_m|} \to p_m,$$
$$)$$

where

$$\gamma_i^j = \eta_i^j \wedge \varphi \wedge \gamma$$
$$S_i = \{\eta_i^1, \ldots, \eta_i^{|S_i|}\} \quad \text{for } i = 1, \ldots, m, \text{ and}$$
$$\gamma = \bigwedge\nolimits_{f \in Sens_a \cap \sigma.T} f \wedge \bigwedge\nolimits_{f \in Sens_a \cap \sigma.F} \neg f \ .$$

We have that $\Phi^*(c, \sigma) = \Phi^*(c_2, \sigma)$. It is easy to see that the set $\{\gamma_1^1, \ldots, \gamma_m^{|S_m|}\}$ spans over S_a and $Sens_a \subseteq (\gamma_i^j)^+ \cup (\gamma_i^j)^-$. Thus, c_2 is potentially regressable. The lemma is proved with $c' = c_2$. $\qquad\square$

Lemma 4.26. Let σ be an a-state, let δ be a p-state, and let c be a conditional plan such that $\Phi^*(c, \sigma) \subseteq ext(\delta)$. There exists a plan c' such that c' is regressable with respect to (σ, δ) and c' is equivalent to c with respect to (σ, δ).

Proof. By induction on $count(c)$, the number of case plans in c.

- Base case: $count(c) = 0$
   This follows from Lemma 4.24.

- Inductive Step: Assume that we have proved the lemma for $count(c) \leq k$. We need to prove the lemma for $count(c) = k + 1$.

  By construction of c, we have two cases

  (1) $c = a; p$ where $p = case \ (\varphi_1 \to p_1, \ldots, \varphi_m \to p_m)$. Here, we have two cases.
     (a) $Sens_a \setminus \sigma = \emptyset$. In this case, we have that there exists some j such that φ_j is satisfied by σ and $\Phi^*(c, \sigma) = \Phi^*(p_j, \sigma)$. Thus, c is equivalent to p_j with respect to (σ, δ). Since $count(p_j) < count(c)$, by the inductive hypothesis and transitivity of the equivalence relation, we conclude that there exists a plan c' such that c' is regressable with respect to (σ, δ) and c' is equivalent to c with respect to (σ, δ).
     (b) $Sens_a \setminus \sigma \neq \emptyset$. Without loss of generality, we can assume that for each $\varphi_i \in S$, there exists (at least) one $\sigma' \in \Phi(a, \sigma)$ such that φ_i is satisfied in σ'. Using Lemma 4.25, we can construct a plan $c_1 = a; case \ (\varphi_1' \to p_1', \ldots, \varphi_n' \to p_n')$ which is potentially regressable and $\Phi^*(c_1, \sigma) = \Phi^*(c, \sigma)$. From the construction of c_1, we know that for each $\sigma' \in \Phi(a, \sigma)$ there exists one and only one j, $1 \leq j \leq n$, such that φ_j' is satisfied in σ'. Applying the inductive hypothesis for (σ', δ) and the plan p_i', we know that there exists a regressable plan q_i which is equivalent to p_i' with respect to (σ', δ). This implies that $c' = a; case \ (\varphi_1' \to q_1, \ldots, \varphi_n' \to q_n)$ is equivalent to c with respect to (σ, δ). Furthermore, every case plan in c' is potentially regressable and each q_i is regressable with respect to (σ', δ). To complete the proof, we will show that c' is not redundant with respect to (σ, δ). Because for each $\sigma' \in \Phi(a, \sigma)$ there exists at most one j such that φ_j' is satisfied in σ', none of the branches can be removed. Since $Sens_a \setminus \sigma \neq \emptyset$ there are more than one a-state in $\Phi(a, \sigma)$. Therefore, we cannot replace c' by one of its branches. This, together with the fact that q_j is not redundant with respect to (σ', δ), implies that c' is not redundant with respect to (σ, δ). The inductive hypothesis is proved for this case as well.
  (2) $c = \alpha; c_1$ where α is a sequence of non-sensing actions and c_1 is a case plan. Let $P_\alpha = \{\alpha' \mid \alpha'$ is a subplan of α and there exists some c_1'' such that $\alpha'; c_1''$ is equivalent to c with respect to $(\sigma, \delta)\}$. Let β be a member of P_α such that $|\beta| = \min\{|\alpha'| \mid \alpha' \in P_\alpha\}$[7]. Since $P_\alpha \neq \emptyset$, β exists. We have that β is a sequence of non-sensing actions, and so, $\Phi^*(\beta, \sigma) = \{\sigma_1\}$. It follows from the above case and the inductive hypothesis that there exists a regressable plan c_1' which is equivalent to c_1 with respect to (σ_1, δ). Consider the plan $c' = \beta; c_1'$. We have that c' is a potentially regressable conditional plan. To complete the proof, we will show that c' is not redundant with respect to (σ, δ). Assume the contrary, we will have three cases:
     (a) There exists a subplan β' of β such that $q = \beta'; c_1'$ is equivalent to c' with respect to (σ, δ). This implies that $\beta'; c_1$ is equivalent to c' with respect to (σ, δ) which contradicts the construction of β.
     (b) There exists a subplan c'' of c_1' such that $q = \beta; c''$ is equivalent to c' with respect to (σ, δ). This implies that c'' is equivalent to c_1' with respect to (σ_1, δ) which contradicts the construction of c_1'.
     (c) There exists a subplan β' of β and a subplan c'' of c_1' such that $q = \beta'; c''$ is equivalent to c' with respect to (σ, δ). This implies that $\beta' \in P_\alpha$ and $|\beta'| < |\beta|$, which is a contradiction on the construction of β. Thus this case cannot happen as well.

---

[7]For a sequence of actions γ, $|\gamma|$ denotes the length of γ.

This shows that c' is not redundant with respect to (σ, δ). So, we have proved that c' is regressable and equivalent to c with respect to (σ, δ). The inductive step is proved for this case.   □

We are now ready to prove the completeness of our regression formulation, which is illustrated by Figure 5.

Theorem 4.27 (Completeness of Regression). Given a planning problem $P = \langle A, O, I, G \rangle$ and a progression solution c of P, there exists a regression solution c' of P such that c' is not redundant and is equivalent to c with respect to (σ_I, δ_G).
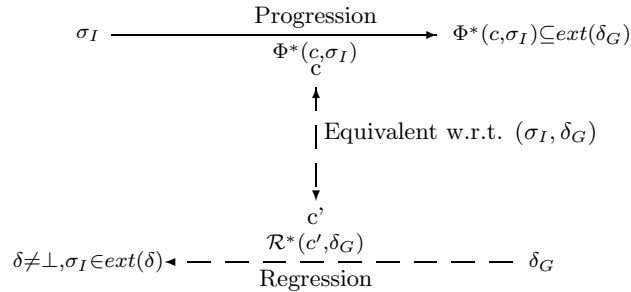
$$\sigma_I \xrightarrow[\underset{c}{\Phi^*(c,\sigma_I)}]{\text{Progression}} \Phi^*(c,\sigma_I) \subseteq ext(\delta_G)$$

Equivalent w.r.t. (σ_I, δ_G)

c'

$\delta \neq \bot, \sigma_I \in ext(\delta) \leftarrow - - \underset{\text{Regression}}{\overset{\mathcal{R}^*(c',\delta_G)}{- - -}} - - - \delta_G$

Figure 5: Illustration of Theorem 4.27.

Proof. Lemma 4.26 implies that there exists a regressable plan c' with respect to (σ_I, δ_G) which is equivalent to c with respect to (σ_I, δ_G). The non-redundancy of c' follows from the fact that it is a regressable plan. The conclusion of the theorem follows directly from Lemma 4.23 and Theorem 4.8.   □

5. RELATED WORK

Waldinger [25] is probably the first to discuss regression in Artificial Intelligence. In his paper, Waldinger uses the concept of regression in *plan modification*. To plan for several goals simultaneously, say P and Q, his strategy was to first find a plan to achieve P, then modify that plan to achieve Q. In order to achieve Q, regression is used to make sure that any action added to the existing plan will not interfere with P. Waldinger's regression is based on the idea of "weakest precondition" proposed by Dijkstra in 1975 [8](see also, e.g., [2, 7]). Intuitively, regression from a logical sentence that is represented by a conjunction of goals, *conj*, via an action, A, yields another logical sentence that encodes what must be true before A is performed to make *conj* true immediately afterwards. This is computed by the formula

$$S' = Prec(A) \cup (S \setminus Add(A)),$$

where S denotes the set of goals in the conjunction *conj*, S' denotes subgoals in the regressed conjunction, $Pre(A)$ denotes the set of preconditions of A, and $Add(A)$ denotes the set of add conditions of A; something similar to what is proposed in [26]. Following Waldinger, Nilsson [15] discusses regression with respect to partially grounded actions and proposes a regression algorithm for plan generation.

Another early effort in formulating regression over simple (non-sensing) actions is due to Pednault [16]. In his Ph.D. thesis [16], Pednault proposed the language ADL (Action

Description Language) that extends STRIPS and allows, amongst other things, conditional effects. In addition, Pednault also presents sound and complete formula-based regression operators for ADL actions. Addressing a similar problem, Reiter [18] also presents a sound and complete formula-based regression formulation over simple actions within the Situation Calculus framework. It reduces reasoning about future situations to reasoning about the initial situation using first-order theorem proving. Regression operators are provided for the formulae, with and without functional fluents.

Scherl and Levesque [19] were probably the first to extend the regression formulation for simple actions to include sensing actions. They directly formalize regression in first order logic, within the framework of Situation Calculus. Their formula-based regression operator is defined with respect to a set of successor state axioms which was based on Moore's formulation of accessible worlds [13]. They show that, for any plan P expressed by a ground situation term s_{gr} (a ground situation term is built on the initial situation by repeatedly applying the function do on it), the axiomatization F of a domain including the successor state axioms F_{ss}, G is an arbitrary sentence then

$$F \models G(s_{gr}) \quad \Leftrightarrow \quad F \setminus F_{ss} \models R^*[G(s_{gr})],$$

where $R^*(\varphi)$ indicates that the regression operator is repeatedly applied until the regressed formulae is unchanged. Intuitively, this shows that the regression is sound and complete. However, Scherl and Levesque do not define regression over conditional plans. Later, Reiter adapts the work of Scherl and Levesque in his book [18]. He does not, however, consider regression on conditional plans. De Giacomo and Levesque [6] consider a generalized action theory where successor state axioms and sensing information are conditionally applicable. For example the following conditional successor state axiom [6] expresses that if a robot is alone in a building, then the status of the door is only defined by the robot's actions *open* and *close*.

$Alone(s) \supset$
$\quad DoorOpen(x, do(a, s)) \equiv$
$\quad\quad a = open(x) \vee (a \neq close(x) \wedge DoorOpen(x, s)).$

Here the *sensor fluent formula $Alone(s)$* expresses the condition that the robot is alone in the building in a situation s, $DoorOpen(x, do(a, s))$ expresses the fact that a door x is open in the situation after the robot performs an action in the situation s, and $DoorOpen(x, s)$ expresses the fact the door x is open in the situation s. Similarly, the following conditional sensed fluent axiom [6] expresses the condition that if the robot is outdoors, then its on-board thermometer always measures the temperature around the robot.

$Outdoor(s) \supset$
$\quad OutDoorTemperature(n, s) \equiv thermometer(s) = n.$

Their formula-based regression is then defined over *histories*. A history is defined as a sequence $(\vec{v_0}).(A_1, \vec{v_1}), \ldots, (A_n, \vec{v_n})$ where each A_i is an action, $\vec{v_i}$ represents a vector of the values $\langle v_{i,1}, \ldots, v_{i,m} \rangle$ and $v_{i,j}$ represents the reading value of jth sensor after the ith action. However, they showed that the regression although sound, does not guarantee completeness in some circumstances. They also did not consider regression on conditional plans.

In another direction, Son and Baral [20] study regression over sensing actions using the high-level action language $\mathcal{A}_K$. In this work, they provide a state-based transition function and a formula-based regression function with respect to the full semantics. Different from the work of [6, 18], Son and Baral define regression over conditional plans. They also prove that their regression formulation is both sound and complete with respect to the transition

function. However in [20], Son and Baral do not consider precondition of actions in their regression formulation.

The regression formalism presented in this paper differs from earlier notion of regression for action theories with sensing actions in [18, 19, 20] in that our definition is a state-based regression formalism while the earlier definitions are formula-based. With regards to regression on conditional plans, we are not aware of any other work except [20]. For regression on non-sensing actions, our definition is close to the formula used in [4].

6. Conclusion, Discussion, and Future Work

In this paper, we developed a state-based regression function in domains with sensing actions, incomplete information, and actions without conditional effects. We also extended the regression function to allow for the regression over conditional plans. We proved the soundness of the extended regression function with respect to the definition of the progression function and developed a relaxed notion of completeness for the regression function.

It is interesting to note that for planning problems described in this paper, the progression function developed in this paper is equivalent to the full semantics for domains with sensing actions and incomplete information and to the 0-approximation developed in [20]. This implies that the regression function $\mathcal{R}$ (and hence $\mathcal{R}^*$) is also complete with respect to the full semantics for planning problems as defined in Section 2.1. Since the complexity of (conditional) planning with respect to the 0-approximation is lower than that with respect to the full semantics of sensing actions, this means that the conditional planning problem for domains presented in this paper has a lower complexity than it is in general. In other words, the complexity of the conditional planning problem presented in this paper in **NP**-complete, whereas the complexity of the conditional planning problem for action theories with conditional effects is Σ_P^2-complete [1]. We observe that this complexity results are somewhat different than the complexity results in [3], as the planning problems in [3] do not contain sensing actions and are complete.

It should be noted that the notion of a conditional plan in this paper is not as general as in [20]. For example, we do not consider plans of the form $c_1; c_2$ where c_1 and c_2 are case plans. This is done to make the presentation of the proofs easier to follow. Indeed, in [22], we proved that all of the theorems in this paper are valid with respect to conditional plans defined in [20].

Finally, we would like to mention that we have developed a regression-based planner, called **CPR**, using the regression formulation proposed in this paper [23]. The planner employs the best first search strategy with a heuristic function similar to the HSP-r heuristic function [4]. Due to the fact that most of the available benchmarks in planning with sensing actions allow disjunction in the initial state and conditional effects, an experimental evaluation of **CPR** against other planners could not be done with respect to the benchmarks. We have therefore developed our own domains to test **CPR**. Our initial experimental result shows that **CPR** performs reasonably well [23]. The code of **CPR** and the domains are available at `http://www.cs.nmsu.edu/~tson/CPR`.

Our main goal in the near future is to extend the regression formalism proposed in this paper to allow conditional effects and disjunctive initial states. This will allow us to extend **CPR** to deal with conditional effects and to evaluate the planning approach based on regression against forward chaining approaches.

Acknowledgment

References

[1] C. Baral, V. Kreinovich, and R. Trejo. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122:241–267, 2000.

[2] E. Best. Semantics of Sequential and Parallel Programs. Prentice Hall, 1996.

[3] T. Bylander. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69:165–204, 1994.
Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122:241–267, 2000.

[4] B. Bonet and H. Geffner. Planning as Heuristic Search. *Artificial Intelligence*, 129(1–2):5–33, 2001.

[5] D. Bryce, S. Kambhampati, and D. Smith. Planning Graph Heuristics for Belief Space Search. Technical report, Arizona State University, Computer Science and Engineering, 2004. `http://www.public.asu.edu/~danbryce/papers/`.

[6] G. De Giacomo and H. Levesque. Projection using regression and sensors. In *Proc. of the Sixteen International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 160–165, 1999.

[7] J. W. De Bakker. Mathematical theory of program correctness. Englewood Cliffs, N.J, Prentice-Hall International, 1980.

[8] E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. In *Communications of the ACM*, 18(8):453457, August 1975.

[9] R. Fikes and N. Nilson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[10] H. Levesque. What is planning in the presence of sensing? In *Proceedings of the 14th Conference on Artificial Intelligence*, pages 1139–1146. AAAI Press, 1996.

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

[12] J. Lobo, S. Taylor, and G. Mendez. Adding knowledge to the action description language $\mathcal{A}$. In *AAAI 97*, pages 454–459, 1997.

[13] R. Moore. A formal theory of knowledge and action. In J. Hobbs and R. Moore, editors, *Formal theories of the commonsense world*. Ablex, Norwood, NJ, 1985.

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

[15] N. Nilson. Principles of Artificial Intelligence. Tioga publishing company, 1980.

[16] E. Pednault. Toward a Mathematical Theory of Plan Synthesis. PhD thesis, Stanford University, 1986.

[17] L. Pryor and G. Collins. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339, 1996.

[18] R. Reiter. KNOWLEDGE IN ACTION: Logical Foundations for Describing and Implementing Dynamical Systems. MIT Press, 2001.

[19] R. Scherl and H. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1-2), 2003.

[20] T.C. Son and C. Baral. Formalizing sensing actions - a transition function based approach. *Artificial Intelligence*, 125(1-2):19–91, January 2001.

[21] P.H. Tu, T.C. Son, and C. Baral. Reasoning and Planning with Sensing Actions, Incomplete Information, and Static Causal Laws using Answer Set Programming. *Theory and Practice of Logic Programming*, 2006.

[22] L.C. Tuan. Regression in the Presence of Incomplete Information and Sensing Actions, and its Application to Conditional Planning. PhD thesis, Arizona State University, 2004.

[23] L.C. Tuan, C. Baral, and T.C. Son. Regression-based Conditional Planning in the Presence of Sensing Actions and Uncertainty in the Initial State. Technical report, Computer Science Department, New Mexico State University, 2005. `http://www.cs.nmsu.edu/TechReports/2005/004.ps`.

[24] L.C. Tuan, C. Baral, X. Zhang, and T.C. Son. Regression With Respect to Sensing Actions and Partial States. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, pages 556–561. AAAI Press, 2004.

[25] R. Waldinger. Achieving several goals simultaneously. *Machine Intelligence*, pages 94–136, 1977.

[26] D. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61, winter 1994.

[27] D. Weld, C. Anderson, and D. Smith. Extending graphplan to handle uncertainty and sensing actions. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press, 1998.