*ECAI 2014*
*T. Schaub et al. (Eds.)*


*doi:10.3233/978-1-61499-419-0-147*

# On The Properties of Belief Tracking for Online Contingent Planning using Regression

**Ronen Brafman** and **Guy Shani**[1]

**Abstract.** Planning under partial observability typically requires some representation of the agent's belief state – either online to determine which actions are valid, or offline for planning. Due to its potential exponential size, efficient maintenance of a belief state is, thus, a key research challenge in this area. The state-of-the-art *factored belief tracking (FBT)* method addresses this problem by maintaining multiple smaller projected belief states, each involving only a subset of the variable set. Its complexity is exponential in the size of these subsets, as opposed to the entire variable set, without jeopardizing completeness. In this paper we develop the theory of regression to serve as an alternative tool for belief-state maintenance. Regression is a well known technique enjoying similar, and potentially even better worst-case complexity, as its complexity depends on the actions and observations that actually took place, rather than all actions and potential observations, as in the FBT method. On the other hand, FBT is likely to have better amortized complexity if the number of queries to the belief state is very large. An empirical comparison of regression with FBT-based belief maintenance is carried out, showing that the two perform similarly.

## 1 Introduction

To plan and act in a partially observable domain, an agent typically maintains some representation of its state of knowledge online. A complete description of the agent's state of knowledge, consisting of the set of possible states of the world (or a distribution over possible states, in the probabilistic case), is called the agent's *belief state*. Many planners for partially observable domains search directly in the space of belief states, known as the *belief space*.

Maintaining and updating an explicit belief state can be expensive because the number of possible states of the world can be exponential in the description size of a single state, i.e. the number of state variables. Thus, directly maintaining sets of states becomes unmanageable both space and time-wise as the problem grows. To alleviate this, methods that maintain a more compact, symbolic description of the set of possible states have been developed, such as methods based on BDDs [1], prime-implicates, CNFs, and DNFs [10]. Unfortunately, symbolic representations also have an exponential worst-case description, and when not, may be expensive to update. Furthermore, every representation that was suggested thus far, while being very compact for certain benchmark problems, demonstrated the worst-case performance on other benchmarks.

Still, planning algorithms can benefit from an important observation [4] — during planning and plan execution it is sufficient for the agent to answer only two types of queries with respect to a belief state: has the goal been achieved, and for each action, is it applicable, i.e., are its preconditions satisfied in this belief state. Furthermore, assuming conjunctive goals and preconditions, one need only need to check whether literals appearing in the goal and in actions preconditions are satisfied in the current belief state.

Bonet and Geffner leverage this insight to introduce a method that maintains multiple small belief states [2, 3], which are abstractions of the real belief state. Each abstraction contains enough information to answer one query, that is, to assess the value of a single variable. Often the value of a single variable depends only on the value of a small subset of the state variables, called *relevant variables*. As such, these abstract beliefs can be considerably smaller, requiring space exponential in the number of relevant variables only, which is known as the problem *causal width*. Using this representation, called *factored belief tracking (FBT)*, Bonet and Geffner show an impressive scaling up to much larger problems.

The CFF algorithm [4] introduced an implicit representation of belief states. It maintains a copy of the state variables for every time point, together with constraints over the value of these variables (as in a SAT encoding of planning problems). This representation grows linearly with the number of actions executed and answering a query regarding the current value of a variable requires solving an UNSAT problem. As information is obtained, the representation and constraints concerning earlier stages can be simplified.

Regression-based belief maintenance takes this lazy approach a step forward, maintaining only the initial belief state, the set of actions executed, and sensed observations [9]. This approach is similar in spirit to the Situation Calculus [5] where a state is represented in terms of the initial state and sequence of actions. Using this information, one could regress the conditions required currently (e.g., $p$) towards the initial belief state. If the regressed condition is implied by the initial state then we know that $p$ holds now. Otherwise, there exists some current possible state that does not satisfy $p$. In earlier work [9], we showed that, empirically, the regression-based method, coupled with some caching, is significantly more efficient than CFF's method over current benchmark problems.

In this paper we provide a detailed description of the regression method and its use for belief-state maintenance. We focus on *online* belief maintenance, where, after having performed a sequence of actions and observing some observations, the agent must determine whether the goal or a precondition literal $l$ hold. This is a slightly simpler task than *offline* belief maintenance, where the agent must consider arbitrary hypothetical sequences, and for each such sequence, not only determine the resulting belief state, but also, determine whether this sequence is possible.

Our first contribution is to extend Rintanen's formalism of regression [7] to handle observations, allowing for the use of regression

[1] Ben Gurion University, {brafman/shanigu}@bgu.ac.il

for belief-state queries in domains with partial observability. Earlier work on this topic [8] considered offline regression of whole branching plans, resulting in a more complex formalism that could be useful for offline planning. We show that regression enjoys similar complexity bounds to the FBT because one can ensure that the regression formula will contain relevant variables only. This occurs without effort when regressing actions, and provided that only observations relevant to the queries are regressed. Finally, we conduct experiments comparing belief update and query time of the regression method to approximate FBT, showing regression to be very efficient, scaling up similarly.

## 2 Background

We define the contingent planning model, its specification language, and review causal belief tracking and the concept of problem width.

**Model:** We focus on contingent planning problems with sensing. A contingent planning problem is a tuple of the form $\langle S, S_0, S_G, A, Tr, \Omega, O\rangle$, where $S$ is a set of states, $S_0 \subseteq S$ is the set of possible initial states, also called the *initial belief state* and is often denoted $b_I$, $S_G \subseteq S$ is the set of goal states, $A$ is a set of action symbols, and $Tr$ is the transition function, such that $Tr(s,a) \subseteq S$ is the set of states that can be reached by applying $a$ in state $s$, $\Omega$ is a set of observation symbols, and $O(a,s') \in \Omega$ is the observation obtained when $s'$ is reached following the application of $a$.

At each point in execution, there is a set of states considered possible, called the current belief state. $b_I$ is the initial belief state, and if $b$ is the current belief state, $a$ is executed, and $o$ is observed then the resulting belief state $\tau(b,a,o)$ is defined as:

$$\tau(b,a,o) = \{s' | s \in b \text{ and }, s' \in Tr(s,a), o \in O(a,s')\} \quad (2.1)$$

That is, states $s'$ that can result from the execution of $a$ in a state in $b$, such that $o$ can be observed. We extend this notation to a sequence $\bar{a}, \bar{o}$ of actions and observations recursively as follows:

$$\tau(b, \bar{a}\cdot a, \bar{o}\cdot o) = \tau(\tau(b,\bar{a},\bar{o}), a, o) \quad (2.2)$$

**Language:** A contingent planning problem is specified as a tuple $\langle P, \mathcal{A}, \varphi_I, G\rangle$. $P$ is a set of propositions, $\mathcal{A}$ is the set of actions, $\varphi_I$ is a propositional formula over $P$ in prime implicate form describing the possible initial states, and $G \subset P$ is the set of goal propositions.

A state of the world $s$ assigns a truth value to all elements of $P$. A *belief-state* is a set of possible states, and the initial belief state, $b_I$, consists of the set of states initially possible, i.e. $S_0 = b_I = \{s : s \models \varphi_I\}$. The goal is to arrive at a belief state in which all propositions in $G$ hold, i.e., $S_G = \{s \in S \text{ such that } s \models g_i \text{ for every } g_i \in G\}$. We assume $\varphi_I$ is in prime implicate form; this does not restrict the generality of the language. Alternatively one can assume multi-valued variables and an initial belief formula in the restrictive form of a conjunction of literals. To get an arbitrary initial state, one can add an artificial "initiating" action [2, 3], moving the complexity of the initial state into that action. There are sound reasons for this choice, but we prefer to use a non-restrictive initial state formula in a computationally efficient form. Unlike methods that have to progress the belief state in PI form, we require it from the initial state only.

A deterministic action, $a \in \mathcal{A}$, is a triple: {*pre*($a$), *effects*($a$), *obs*($a$)}. We shall use the more common $a(s)$ to denote $Tr(s,a)$. The action precondition, *pre*($a$), is a set of literals. The action effects, *effects*($a$), is a set of pairs, $(c_{a,l}, l)$, denoting conditional effects, where $c_{a,l}$ is a propositional formula and $l$ is a literal. For notational convenience, we'll assume one condition $c_{a,l}$ exists for every action $a$ and literal $l$[2]. In practice, $c_{a,l} = false$ for most literals, $l$, i.e., $l$ is typically not a possible conditional effect of $a$, and this pair can be omitted. *obs*($a$) is also a set of pairs, $\{(\omega_{a,o}, o) | o \in \Omega\}$, where $\omega_{a,o}$ is a propositional formula over $P$ and $o \in \Omega$.[3] Thus, $o = O(a,s')$ iff $s' \models \omega_{a,o}$.

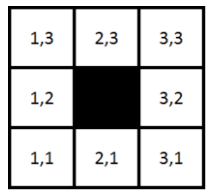


**Figure 1**: Localize $3 \times 3$. The goal is to get to cell $3,3$.

Since one observation must occur following the execution of an action, $\bigvee_{o\in\Omega} \omega_{a,o} = true$ for every $a \in A$. As sensing is deterministic, the $\omega_{a,o}$ for different $o$'s are mutually exclusive. Our implementation uses special *no-obs* observations denoting nothing-observed, but as *no-obs* can be treated like any other observation, we make no special distinction between it and "real" observations.

For deterministic actions, if $s \not\models pre(a)$ then $a(s)$ is undefined. Otherwise, $a(s)$ satisfies $l$ iff either (i) $s \models c_{a,l}$ or (ii) $s \models l \wedge \neg c_{a,\neg l}$.

We assume consistency, i.e., for every proposition $p$ and action $a$: $c_{a,p}$ and $c_{a,\neg p}$ are inconsistent. That is, an action $a$ executed in any state $s$ cannot make both $p$ and $\neg p$ true.

A non-deterministic action is defined as a set of deterministic actions, $a = \{a^1, \ldots, a^m\}$, one of which is non-deterministically selected when $a$ is executed. The actions in this set are restricted to have an identical precondition, i.e., for all $1 \le i,j \le m$, $pre(a^i) = pre(a^j)$. The set of states that are possible following the execution of $a$ in $s$ is thus $a(s) = a^1(s) \cup \cdots \cup a^m(s)$. However, each time $a$ is executed, exactly one of its elements, $a^i$ occurs, and the actual outcome is $a^i(s)$. This means that $o \in O(a,s')$ iff $o \in O(a^i, s')$ for some $1 \le i \le m$.

We restrict our attention to deterministic observations: every non-deterministic observation can be compiled away by adding an observable state variable, whose value changes non-deterministically following the action, representing the observation result.

**Example 1.** *As a running example, consider the* Localize *problem (Figure 1), where an agent in a grid must reach the top-right cell. The agent moves deterministically, and has a sensor that observes nearby walls. In this domain we have 4 movement actions, each with conditional effects modifying the agent location. For example, the* move-up *action would have a conditional effect* $(at_{3,1}, \neg at_{3,1} \wedge at_{3,2})$*, denoting that if the agent was at* $x = 3, y = 1$ *prior to the action, it is at cell* $x = 3, y = 2$ *following the action. The sensor activation action* checking *has conditional effects for specifying nearby walls. For example, it would contain a conditional effect* $(at_{3,1}, \neg wall_{up} \wedge wall_{down} \wedge wall_{right} \wedge \neg wall_{left})$*. There are 2 observations, RED denoting a wall, and GREEN denoting no wall, and 4 sensing actions, with* $\omega_{sense_d,RED} = wall_d$ *and* $\omega_{sense_d,GREEN} = \neg wall_d$*,* $d \in \{up, down, left, right\}$.

[2] In our examples, though, for ease of exposition, we abuse notation and allow conditional effects of the form $(c, e)$ where $c$ is a formula and $e$ is a conjunction of literals.

[3] Formally, $\Omega$ is specified implicitly as the set of observation symbols appearing in actions in $\mathcal{A}$.

## 2.1 Belief Tracking

In some applications belief tracking may require consistent maintenance of the current belief state of the agent. For planning algorithms we can focus on a narrower scope, which requires only answering specific queries concerning the set of possible states.

We distinguish between case of belief tracking for offline planning algorithms and online planning algorithms. In offline planning, given a sequence of actions $\bar{a}$ and a sequence of observations $\bar{o}$, the agent must know whether the sequence is executable starting from $b_I$, that is, whether $\tau(b_I, \bar{a}, \bar{o}) \neq \emptyset$, and whether the goal is satisfied following the sequence, i.e., whether $\tau(b_I, \bar{a}, \bar{o}) \models G$ [2].

In online planning, where an agent computes at each step the next action to perform, the task is slightly different. The agent has already executed a sequence of actions, $\bar{a}$, successfully, and sensed a sequence $\bar{o}$ of observations (i.e., the execution *history*). Thus, obviously $\tau(b_I, \bar{a}, \bar{o}) \neq \emptyset$, and the agent only needs to query, for a given literal $l$, whether $\tau(b_I, \bar{a}, \bar{o}) \models l$.

Computationally, the two problems are closely related. One can check whether a sequence of actions and observations is possible in a belief state by checking whether the preconditions of each action hold following the prefix of actions and observation that precedes it, and whether the condition $\omega_{a,o}$ holds for each action and sensed observation following the relevant prefix. Checking the applicability of an action requires one online belief-tracking query for each of its preconditions. However, checking whether an observation is possible could be complex, as $\omega_{a,o}$ is not necessarily a conjunction. Thus, one can answer an online belief-tracking query using one offline query, but not necessarily vice versa. We further discuss this issue later.

## 2.2 Problem Width

Bonet and Geffner [2] introduce a measure of the complexity of belief tracking, called *causal width*. Essentially, it is the number of variables that must be maintained when answering a belief tracking query regarding the value of some proposition $p$.[4]

**Definition 1** (BG). *$p$ is an immediate cause of a variable $q$, written $p \in Ca(q)$ iff $p \neq q$ and $p$ occurs (possibly negated) in the body $c$ of a conditional effect $(c, l)$ of some action $a$, and $l \in \{q, \neg q\}$. For an observation $o$, $p \in Ca(o)$ is similarly defined, where $c$ is replaced by $\omega_{a,o}$.*

**Definition 2** (BG). *$p$ is causally relevant to $q$ if $p = q$ or $p \in Ca(q)$ or $p \in Ca(r)$ and $r$ is causally relevant to $q$.*

**Definition 3** (BG). *$o \in O$ is evidentially relevant to $q$ if $q$ is casually relevant to $o$.*

**Definition 4** (BG). *$p$ is relevant to $q$ if $p$ is casually or evidentially relevant to $q$, if $p$ and $q$ appear in the same clause in $\varphi_I$[5] , or if $p$ is evidentially or causally relevant to $r$, and $r$ is relevant to $q$.*

Intuitively, the variables relevant to $p$ are variables whose value could impact the value of $p$ in the future because their value determines whether $p$ will be a conditional effect of some action or not; or because these variables impact our ability to make an observation that will impact our belief regarding whether $p$ holds or not.

[4] An alternative notion of width exists [3], which is useful for approximate belief tracking, but less relevant to this paper.

[5] BG's original definition does not consider the case that $p$ and $q$ appear in the same initial clause because of their assumption that all initial clauses are singletons.

**Definition 5** (BG). *The causal width of a variable $p$, $w(p)$ is the size of the context set of $p$: $ctx(p) = \{q | q \text{ is relevant to } p\}$. The causal width of a planning problem $\langle P, \mathcal{A}, \varphi_I, G\rangle$ is $\max\{w(p) | p \in P$ appears in the goal or a precondition$\}$.*

Computing the context set of a variable $p$ is a simple low-order poly-time procedure.

In many problems some variables are always known, or *determined* — their values are known initially and change only deterministically, conditioned on the value of other determined variables. This means that at each time point during the execution of a valid sequence of actions, their value is the same in all possible states. Thus, one can easily track their value independently of all other variables, paying linear time and space for each update. For the purpose of our analysis of belief tracking, we ignore these variables.

We now define a natural extension of width to the online setting, i.e., in the context of an action-observation sequence:

**Definition 6.** *$w_{\bar{a},\bar{o}}(p)$, the width of variable $p$ w.r.t. $\bar{a}$ and $\bar{o}$ is the width of $p$ with respect to the original planning problem, restricted to the set of actions appearing in $\bar{a}$ and the set of observations appearing in $\bar{o}$.*

The following notion will be useful to us later:

**Definition 7.** *Observation $o$ resulting from action $a$ is relevant to $p$ if $\omega_{a,o}$ contains a proposition relevant to $p$.*

# 3 Regression

We review Rintanen's [7] formalization of regression and extend it to address observations. First, we define the applicability of actions. Next we define regression over a single action with no observation, and then extend the results to regression over an observation. Finally, we discuss regression over a sequence of actions and observations.

## 3.1 Applicability

An action is *applicable* in state $s$ if $s$ satisfies its preconditions, i.e., $s \models pre(a)$. An action is *applicable* in a set of states $S$ if $S \models pre(a)$, that is, $\forall s \in S, s \models pre(a)$. An action $a$ is *applicable* given the initial belief state $b_I$ and an action-observation sequence $\bar{a}, \bar{o}$ if $a$ is applicable in $\tau(b_I, \bar{a}, \bar{o})$. Finally, $a_1, \ldots, a_n; o_1, \ldots, o_n$ is *applicable* in a belief state $b$ iff for every $i = 1, \ldots, n$ $a_i$ is applicable in $\tau(b, a_1, \ldots, a_{i-1}, o_1, \ldots, o_{i-1})$.

## 3.2 Regression Without Observations

Let $\phi$ be a propositional formula and $a$ a *deterministic* action. Recall that $c_{a,l}$ is the condition under which $l$ is an effect of $a$. The *regression* of $\phi$ with respect to $a$ is:

$$rg_a(\phi) = pre(a) \wedge \phi_{r(a)} \tag{3.1}$$

$$\phi_{r(a)} = \text{replace each literal } l \text{ in } \phi \text{ by } c_{a,l} \vee (l \wedge \neg c_{a,\neg l}) \tag{3.2}$$

**Example 2.** *Let us assume that the agent has executed a* move-up *action, and now regresses the formula $\phi = at_{3,2}$. The precondition of* move-up *is $\neg wall_{up}$. There is one condition in* move-up *that adds $at_{3,2}$, $(at_{3,1}, \neg at_{3,1} \wedge at_{3,2})$, and there is one condition that removes it, $(at_{3,2}, \neg at_{3,2} \wedge at_{3,3})$. Thus, the regression is $\neg wall_{up} \wedge (at_{3,1} \vee (at_{3,2} \wedge \neg at_{3,2}))$. Simplifying, we get $\neg wall_{up} \wedge at_{3,1}$.*

**Lemma 1.** *1.* $(\phi_1 \wedge \phi_2)_{r(a)} = (\phi_1)_{r(a)} \wedge (\phi_2)_{r(a)}$

2. $(\phi_1 \vee \phi_2)_{r(a)} = (\phi_1)_{r(a)} \vee (\phi_2)_{r(a)}$
3. *If* $\phi$ *is not a literal then* $(\neg\phi)_{r(a)} = \neg(\phi_{r(a)})$

*Proof.* As $\phi_{r(a)}$ is a syntactic manipulation of the formula $\phi$ that is a point-wise replacement of each literal by a formula, the above is immediate. □

**Theorem 1.** *[Rintanen08] Given a formula $\phi$, a deterministic action $a$, and a state $s$, $s \models rg_a(\phi)$ iff $a$ is applicable in $s$ and $a(s) \models \phi$.*

For a *non*-deterministic action $a = \{a^1, \ldots, a^m\}$ define [7]:

$$rg_a(\phi) = rg_{a^1}(\phi) \wedge \cdots \wedge rg_{a^m}(\phi) \tag{3.3}$$

For the non-deterministic case we have:

**Theorem 2.** *[Rintanen08] Let $\phi$ be a formula, $a$ an action, and $s$ a state. Then $s \models rg_a(\phi)$ iff $a$ is applicable in $s$, and for every $s' \in a(s)$, $s' \models \phi$.*

## 3.3 Regression with Observations

We now extend regression to an action and an ensuing observation. Suppose we want to validate that $\phi$ holds following the execution of $a$ in some state $s$ given that we observed $o$. Thus, we need to ensure that following $a$, if $\omega_{a,o}$ holds then $\phi$ holds. This leads to the following definition:

$$rg_{a,o}(\phi) = rg_a(\omega_{a,o} \rightarrow \phi) \tag{3.4}$$

**Theorem 3.** *Given a formula $\phi$, an action $a$, an observation $o$, and a state $s$, $s \models rg_{a,o}(\phi)$ iff $a$ is applicable in $s$ and $\tau(\{s\}, a, o) \models \phi$.*

*Proof.* $s \models rg_{a,o}(\phi)$ iff (by definition) $s \models rg_a(\omega_{a,o} \rightarrow \phi)$ iff (Theorem 2) $a$ is applicable in $s$ and for every $s' \in a(s)$, $s' \models \omega_{a,o} \rightarrow \phi$. By definition, $s' \models \omega_{a,o}$ iff $o = O(a, s')$. Thus, $s \models rg_{a,o}(\phi)$ iff $a$ is applicable in $s$ and for every $s' \in a(s)$, we have that $o = O(a, s')$ implies $s' \models \phi$. To conclude the proof, $\tau(\{s\}, a, o)$ contains precisely all states in $a(s)$ in which it is possible to observe $o$ following $a$. □

The following is an immediate corollary:

**Corollary 1.** *For a belief state $b$, $b \models rg_{a,o}(\phi)$ iff $a$ is applicable in $b$ and $\tau(b, a, o) \models \phi$.*

When $s \models rg_{a,o}(\phi)$ it is not necessarily the case that $o$ can be observed following the execution of $a$ in $s$, only that *if* $o$ is observed then $\phi$ must hold. Thus, the regression of an observation cannot be decomposed: $rg_{a,o}(\phi) \not\equiv rg_a(\omega_{a,o}) \rightarrow rg_a(\phi)$

While $rg_{a,o}(\phi)$ implies $rg_a(\omega_{a,o}) \rightarrow rg_a(\phi)$, the other direction is false for non-deterministic actions. For example, suppose that $a$ has two possible effects, $p, \neg p$, and they are observable. $reg_a(p)$, however, is *false* because there is no condition under which we are *guaranteed* to see $p$ after $a$. For deterministic actions, however:

**Theorem 4.** *Given a formula $\phi$, a* deterministic *action $a$, an observation $o$, and a state $s$, $s \models rg_{a,o}(\phi)$ iff $s \models rg_a(\omega_{a,o}) \Rightarrow s \models rg_a(\phi)$.*

*Proof.* $s \models rg_{a,o}(\phi)$ iff (by definition) $s \models rg_a(\omega_{a,o} \rightarrow \phi)$ iff $a$ is applicable in $s$ and $a(s) \models \omega_{a,o} \rightarrow \phi$; iff $a$ is applicable in $s$ and $a(s) \models \omega_{a,o}$ implies $a(s) \models \phi$;[6] Using Theorem 2 $a$ is applicable in $s$ and $a(s) \models \omega_{a,o}$ iff $s \models rg_a(\omega_{a,o})$ and $a$ is applicable in $s$ and $a(s) \models \phi$ iff $s \models rg_a(\phi)$. Consequently, $s \models rg_{a,o}(\phi)$ iff $s \models rg_a(\omega_{a,o})$ implies $s \models rg_a(\phi)$, as required. □

[6] When $a$ is non-deterministic, only one direction of the last step is valid, i.e., $a(s) \models \phi \rightarrow \psi$ implies $a(s) \models \phi \Rightarrow a(s) \models \psi$, but $a(s) \models \phi \Rightarrow a(s) \models \psi$ does not imply $a(s) \models \phi \rightarrow \psi$.

Finally, regression has a number of useful properties:

**Theorem 5.** *For any two formulas $\phi_1$ and $\phi_2$ we have:*

1. $\phi_1 \equiv \phi_2 \Rightarrow rg_{a,o}(\phi_1) \equiv rg_{a,o}(\phi_2)$
2. $\phi_1 \equiv \phi_2 \Rightarrow rg_a(\phi_1) \equiv rg_a(\phi_2)$
3. $rg_{a,o}(\phi_1 \wedge \phi_2) \equiv rg_{a,o}(\phi_1) \wedge rg_{a,o}(\phi_2)$
4. *For deterministic $a$,* $rg_{a,o}(\phi_1 \vee \phi_2) \equiv rg_{a,o}(\phi_1) \vee rg_{a,o}(\phi_2)$

*Proof.* 1. Follows immediately from $\tau(\{s\}, a, o) \models \phi_1$ iff $\tau(\{s\}, a, o) \models \phi_2$ and Theorem 3.
2. Identical to 1, using Theorem 2 instead of Theorem 3.
3. Suppose $s \models rg_{a,o}(\phi_1 \wedge \phi_2)$. By Theorem 3 $\tau(\{s\}, a, o) \models \phi_1 \wedge \phi_2$, implying $\tau(\{s\}, a, o) \models \phi_1$ and $\tau(\{s\}, a, o) \models \phi_2$. Applying Theorem 3 again, we get $s \models rg_{a,o}(\phi_1)$ and $s \models rg_{a,o}(\phi_2)$. The other direction is identical.
4. Same as 3, noting that for deterministic $a$ $\tau(\{s\}, a, o) \models \phi_1 \vee \phi_2$ implies $\tau(\{s\}, a, o) \models \phi_1$ or $\tau(\{s\}, a, o) \models \phi_2$.

□

## 3.4 Regression Over a Sequence

We extend the definition of regression recursively to a sequence of actions and observations $\bar{a}, \bar{o}$ as follows:

$$rg_{\bar{a}\cdot a, \bar{o}\cdot o}(\phi) = rg_{\bar{a},\bar{o}}(rg_{a,o}(\phi)); \quad rg_{\epsilon,\epsilon}(\phi) = \phi \tag{3.5}$$

where $\epsilon$ is the empty sequence.

Theorem 3 generalizes as follows:

**Theorem 6.** *Given a formula $\phi$, an action-observation sequence $\bar{a}, \bar{o}$, and a belief state $b$, $b \models rg_{\bar{a},\bar{o}}(\phi)$ iff $\bar{a}, \bar{o}$ is applicable in $b$ and $\tau(b, \bar{a}, \bar{o}) \models \phi$.*

*Proof.* Proof by induction on $|\bar{a}|$. The base case is immediate. For the inductive step: $b \models rg_{\bar{a}\cdot a, \bar{o}\cdot o}(\phi)$ iff (by definition of $rg$) $b \models rg_{\bar{a},\bar{o}}(rg_{a,o}(\phi))$ iff (using the inductive hypothesis) $\bar{a}, \bar{o}$ is applicable in $b$ and $\tau(b, \bar{a}, \bar{o}) \models rg_{a,o}(\phi)$. Applying Corollary 1, this holds iff $a$ is applicable in $\tau(b, \bar{a}, \bar{o})$ and $\tau(\tau(b, \bar{a}, \bar{o}), a, o) \models \phi$. As $\tau(\tau(b, \bar{a}, \bar{o}), a, o) = \tau(b, \bar{a} \cdot a, \bar{o} \cdot o)$ the latter is equivalent to: $\bar{a} \cdot a, \bar{o} \cdot o$ is applicable in $b$ and $\tau(b, \bar{a} \cdot a, \bar{o} \cdot o) \models \phi$, as required. □

# 4 Belief Tracking by Regression

Bonet and Geffner [2] track the belief state by progressing, for every $p \in P$, the belief state projected to the context set of $p$, which they call a *beam*. This set is closed under relevance, so it is easy to maintain, and it contains $p$. To determine if $p$ holds, they check whether all belief states in this set satisfy $p$. This yields a sound and complete method for belief tracking, called *factored belief tracking* (FBT). They later [3] proposed an incomplete version of FBT, called *causal belief tracking* (CBT).

**Theorem 7** (BG)**.** *The time and space complexity of online belief tracking over a propositional description using FBT is $O(2^w)$, where $w$ is the width of the problem.*

## 4.1 Method and Properties

We propose an alternative method for online belief tracking based on regression, which is a direct consequence of Theorem 6:

**Theorem 8.** *For any literal $l$ and action-observation sequence $\bar{a}, \bar{o}$ that was applied in $b_I$, we have that $\tau(b_I, \bar{a}, \bar{o}) \models l$ iff $b_I \models rg_{\bar{a}\bar{o}}(l)$.*

**Example 3.** *We illustrate how an agent that has soundly executed two* move-up *actions, cannot grantee that the goal $at_{3,3}$ holds in all possible states. We would regress $\phi = \neg at_{3,3}$ through the action sequence backward, starting with the last action. There is one condition in* move-up *that adds $at_{3,3}$ — $(at_{3,2}, \neg at_{3,2} \wedge at_{3,3})$, and there is no condition in* move-up *that removes $at_{3,3}$, hence, the result of the regression through the last action would be $false \vee (\neg at_{3,3} \wedge \neg at_{3,2}) = \neg at_{3,3} \wedge \neg at_{3,2}$.*

*We can now regress the two literals independently through the first action in the sequence. Focusing on $\neg at_{3,2}$, we see two relevant conditional effects — $(at_{3,1}, \neg at_{3,1} \wedge at_{3,2})$ and $(at_{3,2}, \neg at_{3,2} \wedge at_{3,3})$, the first removing $\neg at_{3,2}$ and the second adding it. Thus, the regression result through the first action is $at_{3,2} \vee (\neg at_{3,2} \wedge \neg at_{3,1}) = at_{3,2} \vee \neg at_{3,1}$ , combined with the regression for $\neg at_{3,3}$, the simplified complete regression is $\neg at_{3,3} \wedge \neg at_{3,2} \wedge \neg at_{3,1}$. If the initial state formula allows the agent to be initially in any place $\phi_I = (oneof\ at_{1,1} ... at_{3,3})$, then there are satisfying assignments to the $\phi_I \wedge rg(\neg at_{3,3})$, such as $at_{1,1}$. Thus, we cannot prove that following two movements upwards we have reached the goal cell $3, 3$.*

Thus, we can now answer online belief queries using regression. However, one practical problem with this method is that $rg_{a,o}(\cdot)$ contains $pre(a)$ and hence when we regress repeatedly over a sequence, as in $rg_{\bar{a},\bar{o}}(\cdot)$, we will also have to regress the variables in $pre(a)$. This can adversely affect the size of $rg_{\bar{a},\bar{o}}(\cdot)$ and the number of variables it involves, leading, in the worst case, to a formula exponential in $P$. Fortunately, this is not necessary for online belief tracking, because the preconditions of already executed actions were already regressed and shown to hold in the initial belief state. Using the $\phi_{r(a)}$ operation we define:

$$rg^*_\epsilon(\phi) = \phi; \quad rg^*_{a\cdot\bar{a}}(\phi) = [rg^*_{\bar{a}}(\phi)]_{r(a)} \tag{4.1}$$

Thus, essentially, $rg^*$ is the same as $rg$, except that we avoid regressing $pre(a)$. As before:

$$rg^*_{a,o}(\phi) = rg^*_a(\omega_{a,o} \rightarrow \phi) \tag{4.2}$$

As with $rg$, $rg^*$ is also extended recursively to sequences.

**Theorem 9.** *For any literal $l$ and action-observation sequence $\bar{a}, \bar{o}$ that is applicable in $b_I$, we have that $b_I \models rg_{\bar{a}\bar{o}}(l)$ iff $b_I \models rg^*_{\bar{a}\bar{o}}(l)$*

*Proof.* By induction on $|\bar{a}|$, exploiting the observation that if $a$ is applicable in $b$ then $b \models pre(a)$ and thus $b \models rg_a(\phi)$ iff $b \models rg^*_a(\phi)$. This immediately extends to $rg_{a,o}$ and $rg^*_{a,o}$ as they are defined using $rg_a$. Formally, the base case (empty sequence) is immediate. Let $l$ be a literal and $a \cdot \bar{a}, o \cdot \bar{o}$ be an action-observation sequence that is applicable in $b_I$. Let $b = \tau(b_I, a, o)$. By the induction hypothesis, observing that if $a \cdot \bar{a}, o \cdot \bar{o}$ is applicable in $b_I$ then $\bar{a}, \bar{o}$ is applicable in $b$, we obtain: $b \models rg_{\bar{a},\bar{o}}(l)$ iff $b \models rg^*_{\bar{a},\bar{o}}(l)$. Thus, $b_I \models rg_{a\bar{a},o\bar{o}}(l)$ iff (by definition of $rg$ on sequences) $b_I \models rg_{a,o}(rg_{\bar{a},\bar{o}}(l))$ iff (by the observation above) $b_I \models rg^*_{a,o}(rg_{\bar{a},\bar{o}}(l))$ iff (by the induction hypothesis) $b_I \models rg^*_{a,o}(rg^*_{\bar{a},\bar{o}}(l))$. In the latter case we show that regression of equivalent formulas is equivalent (Theorem 5). □

Finally, it is not surprising that when verifying the validity of $p$, we do not care about observations that are irrelevant to $p$.

**Lemma 2.** *Let $\bar{o}_1$ be an observation sequence, and let $\bar{o}_2$ be a subsequence of $\bar{o}_1$ that omits observations irrelevant to $p$. Then, $b_I \models rg^*_{\bar{a}\bar{o}_1}(l)$ iff $b_I \models rg^*_{\bar{a}\bar{o}_2}(l)$.*

**Theorem 10.** *For any literal $l$, the time and space complexity of determining whether $l$ is valid following $\bar{a}, \bar{o}$ is $O(2^w)$, where $w = w_{\bar{a},\bar{o}'}(l)$ and $\bar{o}'$ is the subsequence of $\bar{o}$ containing only observation relevant to $l$ .*

*Proof.* For any literal $l$, to determine whether it holds following $\bar{a}, \bar{o}$, we need to compute $rg^*_{\bar{a},\bar{o}'}(l)$ (where $\bar{o}'$ are the observations relevant to $l$) and to check whether $b_I \models rg^*_{\bar{a},\bar{o}}(l)$. By definition of relevance, $rg^*_{\bar{a},\bar{o}}(l)$ contains only propositions relevant to $l$ with respect to the planning problem, restricted to the actions and observations in $\bar{a}$ and $\bar{o}$, which we denote by $w$. The size of the regression formula is at most exponential in $w$. Potentially, its size can grow by a polynomial factor following each regression step, becoming exponential in the length, rather than $w$. However, we can maintain size exponential in $w$ by simplifying the formula following each step, at a cost that is at most exponential in $w$ for each step.

Finally, to check whether $b_I \models rg^*_{\bar{a},\bar{o}}(l)$ we can convert $rg^*_{\bar{a},\bar{o}}(l)$ into CNF (again, in time at most exponential in $w$) and check that each clause in it is entailed by $b_I$. Since $b_I$ is in PI form, this takes polynomial time in the input size for each clause. □

Thus, regression may have a practical advantage over FBT when the sequence of actions has lower width than the original planning problem. We note that the offline query: "is $\bar{a}, \bar{o}$ executable in $b_I$?" is not typical of online planning. In an online planning process, one would query each of the $\bar{a}, \bar{o}$ prefixes earlier, and the $\bar{a}, \bar{o}$ queries will be executed only after all the prefixes are known to be executable. Thus, to determine executability, one needs query only regarding the preconditions of the last action in the sequence.

## 4.2 Implementation

In practice, the actual run-time of regression can be improved.

First, during planning we perform many regression queries that lead to the learning of new facts. For example, we always learn that the preconditions of an executed action are valid. These learned facts can be cached at each step and used to simplify formulas generated when answering future queries.

Second, we can utilize observations that were made following a deterministic action to constrain and simplify the initial state. If we observed $o$ following $a$, we regress $\omega_{a,o}$ through the preceding sequence of action and observations, and obtain a formula $\varphi$ that must hold in the initial state. Thus, we can replace $b_I$ with $b_I \wedge \varphi$.

Sometimes $\varphi$ is a unit literal, e.g., if $o$ is an observation of a static fact. In that case, we can insert it using unit propagation into $\varphi$ maintaining PI form in polynomial time. Sometimes, however, it could be a more complex formula, requiring an exponential price for converting $b_I \wedge \varphi$ to PI form. In our current implementation, we do not maintain a PI form, but rather use CNF and determine validity using UNSAT queries to a SAT solver. Theoretically, these queries can take exponential time to answer, but in practice they are very fast.

This technique has two important advantages. First, simpler initial state formulas imply faster inference in future queries. Second, once an observation is regressed and added to the initial state, we can ignore it when answering future regression queries. Furthermore, we avoid observation relevance analysis, because once the regressed observation has been conjoined with the initial state, we have the the simpler case of regression without observations.

**Example 4.** *Let us assume that the agent has executed the sensor activation action* checking*, and then the* observe-wall-up *action, observing the green light observation. We can now regress $w_{a,o} = \neg wall_{up}$ through the action sequence. The* observe-wall-up *is a sensing action with no effects, thus regressing through it has no effect on the regressed formula. We hence need to regress $\neg wall_{up}$ through the checking action. For this action $c_{a,l} = at_{1,1} \vee at_{1,2} \vee at_{3,1} \vee at_{3,2}$ — the list of cells where there is no wall above the*

*agent. The condition* $c_{a,\neg l} = at_{2,1} \vee at_{1,3} \vee at_{2,3} \vee at_{3,3}$*, and* $\neg c_{a,\neg l} = \neg at_{2,1} \wedge \neg at_{1,3} \wedge \neg at_{2,3} \wedge \neg at_{3,3}$*. Thus, the regressed term* $c_{a,l} \vee (l \wedge \neg c_{a,l}) = at_{1,1} \vee at_{1,2} \vee at_{3,1} \vee at_{3,2} \vee (\neg wall_{up} \wedge \neg at_{2,1} \wedge \neg at_{1,3} \wedge \neg at_{2,3} \wedge \neg at_{3,3}$*, we can now conjoin the initial state formula* $(oneof\ at_{1,1} ... at_{3,3})$ *with this regressed formula, limiting the set of possible initial state only to* $(oneof\ at_{1,1} at_{1,2} at_{3,1} at_{3,2})$*.*

## 5 Empirical Evaluation

We now demonstrate the practical value of regression, showing it to scale up well. We experiment with large benchmarks, that are currently unsolvable using any planner. As such, in all these domains, we use a simple and fast domain-specific heuristic for action selection. In each step the "planner" chooses an action, runs a regression query to check if its preconditions hold, executes it, and runs a second regression query to check if the goal has been reached. If an observation is sensed following the action, the planner also regresses the observed value and caches the resulting information. Thus, in every step, there can be up to 3 different regression operations. We report the average step time, rather than the pure regression time, to be comparable to previous experiments. For every problem, we run 25 iterations, and report the average time in seconds.

Our heuristic is not trivial. For example, in the battleship domain, once a cell containing a ship is hit, we hit its neighboring cells until the entire ship was drowned. We must thus check for a set of cells whether they were hit, or contain a ship. Using caching, these queries are not regressed, and are thus sound but incomplete. That being said, it is less efficient than the heuristic implemented in CBT. For example, we require about 50 shots to solve Battleship $5 \times 5$ while CBT requires about 39 shots.

We compare regression to CBT[3], which is a more advanced, approximate implementation of the ideas behind FBT. This method is sound but incomplete in general. Furthermore, the current CBT code is non-generic, implementing only three domains: Battleship, Wumpus, and Minesweeper, where it performs very well. However, it can be executed only on these domains. Also, its implementation makes use of a manually designed multi-valued variable representation. While the use of multi-valued variables does not impact the worst-case complexity of the method, it is reasonable to believe that it lends additional practical efficiency, compared to the use of a more generic, PDDL-like propositional representation that we use.

Our experiments were run on a Windows Server machine with 24 2.66 GHz cores (although only a single core is used), and 32GB RAM. Regression is implemented in C# while CBT uses Cygwin.

**Table 1**: Comparing decision time (secs) of regression and CBT.

| Domain | Regression | CBT |
|---|---|---|
| Battleship $10 \times 10$ | 4.2E-3 | 5.7E-5 |
| Battleship $20 \times 20$ | 1.0E-2 | 7.4E-5 |
| Battleship $30 \times 30$ | 2.0E-2 | 8.5E-5 |
| Battleship $40 \times 40$ | 3.8E-2 | 9.5E-5 |
| Minesweeper $8 \times 8$ | 7.2E-2 | 8.3E-3 |
| Minesweeper $16 \times 16$ | 2.8E-1 | 1.2E-2 |
| Minesweeper $32 \times 32$ | 7.4E-1 | |
| Large Wumpus $20 \times 20$ | 4.5E-3 | 2.4E-3 |
| Large Wumpus $30 \times 30$ | 6.2E-3 | 4.7E-2 |
| Large Wumpus $40 \times 40$ | 9.7E-3 | 2.8E-3 |
| Large Wumpus $50 \times 50$ | 1.4E-2 | 1.3E-2 |

Nevertheless, as shown in Table 1 our regression-based method which is sound, complete, and uses a generic implementation, accepting domains in a PDDL-like language, does very well. It is able to scale-up to similar domain sizes as CBT, although in the Battleship domain it is much slower. It is interesting to observe that in the Wumpus domain, the only domain in which boolean variables are used by CBT, the two methods are virtually identical in performance. In addition, as shown in Table 2 in domains not supported by the current CBT code, regression-based belief tracking scales very well to domain sizes that cannot be handled by any other method.

**Table 2**: Regression time for challenging benchmark domains.

| Domain | Regression |
|---|---|
| Localize 20 | 4.6E-3 |
| Localize 30 | 3.4E-2 |
| Localize 40 | 7.9E-2 |
| Localize 50 | 1.6E-1 |
| RockSample $8 \times 8$ | 1.0E-4 |
| RockSample $16 \times 16$ | 7.6E-4 |
| RockSample $32 \times 32$ | 5.5E-3 |
| MasterMind 6c,4p | 5.1E-3 |
| MasterMind 8c,4p | 6.2E-3 |
| MasterMind 10c,6p | 1.0E-1 |

## 6 Conclusion

In this paper we discuss the theory of regression, developing it as a practical tool for online belief tracking in contingent domains, showing that it enjoys potentially better worst-case theoretical guarantees than FBT. We evaluate the use of regression empirically, showing that it scales up very well on all current contingent benchmark domains.

Regression naturally enjoys a focus on relevant variables only, which is also the main source of efficiency of FBT. As regression takes a lazy approach, constructing formulas during queries, it may not be as beneficial for planners that require many queries. Repeatedly checking the precondition validity of a large set of actions may well be less efficient using regression than using a DBN [6] or FBT.

The success of approximate CBT techniques points to an interesting line of future work focusing on approximate regression methods. For example, by possibly weakening the regression formula in some cases (recall that we regress $\neg l$, not $l$), maintaining a simple syntactic form, one might be able to farther simplify its computational cost in practice, at the price of some loss of completeness.

**Acknowledgments:** This work was supported by ISF Grant 933/13 and by the Lynn and William Frankel Center for Computer Science.

## REFERENCES

[1] Piergiorgio Bertoli, Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso, 'Mbp: a model based planner', in *IJCAI01 Workshop on Planning under Uncertainty and Incomplete Information*, (2001).

[2] Blai Bonet and Hector Geffner, 'Width and complexity of belief tracking in non-deterministic conformant and contingent planning', in *AAAI*, (2012).

[3] Blai Bonet and Hector Geffner, 'Causal belief decomposition for planning with sensing: Completeness results and practical approximation.', in *IJCAI*, (2013).

[4] Jörg Hoffmann and Ronen I Brafman, 'Conformant planning via heuristic forward search: A new approach', *Artificial Intelligence*, **170**(6), 507–541, (2006).

[5] Hector Levesque, Fiora Pirri, and Ray Reiter, 'Foundations for the situation calculus', *Linköping Electronic Articles in Computer and Information Science*, **3**(18), (1998).

[6] Yan Lin and Marek J. Druzdzel, 'Computational advantages of relevance reasoning in bayesian belief networks', in *UAI*, pp. 342–350, (1997).

[7] Jussi Rintanen, 'Regression for classical and nondeterministic planning', in *ECAI*, pp. 568–572, (2008).

[8] R. Scherl, T. Cao Son, and C. Baral, 'State-based regression with sensing and knowledge', *International Journal of Software and Informatics*, (2009).

[9] Guy Shani and Ronen I Brafman, 'Replanning in domains with partial information and sensing actions', in *IJCAI*, pp. 2021–2026. AAAI Press, (2011).

[10] Son Thanh To, Enrico Pontelli, and Tran Cao Son, 'On the effectiveness of cnf and dnf representations in contingent planning', in *IJCAI*, pp. 2033–2038. AAAI Press, (2011).