A Logic Programming Approach to Regression Based Repair of Incorrect Initial Belief States

Loc Pham[1], Enrico Pontelli[1], Fabio Tardivo[1], and Tran Cao Son[1]

New Mexico State University, Las Cruces NM 88003, USA
{lpham,epontelli,ftardivo,tson}@cs.nmsu.edu

Abstract. This paper explores the challenge of encountering incorrect beliefs in the context of reasoning about actions and changes using action languages with sensing actions. An incorrect belief occurs when some observations conflict with the agent's own beliefs. A common approach to recover from this situation is to replace the initial beliefs with beliefs that conform to the sequence of actions and the observations. The paper introduces a regression-based and revision-based approach to calculate a correct initial belief. Starting from an inconsistent history consisting of actions and observations, the proposed framework **(1)** computes the initial belief states that support the actions and observations and **(2)** uses a belief revision operator to repair the false initial belief state. The framework operates on domains with static causal laws, supports arbitrary sequences of actions, and integrates belief revision methods to select a meaningful initial belief state among possible alternatives.

Keywords: Regression · Action Languages · Incorrect Beliefs · Prolog.

1 Introduction

In reasoning about actions and change, sensing actions have been considered as the mean for agents to *refine* their knowledge in presence of uncertainty and/or incomplete knowledge. In these formalisms, a sensing action helps an agent to determine the truth value of an *unknown* fluent. For example, the action *look* helps the agent to determine whether the light in the kitchen is *on* or off ($\neg on$). If the agent does not know whether the light is on or off, her knowledge about the state of the world is the set of *possible states* that she thinks she might be in, i.e., the set $\{\{on\}, \{\neg on\}\}$. The execution of the *look* action will help the agent to decide whether the current state of the world is $\{on\}$ or $\{\neg on\}$.

Let us assume that S denotes the set of possible states that an agent believes she might be in; the execution of a sensing action a, that determines the truth value of a fluent f, will result in:

- S if the truth value of f is correct in every state in S;
- a subset $S' \subseteq S$, such that each state in S' has the correct value of f and each state in $S \setminus S'$ has the incorrect value of f.

It is important to observe that a sensing action *does not change the world* and its effect is about the knowledge of the agent. Although this fact is true, previous approaches to dealing with sensing actions in action languages or situation calculus, such as those proposed in [14, 11, 16], often make a fundamental implicit assumption: the reasoning agent has *correct* information. This also means that these approaches cannot be directly applied to situations in which the reasoning agent has completely *incorrect* information (or beliefs) about the world. Let us illustrate this with an example.

Example 1. Consider a robot which was **told** that the light in the kitchen is off and it needs to turn the light on. According to the given information, a plan for the robot consists of two actions: go the the kitchen and turn the light on. For the sake of our discussion, let us assume that the action of turning the light on/off can only be executed when the light is off/on.

If, in reality, the light in the kitchen is on, then the proposed plan of actions will fail. The robot goes to the kitchen and sees that the light is on. The robot reasons and comes up with three [1] possible explanations. The first one is that the light turned on by itself. The second one is that the robot's sensing equipment is defective. The third possibility is that **the robot was told something wrong,** i.e., its initial belief about the state of the world is wrong. To check its sensing equipment and see whether the light turns on by itself, the robot flips the switch and sees that the light is off. It waits and then flips the switch again and sees that the light is on. It concludes that its sensing equipment is in good condition and that the light cannot turn on by itself. It realizes that the third possibility is the only acceptable explanation for the *inconsistency* between its beliefs and the real state of the world. It corrects its initial beliefs and moves on.

This simple example illustrates the usefulness of sensing actions in helping an agent to revise their beliefs in real-world situations. Generalizing this idea, it means that agents need to be able to incorporate observations and update their beliefs while executing a plan. In this paper, we propose an approach that combines regression (or reasoning about previous states) and belief revision (or updating the beliefs when new information is available) to allow agents to correct their initial belief state. The main contributions of this paper can be summarized as follows: **(1)** we formalize a general framework based on regression and revision for repairing false beliefs in dynamic domains and develop algorithms for a concrete implementation of the framework; **(2)** we consider the formalization to include support for static causal laws and sensing actions; **(3)** we present an implementation for computing the initial correct belief state.

2 Background: The Action Language $\mathcal{B}_S$

We use a simplified version of the semantics for $\mathcal{B}_S$ in [4] that is similar to the semantics of the language $\mathcal{A}_S$ in [11]. In $\mathcal{B}_S$, an action theory in $\mathcal{B}_S$ is defined over two disjoint sets, a set of actions $\mathbf{A}$ and a set of fluents $\mathbf{F}$. A *fluent literal*

[1] We ignore the possibility that some other agent turns on the light while the robot is moving to the kitchen. This could be identified with the first option.

is either a fluent $f \in \mathbf{F}$ or its negation $\neg f$. A *fluent formula* is a propositional formula constructed from fluent literals.

An action theory is composed of statements of the following forms:

$$e \textbf{ if } \{p_1, \ldots, p_n\} \tag{1}$$

$$a \textbf{ causes } \{e_1, \ldots, e_n\} \textbf{ if } \{p_1, \ldots, p_m\} \tag{2}$$

$$a \textbf{ executable\_if } \{p_1, \ldots, p_n\} \tag{3}$$

$$a \textbf{ determines } f \tag{4}$$

where a is an action, f is a fluent, e, e_i are fluent literals representing *effects* and p_i are fluent literals indicating *preconditions*. (1) represents a *static causal law*; it conveys that whenever the fluent literals $p_1, \ldots, p_n$ hold in a state, then e will also hold in the state. (2) represents a *dynamic causal law*. It states that if $p_1, \ldots, p_m$ hold in a state and action a is executed, then the literals $e_1, \ldots, e_n$ will hold in the resulting state after the execution. (3) encodes an *executability condition* for action a. It states that action a can only be executed in a state where the literals $p_1, \ldots, p_n$ hold. (4) is called a *knowledge producing law*. The execution of the *sensing action* a will ensure that in the resulting state the truth value of f is known.

For simplicity, we assume that sensing actions do not occur in dynamic causal laws. To simplify the notation, we often drop the set notation from the laws; we indicate with R_a the set of laws of the form a **causes** $\{e_1, \ldots, e_n\}$ **if** $\{p_1, \ldots, p_m\}$. Given a static or dynamic law r, we indicate with $e(r)$ its effects and with $p(r)$ its preconditions.

An *action theory* is a pair (D, Ψ_0) where Ψ_0 is a fluent formula, describing the *initial state*, and D, called *action domain*, consists of laws of the form (1)–(4). For convenience, we sometimes denote the set of laws of the form (1) by D_C.

2.1 Transition Function

The semantics of $\mathcal{B}_S$ is based on a transition function; its definition requires some introductory concepts. Given a domain D in $\mathcal{B}_S$, a literal is either a fluent $f \in \mathbf{F}$ or its negation $\neg f$; a set of literals s is said to be *consistent* if for each $f \in \mathbf{F}$ we have that $\{f, \neg f\} \not\subseteq s$. A set of literals s is *complete* if for all $f \in \mathbf{F}$ we have that $f \in s \vee \neg f \in s$.

A consistent set of literals s is *closed* under a set of static causal laws $C \subseteq D_C$ if, for all $c \in C$ we have that $p(c) \subseteq s \Rightarrow e(c) \subseteq s$. With $Cl_C(s)$ we denote the smaller consistent set of literals that contains s and is closed under C. To simplify the notation we omit C when $C = D_C$.

A set of literals s is a *state* when it is complete and closed under D_C. A *belief state* is a set of states; intuitively, a belief state represents the states that an agent thinks she may be in. Given a fluent formula φ and a state s, a belief state Σ, and a set of belief states κ, we define: **(1)** $s \models \varphi$ for a state s if s is a model of φ; **(2)** $\Sigma \models \varphi$ for a belief state Σ if $s \models \varphi$ for each $s \in \Sigma$; **(3)** $\kappa \models \varphi$ for a set of belief states κ if $\Sigma \models \varphi$ for at least one $\Sigma \in \kappa$. Given a fluent formula

φ, let us define $\Sigma_\varphi = \{s \mid s \textit{ is a state, } s \models \varphi\}$. φ is said to be *consistent* if Σ_φ is not empty. The *direct effects* $e(a, s)$ of an action a in a state s are defined as $e(a, s) = \bigcup\{e(r) \mid r \in R_a, s \supseteq p(r)\}$.

The transition function for $\mathcal{B}_S$ maps pairs of action and belief state to sets of belief states. Let us start by defining the transition function Φ for the case of a single state. Let us write, for each fluent $f \in \mathbf{F}$, $\overline{f} = \neg f$, $\overline{\neg f} = f$, and $\overline{s} = \{\bar{l} \mid l \in s\}$ for a set of literals s. Let s be a state and a a non-sensing action executable in state s, $\Phi(a, s) = \{s' \mid s' \textit{ is a state}, s' = Cl((s \cap s') \cup e(a, s))\}$. Let Σ be a belief state and a is an action. If a is not executable in some $s' \in \Sigma$, we define $\Phi(a, \Sigma) = \emptyset$; otherwise,

- If a is a non-sensing action, $\Phi(a, \Sigma) = \{\bigcup_{s \in \Sigma} \Phi(a, s)\}$;
- If a is a sensing action a that determines the fluent f, $\Phi(a, \Sigma) = \{\Sigma_1, \Sigma_2\} \setminus \{\emptyset\}$ where $\Sigma_1 = \{s \in \Sigma \mid f \in s\}$ and $\Sigma_2 = \{s \in \Sigma \mid \neg f \in s\}$.

We are also interested in defining the transition function applied to a sequence of actions; let us define the function $\widehat{\Phi}$ which maps a sequence of actions and a set of belief states to a set of belief states: given a set of belief states κ and a sequence of actions α represented by a list of actions (using Prolog notation) $[a_1, \ldots, a_n] = [a_1 \mid \beta]$:

$$\widehat{\Phi}(\alpha, \kappa) = \begin{cases} \kappa & \text{if } n = 0 \\ \bigcup\limits_{\Sigma \in \kappa} \Phi(a_1, \Sigma) & \text{if } n = 1 \wedge \Phi(a_1, \Sigma) \neq \emptyset \text{ for each } \Sigma \in \kappa \\ \widehat{\Phi}(\beta, \widehat{\Phi}(a_1, \kappa)) & \text{if } n > 1 \text{ and } \widehat{\Phi}(a_1, \kappa) \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases} \tag{1}$$

We assume that for every action theory (D, Ψ_0), Ψ_0 is consistent. Since we are working with a history similar to that discussed in [9], we assume that the domains under consideration are deterministic.

3 Recovering from Inconsistent Histories

The transition function Φ provides a means for an agent to reason and plan in domains with sensing actions and incomplete knowledge about the initial state. It works well for hypothetical reasoning and planning but might be insufficient for an agent to use during the execution of a plan, as it might create discrepancies between the agent's hypothetical beliefs and the real-state of the world. Hence, it will not be help the agent to recover from false beliefs. Let us reconsider the problem in Example 1. We have that the initial belief state of the agent is $\Sigma = \{\{\neg on\}\}$ and the action *look* **determines** *on*. The state of the world is given by $\{on\}$. It is easy to see that the definition above yields $\Phi(look, \Sigma) = \{\{\neg on\}\}$ which indicates that the robot has false beliefs about the world. This is clearly not satisfactory; the robot, after observing that the light is on, should realize that the correct initial belief state is $\{\{on\}\}$ and change it accordingly. This issue becomes even more challenging if the realization of an incorrect initial belief state occurs after the execution of several actions. In this section, we propose a method for the robot to deal with this problem and to correct its initial beliefs.

Definition 1. *Let $T = (D, \Psi_0)$ be an action theory. A* history *of T is a sequence of pairs of actions and observations $\alpha = [(a_1, \psi_1), \ldots, (a_n, \psi_n)]$ where a_i is an action and ψ_i is a fluent formula. We assume that if a_i is a sensing action for the fluent f, then either $\Psi_i \models f$ or $\Psi_i \models \neg f$. We say that the history α is* inconsistent *with T if there exists some k, $1 \leq k \leq n$, such that $\widehat{\Phi}([a_1, \ldots, a_k], \{\Sigma_{\Psi_0}\}) \not\models \psi_k$.*

Intuitively, α indicates that the initial belief state Σ_0 of T is *incorrect.* We note that we overload the word "observation" in the definition of a history α. It does not imply that every action in α is a sensing action. We will assume that actions' effects are perfect (i.e., actions do not fail and do not produce wrong results). The case of uncertain effects will be left for future work. In this paper, we will focus on the following problem:

> *Given an action theory $T = (D, \Psi_0)$ and a history $\alpha = [(a_1, \psi_1), \ldots, (a_n, \psi_n)]$ that is inconsistent with T,* what is the correct initial belief state of T? I.e., what should the initial belief state of T be so that α is consistent with T?

We note that this problem is similar to the problem discussed in the diagnosis literature, such as [3, 4], which is concerned with identifying possible diagnosis given an action theory and a sequence of observations. The difference between this work and diagnosis lies in that this work focuses on the beliefs of agents along a history, whereas works in diagnosis concentrate in identifying defective components of the system represented by the action theory. Our work is closely related to the investigations of iterated belief revision in situation calculus, as in [9, 15]. Our proposed approach combines regression and belief revision. We start with the definition of a regression function. This function is different for sensing and non-sensing actions. We start with the case of non-sensing actions (also known as *ontic actions*).

Regression by non-sensing actions. Let a be a non-sensing action and ψ and φ be conjunctions of fluent literals. We say φ is a *result* of the regression of a from ψ, denoted by $\varphi \stackrel{a}{\to} \psi$, if $\forall s \in \Sigma_\varphi.(\Phi(a, s) \models \psi)$.

Regression by sensing actions. Let a be a sensing action and ψ and φ be conjunctions of fluent literals. We say φ is a *result* of the regression of a from ψ, denoted by $\varphi \stackrel{a}{\to} \psi$, if there exists some $\Sigma \in \Phi(a, \Sigma_\varphi)$ such that $\Sigma \models \psi$.

Observe that the requirement on the regression function for sensing actions differs from its counterpart for non-sensing actions. This is because the result of the execution of a sensing action is not deterministically predictable as for a non-sensing action. We should only guarantee that it is possible to obtain ψ after the execution of a.

We define the *regression of action a from a conjunction of fluent literals ψ*, denoted by $\mathcal{R}(a, \psi)$, by $\mathcal{R}(a, \psi) = \bigvee_{\varphi \stackrel{a}{\to} \psi} \varphi$. $\mathcal{R}(a, \psi)$ is called the *result of the regression of a from ψ*. We say that $\mathcal{R}(a, \psi)$ is undefined and write $\mathcal{R}(a, \psi) =$ *false* if $\{\varphi \mid \varphi \stackrel{a}{\to} \psi\} = \emptyset$.

For an arbitrary formula ψ, we define $\mathcal{R}(a, \psi) = \bigvee_{i=1}^{k} \mathcal{R}(a, \psi_i)$. where $\bigvee_{i=1}^{k} \psi_i$ is the unique, full DNF representation of ψ.

Proposition 1. *For an arbitrary consistent formula ψ such that $\mathcal{R}(a, \psi) \neq$ false, it holds that $\Phi(a, \Sigma_{\mathcal{R}(a,\psi)}) \models \psi$.*

Proof. Assume that $\bigvee_{i=1}^{k} \psi_i$ is the unique, full DNF representation of ψ and $\mathcal{R}(a, \psi) \neq \mathit{false}$. Then, $\Sigma_{\mathcal{R}(a,\psi)} = \bigcup_{i=1}^{k} \Sigma_{\mathcal{R}(a,\psi_i)} \neq \emptyset$. Therefore, $\Phi(a, \Sigma_{\mathcal{R}(a,\psi)}) \models \psi$. $\square$

We illustrate this definition using an example from [9].

Example 2 (Extended Litmus Test). Consider a domain with the fluents $\{$*Acid, Litmus, Blue, Red*$\}$, two dynamic laws for action *dip*, and two static causal laws:

$$\begin{array}{ll} \mathit{dip}\ \textbf{causes}\ \mathit{Red}\ \textbf{if}\ \mathit{Litmus}, \mathit{Acid} & \qquad \neg \mathit{Red}\ \textbf{if}\ \mathit{Blue} \\ \mathit{dip}\ \textbf{causes}\ \mathit{Blue}\ \textbf{if}\ \mathit{Litmus}, \neg \mathit{Acid} & \qquad \neg \mathit{Blue}\ \textbf{if}\ \mathit{Red} \end{array}$$

Consider $\psi = \neg \mathit{Red} \wedge \neg \mathit{Blue}$. Let us compute $\mathcal{R}(\mathit{dip}, \psi)$. Clearly, *Litmus* must be $false$ before the execution of *dip*. For otherwise, the paper would change color. Similarly, both *Red* and *Blue* must be false for the execution of *dip*. As such, $\mathcal{R}(\mathit{dip}, \psi) = \neg \mathit{Litmus} \wedge \neg \mathit{Red} \wedge \neg \mathit{Blue}$.

We extend the regression function in order to deal with a history $\alpha = [(a_1, \psi_1), \ldots, (a_n, \psi_n)]$ for $n \geq 1$ as follows:

- For $n = 1$

$$\widehat{\mathcal{R}}([(a_1, \psi_1)]) = \begin{cases} \mathcal{R}^*(a_1, \psi_1) & \text{if } \mathcal{R}(a_1, \psi_1) \not\equiv \mathit{false} \\ \mathit{false} & \text{otherwise} \end{cases} \tag{2}$$

- For $n > 1$

$$\widehat{\mathcal{R}}([(a_1, \psi_1), \ldots, (a_n, \psi_n)]) = \begin{cases} \widehat{\mathcal{R}}([(a_1, \psi_1), \ldots, (a_{n-1}, \psi_{n-1} \wedge \mathcal{R}^*(a_n, \psi_n)]) \\ \qquad\qquad \text{if } \psi_{n-1} \wedge \mathcal{R}^*(a_n, \psi_n) \not\equiv \mathit{false} \\ \mathit{false} \qquad\qquad \text{otherwise} \end{cases} \tag{3}$$

In (2)-(3), $\mathcal{R}^*(a, \psi)$ denotes $\mathcal{R}(a, \psi)$ when a is a non-sensing action and $\mathcal{R}(a, \psi) \wedge \ell$ when a is a sensing action that senses f and $\ell = f$ if $\psi \models f$, $\ell = \neg f$ if $\psi \models \neg f$.

Given an action theory (D, Ψ_0). Let $\alpha = [(a_1, \psi_1), \ldots, (a_n, \psi_n)]$ be a history and $\widehat{\Phi}(\alpha, \{\Sigma_{\Psi_0}\}) \not\models \psi_n$. We can compute $\widehat{\mathcal{R}}(\alpha)$ and use it to correct the initial belief state. This can be done using a belief revision operator. Let us assume the existence of a belief revision operator $\star$, which maps pairs of formulas to formulas and satisfies the *AGM postulates* [1].

Definition 2. *Let (D, Ψ_0) be an action theory. Let $\alpha = [(a_1, \psi_1), \ldots, (a_n, \psi_n)]$ be a history and $\widehat{\Phi}(\alpha, \{\Sigma_{\Psi_0}\}) \not\models \psi_n$. The corrected initial belief state of (D, Ψ_0) is defined by $\Psi_0 \star \widehat{\mathcal{R}}(\alpha)$.*

There are several proposals for the operator $\star$ (e.g., [5, 6, 13, 18]). In this paper, we will consider two approaches for defining the $\star$ operator. We note that as pointed out in [2], only the operator proposed in [6] satisfies all AGM postulates. In this paper, we make use of the following two operators.

- **Satoh's revision operator** [13]**:** Let Δ be the symmetric difference of two sets. For formulae ψ and φ, we define

$$\Delta^{min}(\psi, \varphi) = min_{\subseteq}(\{s \Delta s' \mid s \in \Sigma_\psi, s' \in \Sigma_\varphi\}).$$

Furthermore, define $\Sigma_{\psi \star \varphi}$ as $\{s \in \Sigma_\varphi \mid \exists s' \in \Sigma_\psi$ such that $s' \Delta s \in \Delta^{min}(\psi, \varphi)\}$.
- **Dalal's belief revision operator** [6]**:** Let λ be a formula. Given s and s' be two states in Σ_λ, let us define $Diff(s, s') = |s \Delta s'|$. For a formula ψ and two arbitrary states s and s',

$$s \sqsubseteq_\psi s' \text{ iff } \exists r \in \Sigma_\psi \text{ s.t. } \forall r' \in \Sigma_\psi.[Diff(r, s) \leq Diff(r', s')]$$

Given two formulas ψ and φ, the revision of ψ by φ is defined by[2] $\psi \star \varphi = \min(\Sigma_\varphi, \sqsubseteq_\psi)$.

Example 3 (Continuation of Example 2). Assume that the initial belief for the domain in Example 2 is specified by

$$\Psi_0 = Litmus \wedge \neg Red \wedge \neg Blue.$$

Consider the history $\alpha = [(dip, \psi)]$ where $\psi = \neg Red \wedge \neg Blue$. Let $s_1 = \{Litmus, Acid, \neg Red, \neg Blue\}$ and $s_2 = \{Litmus, \neg Acid, \neg Red, \neg Blue\}$. We have that $\Sigma_{\Psi_0} = \{s_1, s_2\}$ and $\Phi(dip, \Sigma_{\Psi_0}) = \{\{Litmus, Acid, Red, \neg Blue\}, \{Litmus, \neg Acid, \neg Red, Blue\}\}$ which indicates that the initial belief state is incorrect. We need to identify the correct initial belief in this situation.

The regression of dip from ψ gives us $\mathcal{R}(dip, \psi) = \neg Litmus \wedge \neg Red \wedge \neg Blue$. We want to compute $\Psi_0 \star \varphi$ where $\varphi = \mathcal{R}(dip, \psi)$.

- ***Using Satoh's operator***: $\Sigma_\varphi = \{s_3, s_4\}$ where $s_3 = \{\neg Litmus, Acid, \neg Red, \neg Blue\}$ and $s_4 = \{\neg Litmus, \neg Acid, \neg Red, \neg Blue\}$. We calculate [3]

$$\begin{aligned} \Delta^{min}(\Psi_0, \varphi) &= \min_{\subseteq}\{s_3 \Delta s_1, s_3 \Delta s_2, s_4 \Delta s_1, s_4 \Delta s_2\} \\ &= \min_{\subseteq}\{\{\neg Litmus, Litmus\}, \{Acid, \neg Litmus, Litmus, \neg Acit\}\} \\ &= \{\{\neg Litmus, Litmus\}\} \end{aligned}$$

which leads to $\Psi_0 \star \varphi = \{s_3, s_4\}$. In other words, $\neg Litmus$ is true in the initial belief state.
- ***Using Dalal's operator***: because Σ_{Ψ_0} has only two elements, s_1 and s_2, we have that $s_i \sqsubseteq s_i$ for $i = 3, 4$ and $s_3 \sqsubseteq_{\Psi_0} s_4$ and $s_4 \sqsubseteq_{\Psi_0} s_3$. Therefore, $\min(\Sigma_\varphi, \sqsubseteq_{\Psi_0}) = \{s_3, s_4\}$. In other words, we receive the same results as with Satoh's operator.

We revisit the story from the introduction and illustrate the definition with sensing actions.

[2] The original definition by Dalal identifies the set of formulae which are true in $\min(\Sigma_\varphi, \sqsubseteq_\psi)$.

[3] The results of the computation is the same if states are represented using only positive literals. In [13], $\{\{\neg Litmus, Litmus\}\}$ would be considered as $\{\{Litmus\}\}$

Example 4. A simplified version of the story in the introduction, focused only on the sensing action *look*, can be described by the action theory consisting of the law "*look* **determines** *on*" and the initial belief state specified by the formula $\neg on$. Clearly, the history $[(look, on)]$ is inconsistent with the theory. To correct the initial belief state, we compute $\mathcal{R}(look, on) = True$ and $\mathcal{R}^*(look, on) = on$. $\neg on \star on$ results in on, which shows that our approach allows for the agent to correct its beliefs.

The correctness of our formalization is proved in the next proposition.

Proposition 2. *Given an action theory (D, Ψ_0) and an inconsistent history $\alpha = [(a_1, \psi_1), \ldots, (a_n, \psi_n)]$, for every $k = 1, \ldots, n$, $\widehat{\Phi}([a_1, \ldots, a_k], \Sigma_{\Psi'_0}) \models \psi_k$, where $\Psi'_0 = \Psi_0 \star \varphi$ and $\varphi = \widehat{R}(\alpha)$.*

Proof. Induction by n. **Base**: $n = 1$. Let $\bigvee_{i=1}^{k} \psi_1^i$ be a full DNF representation of ψ_1 and $\varphi = \mathcal{R}(a_1, \psi_1)$. If a_1 is a non-sensing action, then $\Psi'_0 = \Psi_0 \star \varphi$. By definition of regression and $\star$, $\Sigma_{\Psi'_0} \subseteq \Sigma_\varphi$. Thus, $\Phi(a_1, \Sigma_{\Psi'_0}) \models \psi_1$. If a_1 is a sensing action that senses f then $\widehat{R}^*(\alpha) = \varphi \wedge f$ if $\psi_1 \models f$ and $\widehat{R}^*(\alpha) = \varphi \wedge \neg f$ if $\psi_1 \models \neg f$, and $\Psi'_0 = \Psi_0 \star \widehat{R}^*(\alpha)$. In either case, $\Phi(a, \Sigma_{\Psi'_0}) \models \psi_1$. The base case is proved. **Step**: $n \Rightarrow n + 1$. Consider the history $\beta = [(a_1, \psi_1), \ldots, (a_{n-1}, \psi_{n-1} \wedge \mathcal{R}^*(a_n))]$. If β is inconsistent then applying the inductive hypothesis for β, we have that $\widehat{\Phi}([a_1, \ldots, a_k], \Sigma_{\Psi'_0}) \models \psi_k$ for $k = 1, \ldots, n - 1$, and $\widehat{\Phi}([a_1, \ldots, a_{k-1}], \Sigma_{\Psi'_0}) \models \psi_{n-1} \wedge \mathcal{R}^*(a_n)$, which implies $\widehat{\Phi}([a_1, \ldots, a_n], \Sigma_{\Psi'_0}) \models \psi_n$. If β is consistent, it is easy to see that $\widehat{\Phi}([a_1, \ldots, a_k], \Sigma_{\Psi_0}) \models \psi_k$, for $k = 1, \ldots, n$, which contradicts the assumption that α is inconsistent.

4 A Logic Programming Implementation

We are interested in implementing a system that allows us to resolve situations where an agent encounters an inconsistent history. The proposed formalism in the previous section is general and both regression and $\star$ can be implemented in different ways. For example, regression has been mainly implemented by the planning community using an imperative language and $\star$ has been explored using answer set programming [8].

In this paper, we present an implementation using Prolog, as Prolog provides an elegant balance between declarative computational logic and procedural-style encoding. A detailed discussion on the choice of Prolog is included in the last subsection.

Let D be an action domain. Consider a non-sensing action a and $R \subseteq R_a$, let us define

$$eff(R) = \bigcup_{r \in R} [e(r)] \qquad\qquad pre(R) = \bigcup_{r \in R} [p(r)]$$

as the effects and the preconditions of R. We assume that a **executable_if** η_a belongs to D.

4.1 Regression of a Non-Sensing Action

Let us consider an action a and a consistent set of fluent literals ψ [4]. We define

$$R_a^+(\psi) = \{r \in R_a \mid \psi \models p(r)\} \qquad R_a^-(\psi) = \{r \in R_a \mid \psi \models \neg p(r)\}$$

representing the set of dynamic laws that are applicable and not applicable, respectively, when a is executed and ψ is definitely true. We say that φ is a-*splittable* if $R_a = R_a^+(\varphi) \cup R_a^-(\varphi)$.

Proposition 3. *Consider two conjunctions φ and ψ and a non-sensing action a such that $\varphi \stackrel{a}{\to} \psi$. There exists a set of a-splittable conjunctions of literals $\{\varphi_1, \ldots, \varphi_k\}$ such that $\varphi \equiv \bigvee_{i=1}^k \varphi_i$ and $\varphi_i \stackrel{a}{\to} \psi$ for every $i = 1, \ldots, k$.*

Proof. We prove by induction over $k = |R_a \setminus R_a^+(\varphi) \cup R_a^-(\varphi)|$.
Base: $k = 0$, φ is a-splittable and thus the proposition is trivial.
Step: $k \Rightarrow k+1$. Consider some $r \in R_a$ and $r \notin R_a^+(\varphi) \cup R_a^-(\varphi)$. Let $\varphi' = \varphi \cup p(r)$ and $\varphi_l = \varphi \cup \{\bar{l}\}$ for $l \in p(r) \setminus \varphi$. Let Ω be the collection of consistent conjunctions from $\{\varphi'\} \cup \{\varphi_l \mid l \in p(r) \setminus \varphi\}$. It is easy to see that $\varphi \equiv \bigvee_{\lambda \in \Omega} \lambda$. Because $\Sigma_\lambda \subseteq \Sigma_\varphi$ for $\lambda \in \Omega$, we have that $\lambda \stackrel{a}{\to} \varphi$. Observe that each $\lambda \in \Omega$ satisfies that $|R_a^+(\lambda) \cup R_a^-(\lambda)| \leq k$. Using inductive hypothesis, we can conclude the proposition. $\square$

For example, considering $\psi = (l_1 \wedge l_2) \vee (l_3 \wedge l_4)$ and $R_a = \{a$ **causes** $\{l_1\}$ **if** $\{l_2\}$, a **causes** $\{l_3\}$ **if** $\{l_4\}\}$. Two a-splittable formulae are $\varphi_i = \neg l_2 \wedge l_2 \wedge \neg l_4$ and $\varphi_j = \neg l_3 \wedge l_4 \wedge \neg l_2$.

Due to Prop. 3, we will only need to identify a-splittable conjunctions in computing $\mathcal{R}(a, \psi)$. Clearly, if φ is the result of regression from ψ by a then a must be executable in any state $s \in \Sigma_\varphi$, i.e., $\forall_{s \in \Sigma_\varphi}[\eta_a \subseteq s]$.

Definition 3. *A consistent set of literals φ is a* potential regression result *from a conjunction of literals ψ with respect to a if*

- *$Cl(\varphi)$ is a-splittable;*
- *$pre(R_a^+(\varphi)) \cup \eta_a \subseteq Cl(\varphi)$;*
- *$eff(R_a^+(\varphi)) \cup \psi$ is consistent*

This definition guides the implementation during the nondeterministic computing of φ by reducing the number of guessed literals.

Given a potential regression result φ from ψ by a, we observe that the execution of a from any state $s \in \Sigma_\varphi$ would divide φ into three components as shown in Figure 1 where

- δ: the set of inertial literals whose values do not change;
- $\overline{\delta_S}$: the set of literals whose values changed because of the application of *static causal laws* in the resulting state, i.e., the state containing ψ; and
- $\overline{\delta_D}$: the set of literals whose values changed because the effects of the *dynamic laws* in $R_a^+(\varphi)$.

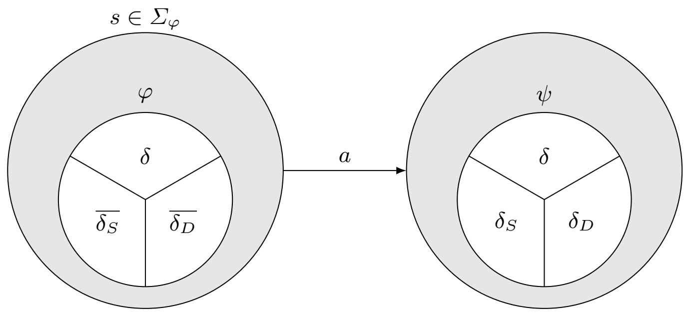

Fig. 1. Intuition for Regression

This leads us to the following definition:

Definition 4. *A consistent set of literals φ is a* computed regression result *from ψ w.r.t. an ontic action a if*

- *φ is a potential regression result from ψ with respect to a*
- *for every γ such that $\gamma \cup \varphi$ is a state, there exists two consistent sets of literals δ, δ_S such that*

$$\delta \subseteq (\varphi \setminus \overline{\mathit{eff}(R_a^+(\varphi))}) \tag{4}$$

$$(\varphi \setminus \overline{\mathit{eff}(R_a^+(\varphi))}) \setminus \delta = \delta_S \tag{5}$$

$$Cl(\delta \cup \gamma \cup \delta_D) \text{ is a state} \tag{6}$$

$$\psi \cup \delta_S \subseteq Cl(\delta \cup \gamma \cup \delta_D) \tag{7}$$

Equation (4) characterizes δ, the inertial literals from φ; (5) identifies δ_S, the non-inertial literals from φ because of static causal laws; and the two equations (6) and (7) guarantee that the execution of a in $\delta \cup \varphi$ results in ψ. This allows to determine whether a potential regression result φ belongs to $\mathcal{R}(a, \psi)$. We prove that computed regression results are really what we need:

Proposition 4. *Let D be a deterministic action theory and a be a non-sensing action. If φ is a computed regression result from ψ with respect to a then $\varphi \xrightarrow{a} \psi$.*

Proof. Consider $s \in \Sigma_\varphi$. This implies that there exists γ such that $\gamma \cup \varphi = s$. It is clear that $\mathit{eff}(R_a^+(\varphi)) = e(a, s)$. Take $s' = Cl(\delta \cup \gamma \cup \mathit{eff}(R_a^+(\varphi)))$. We have $\overline{\delta'} \subseteq s'$. This implies $s \cap s' \subseteq \delta \cup \gamma$. However, both $s \supseteq \delta \cup \gamma$ and $s' \supseteq \delta \cup \gamma$ hold. Therefore, $s \cap s' = \delta \cup \gamma$. So, $s' \in \Phi(a, s)$. Because of D is deterministic, the above proves the proposition. □

The above proposition shows that if there exists no computed regression result from ψ with respect to a, then $\mathcal{R}(a, \psi) = \mathit{false}$. Using Propositions 3-4, we can compute the regression result of an arbitrary formula ψ by (i) computing a full DNF representation of ψ, $\bigvee_{i=1}^k \psi_i$; (ii) computing $\mathcal{R}(a, \psi_i)$; and (iii) returning $\mathcal{R}(a, \psi) = \bigvee_{i=1}^k \mathcal{R}(a, \psi_i)$ if $\mathcal{R}(a, \psi_i) \neq \mathit{false}$ for some i.

[4] Note we freely exchange between sets of literals and conjunctions of literals.

4.2 Regression of a Sensing Action

Assume that a is a sensing action with executability condition η_a and that a only senses one fluent f. Let ψ and φ be two conjunctions of literals such that $\varphi \xrightarrow{a} \psi$, then a must be executable in any state in Σ_φ. Furthermore, f is unknown in φ and, ψ differs from φ only by the observation f or $\neg f$.

Definition 5. *Let a be a sensing action that senses f. A consistent conjunction of literals φ is a* computed regression result *from ψ by a if*
- $\eta_a \subseteq Cl(\varphi)$;
- $\{f, \neg f\} \cap Cl(\varphi) = \emptyset$;
- $\psi \setminus \{f, \neg f\} \subseteq Cl(\varphi)$
- *if $f \in \psi$ then $\varphi \cup \{f\}$ is consistent and if $\neg f \in \psi$ then $\varphi \cup \{\neg f\}$ is consistent.*

Proposition 5. *Let φ be a computed regression result from ψ by a sensing action a. Then, $\varphi \xrightarrow{a} \psi$.*

Proof. Since $\eta_a \subseteq Cl(\varphi)$, we have that $\Sigma_\varphi \models \eta_a$. The last two conditions in Definition 5 implies that $\psi \setminus \{f, \neg f\} \subseteq \gamma$ for every $\gamma \in \Sigma_\varphi$. Consider three cases: (i) $\psi \cap \{f, \neg f\} = \emptyset$. Clearly, we have that $\Phi(a, \Sigma_\varphi) \models \psi$; (ii) $f \in \psi$. The fourth condition implies that $\Sigma_1 = \{\gamma \mid \gamma \in \Sigma_\varphi, f \in \gamma\} \neq \emptyset$. Since $\Sigma_1 \in \Phi(a, \Sigma_\varphi)$ and $\Sigma_1 \models \psi$, the proposition is proved. (iii) $\neg f \in \psi$: similar to (ii). □

The previous proposition and definition provides the basis for computing the regression with sensing actions. Intuitively, φ is the result of regression form $(\psi \setminus \{f, \neg f\}) \cup \eta_a$.

4.3 Implementation in Prolog

This section gives an overview of the implementation and some design decisions in it. The main purpose of the implementation is to guide the development of the definitions in the previous section and validate our ideas as we proceeded. Moreover it gives an overview of strengths and weakness of the theory from a pragmatic point of view.

We use Prolog for different reasons. The simplicity of Prolog in dealing with lists and formulae makes it a suitable platform for computing $\widehat{\mathcal{R}}$ and various belief revision operators. The computation of the regression function $\mathcal{R}$ is inherently non-deterministic, which matches well with Prolog's behavior. Last but not least, Prolog is declarative and modular, which provides a good platform for guaranteeing the correctness of the implementation and the ability to experiment with various belief revision operators. The implementation makes use of Prolog built-ins such as `foldl`, membership checking, union, set difference, etc. and has been tested with SWI-Prolog 8.0.3 and YAP 6.3.3 on Linux x86-64. The complete code is available on GitHub [5].

[5] https://github.com/NMSU-KLAP/Repair-by-Regression

We represent a conjunction of literals as a list and a DNF formula as a list of lists of literals, respectively. The laws (1)–(4) that define an action theory (see Section 2) are expressed in Prolog as:

$$e \textbf{ if } p_1, \ldots, p_n \leftrightarrow \texttt{staticLaw([e],[p}_1\texttt{,...,p}_n\texttt{])}$$
$$a \textbf{ causes } e_1, \ldots, e_n \textbf{ if } p_1, \ldots, p_m \leftrightarrow \texttt{dynamicLaw(a,[e}_1\texttt{,...,e}_n\texttt{],[p}_1\texttt{,...,p}_m\texttt{])}$$
$$a \textbf{ executable\_if } p_1, \ldots, p_n \leftrightarrow \texttt{exeCnd(a,[p}_1\texttt{,...,p}_n\texttt{])}$$
$$a \textbf{ determines } f \leftrightarrow \texttt{senseAct(a,f)}$$

Before we describe the main predicates, we discuss some auxiliary predicates. We use the conventional notation of Prolog with '+'/'-' param to indicate whether the parameter is an input/output parameter. In the following, ψ, ψ', and o are lists of lists of literals, representing formulae in DNF form; a is an action; γ is a list of literals.

- `forceObservation(+`ψ`,+`o`,-`ψ'`)`: returns ψ' that represents the conjunction $\psi \wedge o$; this is needed in the computation of $\widehat{\mathcal{R}}$ (Equation 3); return ***False*** if $\psi \wedge o$ is inconsistent.
- `sensedLiteral(+`a`,+`o`,-Literal)`: returns `Literal`, the fluent literal representing the value of the fluent that a senses given the observation o. Note that we assume that each sensing action only senses one fluent and that if a sensing occurs in a history then the sensed fluent will appear in the observation.
- `minimalPotentialRegressionResult(+`η_a`,+`R_a`,+`ψ`,-`φ`,-Eav)`: returns a minimal potential regression result φ and the effects of executing a in any state containing φ, `Eav`, given η_a, R_a, and ψ according to Definition 3;
- `minimalComputedRegressionResult(+Eav,+`ψ`,+`φ'`,-`φ`)`: returns a minimal (w.r.t. $\subseteq$) computed regression result φ containing φ', given φ', ψ, and `Eav` where φ' is a potential regression result.
- `consistentChk(+`γ`)`: returns ***True*** if γ is consistent and ***False*** otherwise.
- `stateExpansion(+`γ`,-`S`)`: returns a state S containing γ if it exists and ***False*** otherwise.
- `revisionBySatoh(+`ψ`,+`φ`,-Revision)`: returns $\psi \star \varphi$ according to Satoh's definition.
- `revisionByDalal(+`ψ`,+`φ`,-Revision)`: returns $\psi \star \varphi$ according to Dalal's definition.
- `closure(+`φ`, -`φ'`)`: calculate $\varphi' = Cl(\varphi)$.
- `computedRegressionResultChk(+`φ`,+`$\mathit{eff}(R_a^+(\varphi))$`,-`$\psi$`)`: check if φ is a *computed regression result* of ψ (satisfied (4) - (7)).
- `minimalBlockingExpansion(+Init,+Rules,-`φ'`)`: calculate φ' such that $\varphi' \models$ `Init`, φ' is minimal and none of the rules in `Rules` can be executed in φ'.
- `minimalsBySubsets(+List,-MinimalList)`: calculate `MinimalList` such that $\forall s \nexists s' : (s, s' \in \texttt{MinimalList}, s' \subseteq s)$ and `MinimalList` = `List`.
- `rulesSplit(+Rules,+`ψ`,-`R^+`,-`R^-`)`: calculate a split $\langle R^+, R^- \rangle$ from set of rules `Rules` such that this split compatible with ψ.

The main clauses used in the implementation of $\widehat{\mathcal{R}}$ are given below.

- `regression(+`α`,-`φ`) :-`
  `reverse(`α`,`β`), foldl(regressionActionObservation,`β`,[],`φ`).`

  This is the main predicate that returns $\widehat{\mathcal{R}}(\alpha)$ for a history α. It uses the predicate `foldl` to iteratively compute the regression over the suffixes of α.

- `regressionActionObservation(+(`a`,`o`),+`ψ`,-`φ`) :-`
  `forceObservation(`ψ`,`o`,`ψ'`),`
  `(dynamicLaw(`a`,_,_) -> regressionOnticAction(`a`,`ψ'`,`φ`);`
  `(sensedLiteral(`a`,`o`,L), regressionSensingAction(`a`,L,`ψ'`,`φ'`),`
  `findall(`φ''`, (member(`φ'_i`,`φ'`), ord_union(L,`φ'_i`,`φ''`)), `φ`)).`

  This clause returns $\varphi = \mathcal{R}(a, o \wedge \psi)$ if $o \wedge \psi$ is consistent and the regression of a from $o \wedge \psi$ is successful; and fails otherwise. It starts by computing $o \wedge \psi$ and then, depending on whether a is an ontic or a sensing action, computes $\mathcal{R}(a, o \wedge \psi)$ accordingly. Note that we assume that an action is either an ontic or a sensing action but not both.

- `regressionOnticAction(+`a`,+`ψ`,-`φ`) :-`
  `exeCnd(a,`η_a`), findall((a,Ea,Fa), dynamicLaw(a,Ea,Fa), `R_a`),`
  `findall(`φ'`, (member(`ψ_i`,`ψ`),`
  `minimalPotentialRegressionResult(`η_a`,`R_a`,`ψ_i`,`φ''`,Eav),`
  `minimalComputedRegressionResult(Eav,`ψ`,`φ''`,`φ'`)),`φ`).`

  This clause computes $\varphi = \mathcal{R}(a, \psi)$ for an ontic action a and formula ψ. It uses the `minimalPotentialRegres- sionResult/5` and `minimalComputedRegressionResult/4` to identify all possible regression results by a from ψ. Observe that we only compute minimal (with respect to $\subseteq$) regression results since for consistent sets of literals $\varphi_1 \subseteq \varphi_2$, we have that $\Sigma_{\varphi_2} \subseteq \Sigma_{\varphi_1}$. Therefore, if $\Phi(a, \Sigma_{\varphi_1}) \models \psi$ then $\Phi(a, \Sigma_{\varphi_2}) \models \psi$.

- `regressionSensingAction(+`a`,+`ψ`,-`φ`) :-`
  `exeCnd(a,`η_a`), sensedLiteral(a,`ψ`,L),`
  `findall(`φ'`, (member(`ψ_i`,`ψ`), ord_subtract(`ψ_1`,[L,`$\overline{\texttt{L}}$`],`γ`),`
  `ord_union(`η_a`,`γ`,`φ'`), consistentChk(`φ'`)), `φ`)`

  This clause computes $\varphi = \mathcal{R}(a, \psi)$ for a sensing action a and formula ψ.

We conclude the section with an example inspired by the example in [15].

Example 5. Consider an extension of Example 4 with $\mathbf{F} = \{in(k), on(k) \mid k = 1, \ldots, n\}$, and the laws with $x = 1, \ldots, n$, $y = x + 1$ if $x < n$ and $y = 1$ if $x = n$, $z = 1, \ldots, n$, and $z \neq x$:

$leave(x)$ **causes** $in(y)$ **if** $in(x)$	$look(k)$ **determines** $on(k)$	$\neg in(y)$ **if** $in(x)$
$turnOn(k)$ **causes** $on(k)$ **if** $in(k)$	$look(k)$ **executable_if** $in(k)$	

First, let us consider the case $n = 2$. Assume that the initial state is given by $in(2) \wedge \neg on(1)$. Consider the three histories:

```
H1 = [(leave(2),[]),(look(1), [[on(1)]])]
H2 = [(leave(2),[]),(turnOn(1),[]),(leave(1),[]), (look(2),[[on(2)]])]
H3 = [(leave(2),[]),(turnOn(2),[in(2)]), (look(1),[[on(1)]])]
```

The goal `regression(H`$_i$`,Regression`$_i$`)` returns

```
Regression1 = [[-in(1), in(2),on(1)]]
Regression2 = [[-in(1),in(2),on(2),on(1)], [-in(1),in(2),on(2),-on(1)]]
Regression3 = [ ]
```

where the empty list `[ ]` signifies that the regression fails. This is because `look(1)` is executable only if `in(1)` is true, but the observation immediately before the execution of `look(1)` indicates that `in(2)` is true.

With `I = [[-in(1),-on(1)]]`, `revisionByXxx(I,Regression`$_i$`,RIState`$_i$`)`, where `Xxx` stands for Satoh or Dalal, returns

```
RealInitialState1=[[-in(1),in(2),on(1)]]
RealInitialState2=[[-in(1),in(2),on(2),-on(1)]]
RealInitialState3=[ ]
```

We can easily verify that the results are correct.

To experiment with the system on larger problems, we consider the above domain with $n = 2, \dots, 10$ and consider the initial belief state `I=[[in(n),on(n), on(n-1),...,on(1)]]` and the history `H=[(look(n),[[on(n)]]),(leave(n), []),(look(1),[[on(1)]]),(leave(1),[]),...,(leave(n-2),[]),(look(n-1), [[-on(n-1)]])]`.

The problem can be stated as follows. Initially, the robot is in the room n and believes that all lights are on. It makes a tour, from room n to $1, \dots, n-1$. In each room, the robot looks at the light. At the end of the tour, it realizes that its initial belief is incorrect ($\neg on(n-1)$ is observed and it supposed to be $on(n-1)$). We tested with $n = 1, \dots, 10$ and the system returns the result within 30 minutes. We observe that the size of the domain, in terms of the number of fluents and the number of actions, plays a significant role in the performance of the system. For $n = 10$, we have 20 fluents and the number of potential regression results for a non-sensing action (e.g., `leave(k)`) is small but checking whether or not a potential regression result is a computed regression result involves checking the number of possible states given a set of formulae, which could range from 2^1 to 2^{19}. We observe that the system spends most of the time doing just that.

We verify our observation by testing with a domain rich in static casual laws.

Example 6. Consider a pipe line with n sections from [17]. At the left-most end of the pipe line is a gas tank and at the right-most end is a burner. The robot wants to start a flame in the burner. The pipe sections can be either pressured by the tank or un-pressured. Sections connect to each other by valves. Opening a valves causes the section on its right side to be pressured if the section to its left is pressured. And a valve can be opened only if the next valve on the line is closed (for safety reason). Closing a valve causes the pipe section on its right side to be un-pressured. If a valve is open and the section of its left is pressured then the section on its right will be pressured, otherwise (either the valve is closed or the section on the left is un-pressured), the pipe on the right side is un-pressured. The burner will start a flame if the pipe connecting to it is pressured, and the gas tank is always pressured.

This domain can be encoded by a domain with $\mathbf{F} = \{opened(k), pressured(k) \mid k = 1, \ldots, n\}$, and the following laws with $x = 1, \ldots, n; y = 2, \ldots, n$ and $k = 1, \ldots, n-1$:

$$
\begin{array}{cc}
open(x) \textbf{ causes } opened(x) \textbf{ if } \top & sense(x) \textbf{ determines } pressured(x) \\
close(x) \textbf{ causes } \neg opened(x) \textbf{ if } \top & open(k) \textbf{ executable\_if } \neg opended(k+1) \\
\neg pressured(x) \textbf{ if } \neg opened(x) & pressured(y) \textbf{ if } opened(y), pressured(y\neg 1) \\
\neg pressured(y) \textbf{ if } \neg pressured(y\neg 1) & pressured(1) \textbf{ if } opened(1)
\end{array}
$$

Assume that initially, the robot believes the first valve is opened and all other valves are closed. To start the burner, she will open the valves from 2 to n. After opening the n-the valve, she realizes that the burner does not start. It means that her initial belief is incorrect. Regression and revision is required, i.e., we need to compute the result $\widehat{\mathcal{R}}(H)$ where `H = [(open(2),[]), (open(3),[]),..., (open(n),[-pressured(n)])]`. We tested this problem with $n = 2, \ldots, 8$. Again, the performance of the system degrades quickly and the problem of verifying the result of regression, as in Example 5, is also observed. For $n = 8$, the system took more than 1 hour.

The above examples show that a more efficient way of dealing with computing the regression will be required to improve the performance of the system. We leave this challenge for the future work.

5 Related Work and Discussions

The present work is most closely related to studies in iterated belief change in a general setting of transition systems [9] or in situation calculus [15], i.e., iterated belief change in dynamic domains. It is therefore different from studies aimed at defining and/or characterizing a general iterated belief revision in the literature such as those summarized in [7, 12].

The discussion in [9] has been extended to probabilistic setting in [10]. We will focus in comparing our work with the work in [15, 9]. The key differences between our work and [15, 9] are (i) we employ an action language with static causal laws to represent dynamic domains whereas [15] uses situation calculus and [9] uses the general notion of transition system; (ii) we focus on the implementation of the proposed operator and have a fully functional system for correcting the initial belief state; (iii) we consider sensing actions, which are also considered in [15] but not in [9]; and (iv) we do not consider nested beliefs as in [15].

Given an action theory (D, Ψ_0). D can be seen as a transition system whose transitions are defined by (s, a, s') iff $s' \in \Phi(a, s)$. Action histories, belief histories, and observation histories are defined in [9]. A pair (β, o) of an action history

$\beta = [a_1, \ldots, a_n]$ and an observation history $o = [\psi_1, \ldots, \psi_n]$ in their definitions corresponds to a history $\alpha = [(a_1, \psi_1), \ldots, (a_n, \psi_n)]$ in our definition. We do not define belief histories explicitly but implicitly use $[\Phi(a_1, \Psi_0), \widehat{\Phi}([a_1, a_2], \Psi_0), \ldots, \widehat{\Phi}([a_1, \ldots, a_n], \Psi_0)]$ as the belief history corresponding to α. The idea of combining regression and a belief revision operator to correct the initial belief state is also present in [9]. Technically, regression from an observation φ by an action sequence β is defined as the set of states $\varphi^{-1}(\beta)$ that contains every state from which the execution of β results in a state satisfying φ. A revision is then used to identify the correct initial belief and the belief history is adjusted accordingly. The focus of the work in [9] is to define a belief evolution operator that combines belief change and belief revision and study properties of such operator. Therefore, the definitions in [9] are generic and no implementation is available.

Our proposed work could be viewed a concretization of the framework in [9], i.e., we developed and implemented the regression function and two belief revision operators. In addition, we use a language to specify dynamic systems while [9] assumes that the transition system is given. The work in [9] also considered situations when observations can be incorrect.

The work in [15] formalizes reasoning about beliefs in action domains with sensing actions using situation calculus. Unlike earlier approaches to reasoning with sensing actions (e.g., [16, 14]), the formalism in [15] deals with incorrect beliefs. It introduced an explicit binary relation B, called an *accessibility relation*, between situations for modeling beliefs; and a function pl, called *plausibility function*, from situations to natural numbers for modeling the plausibility of situations. Successor states axioms are then defined for B and pl. This, together with the situation calculus foundational axioms, will allow agents for reasoning about beliefs, belief introspection, and awareness of mistakes. Roughly speaking, the plausibility function modifies the original definition of beliefs of an agent by Scherl and Levesque, $Bel(\psi, s) \stackrel{def}{=} \forall s'.[B(s', s) \supset \psi[s']]$, to $Bel(\psi, s) \stackrel{def}{=} \forall s'.[(B(s', s) \wedge (\forall s''.(B(s'', s) \supset pl(s') \leq pl(s''))) \supset \psi[s']]$, which basically states that the agent believes ψ is true in a situation s if ψ is true in any most plausible situations accessible to s. Under this framework, it is shown that sensing actions can be used to correct the incorrect beliefs of agents. Proposition 2 shows that our approach also allows agent to correct beliefs. We focus on the initial belief but the approach can easily be adapted to consider any states in a history. For example, assume that the agent believes that Σ is the current belief state, $\Sigma \models f$, and a is a sensing action that senses f and executable in Σ, then the following holds: if $\Phi(a, \Sigma) \models \neg f$ then the current belief state must be revised by $\neg f$, i.e., $\Sigma \star \neg f$. This result is similar to the Theorem 21 in [15]. Similar to [9], the work in [15] investigates properties of the belief operators while we focus on the development of a system that computes the correct initial belief state or, as indicated, the belief states along a history. We observe that the discussion in [15] does assume that all actions are always executable. A key difference between the framework of [15] and ours is how the approaches deal with static causal laws. Indeed, static causal laws could be compiled to effect axioms and dealt with in the framework of [15] while they are dealt with directly. For example, to work with Example 5, the approach of [15] requires that effects of actions, direct (e.g., $in(y)$ for the action $leave(x)$ or indirect ($\neg in(x)$), must be described as effects of the actions via successor state axioms. Indeed, there are certain advantages in dealing with static causal laws directly (see below).

The proposed work is considered in a setting similar to the one considered in the diagnosis literature such as those in [3, 4]. We assumes that the history is complete and focus on repairing the initial beliefs of the reasoner if the history is consistent. On the other hand, approaches to diagnosis often assume an incomplete history and focus on identifying missing action occurrences that are the source of the history's inconsistency.

The proposed work is considered within the deterministic fragment of the action language $\mathcal{B}_S$ which is $\mathcal{A}_S$ extended with static causal laws. We focus on deterministic domains with static causal laws for two reasons. First, the use of static causal laws allows a natural and compact representation of many domains that would have otherwise required an exponential number of fluents. This covers the majority of benchmarks used in several planning competitions[6]. Furthermore, dealing directly with static causal laws in planning is advantageous and syntactical conditions guaranteeing determinism of domains with static causal laws can be found in [17]. Second, we employ the assumption sets forth in [9] that observations are correct. When domains are non-deterministic, this assumption must be lifted or the revision will need to be weaken as shown below.

Example 7. Consider a domain with the set of fluents $\{f, g, h\}$, an action a with a **causes** f **if** $\neg f$; a **causes** $\neg f$ **if** f and two laws g **if** $f, \neg h$; h **if** $f, \neg g$.

It is easy to see that this domain is non-deterministic. Assume that initially, the reasoner believe is $\bigvee_{s \in \Sigma} s$ where Σ is the set of all possible states. Consider the history $[(a, \psi)]$ with $\psi = f \wedge g \wedge \neg h$. It is easy to see that the regression of a from ψ, let us overload it with $\mathcal{R}(a, \psi)$, should be the set $\{\{\neg f, g, \neg h\}, \{\neg f, \neg g, \neg h\}\}$. Furthermore, the revision of the initial belief state by $\mathcal{R}(a, \psi)$ following either Satoh's or Dalal's approach yields $\mathcal{R}(a, \psi)$. However, the execution of a in $\{\neg f, \neg g, \neg h\}$ gives $\Phi(a, \{\neg f, \neg g, \neg h\}) = \{\{f, g, \neg h\}, \{f, \neg g, h\}\}$ which does not guarantee that the observation is true.

Example 7 shows that the non-determinism of actions leads to the non-determinism of regression, and therefore, would require a relaxed definition of regression or removal of the assumption that observations along the history are correct.

The proposed work is considered the problem when the agent repair/update her belief initial after observed something that contradict with her belief before. We want to update the initial belief instead of just accept the observation and move on because the observation after (an) action/actions can give us some information about the states before that. Let us illustrate this with an example:

Example 8. (Extended of Example 1) Consider a robot which was **told** that the light in the kitchen is off, the power of the house is on and it needs to turn the

[6] E.g., all non-probabilistic domains in www.icaps-conference.org/competitions

light on by a switch. A plan for the robot consists of two actions: go the the kitchen and switch the light on. This example can be encoded by a domain with the fluents $\{onLight, onPower, inKitchen\}$, 1 static causal law, 2 dynamic laws for action *switch*, 1 executability condition and 1 knowledge producing law:

$$
\begin{array}{cc}
goToKitchen \ \textbf{causes} \ inKitchen & switch \ \textbf{executable\_if} \ inKitchen \\
switch \ \textbf{causes} \ onLight \ \textbf{if} \ \neg onLight, onPower & look \ \textbf{determines} \ onLight \\
switch \ \textbf{causes} \ \neg onLight \ \textbf{if} \ onLight, onPower & \neg onLight \ \textbf{if} \ \neg onPower
\end{array}
$$

Similar with the example 1, after executed the plan, the robot realize that the light still off. At this point, if we just move from this observation, the robot may not know/believe that the power of the house could be *off*. By correct the initial belief, the robot should have more right information about the world than just accept the observation and move on (without correct its initial belief).

6 Conclusions

In this paper, we explore the problem of correcting the initial beliefs of an agent who, after executing a sequence of actions and making observations along the history, realizes that her initial beliefs are incorrect. Given an inconsistent history, the approach starts by regressing from the final to the first observation and revising the initial beliefs using the result of the regression. Unlike similar approaches explored in the literature, we consider sensing actions in the presence of static causal laws and propose algorithms for computing the correct initial beliefs. The paper presents an implementation of the algorithms which takes advantage of the declarative nature of Prolog. Our current work aims at improving the performance using tabling, parallelism, and intermediate results minimization. These directions help increase the performance by removing unnecessary computations and utilizing the available hardware. In addition, we intend to explore also other directions as probabilistic settings and ASP implementations. At present, our approach assumes also that observations are correct and actions are perfect. We therefore plan to consider the problem of having observations which are uncertain and imperfect actions in the future.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. JSL **50**(2), 510–530 (1985)
2. Aravanis, T.I., Peppas, P., Williams, M.: Iterated Belief Revision and Dalal's Operator. In: Hellenic Conference on Artificial Intelligence. pp. 26:1–26:4 (2018). https://doi.org/10.1145/3200947.3201038, https://doi.org/10.1145/3200947.3201038
3. Balduccini, M., Gelfond, M.: Diagnostic Reasoning with A-Prolog. Theory and Practice of Logic Programming **3**(4,5), 425–461 (2003)
4. Baral, C., McIlraith, S., Son, T.C.: Formulating diagnostic problem solving using an action language with narratives and sensing. In: KRR. pp. 311–322 (2000)
5. Borgida, A.: Language features for flexible handling of exceptions in information systems. ACM Transactions on Database Systems **10**(4), 563–603 (1985)
6. Dalal, M.: Investigations into theory of knowledge base revision. In: Proc. AAAI. pp. 449–479 (1988)
7. Delgrande, J.P., Peppas, P., Woltran, S.: General belief revision. J. ACM **65**(5), 29:1–29:34 (2018). https://doi.org/10.1145/3203409, https://doi.org/10.1145/3203409
8. Delgrande, J.P., Schaub, T., Tompits, H., Woltran, S.: Belief revision of logic programs under answer set semantics. In: KRR. pp. 411–421 (2008), http://www.aaai.org/Library/KR/2008/kr08-040.php
9. Hunter, A., Delgrande, J.P.: Iterated belief change due to actions and observations. J. Artif. Intell. Res. (JAIR) **40**, 269–304 (2011)
10. Hunter, A., Delgrande, J.P.: Belief change with uncertain action histories. J. Artif. Intell. Res. (JAIR) **53**, 779–824 (2015)
11. Lobo, J., Mendez, G., Taylor, S.: Adding knowledge to the action description language a. In: AAAI 97. pp. 454–459 (1997)
12. Peppas, P.: A panorama of iterated revision. In: David Makinson on Classical Methods for Non-Classical Problems, pp. 71–94. Springer Netherlands (2014)
13. Satoh, K.: Nonmonotonic reasoning by minimal belief revision. In: Proc. FGCS. pp. 455–462. Springer (1988)
14. Scherl, R., Levesque, H.: Knowledge, action, and the frame problem. Artificial Intelligence **144**(1-2) (2003)
15. Shapiro, S., Pagnucco, M., Lespérance, Y., Levesque, H.J.: Iterated belief change in the situation calculus. Artif. Intell. **175**(1), 165–192 (2011)
16. Son, T.C., Baral, C.: Formalizing sensing actions - a transition function based approach. Artificial Intelligence **125**(1-2), 19–91 (January 2001)
17. Tu, P., Son, T., Gelfond, M., Morales, R.: Approximation of action theories and its application to conformant planning. AIJ **175**(1), 79–119 (January 2011)
18. Winslett, M.: Reasoning about action using a possible models approach. In: AAAI. pp. 89–93 (1988)