Int J Software Informatics, Vol.3, No.1, March 2009, pp. 3–29 E-mail: ijsi@iscas.ac.cn
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org
©2009 by Institute of Software, Chinese Academy of Sciences. All rights reserved. Tel: +86-10-62661040

State-Based Regression with Sensing and Knowledge*

Richard Scherl[1], Tran Cao Son[2], and Chitta Baral[3]

[1](CS Department, Monmouth University, West Long Branch, NJ, USA, rscherl@monmouth.edu)

[2](CS Department, New Mexico State University, Las Cruses, NM, USA, tson@cs.nmsu.edu)

[3](CS and Engineering, Arizona State University, Tempe, AZ, USA, chitta@asu.edu)

Abstract This paper develops a state-based regression method for planning domains with sensing operators and a representation of the knowledge of the planning agent. The language includes primitive actions, sensing actions, and conditional plans. The regression operator is direct in that it does not depend on a progression operator for its formulation. We prove the soundness and completeness of the regression formulation with respect to the definition of progression and the semantics of a propositional modal logic of knowledge. The approach is illustrated with a running example that can not be handled by related methods that utilize an approximation of knowledge instead of the full semantics of knowledge as is used here. It is our expectation that this work will serve as the foundation for the extension of work on state-based regression planning to include sensing and knowledge as well.

Key words: regression; plans; knowledge; sensing

Scherl R, Son TC, Baral C. State-Based regression with sensing and knowledge. *Int J Software Informatics*, **2009, 3(1): 3–29. http://www.ijsi.org/1673-7288/3/3.htm**

1 Introduction

Progression and regression are two important reasoning methods in reasoning about actions and change. Progression is defined for computing the possible states resulting from the execution of a plan from a given state. Formally, the progression function could be defined as a mapping

$$F : Actions \times States \longrightarrow 2^{States} \tag{1}$$

where $States$ denotes the set of possible states of the world and $Actions$ denotes the set of actions. This function is then extended to compute the result of the execution of a plan from a given state. Regression, on the other hand, is used to determine the possible states of the world, from which the execution of a given plan results in some states satisfying a predefined formula. Given the progression function F,

* This effort is sponsored by the DTO's AQUAINT program under award number N61339-06-C-0143; the Knowledge Fusion Center of the Army Research Laboratory under contract number DAAD-03-2-0034; NSF under grant number 0412000 and from ONR-MURI number N00014-07-1-1049; NSF grants IIS-0812267, EIA-0220590, and CNS-0454066.
Corresponding author: Richard Scherl, Email: rscherl@monmouth.edu
Manuscript received 2009-03-22; revised 2009-07-06; accepted 2009-07-15.

mathematically, the regression function R should be defined by the inverse of F; i.e., given a formula φ and an action a, the regression function R is defined by a mapping

$$R : Formulas \times Actions \longrightarrow Formulas \tag{2}$$

where $Formulas$ is the set of formulas. Each formula (from the domain and the range of R) characterizes a set of states, such that

$$R(\phi, a) = \psi \text{ if and only if } \forall s_1, s_2.(s_1 \models \psi \wedge s_2 \in F(a, s_1) \rightarrow s_2 \models \phi)$$

where $\models$ denotes the usual entailment relation between states and formulas. For action domains with sensing actions and incomplete information, this is the formula for regression adopted in Refs.[25, 15].

By defining the regression function as the inverse of the progression function, we obtain a generic formalism that is ready for use in action domains in which the progression function is known. However, this definition is not *constructive*, i.e., to compute the result of regression on a formula, one might have to guess the answer and then verify it using the progression function. Given that the complexity of the problem of computing the result of regression is NP-complete[15], which holds even if actions are deterministic, this is not a surprise.

Despite the computational complexity of this problem, several regression formalisms for action theories with sensing actions and incomplete information have been investigated[25, 13, 15, 17, 18, 23] and some have been successfully implemented[15, 23]. Regression formalisms can be classified by their use of Equation (2) in their definition of regression. In some proposals, Refs.[25, 15], the regression function is defined by Equation (2). We call these proposals *indirect definitions* of the regression function. In other formalisms, the regression function is defined *directly*, i.e., no reference to the progression function is used in its definition and Equation (2) serves as a means for the verification of its soundness and completeness[13, 17, 18, 23].

Works that define a direct regression function follow two patterns: one defines the function on formulas and another defines the function on sets of states or formulas in conjunctive normal forms. The later, also referred to as *state-based regression*, has been implemented in a conditional planner[23], which is competitive against several other conditional planners[22]. This result is also in line with the success of regression planners in classical planning[2, 10].

In this paper, we develop a direct regression function for domains that include sensing actions, knowledge, and conditional actions. The work developed here can be seen as an extension of the work of Ref.[23] to be sound and complete with respect to the full semantics of knowledge and actions with conditional effects. To the best of our knowledge, the proposal developed in this paper is the first state-based regression formalism which has these properties. Needless to say, this completeness is obtained at the cost of greater complexity (as discussed from a theoretical perspective in Ref.[1]), but the greater expressivity is needed for many problems such as the illustrative example used in this paper. The completeness (and soundness) of the regression operator is shown with respect to the definition of progression and the semantics of a propositional modal logic of knowledge.

It is our expectation that this work will serve as the foundation for the extension of the promising state-based regression planning methods[10, 2] to domains that include sensing and a representation of the knowledge of the planning agent. The development of conditional planning algorithms based on the regression operator presented here is not discussed in this paper, but forms an important part of our future work in this area.

Our planning language is defined in Section 2 and the semantics based on a progression function is covered in Section 3. An extended example that will be used throughout this paper is introduced in Section 4. Section 5 describes our regression operator and proves the regression method to be sound and complete with respect to progression. Additionally, the extended example is used to illustrate the method. Section 7 discusses related work and finally the conclusions are summarized in Section 8.

2 Language

A planning domain $\mathbf{D} = \langle \mathbf{F}, \mathbf{O_{ns}}, \mathbf{O_{se}}, \mathbf{A}, \mathbf{I}, \mathbf{G} \rangle$ consists of a finite set of propositional fluent symbols $\mathbf{F}$, a finite set of ordinary (nonsensing) action operators $\mathbf{O_{ns}}$, a finite set of sensing action operators $\mathbf{O_{se}}$, a representation $\mathbf{A}$ of the preconditions and effects of these action operators, a specification of the initial state of the world $\mathbf{I}$, and a specification of the goal state $\mathbf{G}$. The propositional fluent symbols include the symbol $\top$ that is true in all interpretations.

The representation of the initial state $\mathbf{I}$ consists of propositions of the form **initially** φ, where φ is an arbitrary propositional formula formed from $\mathbf{F}$. For example: **initially** $\neg \mathrm{P}_3$ and **initially** $\mathrm{P}_1 \vee \mathrm{P}_2$.

The planner is given knowledge of this initial state of the world. Note that in the example to follow, the use of implication in something of the form $\mathrm{P}_1 \rightarrow \mathrm{P}_2$ is merely an abbreviation for $\neg \mathrm{P}_1 \vee \mathrm{P}_2$.

A goal $\mathbf{G}$ (e.g., $\mathrm{P}_1 \wedge \neg \mathrm{P}_2$) is always a conjunction of literals. The goal of the planner is to achieve knowledge of these literals and also know that they are achieved.

The specification of actions (non-sensing actions) $\mathbf{A}$, indicates the effects (with conditions) and also executability preconditions of all actions in $\mathbf{O}$. Both the effects, conditions, and executability conditions are restricted to be conjunctions of literals which we represent as sets of literals. Consider an arbitrary action $\textsc{act}_1$:

Effect: $\{\{\mathrm{P}^1_1, \ldots, \mathrm{P}^1_{n^1}\} \Rightarrow \{\mathrm{Q}^1_1, \ldots, \mathrm{Q}^1_{m^1}\}, \ldots \{\mathrm{P}^j_1, \ldots, \mathrm{P}^j_{n^j}\} \Rightarrow \{\mathrm{Q}^j_1, \ldots, \mathrm{Q}^j_{m^j}\}\}$

ExCond: $\{\mathrm{R}_1, \ldots, \mathrm{R}_o\}$

The effect is a set of condition-effect pairs. The action $\textsc{act}_1$ is executable in a state as long as the conjunction of $\{\mathrm{R}_1, \ldots, \mathrm{R}_o\}$ holds. For each i, the conjunction of the literals $\{\mathrm{Q}^i_1, \ldots, \mathrm{Q}^i_{m^i}\}$ must hold in the successor state if the conjunction of $\{\mathrm{P}^i_1, \ldots, \mathrm{P}^i_{n^i}\}$ holds in the state in which the action begins. It is required that the conditions of the various condition-action pairs be mutually exclusive. Therefore no more than one condition can hold in any single state. It is assumed that the language includes an action $\textsc{noOp}$ that has one condition-effect with the condition as $\top$, an empty consequent, and **ExCond** as $\top$.

Some notation is useful to talk about the specifications of actions. The function **excond**(a) returns the ExCond of action a. The function **effects**(a) returns a set of pairs consisting of the antecedent and consequent of the conditional effect. Given such a pair e, the function **condition**(e) yields a list of literals that constitutes the antecedent or condition of effect e and the function **head**(e) yields the list of literals that constitutes the consequent of e. For a literal l, $\bar{l}$ denotes its complementary literal; for a set of literals S, $\bar{S} = \{\bar{l} \mid l \in S\}$.

There are also sensing actions that determine the truth of a fluent:

$$\textsc{sense}_1 : \textbf{Effect: } \{\} \textbf{ Determines: } \mathrm{P}_1 \textbf{ ExCond: } \{\mathrm{R}_1, \ldots, \mathrm{R}_o\}$$

The sensing action $\textsc{sense}_1$ determines the truth of proposition P_1, called the *sensed-fluent*, and is executable if the conjunction $\{\mathrm{R}_1, \ldots, \mathrm{R}_o\}$ is satisfied. Given a sensing action a, **determines**(a) returns the sensed-fluent of action a. The restriction to a single fluent is not more restrictive than a set of fluents since a sequence of sensing actions can be equivalent to a sensing action that determines a set (i.e., a conjunction) of literals. Note that since we require that the **Effect** component be empty, it is ensured that sensing actions have no effect on the world

Plans are constructed out of a sequence of actions and the if/then constructs are called conditional plans. For simplicity of the presentation of the definitions in this paper, we do impose some restrictions on the form of conditional plans as indicated in the following definition:

Definition 1 (Plan). *Let a be an action.*

1. $[]$ *is a plan.*

2. $a; c$ *is a plan if c is a plan and a is a non-sensing action.*

3. $a; [\textbf{ if } f \textbf{ then } c_1 \textbf{ else } c_2]$ *is a plan if a is a sensing action, which senses f, and c_1 and c_2 are plans.*

It is easy to see that every conditional plan is either a sequence of non-sensing actions or of the from $a_1; \ldots; a_k; a; [\textbf{ if } f \textbf{ then } c_1 \textbf{ else } c_2]$ where $a_1, \ldots, a_k$ is a sequence of non-sensing actions, a is a sensing action, which senses f, and c_1 and c_2 are plans.

The restriction to have sensing actions only occur immediately prior to a conditional construct (and to require that the sensed fluent be identical to the conditional fluent) is made to simplify the proofs. It is not a restriction in the expressivity of the language of plans since one can always replace something of the form $\alpha; sense(f); \gamma$ by $\alpha, sense(f), [\textbf{ if } f \textbf{ then } \gamma \textbf{ else } \gamma]$.

3 Progression

A *state* is a complete (i.e., for each fluent f either f or $\neg f$ is included) and consistent set of fluent literals (i.e., for each fluent f both f and $\neg f$ are not included). It is a propositional representation of the truth (falsity) of propositions in a particular possible world. A *knowledge set* (or *k-set*) is a set of states. A *combined structure* (or *c-structure*) is a pair $\langle s, \Sigma \rangle$ where Σ is a k-set and s is a state belonging to Σ. A *partial state* (or *p-state*) is a consistent set of fluent literals. A *partial structure* (or *p-structure*) is a pair $\langle \delta, \Delta \rangle$ where Δ is a set of p-states and δ is a p-state belonging to Δ. A p-structure $\gamma = \langle \delta, \Delta \rangle$ *extends* a p-structure $\gamma' = \langle \delta', \Delta' \rangle$, denoted by $\gamma' \sqsubseteq \gamma$, if (i) $\delta' \subseteq \delta$; (ii) for each $\lambda \in \Delta$ there exists some $\lambda' \in \Delta'$ such that $\lambda' \subseteq \lambda$; and (iii) for each $\lambda' \in \Delta'$ there exists some $\lambda \in \Delta$ such that $\lambda' \subseteq \lambda$. For a set of p-structures Δ and a p-structure γ, we write $\Delta \trianglerighteq \gamma$ (resp. $\gamma \trianglelefteq \Delta$) if there exists some $\gamma' \in \Delta$ such that $\gamma' \sqsubseteq \gamma$ (resp. $\gamma' \sqsubseteq \gamma$).

Remark 1. Each state is a p-state. Each c-structure is also a p-structure. Thus, we can talk about the relationship $\sqsubseteq$ between p-structures and c-structures.

For example, the following are states if we consider only the fluent symbols F and G: $s_1 = \{\mathrm{F}, \mathrm{G}\}$, $s_2 = \{\neg\mathrm{F}, \mathrm{G}\}$, $s_3 = \{\mathrm{F}, \neg\mathrm{G}\}$, and $s_4 = \{\neg\mathrm{F}, \neg\mathrm{G}\}$. If s is a state (or more generally a p-state), then $s \models l_1$, where l_1 is a literal, means that $l_1 \in s$. The definition of $\models$ can be inductively generalized in the obvious way to $s \models \varphi$ where φ is an arbitrary formula or a set of literals representing a conjunction of literals. Knowledge sets are a representation of the knowledge (ignorance) of the planner. For example, if we consider only the fluent symbols F and G, some possible knowledge sets are: $b_1 = \{s_1, s_2, s_3, s_4\}$, $b_2 = \{s_1, s_4\}$, and $b_3 = \{s_2, s_3\}$. If Σ is a knowledge set, then $\Sigma \models \mathbf{Knows}(l_1)$ means that $\forall s \in \Sigma,\ s \models l_1$. Otherwise $\Sigma \models \neg\mathbf{Knows}(l_1)$. This definition of $\models$ can be inductively generalized in the obvious way to define $\Sigma \models \mathbf{Knows}(\varphi)$, where φ is an arbitrary formula. Given a c-structure $st = \langle s, \Sigma\rangle$, $st \models l_1$, if $l_1 \in s$. Additionally, $st \models \mathbf{Knows}(l_1)$, if $\Sigma \models \mathbf{Knows}(l_1)$. For the definition of the transition function (to be presented next), we need to introduce a structure $\perp$. For any expression Ψ, $\perp \not\models \Psi$. Finally, for every c-structure $\langle s, \Sigma\rangle$, it is required that $s \in \Sigma$. We are therefore using the semantics S5 as the basis of our modal logic of knowledge and there is no need to allow nesting of modal operators.

Given a planning domain, we can define the initial states and initial c-structures as follows:

Definition 2 (Initial state). *A state s is an initial state of a planning domain* **D** *if $s \models \varphi$ for every statement* **initially** φ *in I.*

Definition 3 (Initial c-structure). *A c-structure $\langle s, \Sigma_0\rangle$ is an* initial c-structure *if every $u \in \Sigma_0$ is an initial state.*

It is straight forward to develop an algorithm that converts the conjunction of the φs from each **initially** φ statement into the set of possible initial states.

Some notation will be needed in the machinery to be developed. Two p-states δ and δ' are *compatible* with respect to a set of fluent literals S, denoted by $\delta \sim_S \delta'$ if $S \cap \delta = S \cap \delta'$. A fluent f is *specified* in δ if $\delta \cap \{f, \neg f\} \neq \emptyset$; otherwise, f is *unspecified* in δ. f is *specified* in a set of p-states Δ if it is specified in every $\gamma \in \Delta$. f is *foreign* in a set of p-states Δ if it is unspecified in every $\gamma \in \Delta$. For a fluent f, $\overline{f}$ denotes $\neg f$ and $\overline{\neg f} = f$. For a set of literals L, by $\overline{L}$ we denote the set $\{\overline{l} \mid l \in L\}$.

In progression, a plan is executed starting from an initial structure. We need to specify a transition function $\hat{\Phi}$ from plans and structures into structures. This is based on the specification of a transition function Φ from actions and structures into structures.

Some notation needs to be initially defined so that the transition function can be specified. For a p-state δ and action a, a is executable in δ if $\mathbf{excond}(a) \subseteq \delta$. a is executable in a p-structure $\langle \delta, \Delta\rangle$ if $\gamma \models \mathbf{excond}(a)$ for every $\gamma \in \Delta$. The effect of a in δ is defined by

$$e_a(\delta) = \begin{cases} \mathbf{head}(p) & p \in \mathbf{effects}(a) \text{ and } \delta \models \mathbf{condition}(p) \\ \emptyset & \neg\exists p \in \mathbf{effects}(a) \text{ s.t. } \delta \models \mathbf{condition}(p) \end{cases}$$

The set $e_a(\delta)$ is well-defined because $\mathbf{condition}(p)$ and $\mathbf{condition}(p')$ are mutual exclusive for $p \neq p'$. If $\delta \models \mathbf{condition}(p)$, we say that p is *applicable* in δ. Intuitively, $e_a(\delta)$ is the set of fluent literals that must be true after the execution of a.

Definition 4 (Result). *The result of executing a non-sensing action a in δ*

is defined by

$$Res(a,\delta) = \begin{cases} (\delta \setminus \overline{e_a(\delta)}) \cup e_a(\delta) & \text{if } \delta \models \mathbf{excond}(a) \\ \bot & \text{otherwise} \end{cases}$$

The notation $\bot$ is used to indicate that the execution of a in δ fails. It is easy to see that the following property holds.

Property 1. If $\delta \subseteq \delta'$, a is executable in δ, and p is an effect of a such that $\mathbf{condition}(p) \subseteq \delta$ then $Res(a,\delta) \subseteq Res(a,\delta')$.

The transition function over p-structures and actions is defined as follows.

Definition 5 (Transition Function). *For an action a and a p-structure $\langle \delta, \Delta \rangle$,*

– if a is a non-sensing action and it is executable in $\langle \delta, \Delta \rangle$ then

$$\Phi(a, \langle \delta, \Delta \rangle) = \langle Res(a,\delta), \{Res(a,\delta') \mid \delta' \in \Delta\} \rangle;$$

– if a is a sensing action which senses f (i.e. $\mathbf{determines}(a) = \{f\}$*) executable in $\langle \delta, \Delta \rangle$ and f is either specified or foreign in Δ then*

$$\Phi(a, \langle \delta, \Delta \rangle) = \langle \delta, \{\delta' \mid \delta' \in \Delta, \delta' \sim_{\{f,\neg f\}} \delta, \text{ and } \delta' \models \mathbf{excond}(a)\} \rangle;$$

– otherwise, $\Phi(a, \langle \delta, \Delta \rangle) = \bot$.

Intuitively, the execution of an action in a p-structure results in a p-structure. For non-sensing actions, the resulting p-structure is obtained by progressing each of its p-states. On the other hand, for sensing actions, there are two cases. In the first case, if the sensed fluent is specified in the p-structure, executing the action will result in a new p-structure, in which the fluent is known to be true or known to be false. In the second case, the sensed fluent is foreign in the p-structure, then the progression does not result in any change.[1)] It is easy to see that Φ is identical to the transition function defined in Ref.[5] when its domain is restricted to the set of all c-structures. Following Ref.[5], the function Φ needs to be extended to progress plans.

Definition 6 (Extended Transition Function). *Let c be a plan and $\sigma = \langle \delta, \Delta \rangle$ be a p-structure.*

1. *if $c = []$ then $\hat{\Phi}(c,\sigma) = \sigma$;*
2. *if $c = a; c_1$ where a is a non-sensing action and c_1 is a plan then $\hat{\Phi}(c,\sigma) = \hat{\Phi}(c_1, \Phi(a,\sigma))$;*
3. *if $c = a; [$***if** *f* **then** *c_1* **else** *$c_2]$ where a is a sensing action with* **determines** *$(a) = f$ and c_1 and c_2 are plans then*

$$\hat{\Phi}(c,\sigma) = \begin{cases} \hat{\Phi}(c_1,\sigma') & \text{if } \sigma' \models \mathbf{Knows}(f) \\ \hat{\Phi}(c_2,\sigma') & \text{if } \sigma' \models \mathbf{Knows}(\neg f) \end{cases}$$

where $\sigma' = \Phi(a,\sigma)$.

4. *$\hat{\Phi}(c,\bot) = \bot$.*

1) This is because we assume that the progression is done on the p-structures. This will not be the case if the progression is done on c-structures, which is the case when we progress from initial c-structures.

Extending the notation, we write

$$\hat{\Phi}(c,\Omega) = \begin{cases} \bot & \text{if } \hat{\Phi}(c,\sigma) = \bot \text{ for some } \sigma \in \Omega \\ \bigcup_{\sigma\in\Omega} \hat{\Phi}(c,\sigma) & \text{otherwise} \end{cases}$$

where c is a plan and Ω is a set of p-structures. We will say that c is executable in Ω if $\hat{\Phi}(c,\Omega) \neq \bot$. Given a planning domain $\mathbf{D} = \langle \mathbf{F}, \mathbf{O_{ns}}, \mathbf{O_{se}}, \mathbf{A}, \mathbf{I}, \mathbf{G} \rangle$ a progression solution is defined as follows:

Definition 7 (Progression Solution). *A plan c is a progression solution for a planning domain $\mathbf{D}$ if for every initial c-structure σ_0 of $\mathbf{D}$, $\hat{\Phi}(c,\sigma_0) \neq \bot$ and $\hat{\Phi}(c,\sigma_0) \models \mathbf{Knows}(G)$.*

This definition is illustrated in Fig.1. Intuitively, the planner begins with some knowledge of the possible ways the world can be. The actual world could be any one of the possibilities. The various possibilities (a state of the world and a set of possible states) are represented by the c-structures in Ω_0. In the figure, these are the checkered squares. A plan is a solution, if no matter which of the possibilities is in fact the actual world, an execution of the plan in that world with the knowledge of the possible ways the world could be, yields knowledge of the goal. As shown in Figure 1, the progression of the plan starting with each of the c-structures in Ω_0 yields $\hat{\Phi}(c,\Omega_0)$. Each member of $\hat{\Phi}(c,\Omega_0)$ (the squares filled with dots) must entail $\mathbf{Knows}(G)$ for c to be a progression solution.

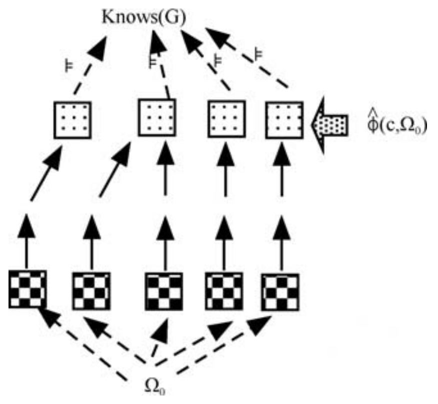


Figure 1. Progression solution

Before looking at a larger example, consider the following simple example:

Example 1. The planning domain is $\mathbf{D}_1 = \langle \{f,g,h\}, \{a\}, \emptyset, \mathbf{A}_1, \mathbf{I}, \{h\} \rangle$, where $\mathbf{A}_1$ defines a single non-sensing action a with two effects

$$p = \{f,g\} \Rightarrow h \qquad \text{and} \qquad q = \{f,\neg g\} \Rightarrow h$$

and $\mathbf{excond}(a) = \top$, $I = \{\mathbf{initially}\ f\}$, and the goal $\mathbf{G} = \{h\}$. There are only four possible initial states

$$s_1 = \{f,g,h\}$$
$$s_2 = \{f,g,\neg h\}$$
$$s_3 = \{f,\neg g,h\}$$
$$s_4 = \{f,\neg g,\neg h\}$$

and therefore $\Sigma_0 = \{s_1, s_2, s_3, s_4\}$. We have that

$$\begin{aligned}
\Phi(a, \langle s_1, \Sigma_0\rangle) &= \langle s_1, \{s_1, s_3\}\rangle \\
\Phi(a, \langle s_3, \Sigma_0\rangle) &= \langle s_3, \{s_1, s_3\}\rangle \\
\Phi(a, \langle s_2, \Sigma_0\rangle) &= \langle s_1, \{s_1, s_3\}\rangle \\
\Phi(a, \langle s_4, \Sigma_0\rangle) &= \langle s_3, \{s_1, s_3\}\rangle
\end{aligned}$$

This implies that a is a progression solution for h.

4 Medical Example

A running medical example, similar to that used in other work in this area[24], is used to illustrate our approach. A patient is ill, but alive. We know that if he is infected with disease 123, then he is also hydrated. But we do not know whether he is infected. We do have a stain, which can be used to test for infection with disease 123. If the result of the staining is blue, then the patient is infected. We have a sensing action to determine whether or not the result of the stain action is blue or not. We can treat disease 123 with medication, but the problem is that if the patient is not hydrated, he will die from the medication. So, it is important to construct the appropriate plan so that the patient is guaranteed not to die.

In this domain ($\mathbf{D}_1$), $\mathbf{O}_{ns}$ contains the following actions: MEDICATE, and STAIN. Additionally, $\mathbf{O}_{se}$ contains INSPECT. The set $\mathbf{F}$ contains DEAD, BLUE,INFECTED, and HYDRATED.

The complete set of 16 possible states for D_1 are as follows[2)]:

$$\begin{aligned}
s_1 &= \{\neg\text{DEAD}, \neg\text{INFECTED}, \neg\text{BLUE}, \neg\text{HYDRATED}\} \\
s_2 &= \{\text{DEAD}, \neg\text{INFECTED}, \neg\text{BLUE}, \neg\text{HYDRATED}\} \\
s_3 &= \{\neg\text{DEAD}, \text{INFECTED}, \neg\text{BLUE}, \neg\text{HYDRATED}\} \\
s_4 &= \{\neg\text{DEAD}, \neg\text{INFECTED}, \text{BLUE}, \neg\text{HYDRATED}\} \\
s_5 &= \{\neg\text{DEAD}, \neg\text{INFECTED}, \neg\text{BLUE}, \text{HYDRATED}\} \\
s_6 &= \{\text{DEAD}, \text{INFECTED}, \neg\text{BLUE}, \neg\text{HYDRATED}\} \\
s_7 &= \{\neg\text{DEAD}, \text{INFECTED}, \text{BLUE}, \neg\text{HYDRATED}\} \\
s_8 &= \{\neg\text{DEAD}, \neg\text{INFECTED}, \text{BLUE}, \text{HYDRATED}\} \\
s_9 &= \{\text{DEAD}, \neg\text{INFECTED}, \text{BLUE}, \neg\text{HYDRATED}\} \\
s_{10} &= \{\text{DEAD}, \neg\text{INFECTED}, \neg\text{BLUE}, \text{HYDRATED}\} \\
s_{11} &= \{\neg\text{DEAD}, \text{INFECTED}, \neg\text{BLUE}, \text{HYDRATED}\} \\
s_{12} &= \{\text{DEAD}, \text{INFECTED}, \text{BLUE}, \neg\text{HYDRATED}\} \\
s_{13} &= \{\neg\text{DEAD}, \text{INFECTED}, \text{BLUE}, \text{HYDRATED}\} \\
s_{14} &= \{\text{DEAD}, \neg\text{INFECTED}, \text{BLUE}, \text{HYDRATED}\} \\
s_{15} &= \{\text{DEAD}, \text{INFECTED}, \neg\text{BLUE}, \text{HYDRATED}\} \\
s_{16} &= \{\text{DEAD}, \text{INFECTED}, \text{BLUE}, \text{HYDRATED}\}
\end{aligned}$$

We have the following axiomatization of the initial state:

2) Note that these are being explicitly listed for expository purposes. The method does not need to construct the entire state space.

initially $\neg$DEAD $\wedge$ $\neg$BLUE
initially INFECTED $\rightarrow$ HYDRATED

and the goal is to have the patient not dead and not infected:

Goal :$\neg$DEAD $\wedge$ $\neg$INFECTED

The following are initial states:

$$\begin{aligned}s_1 &= \{\neg\text{DEAD}, \neg\text{INFECTED}, \neg\text{BLUE}, \neg\text{HYDRATED}\}\\ s_5 &= \{\neg\text{DEAD}, \neg\text{INFECTED}, \neg\text{BLUE}, \text{HYDRATED}\}\\ s_{11} &= \{\neg\text{DEAD}, \text{INFECTED}, \neg\text{BLUE}, \text{HYDRATED}\}\end{aligned}$$

Therefore, the knowledge set that we need to consider is $\{s_1, s_5, s_{11}\}$. The patient is not dead ($\neg$DEAD) and the stain is not blue ($\neg$BLUE) in all three states. In one state the patient is infected (INFECTED) and hydrated (HYDRATED), in another he is not infected and not hydrated, and in the other he is hydrated and not infected. So, we have the following three initial c-structures:

$$\langle s_1, \{s_1, s_5, s_{11}\}\rangle \quad \langle s_5, \{s_1, s_5, s_{11}\}\rangle \quad \langle s_{11}, \{s_1, s_5, s_{11}\}\rangle$$

Two of the initial states (s_1 and s_5) are already satisfying the goal. The third one (s_{11}) does not. Thus, we must perform actions that do not undo the goal requirements in the first two and changes the third one to a state that satisfies the goal. Additionally, in the resulting structures, the planning agent must know that the goal holds. The axiomatization **A** of the actions are as follows:

- STAIN: **Effect** : $\{\{\text{INFECTED}\} \Rightarrow \{\text{BLUE}\}\}$
  ExCond : $\neg$BLUE

- MEDICATE: **Effect** : $\{\{\text{HYDRATED}\} \Rightarrow \{\neg\text{INFECTED}\},$
  $\{\neg\text{HYDRATED}\} \Rightarrow \{\text{DEAD}\}\}$
  ExCond :$\top$

- INSPECT: **Effect:** $\emptyset$
  Determines: BLUE
  ExCond: $\top$

A plan to accomplish the goal is as follows:

$$c = [\text{STAIN}; \text{INSPECT}; [\textbf{if } \text{BLUE} \textbf{ then } \text{MEDICATE} \textbf{ else } \text{NOOP}]].$$

Let

$$c_1 = [\text{INSPECT}; [\textbf{if } \text{BLUE} \textbf{ then } \text{MEDICATE} \textbf{ else } \text{NOOP}]].$$

The plan ensures that medicate is only applied in the correct state. The idea is that we want to stain and inspect so we can differentiate the initial states so that we only medicate in one of them because medicating in the other prevents achievement of the goal. To see how the plan works, let us consider the step by step progression of each of the initial c-structures. First consider:

$$\begin{aligned}\hat{\Phi}(c, \langle s_{11}, \{s_1, s_5, s_{11}\}\rangle) &= \hat{\Phi}(c_1, \langle s_{13}, \{s_{13}, s_1, s_5\}\rangle)\\ &= \hat{\Phi}(\text{MEDICATE}, \langle s_{13}, \{s_{13}\}\rangle)\\ &= \langle s_8, \{s_8\}\rangle\\ &= \sigma'\end{aligned}$$

Note that $\sigma' \models \mathbf{Knows}(G)$. Now consider:

$$\begin{aligned}\hat{\Phi}(c, \langle s_1, \{s_{11}, s_1, s_5\}\rangle) &= \hat{\Phi}(c_1, \langle s_1, \{s_{13}, s_1, s_5\}\rangle)\\ &= \langle s_1, \{s_1, s_5\}\rangle\\ &= \sigma''\end{aligned}$$

Note that $\sigma'' \models \mathbf{Knows}(G)$.

The other initial structure works similarly. So, we have that

$$\begin{aligned}\hat{\Phi}(c, \langle s_1, \{s_{11}, s_1, s_5\}\rangle) &= \langle s_1, \{s_1, s_5\}\rangle\\ \hat{\Phi}(c, \langle s_5, \{s_{11}, s_1, s_5\}\rangle) &= \langle s_5, \{s_1, s_5\}\rangle\\ \hat{\Phi}(c, \langle s_{11}, \{s_1, s_5, s_{11}\}\rangle) &= \langle s_8, \{s_8\}\rangle\end{aligned}$$

Since the goal is satisfied by s_1, s_5, and s_8, we conclude that c is a progression solution (plan) for the domain $\mathbf{D}_1$.

5 Regression

Various formalisms on regression for reasoning about actions have been developed (e.g. [11, 13, 17, 18]). Regression has also formed the basis for a number of regression-based planners (e.g.[2, 23]). In these works, regression was designed to ignore actions that do not directly contribute to the current goal. In other words, they would consider only "useful" (i.e., those that help achieve the goal) actions in the regression procedure. But with these restricted forms of regression, there are plans found by progression based planners that can not be found by a regression based planner. This can be seen in the following example[3) :

Example 2. Consider $\mathbf{D}_2 = \langle \{f, h\}, \{a, b\}, \emptyset, \mathbf{A}_2, \emptyset, \{h\}\rangle$, where $\mathbf{A}_2$ is defined by

$$\mathbf{excond}(a) = \mathbf{excond}(b) = \top,$$

and $\mathbf{effects}(a) = \{\{f\} \Rightarrow \{h\}\}$ and $\mathbf{effects}(b) = \{\{\neg f\} \Rightarrow \{h\}\}$.

It is easy to see that both $[a; b]$ and $[b; a]$ are progression solutions for $\mathbf{D}_2$.

Now let us try to find the plan $[b; a]$ by regression, following an adaptation of the formalism in Ref.[2] for $\mathbf{D}_2$. In this formalism, only "useful actions" are considered in the regression process.

Intuitively, we should start with the regression of a in $\{h\}$. This will result in $\{f\}$, i.e., to achieve h by means of a, we need to achieve f. As $\{f\}$ is not satisfied by

3) This example does not imply that the completeness result in Ref.[23] is incorrect since their work does not consider actions with conditional effects.

the initial condition, another regression step is needed. Because f is not present in the head of any effect of a or b, neither a nor b will be considered as "useful" for the goal of achieving f. This implies that $[b; a]$ cannot be found by a regression formalism considering only "useful" actions in its reasoning. The same argument can be made for $[a; b]$. As such, *regression formalisms under the restriction of using only "useful" actions are generally incomplete* with respect to the complete semantics.

In this work, we define a regression formalism that is a truly reversal of the progression. Even though our later goal is to use this work in planners (where for efficiency purposes it may be necessary to restrict regression), our current purpose (being theoretical in nature) is to establish the equivalence between progression and regression. We will present our formulation in a series of definitions. We start with the definition of the regression of a non-sensing action in a p-state, a set of p-states, and a p-structure. This is followed by the definition of the regression of a sensing action in a set of p-structures. Finally, we define the extended regression function, which allows for the regression of conditional plans and can be seen as the counter part to the extended transition function.

5.1 Regression with non-sensing actions

In defining the regression function, the first question we need to answer is "when can an action a be regressed in a p-state δ?" Assume that a is a non-sensing action. Intuitively, the regression of a in δ should result in δ' such that $\delta \subseteq Res(a, \delta')$. From the definition of the function Res, we know that there are two possibilities: (i) there exists an effect p of a which is applicable in δ'; or (ii) otherwise none is applicable. The first case implies that (a) there exists no literal in $\mathbf{head}(p)$ such that its negation belongs to δ; and (b) if a literal l belongs to $\mathbf{excond}(a) \cup \mathbf{condition}(p)$ (l is true in δ') and its negation $\bar{l}$ belongs to δ (l is false in δ) then $\bar{l}$ should belong to $\mathbf{head}(p)$. In the second case, we have that $\mathbf{excond}(a)$ must not be false in δ and for every effect p of a, its precondition, $\mathbf{condition}(p)$, must be false in δ'. The following definition reflects these conditions (the cases (i) and (ii) correspond to Items 1 and 2, respectively.)

Definition 8 (Regressable). *A non-sensing action a is* regressable *in a p-state δ if either 1 or 2 holds:*

1. *there exists some $p \in \mathbf{effects}(a)$ such that*

- $\overline{\mathbf{head}(p)} \cap \delta = \emptyset$,
- $\overline{\mathbf{condition}(p)} \cap \delta \subseteq \mathbf{head}(p)$, *and*
- $\overline{\mathbf{excond}(a)} \cap \delta \subseteq \mathbf{head}(p)$.

We say that a is regressable via p in δ in this case.

2. $\overline{\mathbf{excond}(a)} \cap \delta = \emptyset$ *and there exists a p-state δ' such that $\delta \cup \mathbf{excond}(a) \subseteq \delta'$ and for every $p \in \mathbf{effects}(a)$, $\overline{\mathbf{condition}(p)} \cap \delta' \neq \emptyset$.*

A non-sensing action a is regressable *in a p-structure $\langle \delta, \Delta \rangle$ if it is regressable in every p-state belonging to Δ.*

The next step is to define the result of the regression of an action a in a p-state δ. Let us denote it with δ'. Clearly, we must have that a is executable in δ'. Furthermore, if an effect p of a is applicable in δ' then $\mathbf{condition}(p)$ must be satisfied in δ'. In this case, $\mathbf{head}(p)$ need not be present in δ' since it will be added to $Res(a, \delta')$ (Item

1, Def. 8). On the other hand, if none of the effects of a is applicable in δ' then for every $p \in \textbf{effects}(a)$, δ' should contain at least some elements in $\overline{\textbf{condition}(p)}$ (Item 2, Def. 8). This leads to the following definition.

Definition 9 (Regression (non-sensing) in a P-State). *Let a be a non-sensing action regressable in the p-state δ. Let*

$$r_a^1(\delta) = \{Reg(a,p,\delta) \mid p \in \textbf{effects}(a) \text{ s.t. } a \text{ is regressable via } p \text{ in } \delta\}$$

where $Reg(a,p,\delta) = (\delta \backslash \textbf{head}(p)) \cup \textbf{condition}(p) \cup \textbf{excond}(a)$; and

$$r_a^2(\delta) = \left\{ \delta' \left| \begin{array}{l} \exists\gamma.[\gamma \subseteq \bigcup_{p \in \textbf{effects}(a)} \overline{\textbf{condition}(p)}, \\ \delta' = \delta \cup \textbf{excond}(a) \cup \gamma, \\ \delta' \text{ is consistent}, \\ \forall p \in \textbf{effects}(a).[\delta' \cap \overline{\textbf{condition}(p)} \neq \emptyset]] \end{array} \right. \right\}$$

We say that $r_a(\delta) = r_a^1(\delta) \cup r_a^2(\delta)$ is the set of p-states resulting from the regression of a in δ.

For simplicity of the presentation, let us introduce a special effect, called NoEffect. We assume that NoEffect $\in \textbf{effects}(a)$ for every a and use $Reg(a, \text{NoEffect}, \delta)$ to refer to a p-state belonging to $r_a^2(\delta)$. It is easy to see that a is regressable in δ if and only if $r_a(\delta) \neq \emptyset$.

Example 3. For domain $\mathbf{D}_2$ in Example 2 and $\delta = \{h\}$, we have that $r_a^1(\delta) = \{\{f\}\}$ and $r_a^2(\delta) = \{\{h, \neg f\}\}$.

Lemma 1. *If a non-sensing action a is regressable in δ then for each $\gamma \in r_a(\delta)$ and $\gamma \subseteq \gamma'$, $\delta \subseteq Res(a, \gamma')$.*

Proof: Consider two cases:

- $\gamma \in r_a^1(\delta)$ we have that $\gamma = (\delta \setminus \textbf{head}(p)) \cup \textbf{condition}(p) \cup \textbf{excond}(a)$ for some $p \in \textbf{effects}(a)$. Thus, a is executable in γ and $e_a(\gamma) = \textbf{head}(p)$ and $Res(a,\gamma) = (\gamma \setminus \overline{\textbf{head}(p)}) \cup \textbf{head}(p)$. This implies that $\delta \subseteq Res(a,\gamma)$ since substituting for γ we get

  $$Res(a,\gamma) = ((\delta \setminus \textbf{head}(p)) \cup \textbf{condition}(p) \cup \textbf{excond}(a) \setminus \overline{\textbf{head}(p)}) \cup \textbf{head}(p).$$

  This simplifies to

  $$Res(a,\gamma) = (\delta \cup (\textbf{condition}(p) \setminus \overline{\textbf{head}(p)}) \cup (\textbf{excond}(a) \setminus \overline{\textbf{head}(p)}) \cup \textbf{head}(p))$$

  given that $\overline{\textbf{head}(p)} \cap \delta = \emptyset$.

  Because $\gamma \subseteq \gamma'$, we have that $\textbf{excond}(a) \cup \textbf{condition}(p) \subseteq \gamma'$. This implies that $Res(a,\gamma) \subseteq Res(a,\gamma')$, which proves the lemma for this case.

- $\gamma \in r_a^2(\delta)$ we have that $Res(a,\gamma) = \gamma$, $Res(a,\gamma') = \gamma'$, and so $\delta \subseteq \gamma'$ by definition.

Corollary 1. *If a non-sensing action a is regressable in δ then $r_a(\delta) \neq \emptyset$. For this reason, this condition can be used as a necessary condition for regressable as well.*

Example 4. For domain $\mathbf{D}_1$ in Example 1, consider $\delta = \{h\}$. We have that

$$r^1_a(\delta) = \{\{f, g\}, \{f, \neg g\}\}$$

and

$$r^2_a(\delta) = \{\{h, \neg f\}, \{h, \neg f, g\}, \{h, \neg f, \neg g\}\}.$$

It should be mentioned that $r_a(\delta)$ might contain some p-states which are superset of other elements in $r_a(\delta)$. Due to the fact that if an action a is regressable in a p-state δ then it is regressable in a p-state $\delta' \subseteq \delta$, the presence of such elements do not have any impact on the theoretical results presented in this paper. Eliminating this redundancy will be important in the development of a regression-based planner and will therefore be one of our future concerns.

We now extend regression to a set of p-states.

Definition 10 (Regression (non-sensing) in a Set of p-States). *Let Δ be a set of p-states and a be a non-sensing action regressable in every $\delta \in \Delta$. The regression of a in Δ, denoted by $r_a(\Delta)$, is given by*

$$\begin{aligned} r_a(\Delta) = \{\Delta' \mid & (\forall \delta' \in \Delta'. \exists \delta \in \Delta \text{ s.t. } \delta' \in r_a(\delta)) \text{ and} \\ & (\forall \delta \in \Delta. \exists \delta' \in \Delta' \text{ s.t. } \delta' \in r_a(\delta))\} \end{aligned}$$

The first condition implies that each element in Δ' must be the result of the regression of some element in Δ and the second condition makes sure that nothing extra is introduced into Δ'. If a is not regressable in Δ, we write $r_a(\Delta) = \emptyset$. We are now ready to define the result of the regression of an action in a p-structure.

Definition 11 (Regression of Non-Sensing Actions). *Let a be a non-sensing action and $\sigma = \langle \delta, \Delta \rangle$ be in the p-structure. We define*

$$\mathcal{R}(a, \sigma) = \begin{cases} \{\langle \delta', \Delta' \rangle \mid \delta' \in r_a(\delta),\ \Delta' \in r_a(\Delta),\ \delta' \in \Delta'\} & \\ \qquad\qquad\qquad\qquad \text{if } a \text{ is regressable in } \sigma & \\ \emptyset \qquad\qquad\qquad\quad\ \text{otherwise} & \end{cases}$$

For a set of p-structures Ω,

$$\mathcal{R}(a, \Omega) = \bigcup_{\sigma \in \Omega} \mathcal{R}(a, \sigma)$$

Remark 2. We consider NoOp as a special non-sensing action and define $\mathcal{R}(\text{NoOp}, \sigma) = \{\sigma\}$ for every p-structure σ.

As a simple illustration, consider the following:

Example 5. Continue with the domain $\mathbf{D}_1$ in Example 1, let $\delta = \{h\}$, $\Delta = \{\delta\}$, and $\sigma = \langle \delta, \Delta \rangle$, we have that $r_a(\Delta) = r_a(\delta) = \{\{f, g\}, \{f, \neg g\}, \{h, \neg f\}, \{h, \neg f, g\}, \{h, \neg f, \neg g\}\}$. This leads to $\mathcal{R}(a, \sigma) = \langle \delta', \Delta' \rangle$ where $\delta' \in r_a(\delta)$ and $\Delta' \in r_a(\Delta)$ with $\delta' \in \Delta'$. Some of the members in $\mathcal{R}(a, \sigma)$ are $\langle \{f, g\}, \{\{f, g\}\} \rangle$, $\langle \{f, \neg g\}, \{\{f, \neg g\}\} \rangle$, $\langle \{f, g\}, \{\{f, g\}, \{f, \neg g\}\} \rangle$, etc.

Lemma 2. *Let a be a non-sensing action regressable in the p-structure σ. Then, for every ω such that $\mathcal{R}(a, \sigma) \trianglerighteq \omega$, a is executable in ω and $\sigma \sqsubseteq \Phi(a, \omega)$.*

Proof: Recall that $\mathcal{R}(a,\sigma) \trianglerighteq \omega$ means that there exists some $\sigma' \in \mathcal{R}(a,\sigma)$ such that $\sigma' \sqsubseteq \omega$. First, we will show that a is executable in σ' and $\sigma \sqsubseteq \Phi(a,\sigma')$. Let $\sigma = \langle \delta, \Delta \rangle$, $\sigma' = \langle \delta', \Delta' \rangle$, and $\Phi(a,\sigma') = \langle \delta'', \Delta'' \rangle$. Lemma 1 implies that $\delta \subseteq Res(a,\delta') = \delta''$. For each $\gamma \in \Delta$, we know that there exists some $\gamma' \in \Delta'$ such that $\gamma' \in r_a(\gamma)$. Thus, we have that $\gamma \subseteq Res(a,\gamma') \in \Delta''$. For each $\gamma'' \in \Delta''$ there exists some $\gamma' \in \Delta'$ such that $\gamma'' = Res(a,\gamma')$. Since $\gamma' \in \Delta'$ there exists some $\gamma \in \Delta$ such that $\gamma' \in r_a(\gamma)$. Lemma 1 implies that $\gamma \subseteq Res(a,\gamma') = \gamma''$. The three conclusions imply that $\sigma \sqsubseteq \Phi(a,\sigma')$.

Now, consider ω. Because $\sigma' \sqsubseteq \omega$, we have that a is executable in ω. The definition of the function Res and of $\sqsubseteq$ imply that $\Phi(a,\sigma') \sqsubseteq \Phi(a,\omega)$, which proves the conclusion of the lemma. □

Corollary 2. *Let a be a non-sensing action regressable in the p-structure σ such that $\sigma \models$ **Knows**(φ). Then, for each σ' such that $\mathcal{R}(a,\sigma) \trianglerighteq \sigma'$, a is executable in σ' and $\Phi(a,\sigma') \models$ **Knows**(φ).*

Proof: This follows immediately from Lemma 2. □

5.2 Regression with sensing actions

Now it is necessary to define both regressability and regression for sensing actions. The main difference between non-sensing actions and sensing actions is that sensing actions do not change the world while non-sensing actions do. This leads to the following definition.

Definition 12 (Regressable for Sensing Actions). *Let a be a sensing action which determines f. We say*

- *a is regressable in a p-state δ if $\overline{\mathbf{excond}(a)} \cap \delta = \emptyset$.*
- *a is regressable in a set of p-states Δ if it is regressable in every $\delta \in \Delta$ and $\delta \sim_{\{f,\neg f\}} \delta'$ for every pair δ, δ' in Δ.*
- *a is regressable in a p-structure $\sigma = \langle \delta, \Delta \rangle$ if it is regressable in Δ.*

The first condition guarantees that δ can be extended to a p-state in which a is executable. The second condition ensures that if a is regressable in Δ then either (*i*) $\Delta \models$ **Knows**(f); (*ii*) $\Delta \models$ **Knows**$(\neg f)$; or (*iii*) f is foreign to Δ. To continue we need the following notation:

Definition 13. *A p-structure $\sigma = \langle \delta, \Delta \rangle$ agrees with a literal l (written as $\sigma \gg l$) if for every $\delta' \in \Delta$, either $l \in \delta'$ or $\bar{l} \notin \delta'$.*

Definition 14. *If $\sigma \gg l$, $\sigma + l$ denotes the p-structure $\langle \delta \cup \{l\}, \{\delta' \cup \{l\} \mid \delta' \in \Delta\}\rangle$; for a set of p-structures Ω, $\Omega + l = \{\sigma + l \mid \sigma \in \Omega\}$.*

The above definition is extended to a set of fluent literals S in the obvious way. Note that l is known to be true in the p-structure $\sigma + l$. For a set of p-structures Ω and a formula φ, we write $\Omega \models$ **Knows**(φ) to indicate that $\sigma \models$ **Knows**(φ) for each p-structure $\sigma \in \Omega$.

To define the regression of sensing actions, observe that given a p-structure $\sigma = \langle \delta, \Delta \rangle$ and a sensing action a, the execution of a in σ will result in $\sigma' = \langle \delta, \Delta' \rangle$ with $\Delta' \subseteq \Delta$ and every $\delta' \in \Delta'$, $\delta' \sim_{\{f,\neg f\}} \delta$ where **determines**$(a) = f$. Note that in the initial set-up of a planning problem, each of the different possible states of the world become a state s in a c-structure $\langle s, \Sigma \rangle$. The k-set Σ is a set of all the initial possible states. Therefore progression begins with a set of structures and as sensing occurs, the k-sets of each of these c-structures are pruned, representing the increase in knowledge. As regression is intended to reverse the process, it needs to increase the k-sets. This, however, is done only when we need to regress a sensing action in a conditional plan. As such, the regression of a sensing action should start with two set of p-structures, each corresponding to a branch of the plan. Recall that we have required that sensing actions that sense a fluent f only occur immediately prior to a conditional construct that branches on f. For a set of p-structures Ω, the set of all p-states occurring in Ω is denoted by $b(\Omega) = \bigcup_{\langle \gamma, \Gamma \rangle \in \Omega} \Gamma$.

Definition 15 (Regression of Sensing Actions). *Let a be a sensing action which senses f and Ω_1 and Ω_2 be two set of p-structures such that $\Omega_1 \models$ **Knows**(f) and $\Omega_2 \models$ **Knows**$(\neg f)$. The result of regression a in Ω_1 and Ω_2, denoted by $\mathcal{R}(a, \Omega_1, \Omega_2)$, and is defined as follows.*

$$\mathcal{R}(a, \Omega_1, \Omega_2) = \left\{ \begin{array}{l} \langle \delta, \Delta' \rangle + \mathbf{excond}(a) \mid \exists \langle \delta, \Delta \rangle \in \Omega_1 \cup \Omega_2 \text{ and} \\ \qquad a \text{ is regressable in } \langle \delta, \Delta \rangle \\ \qquad \Delta \subseteq \Delta' \subseteq \Delta \cup \{\gamma \mid \gamma \in b(\Omega_1 \cup \Omega_2) \text{ s.t. } \delta \not\sim_{\{f, \neg f\}} \gamma \\ \qquad\qquad\qquad a \text{ is regressable in } \gamma\} \end{array} \right\}$$

Consider the planning problem in which the p-states include a, b, c, and d, where it is the case that $f \in a$, $f \in d$, $\neg f \in b$, and $\neg f \in c$. Now consider the case where we are regressing a sensing action that determines f and $\Omega_1 = \{\langle a, \{a, d\} \rangle\}$ and $\Omega_2 = \{\langle b, \{b, c\} \rangle\}$. Note that $\Omega_1 \models$ **Knows**(f) and $\Omega_2 \models$ **Knows**$(\neg f)$. We need to regress to all possible p-structures that would yield the contents of Ω_1 and Ω_2 through progression of the sensing action. The sensing action may have been vacuous, so we need to include the contents of Ω_1 and Ω_2, along with structures representing all possible ways to weaken the knowledge found in those structures. So, the result needs to be $\langle a, \{a, d\} \rangle$, $\langle a, \{a, d, b\} \rangle$, $\langle a, \{a, d, c\} \rangle$, $\langle a, \{a, d, b, c\} \rangle$, $\langle b, \{b, c\} \rangle$, $\langle b, \{b, c, a\} \rangle$, $\langle b, \{b, c, d\} \rangle$, and $\langle b, \{b, c, a, d\} \rangle$. Note that progression of these structures (and nothing more) through the sensing action that determines f yields precisely the contents of $\Omega_1 \cup \Omega_2$.

5.3 Regression over plans

We next extend $\mathcal{R}$ to define the regression $\widehat{\mathcal{R}}$ on conditional plans.

Definition 16 (Extended Regression Function). *Let Ω be a set of p-structures and c be a plan. We define $\mathcal{R}(c, \emptyset) = \emptyset$ for every c and extend the $\mathcal{R}$ function to $\widehat{\mathcal{R}}$ as follows:*

1. *For $c = []$, $\widehat{\mathcal{R}}([], \Omega) = \Omega$.*

2. *For $c = a; c'$ where c' is a plan and a is a non-sensing action,*

$$\widehat{\mathcal{R}}(c, \Omega) = \bigcup_{\sigma \in \widehat{\mathcal{R}}(c', \Omega)} \mathcal{R}(a, \sigma).$$

3. *For $c = a; [$**if** f **then** c_1 **else** $c_2]$ where c_1 and c_2 are two plans and a is a sensing action with* **determines**$(a) = f$, *let $R_1 = \widehat{\mathcal{R}}(c_1, \Omega)$ and $R_2 = \widehat{\mathcal{R}}(c_2, \Omega)$,*

(a) if $R_1 \cup R_2 = \emptyset$ *then* $\widehat{\mathcal{R}}(c, \Omega) = \emptyset$

(b) otherwise, $\widehat{\mathcal{R}}(c, \Omega) = \mathcal{R}(a, (R_1 + f), (R_2 + \bar{f}))$.

We are now ready to define the notion of regression solution.

Definition 17 (Regression Solution). *Let* $\mathbf{D} = \langle \mathbf{F}, \mathbf{O_{ns}}, \mathbf{O_{se}}, \mathbf{A}, \mathbf{I}, \mathbf{G} \rangle$ *be a planning problem and* $\Omega = \{\langle \mathbf{G}, \{\mathbf{G}\}\rangle\}$*. A conditional plan c is a regression solution for* $\mathbf{D}$ *if (i)* $\widehat{\mathcal{R}}(c, \Omega) \neq \emptyset$ *and (ii) for every initial c-structure* σ*,* $\widehat{\mathcal{R}}(c, \Omega) \trianglerighteq \sigma$*.*

This definition is illustrated in Fig.2. We see that regression begins with the goal p-structure, $\langle \mathbf{G}, \{\mathbf{G}\}\rangle$. The process of regression continues, in general increasing the number of p-structures being regressed. Finally, the regression of the plan c, terminates and yields the set of p-structures $\widehat{\mathcal{R}}(c, \Omega)$. These are represented by the circles filled with dots. For the plan c to be a regression solution, it must be the case that for each initial c-structure $\sigma_0 \in \Omega_0$ (represented by the checkered squares), there must be a $\sigma \in \widehat{\mathcal{R}}(c, \Omega)$ such that $\sigma \sqsubseteq \sigma_0$.

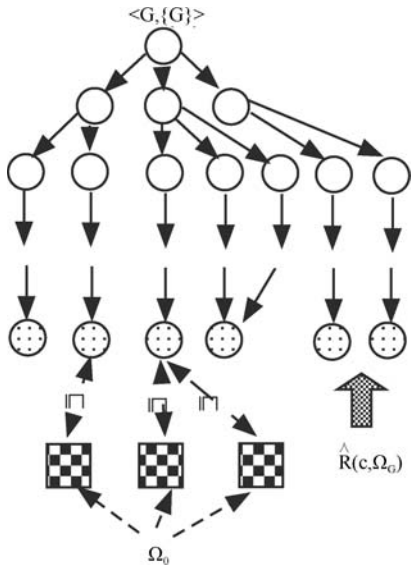


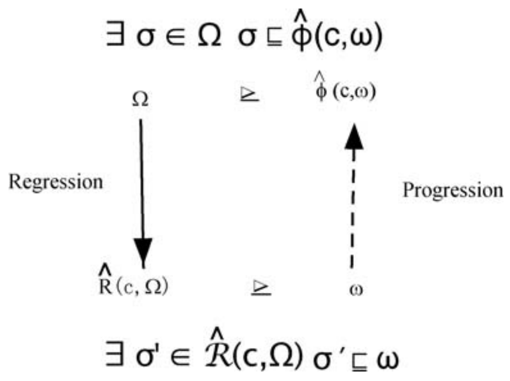
Figure 2. Regression solution

In the following, we will prove the soundness and completeness of our regression formalism.

5.4 Soundness

Soundness is established through a series of lemmata. Our goal is Lemma 4, which is illustrated in Fig.3. Given that a plan c is regressable in Ω and ω is a p-structure such that for some $\sigma' \in \widehat{\mathcal{R}}(c, \Omega)$, $\sigma' \sqsubseteq \omega$, we can conclude that the progression of c starting in ω wil be in the relation $\sigma \sqsubseteq \hat{\Phi}(c, \omega)$ for some $\sigma \in \Omega$. From this soundness follows, since anything known in σ (in particular G, the goal) will be known in $\hat{\Phi}(c, \omega)$. We first prove this property for a sequence of non-sensing actions, and then a conditional plan in Lemma 4.

Lemma 3. *Let* $c = a_1; \ldots; a_k$ *be a sequence of non-sensing actions regressable in the p-structure* σ *and* σ' *be a p-structure such that* $\widehat{\mathcal{R}}(c, \sigma) \trianglerighteq \sigma'$*. Then, c is executable in* σ' *and* $\sigma \sqsubseteq \hat{\Phi}(c, \sigma')$*.*



$\exists \sigma \in \Omega \; \sigma \sqsubseteq \hat{\Phi}(c,\omega)$
Ω
$\trianglerighteq$
$\hat{\Phi}(c,\omega)$
Regression
Progression
$\hat{R}(c,\Omega)$
$\trianglerighteq$
ω
$\exists \sigma' \in \hat{\mathcal{R}}(c,\Omega) \; \sigma' \sqsubseteq \omega$

Figure 3. Illustration of Lemma 4

Proof: By induction on k. The case for $k = 1$ is Lemma 2. Assume that it is correct for $k-1$. We prove that it holds for k.

Let $c' = a_1; \ldots; a_{k-1}$. Because $\widehat{\mathcal{R}}(c,\sigma) \trianglerighteq \sigma'$, we know that there exists some $\omega \in \mathcal{R}(a_k, \sigma)$ such that $\widehat{\mathcal{R}}(c',\omega) \trianglerighteq \sigma'$. The inductive hypothesis implies that c' is executable in σ' and $\omega \sqsubseteq \hat{\Phi}(c',\sigma')$. By Lemma 2, we have that a_k is executable in $\hat{\Phi}(c',\sigma')$ and $\sigma \sqsubseteq \Phi(a_k, \hat{\Phi}(c',\sigma'))$. Thus, c is executable in σ' and $\sigma \sqsubseteq \Phi(a_k, \hat{\Phi}(c',\sigma')) = \hat{\Phi}(c,\sigma')$.$\square$

To illustrate regression of a sequence of non-sensing actions through a set of p-structures, consider the following example:

Example 6. Let's continue with Example 3, let $\sigma_G = \langle \{h\}, \{\{h\}\} \rangle$, and $\Omega = \{\sigma_G\}$. Let $\delta_1 = \{f\}$ and $\delta_2 = \{\neg f, h\}$. We have that $r_a(\{h\}) = \{\delta_1, \delta_2\}$. It is easy to verify that $r_b(\delta_1) = \{f\}$ and $r_b(\delta_2) = \{\neg f\}$. This allows us to conclude that $\widehat{\mathcal{R}}([b;a], \Omega) \neq \emptyset$ and for each initial c-structure σ_0 there exists some $\gamma \in \widehat{\mathcal{R}}([b;a],\Omega)$ such that $\gamma \sqsubseteq \sigma_0$, i.e., $[b;a]$ is indeed a regression solution for $\mathbf{D}_2$.

With the help of Lemma 3, we can prove the soundness of regression over conditional plans. By Definition 1, each conditional plan begins with a sequence of non-sensing actions and this may be followed by a sensing action and a conditional statement. This provides a structure for conditional plans and allows for the proof to be done inductively over the number of **if** statements, denoted by $branch(c)$, occurring in the regression solution. Formally, it is $branch([]) = 0$; $branch(c) = 0 + branch(c')$ if $c = a; c'$ where a is a primitive action and c' is a conditional plan; and $branch(c) = 1 + \max\{branch(c_1), branch(c_2)\}$ if $c = a; [\textbf{ if } f \textbf{ then } c_1 \textbf{ else } c_2]$ where a is a sensing action and c_1 and c_2 are conditional plans.

Lemma 4. *Let c be a plan regressable in Ω and ω be a p-structure such that $\widehat{\mathcal{R}}(c,\Omega) \trianglerighteq \omega$. Then, c is executable in ω and $\Omega \trianglerighteq \hat{\Phi}(c,\omega)$.*

Proof: Inductive on $branch(c)$.

- **Base:** $branch(c) = 0$ implies that c is a sequence of non-sensing actions. This follows from Lemma 3.
- **Inductive Step:** The inductive step requires us to consider the following plan

$$c = \alpha; a; [\textbf{ if } f \textbf{ then } c_1 \textbf{ else } c_2]$$

where α is a sequence of non-sensing actions, a is a sensing action, and c_1 and c_2 are conditional plans with $branch(c_1) < branch(c)$ and $branch(c_2) < branch(c)$.

Let $\widehat{\mathcal{R}}(c_1, \Omega) = \Omega_1$ and $\widehat{\mathcal{R}}(c_2, \Omega) = \Omega_2$. Let

$$\Omega_3 = \mathcal{R}(a, (\Omega_1 + f), (\Omega_2 + \bar{f})).$$

We have that

$$\widehat{\mathcal{R}}(c, \Omega) = \widehat{\mathcal{R}}(\alpha, \Omega_3) = \Omega_c$$

Given that $\widehat{\mathcal{R}}(c, \Omega) \trianglerighteq \omega$, by Lemma 3, we conclude that α is executable in ω and there exists some $\gamma \in \Omega_3$ such that $\gamma \sqsubseteq \hat{\Phi}(\alpha, \omega)$. There are two cases:

- $\gamma \in \Omega_1 + f$. Thus, there exists some $\gamma' \in \Omega_1$ such that $\gamma' + f = \gamma$. By the inductive hypothesis, c_1 is executable in γ and there exists some $\sigma \in \Omega$ such that $\sigma \sqsubseteq \hat{\Phi}(c_1, \gamma')$. This, together with $\gamma \sqsubseteq \hat{\Phi}(\alpha, \omega)$, implies that $\sigma \sqsubseteq \hat{\Phi}(c, \omega)$;
- $\gamma \in \Omega_2 + \bar{f}$. The same argument allows us to conclude that there exists some $\sigma \in \Omega$ such that $\sigma \sqsubseteq \hat{\Phi}(c, \omega)$;

Combining the above two cases, we have the conclusion of the inductive step.

□

This leads to the soundness theorem.

Theorem 1 (Soundness). *For every planning problem* $\mathbf{D} = \langle \mathbf{F}, \mathbf{O_{ns}}, \mathbf{O_{se}}, \mathbf{A}, \mathbf{I}, \mathbf{G} \rangle$*, every regression solution of a planning problem* $\mathbf{D}$ *is a progression solution of* $\mathbf{D}$.

Proof: Let $\Omega_G = \{\langle \mathbf{G}, \{\mathbf{G}\} \rangle\}$. Consider a plan c which is a regression solution for $\mathbf{D}$. Then we know that (i) $\widehat{\mathcal{R}}(c, \Omega_G) \neq \emptyset$ and (ii) for every initial c-structure σ, there exists some p-structure σ' in $\widehat{\mathcal{R}}(c, \Omega)$ such that $\sigma' \sqsubseteq \sigma$. Lemma 4 implies that $\langle \mathbf{G}, \{\mathbf{G}\} \rangle \sqsubseteq \hat{\Phi}(c, \sigma)$. For each such σ, $\hat{\Phi}(c, \sigma) \models \mathbf{Knows}(G)$. Therefore c is a progression solution. □

5.5 Completeness

The next step is to prove completeness of the regression method. It will be helpful to extend the notation $\trianglerighteq$ to two sets of p-structures as follows: We write $\Omega' \trianglerighteq \Omega$, where Ω and Ω' are two sets of p-structures, to indicate that for every $\sigma \in \Omega$ there exists a $\sigma' \in \Omega'$ such that $\sigma' \sqsubseteq \sigma$.

We prove completeness of the regression method through a series of lemmata, beginning with sequences of actions and continuing to conditional plans. The final step is Lemma 10, which is illustrated in Fig.4. It is given that c is a progression solution for an initial set of c-structures Ω, and Ω' is a set of p-structures such that $\Omega' \trianglerighteq \hat{\Phi}(c, \Omega)$. Then, we can conclude that c is regressable in Ω' and $\widehat{\mathcal{R}}(c, \Omega') \trianglerighteq \Omega$. Completeness follows from this property since everything known in Ω' (in particular G) is also known in $\hat{\Phi}(c, \Omega)$.

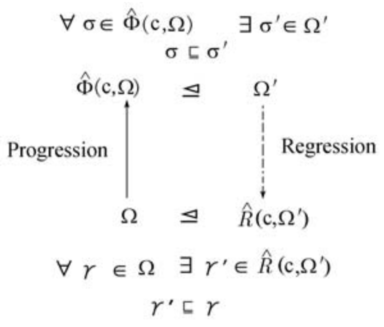

$\forall \sigma \in \hat{\Phi}(c,\Omega)$ $\exists \sigma' \in \Omega'$
$\sigma \sqsubseteq \sigma'$
$\hat{\Phi}(c,\Omega)$ $\trianglelefteq$ Ω'
Progression
Regression
Ω $\trianglelefteq$ $\hat{R}(c,\Omega')$
$\forall \gamma \in \Omega$ $\exists \gamma' \in \hat{R}(c,\Omega')$
$\gamma' \sqsubseteq \gamma$

Figure 4. Illustration of Lemma 10

The first lemma relates progression and regression with respect to the execution of an action in a state.

Lemma 5. *Let s be a state and a be a non-sensing action executable in s. Then, a is regressable in every $\delta \subseteq Res(a, s)$ and there exists some $\delta' \in r_a(\delta)$ such that $\delta' \subseteq s$.*

Proof: The fact that a is executable in s implies that $\mathbf{excond}(a) \subseteq s$. Consider two cases:

- There exists an effect p of a such that $s \models \mathbf{condition}(p)$. This means that $\mathbf{condition}(p) \subseteq s$ and $e_a(s) = \mathbf{head}(p)$. Thus, $Res(a, s) = (s \setminus \overline{\mathbf{head}(p)}) \cup \mathbf{head}(p)$. We will show that a is regressable via p in δ. Obviously,
  - $\overline{\mathbf{head}(p)} \cap \delta = \emptyset$,
  - $\overline{\mathbf{condition}(p)} \cap \delta \subseteq \overline{\mathbf{condition}(p)} \cap Res(a, s) \subseteq \mathbf{head}(p)$, and
  - $\overline{\mathbf{excond}(a)} \cap \delta \subseteq \overline{\mathbf{excond}(a)} \cap Res(a, s) \subseteq \mathbf{head}(p)$.

  Hence, the first condition in Definition 8 allows us to conclude that a is regressable via p in δ and Definition 9 allows us to conclude that $Reg(a, p, \delta) \in r^1_a(\delta)$ and $Reg(a, p, \delta) \subseteq s$.
- There exists no effect p of a such that $s \models \mathbf{condition}(p)$. Hence, $Res(a, s) = s$. It is easy to see that $s \in r^2_a(\delta)$.

In both cases, we have that $r_a(\delta) \neq \emptyset$ and there exists some $\delta' \in r_a(\delta)$ such that $\delta' \subseteq s$. □

The next lemma relates progression and regression with respect to the execution of an action in a p-structure.

Lemma 6. *Let $\gamma = \langle s, \Sigma \rangle$ be a c-structure and a be a non-sensing action executable in γ. Then, a is regressable in every $\gamma' \sqsubseteq \Phi(a, \gamma)$ and there exists some $\gamma'' \in \mathcal{R}(a, \gamma')$ such that $\gamma'' \sqsubseteq \gamma$.*

Proof: Assume that $\gamma' = \langle \delta, \Delta \rangle$. We have that for each $\rho \in \Delta$ there exists some $v \in \Sigma$ such that a is executable in v and $\rho \subseteq Res(a, v)$. Lemma 5 implies that a is regressable in ρ for every $\rho \in \Delta$. This means that a is regressable in Δ, and hence, in γ'.

Consider a $u \in \Sigma$, by definition of $\sqsubseteq$, there exists some $\rho \in \Delta$ such that $\rho \subseteq Res(a, u)$. Again, Lemma 5 implies that there exists some $\delta' \in r_a(\rho)$ such that $\delta' \subseteq u$. This allows us to construct $\gamma'' \in \mathcal{R}(a, \gamma')$ such that $\gamma'' \sqsubseteq \gamma$. $\square$

The above result is extended to a sequence of non-sensing actions as follows.

Lemma 7. *Let $\gamma = \langle s, \Sigma \rangle$ be a c-structure and $a_1; \ldots; a_k$ be a sequence of non-sensing actions executable in γ. Then, a is regressable in every $\gamma' \sqsubseteq \hat{\Phi}(a, \gamma)$ and there exists some $\gamma'' \in \widehat{\mathcal{R}}(a_1; \ldots; a_k, \gamma')$ such that $\gamma'' \sqsubseteq \gamma$.*

Proof: Inductive on k. The base case follows from Lemma 6. We now prove the inductive step. Assume that the lemma is correct for $k-1$, $k \geqslant 1$. We prove it for k. Let $\gamma_0 = \Phi(a_1, \gamma)$. We have that $a_2; \ldots; a_k$ is executable in γ_0 and there exists some $\theta \in \widehat{\mathcal{R}}(a_2; \ldots, a_k, \gamma')$, where $\gamma' = \hat{\Phi}(a_2; \ldots; a_k, \gamma_0)$, such that $\theta \sqsubseteq \gamma_0$. Applying the base case to γ, γ_0, and θ yields the conclusion of the lemma. $\square$

The next lemma proves the above result for sequences of non-sensing actions over a set of c-structures.

Lemma 8. *Let Ω be a set of c-structures and c be a sequence of non-sensing actions executable in Ω. Let Ω' be a set of p-structures such that $\Omega' \trianglerighteq \hat{\Phi}(c, \Omega)$. Then, c is regressable in Ω' and $\widehat{\mathcal{R}}(c, \Omega') \trianglerighteq \Omega$.*

Proof: Consider an arbitrary $\sigma \in \Omega'$, Lemma 7 implies that c is regressable in σ. Thus, $\widehat{\mathcal{R}}(c, \Omega') \neq \emptyset$, i.e., c is regressable in Ω'.

Furthermore, for each $\gamma \in \Omega$, there exists some $\sigma \in \Omega'$ such that $\sigma \sqsubseteq \hat{\Phi}(c, \gamma)$. Again, Lemma 7 implies that there exists some $\gamma' \in \widehat{\mathcal{R}}(c, \sigma) \subseteq \widehat{\mathcal{R}}(c, \Omega')$ such that $\gamma' \sqsubseteq \gamma$. $\square$

To complete the proof of completeness, we need a lemma concerning the relationship between the progression and regression of a sensing action. Before doing so, let us introduce the notion of a *full* set of p-structures. We say that a set of p-structures Ω is *full* if $\{\delta \mid \exists \langle \delta, \Delta \rangle \in \Omega\} = \bigcup_{\langle \delta, \Delta \rangle \in \Omega} \Delta$. This means that for each p-state γ occurring in Ω there exists some p-structure $\langle \gamma, \Delta \rangle$ in Ω. It is clear that if an action a is executable in Ω, and Ω is *full*, then $\Phi(a, \Omega)$ is also *full*. Additionally, we extend our notation $\Gamma' \trianglerighteq \Gamma$, where Γ and Γ' are sets of p-structures to indicate that (i) for each $\gamma \in \Gamma$ there exists some $\gamma' \in \Gamma'$ such that $\gamma' \subseteq \gamma$; and (iii) for each $\gamma' \in \Gamma'$ there exists some $\gamma \in \Gamma$ such that $\gamma' \subseteq \gamma$. We now prove the following lemma:

Lemma 9. *Let a be a sensing action with* **determines** $(a) = f$ *and Ω a* full *set of p-structures such that f is not foreign in Ω. Assume that a is executable in Ω and $\Phi(a, \Omega) = \Omega_p \cup \Omega_n$ where $\Omega_p \models$* **Knows**(f) *and $\Omega_n \models$* **Knows**$(\bar{f})$. *Let Ω'_p and Ω'_n be two sets of p-structures such that $\Omega'_p \trianglerighteq \Omega_p$ and $\Omega'_n \trianglerighteq \Omega_n$. Then, $\mathcal{R}(a, \Omega'_p + f, \Omega'_n + \bar{f}) \trianglerighteq \Omega$.*

Proof: Let $\Omega_1 = \Omega'_p + f$ and $\Omega_2 = \Omega'_n + \bar{f}$. We have that $\Omega_1 \models$ **Knows**(f) and $\Omega_2 \models$ **Knows**$(\bar{f})$. Thus,

$$\mathcal{R}(a, \Omega_1, \Omega_2) = \left\{ \begin{array}{l} \langle \delta, \Delta' \rangle + \mathbf{excond}(a) \mid \exists \langle \delta, \Delta \rangle \in \Omega_1 \cup \Omega_2 \text{ and} \\ \qquad a \text{ is regressable in } \langle \delta, \Delta \rangle \\ \qquad \Delta \subseteq \Delta' \subseteq \Delta \cup \{\gamma \mid \gamma \in b(\Omega_1 \cup \Omega_2) \text{ s.t. } \delta \not\sim_{\{f, \neg f\}} \gamma \\ \qquad\qquad a \text{ is regressable in } \gamma\} \end{array} \right\}$$

First, we observe that $b(\Omega_1 \cup \Omega_2) \trianglerighteq b(\Omega_p \cup \Omega_n)$ since $\Omega'_p \trianglerighteq \Omega_p$ and $\Omega'_n \trianglerighteq \Omega_n$. Now, consider some $\sigma = \langle \delta, \Delta \rangle \in \Omega$. Without loss of generality, we can assume that $\delta \models f$. We know that there exists $\Delta_p \models \mathbf{Knows}(f)$ and $\Delta_n \models \mathbf{Knows}(\bar{f})$ such that $\Delta = \Delta_p \cup \Delta_n$. The fact that a is executable in Ω and Ω is *full* implies that a is executable in Δ. Thus, $\Phi(a, \sigma) = \langle \delta, \Delta_p \rangle \in \Omega_p$.

Because $\Omega'_p \trianglerighteq \Omega_p$, there exists some $\sigma' \in \Omega'_p$ such that $\sigma' = \langle \delta', \Delta'_p \rangle$ and $\langle \delta', \Delta'_p \rangle \sqsubseteq \langle \delta, \Delta_p \rangle$. This implies that for every $\gamma' \in \Delta'_p$ there exists some $\gamma \in \Delta_p$ such that $\gamma' \subseteq \gamma$. Since a is executable in Ω, for every γ in $b(\Omega_1 \cup \Omega_2)$, $\gamma \models \overline{\mathbf{excond}(a).}$ Therefore, since $b(\Omega_1 \cup \Omega_2) \trianglerighteq b(\Omega_p \cup \Omega_n)$, for every γ' in $b(\Omega_p \cup \Omega_n)$, $\gamma' \cap \overline{\mathbf{excond}(a)} = \emptyset$. This means that a is regressable in every $\gamma' \in \Delta'_p$. Thus, a is regressable in $\langle \delta', \Delta'_p \rangle + f$.

Clearly, there exists a $\theta \in \mathcal{R}(a, \Omega'_p + f, \Omega'_n + \bar{f})$ such that $\theta \sqsubseteq \langle \delta, \Delta_p \rangle \in \Omega_p$, namely $\langle \delta', \Delta'_p \rangle + f + \mathbf{excond}(a)$. Note that since a is executable in Ω, the addition of $\mathbf{excond}(a)$ does not affect the $\sqsubseteq$ relation. Similarly, since $\Delta_p \models f$, the addition of f does not affect the $\sqsubseteq$ relation.

Note that $\sigma = \langle \delta, \Delta \rangle = \langle \Delta_p \cup \Delta_n \rangle$. Let us now consider the set Δ_n. Since $b(\Omega_1 \cup \Omega_2) \trianglerighteq b(\Omega_p \cup \Omega_n)$, we know that there exsits some set $\Delta'_n \in b(\Omega_1 \cup \Omega_2)$, such that $\Delta'_n \sqsubseteq \Delta_n$. Note that since a is executable in Δ_n, the addition of $\mathbf{excond}(a)$ to Δ'_n does not affect the $\sqsubseteq$ relation. Similarly, since $\Delta_n \models \neg f$, the addition of $\neg f$ to Δ'_n does not affect the $\sqsubseteq$ relation. Therefore $\Delta'_n + \mathbf{excond}(a) + \neg f \sqsubseteq \Delta_n$.

By definition, $\langle \delta', \Delta'_p + f \cup \Delta'_n + \neg f \rangle + \mathbf{excond}(a) \in \mathcal{R}(a, \Omega_1, \Omega_2)$. The arguments in the paragraph above together with the fact that $\langle \delta', \Delta'_p \rangle \sqsubseteq \langle \delta, \Delta_p \rangle$, allow us to conclude that $\theta' = \langle \delta', \Delta'_p + f \cup \Delta'_n + \neg f \rangle + \mathbf{excond}(a) \sqsubseteq \langle \delta, \Delta \rangle = \sigma$. Since we have shown that for every $\sigma \in \Omega$, there exists some $\theta' \in \mathcal{R}(a, \Omega'_p + f, \Omega'_n + \bar{f})$ such that $\theta' \sqsubseteq \sigma$, the lemma is proven. □

The next lemma combines the above results to consider a conditional plan with an arbitrary level of branching.

Lemma 10. *Let Ω be a* full *set of c-structures and c be a conditional plan executable in Ω. Let Ω' be a set of p-structures such that $\Omega' \trianglerighteq \hat{\Phi}(c, \Omega)$. Then, c is regressable in Ω' and $\widehat{\mathcal{R}}(c, \Omega') \trianglerighteq \Omega$.*

Proof: Inductive on *branch*(c). Base case follows from Lemma 8. The inductive step requires us to consider the following plan

$$c = \alpha; a; [\ \mathbf{if}\ \ f\ \mathbf{then}\ \ c_1\ \mathbf{else}\ \ c_2]$$

where α is a sequence of non-sensing actions, a is a sensing action, and c_1 and c_2 are conditional plans with *branch*$(c_1) <$ *branch*(c) and *branch*$(c_2) <$ *branch*(c).

Let $c' = \alpha; a$ and $d = a; [\ \mathbf{if}\ \ f\ \mathbf{then}\ \ c_1\ \mathbf{else}\ \ c_2]$.

Since c is executable in Ω, c' must be executable in each c-structure in Ω. Let $\Omega_* = \hat{\Phi}(c', \Omega)$. We know that for each $\sigma_* \in \Omega_*$ either $\sigma_* \models \mathbf{Knows}(f)$ or $\sigma_* \models \mathbf{Knows}(\bar{f})$. Group the $\sigma_* \in \Omega_*$ into two groups Ω_p and Ω_n such that $\Omega_p \models \mathbf{Knows}(f)$ and $\Omega_n \models \mathbf{Knows}(\bar{f})$, and $\Omega_* = \Omega_n \cup \Omega_p$.

Clearly, c_1 is executable in Ω_p and c_2 is executable in Ω_n. Therefore, there exists $\Omega'_p \subseteq \Omega'$ and $\Omega'_n \subseteq \Omega'$ such that $\Omega'_p \trianglerighteq \hat{\Phi}(c_1, \Omega_p)$ and $\Omega'_n \trianglerighteq \hat{\Phi}(c_2, \Omega_n)$. By the inductive hypothesis, we have that $\widehat{\mathcal{R}}(c_1, \Omega'_p) \neq \emptyset$ and $\widehat{\mathcal{R}}(c_2, \Omega'_n) \neq \emptyset$. Thus, $\widehat{\mathcal{R}}(c_1, \Omega') \neq \emptyset$ and $\widehat{\mathcal{R}}(c_2, \Omega') \neq \emptyset$. Hence by the inductive hypothesis, $\widehat{\mathcal{R}}(c_1, \Omega'_p) \trianglerighteq \Omega_p$ and $\widehat{\mathcal{R}}(c_2, \Omega'_p) \trianglerighteq \Omega_n$.

Since c is executable in Ω, α must be executable in each c-structure in Ω. Note that by assumption Ω is *full*, and therefore $\hat{\Phi}(\alpha, \Omega)$ is also full. Then by Lemma 9, it follows that $\mathcal{R}(a, \Omega'_p + f, \Omega'_n + \bar{f}) \trianglerighteq \phi(\alpha, \Omega)$. Since $\widehat{\mathcal{R}}(d, \Omega') \supseteq \mathcal{R}(a, \widehat{\mathcal{R}}(c_1, \Omega'_p), \widehat{\mathcal{R}}(c_2, \Omega'_n))$, we conclude that $\widehat{\mathcal{R}}(d, \Omega') \trianglerighteq \hat{\Phi}(\alpha, \Omega)$. Lemma 8 then allows us to conclude that $\widehat{\mathcal{R}}(c, \Omega') \trianglerighteq \Omega$. This proves the lemma. □

In the next theorem, we prove the completeness of regression.

Theorem 2 (Completeness). *For every planning problem* $\mathbf{P} = \langle \mathbf{F}, \mathbf{O_{ns}}, \mathbf{O_{se}}, \mathbf{A}, \mathbf{I}, \mathbf{G} \rangle$*, every progression solution of* $\mathbf{D}$ *is a regression solution of* $\mathbf{D}$.

Proof: Let Ω_0 be the set of initial c-structures and c be a progression solution of $\mathbf{D}$. Note that by constrution Ω_0 is *full*. We have that $\hat{\Phi}(c, \Omega_0) \models \mathbf{Knows}(\mathbf{G})$. Let $\Omega_G = \{\langle \mathbf{G}, \{\mathbf{G}\} \rangle\}$. Lemma 10 indicates that c is regressable in Ω_G since $\Omega_G \trianglerighteq \hat{\Phi}(c, \Omega_0)$. Furthermore, we can conclude that $\widehat{\mathcal{R}}(c, \Omega_G) \trianglerighteq \Omega_0$. Therefore for each $\sigma_0 \in \Omega_0$, there exists some $\gamma \in \widehat{\mathcal{R}}(c, \Omega_G)$ such that $\gamma \sqsubseteq \sigma_0$, which proves that c is indeed also a regression solution. □

5.6 Example continued

We conclude with the continuation of our running example; the medical domain $\mathbf{D}_1$. Let

$$\sigma_G = \{\neg\textsc{Dead}, \neg\textsc{Infected}\},$$

and

$$c = \textsc{stain}; \textsc{inspect}; [\textbf{if } \textsc{Blue} \textbf{ then } \textsc{medicate} \textbf{ else } \textsc{noOp}].$$

For $\Omega = \{\langle \sigma_G, \{\sigma_G\} \rangle\}$, we want to compute:

$$\widehat{\mathcal{R}}(c, \Omega)$$

Let

$$\begin{aligned}
\delta_1 &= \{\neg\textsc{Dead}, \textsc{Hydrated}, \textsc{Blue}\},\\
\delta_2 &= \{\neg\textsc{Dead}, \neg\textsc{Infected}, \neg\textsc{Blue}\}, \text{ and}\\
\delta_3 &= \{\neg\textsc{Dead}, \textsc{Infected}, \textsc{Hydrated}, \neg\textsc{Blue}\}.
\end{aligned}$$

The initial computation is as follows.

- $\textsc{medicate}$ is regressable via the effect $p_1 = \textsc{Hydrated} \Rightarrow \neg\textsc{Infected}$ in G and is not regressable via the effect $p_2 = \neg\textsc{Hydrated} \Rightarrow \textsc{Dead}$ in G. Furthermore,

  $$r^1_{\textsc{medicate}}(\sigma_G) = \{Reg(\textsc{medicate}, p_1, \sigma_G)\} = \{\{\neg\textsc{Dead}, \textsc{hydrated}\}\}$$

  and $r^2_{\textsc{medicate}}(\sigma_G) = \emptyset$. This gives us

  $$\mathcal{R}(\textsc{medicate}, \Omega) = \{\langle \{\neg\textsc{Dead}, \textsc{Hydrated}\}, \{\{\neg\textsc{Dead}, \textsc{hydrated}\}\} \rangle\}$$

- Let $c_1 = \textsc{inspect}; [\textbf{ if } \textsc{Blue} \textbf{ then } \textsc{medicate} \textbf{ else } \textsc{NoOp}]$.
  We have $\mathcal{R}(\textsc{medicate}, \Omega) + \textsc{Blue} = \{\langle \delta_1, \{\delta_1\} \rangle\}$ and $\mathcal{R}(\textsc{NoOp}, \Omega) + \neg\textsc{Blue} = \{\langle \delta_2, \{\delta_2\} \rangle\}$. Then,

  $$\widehat{\mathcal{R}}(c_1, \Omega) = \mathcal{R}(\textsc{inspect}, \{\langle \delta_1, \{\delta_1\} \rangle\}, \{\langle \delta_2, \{\delta_2\} \rangle\})$$

Therefore, the first step creates the following formula to be regressed further:

$$\widehat{\mathcal{R}}([\textsc{stain}], \mathcal{R}(\textsc{inspect}, \{\langle \delta_1, \{\delta_1\} \rangle\}, \{\langle \delta_2, \{\delta_2\} \rangle\})) \tag{3}$$

The next step involves regression of the sensing action. We have

$$\widehat{\mathcal{R}}(\text{INSPECT}, \{\langle\delta_1, \{\delta_1\}\rangle\}, \{\langle\delta_2, \{\delta_2\}\rangle\}) = \left\{ \begin{array}{c} \langle\delta_1, \{\delta_1\}\rangle, \langle\delta_2, \{\delta_2\}\rangle, \\ \langle\delta_1, \{\delta_1, \delta_2\}\rangle, \langle\delta_2, \{\delta_1, \delta_2\}\rangle \end{array} \right\} = \Omega_1$$

This follows from the second item in Definition 16. Sentence (3) regresses to (4).

$$\widehat{\mathcal{R}}([\text{STAIN}], \Omega_1) \tag{4}$$

Now there is one action left. Notice that STAIN is regressable in both δ_1 and δ_2. It is regressable via $q = \text{INFECTED} \Rightarrow \text{BLUE}$ in δ_1 and via the second condition (Definition 8) in δ_2. We have that $Reg(\text{STAIN}, q, \delta_1) = \{\neg\text{DEAD}, \text{INFECTED}, \text{HYDRATED}, \neg\text{BLUE}\} = \delta_3$. Thus:

$$r^1_{\text{STAIN}}(\delta_1) = \{\delta_3\} \quad \text{and} \quad r^2_{\text{STAIN}}(\delta_1) = \emptyset$$

and

$$r^1_{\text{STAIN}}(\delta_2) = \emptyset \quad \text{and} \quad r^2_{\text{STAIN}}(\delta_2) = \{\delta_2\}.$$

This gives us

$$\widehat{\mathcal{R}}(c, \Omega) = \{\langle\delta_3, \{\delta_3\}\rangle, \langle\delta_2, \{\delta_2\}\rangle, \langle\delta_3, \{\delta_3, \delta_2\}\rangle, \langle\delta_2, \{\delta_3, \delta_2\}\rangle\}$$

So finally sentence (4) regresses to:

$$\widehat{\mathcal{R}}([], \{\langle\delta_3, \{\delta_3\}\rangle, \langle\delta_2, \{\delta_2\}\rangle, \langle\delta_3, \{\delta_3, \delta_2\}\rangle, \langle\delta_2, \{\delta_3, \delta_2\}\rangle) \tag{5}$$

We now have the following four structures for testing whether we have a regression solution:

$$\begin{aligned} \omega_1 &= \langle\delta_3, \{\delta_3, \delta_2\}\rangle \\ \omega_2 &= \langle\delta_2, \{\delta_2, \delta_3\}\rangle \\ \omega_3 &= \langle\delta_3, \{\delta_3\}\rangle \\ \omega_4 &= \langle\delta_2, \{\delta_2\}\rangle \end{aligned}$$

Since $\omega_1 \sqsubseteq \langle s_{11}, \{s_{11}, s_1, s_5\}\rangle$, $\omega_2 \sqsubseteq \langle s_5, \{s_{11}, s_1, s_5\}\rangle$, and $\omega_2 \sqsubseteq \langle s_1, \{s_{11}, s_1, s_5\}\rangle$, c is a regression solution for $\mathbf{D}$.

6 Related Work

In recent years, there has been increasing attention given to the importance of sensing actions and modeling planning agents with incomplete knowledge[3, 12, 18]. The approach of Graphplan has been extended to handle knowledge and sensing actions[24]. The situation calculus and the closely related fluent calculus have been extended to include knowledge and sensing[17, 19, 13, 20]. Additionally, planning methods have been developed within the situation calculus[5, 4]. A number of agent control languages with sensing and knowledge as well as various programming language constructs have been proposed[7, 17, 13, 21].

Regression as a method for the verification and construction of plans has a long history[9, 16, 8, 11]. Within the situation calculus, regression has been used to verify the correctness of plans with knowledge, sensing, and conditionals[17, 13]. Regression also forms the basis for the agent programming language GoLog[7, 13]. These approaches regress formulas of the situation calculus.

Regression over states, on the other hand, has been used in a number of successful methods within the automated planning area[2]. The language used in this paper is based upon that developed in Ref.[18]. The latter work proposes a progression method for such plans and also a number of approximations of knowledge. In Ref.[23], Tuan, Baral and Son assume the 0-approximation of knowledge and develop a method of regression for plans and prove it complete with respect to the progression operator. Here regression works by regressing over states as contrasted with Ref.[17] which uses regression over formulas.

It should be noted that the 0-approximation of knowledge is not complete. For example, the medical domain used in this paper could not be formulated with the 0-approximation of knowledge. In this paper, building upon Ref.[23], a regression method on states is developed for full knowledge. Needless to say, this has to be done at a cost of greater complexity[1].

Herzig, Lang and Marquis[25] have developed a very general framework for planning in partially observable domains with knowledge and sensing actions. They represent sets of states with the strongest possible epistemic atom $\mathbf{K}(\varphi)$ that is satisfied by the set of states. The framework is designed to be compatable with a wide variety of action formalisms. They propose a method of regression which is defined in terms of the progression operator. So, regression is not constructive, but rather the regressed formula needs to be guessed and then proven correct with the progression operator. They characterize their regression method for non-sensing actions as abductive. Their regression of sensing actions is similar to that found in Ref.[17] as they are regressing formulas. But note that the formulas in Ref.[17] are those of a 1'st order language that explicitly talks of possible worlds and accessibility relations.

In the field of automated planning, Rintanen's work in Refs.[14, 15] has many relatively superficial differences with our approach, but upon deeper analysis it can be seen to be strongly related to our work. It proposes different algorithms for backward searching for a plan in partially observable domains, i.e., regression algorithms for planning.

Rintanen does not have a concrete textual representation of plans and actions as we do here (and in languages such as GoLog and Flux). In his work, plans are directed acyclic graphs, whose nodes represent belief states, while they are sequences of actions with **if-then** constructs in our notation. Sensing actions are not explicitly mentioned, but are represented through observation classes. Therefore, there is no notion of sensing actions with preconditions. For example, if a domain has a single sensing action which determines f, then the state space is divided into two observation classes; one class consists of all states in which f is true and the other consists of all states in which f is false. Non-sensing actions are represented by operators with executability conditions and conditional effects. For example, an operator given by the pair $\langle f, g_1 \wedge g_2 \lhd \neg l \rangle$ represents an action whose executability condition is f and whose effect is $\{g_1, g_2\} \Rightarrow \{\neg l\}$ in our notation. Rintanen also considers non-deterministic actions but does not consider sensing actions with preconditions.

In each iteration of his regression algorithm, the set of current belief states is put through two steps of computation. In the first step, regression is executed over operators (or non-sensing actions). Given a belief state B, the regression of an operator o is defined by its strong preimage and is the maximal set of states from which a state in B is always reached by the execution of o. Rintanen uses a BDD formula based representation of belief states. Although the formulation is stated in terms of belief states, Rintanen's regression is formula based and is similar to the formulation found in Refs.[17, 18]. He developed algorithms for computing the strong preimages of a belief state when it (the belief state) is represented as a formula.

The second step in the regression algorithm is done by an operator $\otimes$ which takes a set of belief states and creates a new set of belief states. Intuitively, the result of the operator $\otimes$ over two belief states B_1 and B_2 are maximal subsets of $B_1 \cup B_2$ from which either B_1 or B_2 could be selected based on the observations that can be made. This is similar to our formulation of regression with sensing actions and appears to be related to that of Ref.[25] as well. But in his framework, the operator $\otimes$ needs to be executed in every iteration of the search algorithm, while in our framework, the equivalent computation is done only when a regression over a sensing action is made.

Rintannen[14, 15] has impressive results concerning the performance of his approach in planning. His use of BDD formulas to represent states might yield an advantage (in general or in certain circumstances) compared to the representation used in this paper as an implementation technique. On the other hand, our constructive regression method may (in general or in certain circumstances) perform better. The thesis of Ref.[22] contains some experimental comparisons between the planner presented in Ref.[14] and the implementation of the state-based regression given in Ref.[23] which show that state-based regression formulation could be implemented in a conditional planner competitive with YAK, the planner discussed in Ref.[14].

The question of which method yields a better implementation is a very different topic from the foundational contribution of this paper. It is something that we regard as unsettled and is a topic for future research and a different paper. Some preliminary work on the properties of an implementation based on our approach may be found in Ref.[22].

The literature survey reveals underlying similarity between the diverse approaches to the regression of conditional plans with knowledge and sensing. We find the following methods for representing the objects of regression:

- Formulas of 1'st order logic that describe possible worlds and their accessibility relations[17, 13].
- Modal formulas reduced to a single level of nesting[25].
- BDD formulas[14, 25].
- States[23].

This paper fills out the last method, providing a directly defined state-based regression approach for full knowledge.

7 Summary and Future Work

This paper develops a constructive state-based regression method for planning domains with sensing operators and a full representation of the knowledge of the planning agent. The language includes primitive actions, sensing actions, and conditional plans. We prove the soundness and completeness of the regression formulation with respect to the definition of progression and the semantics of a modal logic of knowledge. This work can serve as a basis for future work on regression based planning. It is in this context where comparisons with related approaches will be most instructive. Our future work with implementations of our approach will compare its efficiency for both plan verification and planning with other methods.

Acknowledgements

Chitta Baral and Richard Scherl acknowledge support from DTO's AQUAINT program under award number N61339-06-C-0143. Richard acknowledges additional support from the Knowledge Fusion Center of the Army Research Laboratory under contract number DAAD-03-2-0034. Chitta acknowledges additional support from NSF under grant number 0412000 and from ONR-MURI number N00014-07-1-1049. Son Tran acknowledges support from NSF grants IIS-0812267, EIA-0220590, and CNS-0454066. We also thank the anonymous reviewers for their helpful suggestions.

References

[1] Baral C, Kreinovich V, Trejo R. Computational complexity of planning and approximate planning in the presence of incompleteness. Artificial Intelligence, 2000, 122(1-2): 241–267.

[2] Bonet B, Geffner H. Planning as heuristic search. Artificial Intelligence, 2001, 129: 5–33.

[3] Etzioni O, Hanks S, Weld D, Draper D, Lesh N, Williamson M. An approach to planning with incomplete information. In: Proc. of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR92). 1992. 115–125.

[4] Lakemeyer G. On sensing and off-line interpreting in GOLOG. Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter, Spr., 1999. 173–189.

[5] Lespérance Y. An approach to the synthesis of plans with perception acts and conditionals. In: Working Notes of the Canadian Workshop on Distributed AI, 1994.

[6] Levesque H. What is planning in the presence of sensing? In: Proc. of the Thirteenth National Conference on Artificial Intelligence, 1139–1146, 1996.

[7] Levesque H, Reiter R, Lespérance Y, Lin F, Scherl R. GOLOG: A logic programming language for dynamic domains. Journal of Logic Programming, 1997, 31: 59–83.

[8] Manna Z, Waldinger R. How to clear a block: A theory of plans. Journal of Automated Reasoning, 1987, 3: 343–377.

[9] McDermott D. Regression planning. International Journal of Intelligent Systems, 1991, 6: 356–416.

[10] Nguyen XL, Kambhampati S, Nigenda RS. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. Artificial Intelligence, 2002, 135: 73–123.

[11] Pednault E. Toward a Mathematical Theory of Plan Synthesis [Ph.D. Thesis]. Stanford University, 1986.

[12] Pryor L, Collins G. Planning for contingencies: a decision based approach. Journal of AI Research, 1996.

[13] Reiter R. Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press, Cambridge, Massachusetts, 2001.

[14] Rintanen J. Backward plan construction for planning with partial observability. In: Proc. of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS02). 2002. 173–182.

[15] Rintanen J. Conditional Planning in the discrete belief space. In: Proc. of the 19th International Joint Conference on Artificial Intelligence. 2005. 1260–1265.

[16] Rosenschein S. Plan synthesis: A logical perspective. In: Proc. of the International Joint Conference on Artificial Intelligence. 1981. 331–337.

[17] Scherl R, Levesque H. Knowledge, action, and the frame problem. Artificial Intelligence, 2003, 144: 1–39.

[18] Son TC, Baral C. Formalizing sensing actions – a transition function based approach. Artificial Intelligence, 2001, 125: 19–91.

[19] Thielscher M. Representing the Knowledge of a Robot. In: Proc. of the International Conference on Principles of Knowledge Representation and Reasoning (KR). 2000. 109–120.

[20] Thielscher M. Inferring Implicit State Knowledge and Plans with Sensing Actions. In: Proc. of the German Annual Conference on Artificial Intelligence, 2001.

[21] Thielscher M. FLUX: A logic programming method for reasoning agents. Theory and Practice of Logic Programming, 2005, 5(4-5): 533–565.

[22] Tuan L. Regression in the presence of incomplete information and sensing actions, and its application to conditional planning [Ph.D. Thesis]. Arizona State University, 2004.

[23] Tuan LC, Baral C, Son TC. A state-based regression formulation for domains with sensing actions and incomplete information. Logical methods in Computer Science, 2006, 2(4).

[24] Weld D, Anderson C, Smith D. Extending graphplan to handle uncertainty & sensing actions. In: Proc. of the Fifteenth National Conference on Artificial Intelligence, 1998.

[25] Herzig A, Lang J, Marquis P. Action representation and partially observable planning in epistemic logic. In: Proc. of the Eighteenth International Conference on Artificial Intelligence, 2003.