Artificial Intelligence 241 (2016) 131–152

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Online belief tracking using regression for contingent planning

Ronen I. Brafman, Guy Shani *

Ben-Gurion University of the Negev, Be'er Sheva, Israel

ARTICLE INFO

Article history:
Received 22 March 2015
Received in revised form 27 July 2016
Accepted 19 August 2016
Available online 7 September 2016

Keywords:
Contingent planning
Partial observability
Non-deterministic planning
Regression
Belief

ABSTRACT

In online contingent planning under partial observability an agent decides at each time step on the next action to execute, given its initial knowledge of the world, the actions executed so far, and the observation made. Such agents require some representation of their belief state to determine which actions are valid, or whether the goal has been achieved. Efficient maintenance of a belief state is, given its potential exponential size, a key research challenge in this area. In this paper we develop the theory of regression as a useful tool for belief-state maintenance. We provide a formal description of regression, discussing various alternatives and optimization techniques, and analyze its space and time complexity. In particular, we show that, with some care, the regressed formula will contain variables relevant to the current query only, rather than all variables in the problem description. Consequently, under suitable assumptions, the complexity of regression queries is at most exponential in its *contextual width*. This parameter is always upper bounded by Bonet and Geffner's *width* parameter, introduced in their state-of-the-art *factored belief tracking (FBT)* method. In addition, we show how to obtain a poly-sized circuit representation for the online regression formula even with non-deterministic actions. We provide an empirical comparison of regression with FBT-based belief maintenance, showing the power of regression for online belief tracking. We also suggest caching techniques for regression, and demonstrate their value in reducing runtime in current benchmarks.

1. Introduction

Agents that act in the real world are often limited to partial knowledge about the world, obtained through their sensors. This is typically known as planning under partial observability. Agents that plan in a partially observable domain, typically maintain some representation of their 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 probability 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 [3], prime-implicates, CNF, and DNF [27]. Unfortunately, symbolic representations also have an exponential worst-case

* Corresponding author.
E-mail addresses: brafman@cs.bgu.ac.il (R.I. Brafman), shanigu@bgu.ac.il (G. Shani).

http://dx.doi.org/10.1016/j.artint.2016.08.005
0004-3702/

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 [15] – 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 this action applicable?* That is, whether the action's precondition is satisfied in the belief state.

The regression-based method we study takes a lazy approach to belief maintenance, maintaining only the initial belief state, the set of actions executed, and sensed observations. This approach is similar in spirit to the Situation Calculus [16] 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 [8], we implemented this approach within an online contingent planner and showed that, empirically, the regression-based method, coupled with some caching, is highly efficient on current benchmark problems. In this paper we provide detailed description and analysis of regression-based belief-state maintenance, focusing on *online* belief maintenance – which is sometimes called *filtering* [2,25]. In online belief maintenance, the agent, after having performed a sequence of actions and observing some observations, must determine whether the goal or a precondition literal l holds. This is a somewhat 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.

There is a long line of research on regression in the area of KR (e.g., [21,13,24]). Regression is also a key component of the circuit-based filtering approach proposed by Shahaf and Amir [25]. Yet, none of the previous regression-based methods have been shown to be empirically efficient or competitive for modern, state-space planning, whereas the method we describe has been implemented as part of a number of state-of-the-art contingent planners [7,8].

Our first contribution is to extend Rintanen's formalism of regression [22] to handle observations, allowing us to use regression for online belief-state queries in domains with partial observability. Exploiting our focus on online belief maintenance, we obtain a method that is on par with the state of the art, as our empirical evaluation shows. On the theoretical side, we exploit the notion of width [19,5,6] to provide a tighter, width-based bound on the complexity of regression. We show that regression enjoys similar, and potentially better, complexity bounds to those of the state-of-the-art *factored belief tracking (FBT)* method [6]. FBT maintains an explicit model of a factored belief state which utilizes a notion of relevant variables. We show that one can ensure that the regression formula, too, will contain relevant variables only. This occurs naturally when regressing actions, but requires more care for regression of observations. Moreover, the worst-case complexity of regression depends only on the actions that actually appear in the sequence, as opposed to the entire set of actions in the progression-based FBT, potentially leading to a lower width notion, which we call *contextual width*.

Then, we use a well-known technique to compile non-deterministic actions into deterministic actions. Now, with deterministic actions only, one can easily update the initial state formula with the regression of each observation once, ignoring these observations afterward while maintaining completeness. This compilation leads to a surprising result – the existence of a poly-sized circuit representation of the regression formula (assuming polynomial horizon), which typically requires exponential space in other forms. Earlier work considered this to be possible only for domains with deterministic actions.

Next, we empirically evaluate our method, comparing the belief update and the query time of the regression method to approximate FBT, showing regression to be very efficient, scaling up similarly. We also empirically analyze the effect of simple caching – the main optimization to regression we suggest. We conclude with a discussion of related work.

2. Background

We begin by defining the contingent planning model and its specification language, following in most places the definitions of Bonet and Geffner (BG henceforth) [6]. We then review the various concepts of problem width introduced by Bonet and Geffner.

2.1. Model

We focus on contingent planning problems with sensing. A contingent planning problem is a tuple of the form $\langle S, b_I, S_G, A, Tr, \Omega, O\rangle$, where S is a set of states, $b_I \subseteq S$ is the set of possible initial states, also called the *initial belief state*, $S_G \subseteq S$ is the set of goal states, A is a set of action symbols, and Tr is a transition function, such that $Tr(s,a) \subseteq S$ is the set of states that can be reached by applying a in state s, Ω 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. Without loss of generality, we assume an observation follows each action.

In this paper we allow for non-deterministic outcomes of actions, but restrict ourselves to deterministic observations. Semantically, this is not a limiting assumption, as non-deterministic observations can be compiled away using non-deterministic actions; If $s' \in Tr(a,s)$ and $O(a,s') = \{o_1, o_2\}$, copy s' into two new states s'_1, s'_2 such that $s'_1, s'_2 \in Tr(a,s)$. Define $O(a,s'_1) = o_1$ and $O(a,s'_2) = o_2$. Transitions from s_1 and s_2 are identical to those from s.

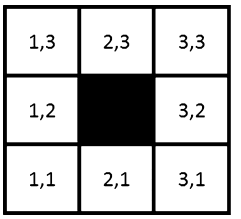


Fig. 1. Localize 3×3. The goal is to get to cell 3, 3.

At each point in execution, there is a set of possible states, called the current *belief-state*. The initial belief state – the set of initially possible states – is denoted b_I. 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:

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

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

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

As an observation always follows an action, $|\bar{a}| = |\bar{o}|$.

2.2. Language

A contingent planning problem is specified as a tuple $\langle P, \mathcal{A}, \varphi_I, \varphi_G \rangle$. P is a set of atomic boolean propositions, $\mathcal{A}$ is the set of actions, φ_I is a propositional formula over P in conjunctive normal form that represents the initial state, and φ_G is a propositional formula over P that represents the goal condition.

A state of the world s assigns a truth value to all elements of P. The initial belief state, b_I, consists of the set of states that satisfy φ_I, i.e. $b_I = \{s : s \models \varphi_I\}$. The goal is to achieve a belief state in which φ_G holds, i.e., $b \models \varphi_G$.

A deterministic action, $a \in \mathcal{A}$, is a triple: $\{pre(a), effects(a), obs(a)\}$. We shall use the common $a(s)$ to denote $Tr(s, a)$ in the deterministic case. The action precondition, $pre(a)$, is a propositional formula. 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 assume that only one condition $c_{a,l}$ exists for every action a and literal l. 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 $o \in \Omega$. Formally, Ω is specified implicitly as the set of observation symbols appearing in actions in $\mathcal{A}$. Thus, $o = O(a, s')$ iff $s' \models \omega_{a,o}$.

Formally, an observation always occurs following the execution of an action. Thus, for every $a \in A$, $\bigvee_{o \in \Omega} \omega_{a,o} \equiv true$. Furthermore, sensing is deterministic and therefore $\omega_{a,o}$ for different observations must be mutually exclusive. In practice, not all actions are followed by a meaningful observation. This can be captured by a special, uninformative, *no-obs* observation. If a is followed by a *no-obs*, then $\omega_{a,no\text{-}obs} = true$, i.e., it is the only observation possible following a. 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 that the conditions are mutually exclusive, i.e., for every atomic proposition p and action a: no state in which both $c_{a,p}$ and $c_{a,\neg p}$ hold is reachable. That is, an action a executed in a reachable 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 – as the applicability of the action does not depend on the outcome, and identical observations – as the observation depends on the identity of the action, which in this case is the non-deterministic a, and the outcome state. Formally, we require that for all $1 \leq i, j \leq m$, $pre(a^i) = pre(a^j)$ and $obs(a^i) = obs(a^j)$. As in the deterministic case, if $s \not\models pre(a)$ then $a(s)$ is undefined. 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 $a^i(s)$.

As we explained earlier, semantically, non-deterministic observations are not a limitation, and the approach we use to compile them away applies here as well. A simple scheme would be to add $\log(|\Omega|)$ new variables to the state, and use them to encode the observation as part of the state. The values of these variables can be non-deterministically determined, but the agent would deterministically observe their value. These new variables do not influence the transition to a new state. Of course, for specific domains, less generic methods could be more efficient.

Example 1. As a running example, consider the *Localize* problem (Fig. 1), where an agent in a grid must reach the top-right cell. In our examples, for ease of exposition, we will abuse notation and allow conditional effects of the form (c, e) where c is a formula and e is a conjunction of literals.

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 $\omega_{sense_d,GREEN} = \neg wall_d$, $d \in \{up, down, left, right\}$.

2.3. Online 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 the 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, the agent must determine whether the sequence is executable – technically, whether $\tau(b_I, \bar{a}, \bar{o})$ is non-empty – and whether the goal is satisfied following the sequence, i.e., whether $\tau(b_I, \bar{a}, \bar{o}) \models \varphi_G$.

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*). This immediately implies that $\tau(b_I, \bar{a}, \bar{o})$ is non-empty, and so the agent only needs to query, for a given precondition or goal formula ϕ whether $\tau(b_I, \bar{a}, \bar{o}) \models \phi$.

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.

Finally, an important advantage of online belief tracking, discussed in Section 6, is that it allows us to cheaply transform non-deterministic actions into deterministic actions. We need only add a number of new variables that is linear in the number of non-deterministic actions executed so far. Compiling non-deterministic actions in the offline case is problematic, as one new variable must be added for every potential action instance. This usually forces researchers to place some limitation, e.g., on the number of instances executed from each action, or requires that effects of different instances of the same action be correlated.

2.4. Relevance and width

Bonet and Geffner [6] introduce a measure of the complexity of belief tracking, called *problem width*. Essentially, it is the number of variables that must be maintained when answering a belief tracking query regarding the value of some state variable p.

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 $c_{a,l}$ of a conditional effect $(c_{a,l}, l)$ of some action a, and $l \in \{q, \neg q\}$. For an observation o, $p \in Ca(o)$ is similarly defined, where $c_{a,l}$ 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).* p is initially relevant to q if p and q appear in the same clause in φ_I.[1]

Definition 4 *(BG).* $o \in O$ is evidentially relevant to q if q is causally relevant to o.

Definition 5 *(BG).* p is relevant to q if p is causally, evidentially, or initially relevant to q, or if p is evidentially, causally or initially 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 affect our ability to make an observation that will revise our belief regarding whether p holds or not.

[1] BG's original definition does not consider the case that p and q appear in the same initial clause as they assume that all initial clauses are singletons.

Definition 6 *(BG).* The width of a variable p, $w(p)$ is the number variables relevant to p: $w(p) = |\{q|q \text{ is relevant to } p\}|$.

Definition 7 *(BG).* The width of a planning problem $\langle P, \mathcal{A}, \varphi_I, \varphi_G \rangle$ is $\max\{w(p)|p \in P$, and p appears in the goal or a precondition of some action $a\}$.

Computing the relevant variables is a simple low-order poly-time procedure.
The notion of causal width will later become helpful:

Definition 8 *(BG).* The causal width of a variable p, $w_c(p)$ is the number variables that are causally relevant to p: $w_c(p) = |\{q|q \text{ is causally relevant to } p\}|$.

Definition 9 *(BG).* The causal width of a planning problem $\langle P, \mathcal{A}, \varphi_I, \varphi_G \rangle$ is $\max\{w_c(p)|p \in P$ is an observed variable or p appears in the goal or a precondition of some action $a\}$.

Example 2. In the Localize problem, the *move-up* action has a conditional effect $(at_{3,2}, at_{3,3})$. Hence, $at_{3,2}$ is an immediate cause of $at_{3,3}$. The action has another conditional effect $(at_{3,1}, at_{3,2})$, making $at_{3,1}$ causally relevant to $at_{3,3}$.

The *checking* action contains a conditional effect $(at_{3,3}, wall_{up})$, making $at_{3,3}$ causally relevant to $wall_{up}$. As $\omega_{sense_{up},RED} = wall_{up}$, $wall_{up}$ is an immediate cause of the observation RED, and $at_{3,3}$ is causally relevant to RED. As such, RED is evidentially relevant to $at_{3,3}$.

Furthermore, as the observation $GREEN$ is also similarly relevant for $at_{3,2}$, and $at_{3,2}$ is relevant for $at_{3,3}$, $GREEN$ is also evidentially relevant for $at_{3,3}$.

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, costing linear time and space for each update. For the purpose of our analysis of belief tracking, like Bonet and Geffner, we ignore these variables.

3. Regression

Regression is the process of answering queries concerning the current state, by reasoning backwards through a sequence of actions and observations, combined with a description of the initial state. The basis of regression operations is the substitution of a variable by an expression that describes its new value. This was used in the assignment axioms of the Hoare calculus [14] and later by Dijkstra [10] for computing weakest preconditions.

In the area of planning, we build on two relevant lines of work on regression: Rintanen [22] and Shahaf and Amir [25] offer theoretically well-founded and practical tools for the use of regression in planning. Rintanen provides an efficient, theoretically sound definition for the regression formula in the context of state-space planning without sensing actions. Shahaf and Amir focus on filtering, which fits well with our online perspective, providing an efficient (polynomial) circuit representation for the regression formula. This is to be contrasted with the standard representation in which the formula's size can grow exponentially with the number of regression steps. However, it should be noted that the SAT-based encoding Ferraris and Giunchiglia [12] used by CFF Hoffmann and Brafman [15] is polynomial, too.

We now review Rintanen's 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 $\forall s \in S, s \models pre(a)$, denoted $S \models pre(a)$. 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, $\bar{a} = a_1, \ldots, a_n; \bar{o} = o_1, \ldots, o_n$ is *applicable* in a belief state b iff a_i is applicable in $\tau(b, a_1, \ldots, a_{i-1}, o_1, \ldots, o_{i-1})$, for every $i = 1, \ldots, n$.

3.2. Regression without observations

Let ϕ 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, and that $a(s)$ satisfies l iff either $s \models c_{a,l}$ or $s \models l \wedge \neg c_{a,\neg l}$. Hence, following Rintanen [22], we define the *regression* of ϕ with respect to a as:

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

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

The second equation may seem ambiguous, but as we assume that $c_{a,l} \rightarrow \neg c_{a,\neg l}$ in all reachable beliefs, then $\neg(c_{a,l} \vee (l \wedge \neg c_{a,\neg l})) \equiv c_{a,\neg l} \vee (\neg l \wedge \neg c_{a,l})$.

Clearly, all variables that appear in $rg_a(\phi)$ are relevant to some primitive proposition that appears in ϕ.

Example 3. In the Localize problem, let us regresses the formula $\phi = at_{3,2}$ through the *move-up* action. 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. *The regression of a compound formula can be decomposed:*

1. $(\phi_1 \wedge \phi_2)_{r(a)} \equiv (\phi_1)_{r(a)} \wedge (\phi_2)_{r(a)}$
2. $(\phi_1 \vee \phi_2)_{r(a)} \equiv (\phi_1)_{r(a)} \vee (\phi_2)_{r(a)}$
3. $(\neg\phi)_{r(a)} \equiv \neg(\phi_{r(a)})$

Proof. As $\phi_{r(a)}$ is a syntactic manipulation of the formula ϕ that is a point-wise replacement of each literal by a formula, the above is immediate, except for the case where ϕ is a literal in (3). For example:

$$(l_1 \wedge l_2)_{r(a)} = ((c_{a,l_1} \vee (l_1 \wedge \neg c_{a,\neg l_1})) \wedge (c_{a,l_2} \vee (l_2 \wedge \neg c_{a,\neg l_2}))) = (l_1)_{r(a)} \wedge (l_2)_{r(a)} \tag{3.3}$$

If $\phi = p$ for an atomic proposition, then: $(\neg p)_{r(a)} = c_{a,\neg p} \vee (\neg p \wedge \neg c_{a,p})$, by definition. $\neg(p_{r(a)}) = \neg(c_{a,p} \vee (p \wedge \neg c_{a,\neg p})) \equiv \neg c_{a,p} \wedge (\neg p \vee c_{a,\neg p}) \equiv (\neg c_{a,p} \wedge \neg p) \vee (\neg c_{a,p} \wedge c_{a,\neg p})$. However, $c_{a,p}$ and $c_{a,\neg p}$ are mutually exclusive. Hence, $\neg(p_{r(a)}) \equiv (\neg c_{a,p} \wedge \neg p) \vee c_{a,\neg p} = (\neg p)_{r(a)}$. For $\phi = \neg p$, the proof is identical. □

Theorem 1. *[22] Given a formula ϕ, a deterministic action a, and a state s, $s \models rg_a(\phi)$ iff a is applicable in s and $a(s) \models \phi$.*

Proof. By structural induction on the formula ϕ.

- Base case: let $\phi = l$ for some literal l, and $s \models rg_a(\phi)$. As $rg_a(\phi) = pre(a) \wedge \phi_{r(a)}$, then a is applicable in s. As ϕ is a literal, $\phi_{r(a)} = c_{a,l} \vee (l \wedge \neg c_{a,\neg l})$, thus either s satisfies $c_{a,l}$, the condition for l to hold following a, or $s \models l$ and $s \not\models c_{a,\neg l}$, the condition for $\neg l$. Thus, based on the semantics of action execution, $a(s) \models l$.
  For the other direction, suppose that a is applicable in s and $s \not\models rg_a(l)$. Thus:

  $$s \models \neg[c_{a,l} \vee (l \wedge \neg c_{a,\neg l})] \tag{3.4}$$

  $$s \models \neg c_{a,l} \wedge (\neg l \vee c_{a,\neg l}) \tag{3.5}$$

  $$s \models (\neg c_{a,l} \wedge \neg l) \vee (\neg c_{a,l} \wedge c_{a,\neg l}) \tag{3.6}$$

  In both cases, based on the semantics of action execution, $\neg l$ holds in $a(s)$, i.e., $a(s) \not\models l$, as required.
- Let $\phi = \phi_1 \wedge \phi_2$. From Lemma 1, $rg_a(\phi_1 \wedge \phi_2) \equiv rg_a(\phi_1) \wedge rg_a(\phi_2)$. Following the induction hypothesis, $a(s) \models \phi_1$ and $a(s) \models \phi_2$ and a is applicable in s. The other direction is similar. The same holds for $\phi = \phi_1 \vee \phi_2$.
- Let $\phi = \neg\psi$ where ψ is not a literal, and suppose that $s \models rg_a(\neg\psi)$. Thus, a is applicable in s. By Lemma 1.3, we know that $s \models \neg rg_a(\psi)$. By the induction hypothesis, and as a is applicable in s this implies that $a(s) \not\models \psi$, and thus $a(s) \models \neg\psi$ as $a(s)$ is a single state. For the other direction, suppose that a is applicable in s and $a(s) \models \neg\psi$. Thus, $a(s) \not\models \psi$. By the induction hypothesis, $s \not\models rg_a(\psi)$, i.e., $s \models \neg rg_a(\psi) = \neg pre(a) \vee \neg(\psi_{r(a)})$. As a is applicable in s, we must have that $s \models \neg(\psi_{r(a)})$. By Lemma 1.3, we have $s \models (\neg\psi)_{r(a)}$. Thus, $s \models rg_a(\neg\psi)$. □

3.3. Non-deterministic actions

For a non-deterministic action $a = \{a^1, \ldots, a^m\}$ we define [22]:

$$rg_a(\phi) = rg_{a^1}(\phi) \wedge \cdots \wedge rg_{a^m}(\phi) \tag{3.7}$$

Theorem 2. *[22] Let ϕ be a formula, let a be a non-deterministic action, and s a state. Then $s \models rg_a(\phi)$ iff a is applicable in s, and for every $s' \in a(s)$, $s' \models \phi$.*

Proof. We know that $s \models rg_{a^j}(\phi)$ iff a^j is applicable in s and $a^j(s) \models \phi$. Thus, $s \models rg_{a^1}(\phi)$ and $\cdots$ and $s \models rg_{a^m}(\phi)$ iff $a^1, \cdots, a^m$ are applicable in s and $a^1(s) \models \phi$ and $\cdots$ and $a^m(s) \models \phi$. Thus, $s \models rg_a(\phi)$ iff a is applicable in s and $a(s) \models \phi$. □

3.4. Regression with observations

We now extend regression to an action and an ensuing observation. Suppose we want to validate that ϕ 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 ϕ holds. That is:

$$rg_{a,o}(\phi) = rg_a(\omega_{a,o} \rightarrow \phi) \tag{3.8}$$

When $a = \{a^1, \ldots, a^k\}$ is non-deterministic, then:

$$rg_{a,o}(\phi) = \bigwedge_{1 \le i \le k} rg_{a^i,o}(\phi) = \bigwedge_{1 \le i \le k} rg_{a^i}(\omega_{a^i,o} \rightarrow \phi) \tag{3.9}$$

Theorem 3. *Given a formula ϕ, 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 $\tau(b, a, o) \models \phi$.*

For deterministic actions, we have:

Theorem 4. *Given a formula ϕ, 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 (1) a is applicable in s, and (2) $a(s) \models \omega_{a,o} \rightarrow \phi$; iff (1') a is applicable in s, and (2') $a(s) \models \omega_{a,o}$ implies $a(s) \models \phi$.[2] Using Theorem 2, we have (i) a is applicable in s and $a(s) \models \omega_{a,o}$ iff $s \models rg_a(\omega_{a,o})$, and (ii) a is applicable in s and $a(s) \models \phi$ iff $s \models rg_a(\phi)$. Combining (1') and (2') with (i) and (ii), we get: $s \models rg_{a,o}(\phi)$ iff $s \models rg_a(\omega_{a,o})$ implies $s \models rg_a(\phi)$, as required. □

Theorem 4 does not apply to a non-deterministic action a. The proof requires that if $s \models rg_{a,o}(\phi)$ then o will be observed following the execution of a in s. But for non-deterministic actions, it only holds that if o is observed then ϕ must be satisfied. While $rg_{a,o}(\phi)$ implies that $rg_a(\omega_{a,o}) \rightarrow rg_a(\phi)$, the other direction does not hold for non-deterministic actions. For example, suppose that a has two possible effects, p, and $\neg p$, which are observable. $reg_a(p)$ is *false* because there is no condition under which we are *guaranteed* to see p after a. Thus: $rg_{a,o}(\phi) \not\equiv rg_a(\omega_{a,o}) \rightarrow rg_a(\phi)$.

A different problem arises when regressing $\phi = false$ with an impossible observation, as in $rg_{a,o}(false)$, where o contradicts an effect of a. This would lead to counter-intuitive results. However, in the online setting which we focus on, we regress only with actual, and hence possible, observations, avoiding this problem.

Finally, regression has a number of useful properties:

Theorem 5. *For any two formulas ϕ_1 and ϕ_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, $\tau(\{s\}, a, o)$ contains a single state, and thus, $\tau(\{s\}, a, o) \models \phi_1 \vee \phi_2$ implies $\tau(\{s\}, a, o) \models \phi_1$ or $\tau(\{s\}, a, o) \models \phi_2$. □

[2] 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 $a(s) \models \phi \rightarrow \psi$.

3.5. Regression over a sequence

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

$$rg_{\bar{a}\cdot a,\bar{o}\cdot o}(\phi) = rg_{\bar{a},\bar{o}}(rg_{a,o}(\phi)); \quad rg_{\epsilon,\epsilon}(\phi) = \phi \tag{3.10}$$

where ϵ is the empty sequence.

Theorem 3 generalizes as follows:

Theorem 6. *Given a formula ϕ, 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. Efficient belief tracking using regression

Given Theorem 6, we can now use regression as a convenient method for online belief tracking, which is the main contribution of this paper.

Corollary 2. *For any propositional formula ϕ and any action–observation sequence $\bar{a}, \bar{o}$ that was applied in b_I, $\tau(b_I, \bar{a}, \bar{o}) \models \phi$ iff $b_I \models rg_{\bar{a}\bar{o}}(\phi)$.*

That is, following a valid sequence of actions and observations, querying the current belief state for a given formula ϕ is identical to regressing ϕ through the sequence of actions and observations, and checking whether the regressed formula is satisfied over the initial belief. Thus, there is no need to forward update the belief state – one could just maintain the sequence of already executed actions and observations, and use regression to answer any needed query.

4.1. Ignoring preconditions

Recall that in online belief tracking there are two types of queries – checking whether an action can be applied, i.e., checking whether the action precondition holds, and checking whether the goal holds in the current belief state.

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:

$$rg^*_a(\phi) = \phi_{r(a)} \tag{4.1}$$

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.

$$rg^*_\epsilon(\phi) = \phi; \quad rg^*_{a\cdot\bar{a}}(\phi) = rg^*_a(rg^*_{\bar{a}}(\phi)) \tag{4.3}$$

Theorem 7. *For any propositional formula ϕ and action–observation sequence $\bar{a}, \bar{o}$ that is applicable in b_I, we have that $b_I \models rg_{\bar{a}\bar{o}}(\phi)$ iff $b_I \models rg^*_{\bar{a}\bar{o}}(\phi)$.*

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 ϕ be a propositional formula 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: $b \models rg_{\bar{a},\bar{o}}(\phi)$ iff $b \models rg^*_{\bar{a},\bar{o}}(\phi)$. Thus, $b_I \models rg_{a\bar{a},o\bar{o}}(\phi)$ iff (by definition of rg on sequences) $b_I \models rg_{a,o}(rg_{\bar{a},\bar{o}}(\phi))$ iff (by the observation above) $b_I \models rg^*_{a,o}(rg_{\bar{a},\bar{o}}(\phi))$ iff (by the induction hypothesis) $b_I \models rg^*_{a,o}(rg^*_{\bar{a},\bar{o}}(\phi))$. In the latter case we show that regression of equivalent formulas is equivalent (Theorem 5). □

Example 4. We now illustrate how an agent that has soundly executed two *move-up* actions in our localize 3×3 example, cannot guarantee that the goal $at_{3,3}$ holds in all possible states. We regress $l = \neg at_{3,3}$ 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 *move-up* 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, i.e., $\phi_I = (\text{one-of } at_{1,1} ... at_{3,3})$, 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.

4.2. Ignoring irrelevant observations

We have already seen that, without observations, the regression formula contains only variables relevant to the regressed formula. However, the regression of observations that were received throughout the execution of an action sequence may involve variables that appear in $\omega_{a,o}$, yet are not relevant for any primitive proposition in ϕ. Recall that, earlier, we defined observation o resulting from action a as being *relevant* to p if $\omega_{a,o}$ contains an atomic proposition relevant to p.

It is not surprising that when verifying the validity of ϕ, we can ignore observations that are irrelevant to l. To model the removal of irrelevant observations, we define, for every action a, a new artificial action $a_{no\text{-}obs}$, identical to a except that $obs(a) = (true, no\text{-}obs)$. Recall that *no-obs* is an empty observation, such that $\omega_{a,no\text{-}obs} = true$, and thus provides no information. Now, to model the removal of an irrelevant observation of an action a that appears in the sequence executed, we replace a with $a_{no\text{-}obs}$.

Lemma 2. *Let $\bar{a}, \bar{o}$ be a sequence of actions and observations made, and let $\bar{a}', \bar{o}'$ be a corresponding sequence where every action a associated with an observation o that is irrelevant to all primitive propositions in ϕ is replaced by $a_{no\text{-}obs}$. Then, ϕ holds following $\bar{a}, \bar{o}$ iff $b_I \models rg^*_{\bar{a}'\bar{o}'}(\phi)$.*

Proof. Let us start with the case of a single deterministic action a and observation o, such that o is irrelevant to all propositions in ϕ. From our previous results we have that l holds following a, o iff $b_I \models rg^*_{a,o}(\phi)$ iff $b_I \models rg^*_a(\omega_{a,o} \rightarrow \phi)$. We have that ϕ holds following $a_{no\text{-}obs}$ iff $b_I \models (true \rightarrow rg^*_{a_{no\text{-}obs},no\text{-}obs}(\phi))$ iff $b_I \models (true \rightarrow rg^*_a(\phi))$, i.e., iff $b_I \models rg^*_a(\phi)$ (because, naturally, the initial state formula is satisfiable). Thus, we need to show that $b_I \models rg^*_a(\omega_{a,o} \rightarrow \phi)$ iff $b_I \models rg^*_a(\phi)$ when o is irrelevant to all propositions in ϕ.

By definition, $rg^*_{a,o}(\phi) \equiv \{\omega_{a,o} \rightarrow \phi\}_{r(a)} \equiv \{\omega_{a,o}\}_{r(a)} \rightarrow \{\phi\}_{r(a)}$ because the $_{r(a)}$ operation manipulates each literal independently and a is deterministic. Moreover, as noted earlier, by definition of relevance, $\{\phi'\}_{r(a)}$ contains only literals relevant to propositions in ϕ'.

Thus, we need to show that: $b_I \models \{\omega_{a,o}\}_{r(a)} \rightarrow \{\phi\}_{r(a)}$ iff $b_I \models \{\phi\}_{r(a)}$, i.e., that $\varphi_I \wedge \{\omega_{a,o}\}_{r(a)} \models \{\phi\}_{r(a)}$ iff $\varphi_I \models \{\phi\}_{r(a)}$. φ_I can be partitioned into clauses that contain literals relevant to some proposition in ϕ *only*, $\varphi_{I,rel(\phi)}$, and clauses that contain no literal relevant to ϕ, $\varphi_{I,irrel(\phi)}$. This follows from the definition of relevance, where if l_1 is relevant to l, and l_2 appears with l_1 in the same clause of φ_I, then l_2 is relevant to l. Thus, we have to show that: $\varphi_{I,irrel(\phi)} \wedge \varphi_{I,rel(\phi)} \wedge \{\omega_{a,o}\}_{r(a)} \models \{\phi\}_{r(a)}$ iff $\varphi_{I,irrel(\phi)} \wedge \varphi_{I,rel(\phi)} \models \{\phi\}_{r(a)}$. However, from logic we know that if ϕ, ψ are two formulas over disjoint sets of atomic propositions, and φ is a formula that contains only atomic propositions that appear in ϕ then $\phi \wedge \psi \models \varphi$ iff $\phi \models \varphi$. Thus, $\varphi_I \wedge \{\omega_{a,o}\}_{r(a)} \models \{\phi\}_{r(a)}$ iff $\varphi_{I,irrel(\phi)} \wedge \varphi_{I,rel(\phi)} \wedge \{\omega_{a,o}\}_{r(a)} \models \{\phi\}_{r(a)}$ iff $\varphi_{I,irrel(\phi)} \wedge \varphi_{I,rel(\phi)} \models \{\phi\}_{r(a)}$ iff $\varphi_I \models \{\phi\}_{r(a)}$, as required.

Next, assume that $a = \{a^1, \ldots, a^m\}$ is non-deterministic. Recall that the observations associated with the component a^i's are identical, by definition. Using the same reasoning as above, we have that for every $1 \leq i \leq m$: $b_I \models rg^*_{a_i}(\omega_{a_i,o} \rightarrow \phi)$ iff $b_I \models rg^*_{a_i}(\phi)$. That is: for every $1 \leq i \leq m$: $b_I \models rg^*_{a_i,o}(\phi)$ iff $b_I \models rg^*_{a_i}(\phi)$, which implies $b_I \models rg^*_{a,o}(\phi)$ iff $b_I \models rg^*_a(\phi)$, as required.

The general case follows by induction, as $rg^*_{\bar{a}\cdot a,\bar{o}\cdot o}(\phi) = rg^*_{\bar{a},\bar{o}}(rg^*_{a,o}(\phi))$. By the induction hypothesis, $rg^*_{\bar{a},\bar{o}}(rg^*_{a,o}(\phi)) = rg^*_{\bar{a}',\bar{o}'}(rg^*_{a,o}(\phi))$. If o is relevant to some proposition in ϕ, then this implies $rg^*_{\bar{a}\cdot a,\bar{o}\cdot o}(\phi) = rg^*_{\bar{a}'\cdot a,\bar{o}'\cdot o}(\phi)$ as required, since $\bar{o}' \cdot o$ is the correct sequences of observations in this case. If o is irrelevant to all propositions in ϕ, then we know from above that $rg^*_{a,o}(\phi) = rg^*_{a_{no\text{-}obs},no\text{-}obs}(\phi)$ and this implies $rg^*_{\bar{a}\cdot a,\bar{o}\cdot o}(\phi) = rg^*_{\bar{a}'\cdot a_{no\text{-}obs},\bar{o}'\cdot no\text{-}obs}(\phi)$, as required. □

Thus, regression with observations, as long as irrelevant observations are ignored, also involves only variables relevant to some proposition that appears in the regressed formula.

5. Complexity analysis

Our analysis will focus on the complexity per step under the assumption that the action sequence is of polynomial length. If the action sequence is exponential then both methods must perform exponentially many updates. However, in the worst case, the regression method will suffer even more, because while FBT pays a fixed price per step, the regression method's cost increases (polynomially) with the length of the action sequence.

As we shall see, regression allows us to use a more refined, context sensitive width notion, that is never larger than BG's notion of width. We start by introducing a natural extension of BG's width notion to the online setting, i.e., in the context of an action–observation sequence. Later, we farther refine this notion using additional properties of regression.

Definition 10. $w_{\bar{a},\bar{o}}(p)$, the *contextual 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 $\bar{o}$.

Theorem 8. *For any literal l, the time complexity of computing $rg^*_{\bar{a}',\bar{o}'}(\phi)$ is $O(2^{cw}|\bar{a}|)$, and the space complexity is $O(2^{cw})$, where $w = w_{\bar{a}',\bar{o}'}(l)$ is the contextual width of l in the context of the sequence $\bar{a}', \bar{o}'$, ignoring observations irrelevant to l.*

Proof. For every literal l in ϕ, we have seen that $rg^*_{\bar{a}',\bar{o}'}(l)$ contains only atomic propositions relevant to l with respect to the planning problem, restricted to the actions and observations in $\bar{a}'$ and $\bar{o}'$, where observations irrelevant to l are excluded. We denote the number of such relevant propositions by w. Below we heavily rely on restricting our attention to only w primitive propositions, and consequently to a number of possible models that is exponential in w, rather than in the total number of primitive propositions. This means that, while conversion from DNF to CNF and vice versa can cause an exponential increase in formula size, as long as we are careful to remove repeated/subsumed terms, this is still bounded by $\exp(w)$.

The size of the regression formula is initially 1, corresponding to the literal l. In each regression step, we replace a literal by a formula of the form $c_{a,l} \vee (l \wedge \neg c_{a,\neg l})$. We convert the latter into CNF, which results (following the above) in a formula with size that is at most exponential in w. It is easy to maintain the regression formula in CNF – as we replace a literal in a CNF formula by another CNF formula, we just have to distribute the disjuncts in the respective clause. This is polynomial in the size of the formula and the CNF representation of the inserted formula.

However, as the above transformation is applied $|\bar{a}|$ times during the regression process, the formula size can become exponential in $|\bar{a}|$, the length of the action sequence, rather than in w. However, we can maintain a formula size exponential in w by simplifying the formula following each step, removing subsumed and repeated clauses, at a cost that is at most exponential in w for each step. Specifically, the number of non-subsumed disjuncts is at most 2^{2w} and we can remove each subsumed disjunct in time proportional to the number of clauses squared, which is thus bounded by 2^{4w}. This gives us the desirable space complexity. For the time complexity, as we perform this simplification at every step, we have to multiply the simplification cost 2^{4w} by the sequence length $|\bar{a}|$. □

Corollary 3. *For any formula ϕ, the time complexity of computing $rg^*_{\bar{a}',\bar{o}'}(\phi)$ is $O(2^{cw}|\bar{a}|\,|\phi|)$, and the space complexity is $O(2^{cw}|\phi|)$, where $w = \max_{l\in\phi} w_{\bar{a}',\bar{o}'}(l)$ and $w_{\bar{a}',\bar{o}'}(l)$ is as above.*

Proof. This follows immediately from Theorem 8 and the decomposability of the regression formula as shown in Lemma 1. We need to replace every literal in ϕ by its regression formula. □

Presentations support rapid, poly-time validity checks. Examples of such forms are formulas in prime-implicate form [9] as well as conjunctions of literals. Representing the initial belief using such a form allows us to bound the complexity of the satisfiability query that follows the regression.

Theorem 9. *If the form of φ_I allows for polynomial clause validity checks then the complexity of determining whether l is valid following $\bar{a}, \bar{o}$ is polynomial in $|\bar{a}|$ and at most exponential in $w_{\bar{a}',\bar{o}'}(l)$.*

Proof. Based on Lemma 2, determining whether a literal l holds following $\bar{a}, \bar{o}$, can be determined by checking whether $\varphi_I \models rg^*_{\bar{a}',\bar{o}'}(l)$. As we have seen, the time required to compute $rg^*_{\bar{a}',\bar{o}'}(l)$, as well as its size, are polynomial in $|\bar{a}|$ and exponential in $w = w_{\bar{a}',\bar{o}'}(l)$. Furthermore, if the latter formula is in CNF then it has at most a number of clauses exponential in w. Given our assumption on the form of φ_I, the result follows. □

Online belief tracking replaces the offline query: "is $\bar{a}, \bar{o}$ executable in b_I?" with a sequence of queries for the precondition of each action in the sequence separately. Thus, to determine executability, one needs query only regarding the preconditions of the last action in the sequence. It might well be that some relevant variables for the precondition of an action do not appear in the specific actions that were executed, and the regression would not involve these variables, reducing the practical width for the specific query.

5.1. A tighter definition of contextual width

The contextual width $w_{\bar{a},\bar{o}}(p)$, as defined so far, is based on BG's notion of width, restricted to a subset of actions and observations. A closer look, though, reveals that it can be much smaller.[3]

At the extreme, imagine that the last observation in $\bar{o}$ reveals the value of p. In that case, the value of earlier actions and observations is not needed for determining whether p or $\neg p$ holds. More generally, some observations can provide information that makes earlier actions irrelevant to the current regressed query. To better define this, let us consider a more refined definition of context:

Definition 11. The a, o context of p, denoted $ctx_{a,o}(p)$, consists of all atomic propositions q that are relevant to p given the singleton action set $\{a\}$ and the singleton observation set $\{o\}$ (set)minus the set of propositions $\{r\}$ such that $\omega_{a,o}$ implies r or $\neg r$.

The $\bar{a} \cdot a, \bar{o} \cdot o$ context of p, denoted $ctx_{\bar{a}\cdot a,\bar{o}\cdot o}(p)$, is defined recursively as $\bigcup_{r \in ctx_{a,o}(p)} ctx_{\bar{a},\bar{o}}(r)$. We define $w^*_{\bar{a},\bar{o}}(p) = |ctx_{\bar{a},\bar{o}}(p)|$.

The following is straightforward from the definition of ctx and $ctx_{\bar{a},\bar{o}}$.

Lemma 3. *For any sequence $\bar{a}, \bar{o}$ and an atomic proposition p, we have that $ctx_{\bar{a},\bar{o}}(p) \subseteq ctx(p)$. Therefore, $w^*_{\bar{a},\bar{o}}(p) \leq w_{\bar{a},\bar{o}}(p) \leq w(p)$.*

Corollary 4. *We can replace $w_{\bar{a},\bar{o}}$ in Theorem 8 by $w^*_{\bar{a},\bar{o}}$.*

Proof. Using the recursive definition of regression over sequences, and focusing on relevant observations only, $b_I \models rg^*_{\bar{a}\cdot a,\bar{o}\cdot o}(l)$ iff $b_I \models rg^*_{\bar{a},\bar{o}}(rg^*_{a,o}(l))$ iff $b_I \models rg^*_{\bar{a},\bar{o}}(\omega_{a,o} \rightarrow rg^*_a(l))$. $rg^*_a(l)$ contains only variables relevant to l with respect to action a. If r is such a variable and $\omega_{a,o}$ implies r we can simplify $rg^*_a(l)$ by replacing r with *true*. If $\omega_{a,o}$ implies $\neg r$, we replace it by *false*. In both cases, we maintain logical equivalence. Thus, r will not be carried backwards through the regression at this step. Of course, r could reoccur later in the regression process at some other step, but that will be accounted for in the definition of that particular step. Consequently, the set of variables that will appear in $rg^*_{\bar{a}\cdot a,\bar{o}\cdot o}(l)$ will be contained in $ctx_{\bar{a}\cdot a,\bar{o}\cdot o}(l)$. □

The above result can give us some insight into the potential advantage of caching. In planning, one must perform multiple regression queries at each step. One for every precondition of a potentially applicable action, and for every sub-goal. If following the regression query the value of some variable at time step t becomes known, we can cache this information. In the worst case, the amount of information cached is linear in the length of the sequence. These cached values can be used to simplify the regression formula much like observations are used in the above analysis. Of course, in the worst-case, the context of a propositional variable contains no variable that was cached, hence the worst-case complexity remains unchanged. In practice, as our experiments demonstrate, caching is quite valuable.

5.2. Regression vs. factored belief tracking

Bonet and Geffner show that the complexity of the FBT method is exponential in the width of the problem:

Theorem 10 *(BG). If φ_I is a conjunction of literals than 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 planning problem.*

Compare this result to Theorem 9, and note that under the assumption that φ_I is a conjunction of literals, clausal entailment is polynomial. Furthermore, note that with such an initial state, the initial formula plays no role in the definition of relevance, and so our definition becomes identical to that of BG. Thus, if the action sequence is of polynomial size, then the complexity of both methods is exponential in the width of the problem, except that in the case of regression, it is the contextual width, which is always bounded above by the width, and can often be smaller. Thus, in terms of the bounds, regression is superior, as long as the action sequence is not too long.

The original definition of width in BG's work is with respect to multi-valued variables, whereas we focus on the Boolean case. Thus, one might ask what happens if a Boolean domain is compiled into a multi-valued one and FBT is applied, or if a multi-valued domain is compiled into a Boolean one, and regression is applied, as width is not necessarily invariant under such compilations. A multi-valued domain may have smaller width than its corresponding Boolean domain, and even smaller than the contextual width of the Boolean domain for various action–observation sequence.

As far as the complexity analysis is concerned, compiling a domain from multi-valued variables to Boolean variables increases width because, to the best of our knowledge, existing methods cannot exploit the additional constraints imposed

[3] We thank the reviewers for pointing this out.

on Boolean variables that correspond to multi-valued domains. For example, if *loc* is a multi-valued variable denoting the location of some object, then as we increase its size linearly, its width remains the same, while the number of possible belief states (which FBT tracks) increases polynomially. In its natural Boolean representation, where new propositions denote that the object is in a particular location, the number of variables increases linearly, and so does the width. And while the set of legal truth assignments increases only polynomially, we are not aware of methods for enhancing regression with invariants that will ensure that the size of formula does not grow exponentially in w, and perhaps more importantly, that the validity check is guaranteed to exploit this information.[4] This is an interesting open issue.

In the experimental evaluation, we compare regression on Boolean domains to FBT on multi-valued domains. Despite the fact that the latter may have smaller width, regression appears to perform very well.

6. Compiling away non-deterministic actions

We have previously explained how to handle regression with non-deterministic actions. In this section we suggest an alternative approach to handling non-deterministic actions by compiling them into deterministic actions. In theory, this compilation can have a high cost. In practice, it allows us to decouple observations from the conditions we are checking, and to simplify future regressions.

The reason to compile away non-deterministic actions is that deterministic actions have two important properties:

1. $rg_{a,o}(\phi_1 \vee \phi_2) \equiv rg_{a,o}(\phi_1) \vee rg_{a,o}(\phi_2)$
2. $s \models rg_{a,o}(\phi)$ iff $s \models rg_a(\omega_{a,o}) \Rightarrow s \models rg_a(\phi)$.

The first property, together with the factoring of regression over conjunctions, implies that we can factor the regression formula and regress it in parts, reusing similar parts. This is the essential idea behind the circuit construction of Shahaf and Amir [25] which we discuss later.

The second property has important practical implications. It implies that following an observation o via action a, we can conjoin the regression of $\omega_{a,o}$ to the initial state independently of the condition ϕ we are checking. Thus, once the regression of $\omega_{a,o}$ is conjoined with the initial state, we can simply ignore this observation in future regressions. In practice, the regression of observation conditions often simplifies the initial state considerably, making future validity checks simpler. In theory, however, it can complicate the initial state.

To replace non-deterministic actions with deterministic ones we use a classical compilation scheme that replaces the uncertainty on the effect of the actions with uncertainty on the initial state. For simplicity, let us assume that a non-deterministic action a has only two possible options: $a = \{a^1, a^2\}$. We add a new atomic proposition, p_a, whose value is unknown initially. We now change a into a deterministic action a^{det} with two conditional effects: if p_a then a^1 and if $\neg p_a$ then a^2.

Formally, define $c_{a,l}$ (the condition under which l becomes true following a) to be

$$c_{a,l} = (p_a \wedge c_{a^1,l}) \vee (\neg p_a \wedge c_{a^2,l}) \tag{6.1}$$

Example 5. In our *localize* example, we add a *drift* action, moving the agent non-deterministically vertically to one of the neighboring cells. Formally, $drift = \{drift^{up}, drift^{down}\}$, where $drift^{up}$ contains a condition $(at_{3,2}, at_{3,3} \wedge \neg at3, 2)$ and $drift^{down}$ contains a condition $(at_{3,2}, at_{3,1} \wedge \neg at3, 2)$. We add a new atomic proposition p_{drift_v}, and the new action $drift_v^{det}$ will contain two conditional effects $(at_{3,2} \wedge p_{drift_v}, at_{3,3} \wedge \neg at3, 2)$ and $(at_{3,2} \wedge \neg p_{drift_v}, at_{3,1} \wedge \neg at3, 2)$.

This compilation must be carried out for every action *instance*. That is, if a appears three times in the sequence at times t_1, t_2, t_3, then we need three different variables $p_{a_{t_1}}, p_{a_{t_2}}, p_{a_{t_3}}$, to allow for the outcome at t_1 to differ from the outcome at t_2, etc. For offline planning, this compilation requires a number of additional variables that is exponential in the problem horizon. In online planning and belief tracking, however, we can add these variables lazily, only when a non-deterministic action is chosen for execution. Thus, the number of atomic propositions that we need to add equals the number of non-deterministic actions performed so far, and is therefore at most linear in the sequence length.

Once we compile non-deterministic actions into deterministic ones, we can perform regression over the sequence of actions as before. The complexity analysis remains as above, except that $w_{\bar{a},\bar{o}}(p)$, the width of an atomic proposition p can become larger by $|\bar{a}|$ because each non-deterministic action may introduce a new variable when compiled, which may be relevant for the regressed literal.

[4] Certainly, invariants could be used to simplify the formula, and one can envision a representation that exploits the multi-valued nature of variables (e.g., associating them with sets of possible values). However, we are not aware of methods for satisfiability and validity checking that provide the required complexity guarantees.

6.1. Regressing observations independently

We have seen above that $rg^*_{a,o} = rg^*_a(\omega_{a,o} \rightarrow \phi)$, and discussed how irrelevant observations to the currently regressed literal l can be ignored during the regression. We now suggest a different method for handling observations, and analyze its theoretical and practical implications. Specifically, we suggest to regress observations as they are received, conjoin the regressed formula to the initial belief state, and ignore the observation during later regressions.

That is, following each observation o, we regress it to the initial state, obtaining $\varphi_o = rg^*_{\bar{a},\bar{o}}(\omega_{a,o})$. Now, we conjoin φ_o with φ_I obtaining an updated initial state formula. To verify whether l is valid following $\bar{a}, \bar{o} = a_1, \ldots, a_k; o_1 \ldots o_k$, we check whether $\varphi_I \wedge rg^*_{a_1}(\omega_{a_1,o_1}) \wedge \cdots \wedge rg^*_{a_1,\ldots,a_k}(\omega_{a_k,o_k}) \models rg^*_{a_1,\ldots,a_k}(l)$.

This method has several advantages:

1. Observations are regressed only once, and the cost of their regression is amortized over all future regressions.
2. The regression formula is simpler and smaller, as it no longer refers to the observation conditions ($\omega_{a,o}$). This means that variables that will appear in it are only those that are causally relevant to the regressed literal, which is of course a subset of the set of relevant variables. The complexity is no longer exponential in the width, but rather, in the *causal width*, which can be much smaller (and even smaller given the context-based definition of width). We use $w^c_{\bar{a},\bar{o}}(l)$ to denote the *contextual causal width*.
3. As the regression of the formula $\omega_{a,o} \rightarrow \phi$ makes the decomposition of the regression operator more complex, regression over deterministic actions avoiding the observations decomposes conveniently.
4. In many cases, $\varphi_I \wedge rg^*_{\bar{a}}(\omega_{a,o})$ creates a formula that is simpler than the original φ_I, reducing the cost of future entailment queries.

Example 6. In the localize domain, 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 (one-of $at_{1,1} \ldots at_{3,3}$) with this regressed formula, limiting the set of possible initial state only to (one-of $at_{1,1}$ $at_{1,2}$ $at_{3,1}$ $at_{3,2}$).

The major theoretical disadvantage of this method is that the initial state formula must be updated following every observation with the regressed observation condition. For example, formulas in PI form, which are convenient for SAT queries, do not support simple conjunction. Thus, if we seek to maintain the PI form of the initial state, the price is worst-case exponential in the entire set of variables. If we use a CNF representation, however, the conjunction is worst-case exponential in the number of variables which appear in the regressed formula, which is no greater than the width of the problem, but with CNF, entailment can be exponential.

One may worry that, as we add new clauses or change the initial state, non-relevant atomic propositions will become relevant to each other, as l is relevant to p if it appears in an initial clause with another literal relevant to p. However, the new formulas conjoined with the φ_I have the form $rg^*_{a_1,\ldots,a_k}(\omega_{a_k,o_k})$, i.e., they are obtained via regression over actions only. Consequently, by definition of regression, they refer only to atomic propositions relevant to $\omega_{a,o}$. Thus, the set of atomic propositions relevant to an atomic proposition p remains the same as in the original problem, with the exception of the new determinization propositions, p_a.

The following is a simple corollary of Theorem 8:

Theorem 11. *The time and space complexity of determining whether some literal l is valid following $\bar{a}, \bar{o}$ is $O(2^{cw} \cdot |\bar{a}| \cdot poly(|b_I^{\bar{a},\bar{o}}|))$, where $w = w^c_{\bar{a}'}$ is the contextual causal width of l with respect to the set of actions in $\bar{a}'$, the determinized sequence of actions executed, and $|b_I^{\bar{a},\bar{o}}|$ is the size of the initial state formula conjoined with regressed, past observations, and converted into PI form or some other form supporting polynomial time validity checks.*

Proof. This follows immediately from Theorem 8 after we notice that the regressed formula will only contain causally relevant variables that appear in the determinized sequence $\bar{a}'$, as we no longer need to regress through o. □

As noted, the size of the initial state formula, modified by conjunction with a regressed observation o can increase. In principle, if the initial state formula is maintained in a convenient form that supports polynomial-time conjunction, an equivalent formula whose size is polynomial in $|b_I|$ and $2^{w^c_{\bar{a}',\bar{o}}}$ exists. This is true for a number of forms (see Darwiche and Marquis [9]). In particular, for OBDD$_<$, both conjunction and validity checks are polytime. However, even in this case, unless a stronger property of unbounded conjunction holds, we cannot prove that an unbounded sequence of updates to the initial state formula yields a poly-sized formula. Thus, in principle, the size can become exponential in the length of $|\bar{o}|$.

In practice, however, regressed observations often simplify the initial state rather than complicate it, as they often establish the value of some proposition at the initial state. For example, in the localize domain, observing that there is no wall above the agent, we know that any location where there is a wall above the agent is impossible. Thus, the set of currently possible locations is reduced following every observation.

6.2. Representing beliefs using circuits

Shahaf and Amir [25] suggest logical circuits as a convenient form for representing beliefs. A logical circuit is a directed acyclic graph, where leaves represent variables, and inner nodes are tagged with a logical operation, such as $\wedge, \vee, \neg$. An advantage of circuits, aside from their compact form, is that specialized SAT procedures that exploit the circuit structure can lead to efficient computation of SAT and UNSAT queries. Shahaf and Amir show how a poly-sized circuit can be constructed to represent the belief given a sequence of deterministic actions. They, as well as others [22,25], were pessimistic about extending logical circuits to non-deterministic actions, while preserving a polynomial size.

Surprisingly, our compilation scheme from non-deterministic actions into deterministic actions allows us to prove the existence of a polynomial size circuit representation of the regression formula. Using this compilation, it is easy to see that the circuit construction works for non-deterministic actions too, as long as we pay the price of determinization.

Theorem 12. *Let l be a literal and $\bar{a}, \bar{o}$ an action–observation sequence. Then, a circuit C can be constructed in time and space $O(poly(|\varphi_l| + |\bar{a}| \cdot ActDesc))$, where* ActDesc *is the maximal description size of an action, such that l holds following $\bar{a}, \bar{o}$ iff C is unsatisfiable.*

Proof. This result is an immediate consequence of Theorem 5.5 of Shahaf and Amir [25] combined with the idea of action determinization that we described above. Their result holds for deterministic actions, and the determinization step can only increase the description of the action polynomially. □

We also provide a direct, recursive construction, which given the machinery developed in this paper may be more appropriate in this context than Shahaf and Amir's description. The basic intuition should be clear: regression over deterministic action sequences decomposes over disjunctions and conjunctions, so we can reuse sub-parts of the formula.

Theorem 13. *Let l be a literal and $\bar{a}, \bar{o}$ an action–observation sequence and assume that the conditions $c_{a,l}, \omega_{a,l}$ have size polynomial in the number of variables appearing in them. Then, $rg^*_{\bar{a},\bar{o}}(l)$ can be represented by a circuit with size $O(|\bar{a}| \cdot poly(w_{\bar{a},\bar{o}}))$.*

Proof. We shall assume that for all a, l, $\omega_{a,l}$ and $c_{a,l}$ are in negation normal form. Converting a formula to negation normal form doubles it size, at most.

We begin by describing the circuit construction, then we analyze the circuit size. The proof is by induction on $|\bar{a}|$ and it yields a simple recursive construction. For the base case $\bar{a} = \epsilon$, and the circuit contains l only.

For the inductive step: we explain how we take circuits constructed for the set of literals relevant to l and the sequence $\bar{a}, \bar{o}$ and build a circuit for l given $\bar{a}a, \bar{o}o$.

First, if a is a non-deterministic action, we transform it into a deterministic action. That will only affect the size of the various $c_{a,l}$ polynomially. Note, also, that the variable added does not impact any part of the circuit constructed earlier.

Now, given a constructed circuit for $\bar{a}, \bar{o}$ we consider $rg^*_{\bar{a}\cdot a,\bar{o}\cdot o}(l)$. By definition, $rg^*_{\bar{a}\cdot a,\bar{o}\cdot o}(l) = rg^*_{\bar{a},\bar{o}}(rg^*_{a,o}(l)) \equiv rg^*_{\bar{a},\bar{o}}(rg^*_a(\omega_{a,o} \to l))$. By our inductive assumption, we know that for every literal l, we can construct a circuit $rg^*_{\bar{a},\bar{o}}(l)$. By definition, $rg^*_a(\omega_{a,o} \to l) = (\omega_{a,o} \to l)_{r(a)}$. Recall that (Equation (3.2))

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

Thus, $rg^*_{\bar{a}\cdot a,\bar{o}\cdot o}(l) = rg^*_{\bar{a},\bar{o}}((\omega_{a,o} \to l)_{r(a)})$. Next, we convert $(\omega_{a,o} \to l)_{r(a)}$ to negation normal form. Denote the resulting formula by ϕ. Now, we use Theorem 5, which tells us that ϕ can be decomposed to some Boolean formula with conjunctions and disjunctions over basic elements of the form $rg^*_{\bar{a},\bar{o}}(l)$. Since we have a circuit for each of these elements, we now only need to combine the existing elements with some *and* and *or* gates to get the desired circuit.

Next, we analyze the circuit size. First, note that given the definition of relevance, if l' is relevant to l given $\bar{a} \cdot a, \bar{o} \cdot o$, and l'' is relevant to l' given $\bar{a} \cdot a, \bar{o} \cdot o$, then l'' is relevant to l given $\bar{a} \cdot a, \bar{o} \cdot o$. And furthermore, if l' is relevant to l given some prefix of $\bar{a} \cdot a, \bar{o} \cdot o$, then it is also relevant given $\bar{a} \cdot a, \bar{o} \cdot o$. Now, to build the circuit for $rg^*_{\bar{a}\cdot a,\bar{o}\cdot o}(l)$ we maintain circuits for all the literals relevant to l given $\bar{a} \cdot a, \bar{o} \cdot o$. Their number if $w = o(w_{\bar{a}a,\bar{o}o}(l))$. As described above, moving from a set of circuits for any such l', given a prefix of length m of the sequence of actions and observations, to one representing the prefix of length $m+1$, can be done by adding a single circuit of size k. This circuit corresponds to $reg^*_{a,o}(l)$, for the most recent action observation pair, a, o. We do this for all w literals, and repeat it a number of times that equals to the length of the sequence. Thus, the overall size is polynomial in $w \cdot k \cdot |\bar{a}|$.

It remains to show that k is $O(poly(w))$. But k is the size of a circuit that represents $(\omega_{a,o} \to l')_{r(a)}$. (It is the circuit whose inputs are the various $rg^*_{\bar{a},\bar{o}}(l')$ which is constructed in the last step.) The latter, by assumption, is $O(poly(w))$, and its conversion to negation normal form maintains this property. □

Algorithm 1: Revising the initial belief and cached known facts.

```
1  ReviseInitialBelief(φ, ā_t, F_t)
       Input: φ: an observed formula,
              ā_t: an action sequence of length t, F_t: known facts at t
2      φ_t ← φ
3      for i = t down to 1 do
4          if IsTrue(φ, F_t) then
5              return
6          forall unit clause c ∈ φ add c to F_t
7          φ_{t−1} ← rg*_{a_t}(φ_t)
8      Convert φ_0 into CNF form
9      b_I ← b_I ∧ φ_0
10     K_0 ← all unit clauses in b_I
11     for i = 1 to t do
12         Apply a_i over K_{t−1}
13         Add effect(a) to F_t
```

7. Empirical evaluation

We start by discussing a few optimizations used in our code, and then proceed to empirically evaluate the regression method and compare it to BG's beam-tracking method.

7.1. Practical optimizations

The run-time of regression can be improved considerably by using a number of optimizations. Two of these methods were briefly discussed earlier: caching and initial-state simplification. 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. We maintain at each time step t the set of facts F_t that are known at time t. Each successful precondition regression query may add new facts to F_t.

Second, we utilize observations to constrain and simplify the initial state as explained earlier. Often the regressed formula φ is a unit literal, e.g., if o is an observation of a static fact. We then use unit propagation to discover additional facts about the set of possible initial states. The learned facts are then propagated forward through the executed action–observation sequence to update the F_t cache (Algorithm 1).

Sometimes, however, φ 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 the MiniSAT SAT solver [11]. Theoretically, these queries can take exponential time to answer, but in practice they are very fast.

Instead of checking whether $b_I \models rg^*_{\bar{a}}(l)$, for a precondition or goal literal l, we regress $\neg l$, obtaining $rg^*_{\bar{a}}(\neg l)$. Then we run a SAT query for $b_I \wedge rg^*_{\bar{a}}(\neg l)$. If there is no solution to this SAT query, then l must hold following $\bar{a}$ (Algorithm 2).

We also use some additional caching mechanisms; as the actual PDDL problem specification allows a literal l to appear in the effect of many conditional effects of an action a, we maintain a mapping from each literal l to the conditions it appears in, for every executed action a. This allows us in future regressions to rapidly access only the conditional effects that are relevant to the current query. We also maintain a mapping from each literal to the clauses where it appears in the initial state formula b_I. Recalling that relevance is defined recursively, and that if p appears with q in an initial clause and r appears with p, then r is relevant to q, this allows us to run SAT queries only on variables relevant to the currently regressed formula.

7.2. Experimental set-up

We now demonstrate the practical value of regression, showing it to scale up well. We experiment with the more interesting benchmark domains from the contingent planning literature. Table 1 shows the properties of the various benchmarks. As can be seen, these benchmarks vary greatly on their properties. For examples, some domains exhibit many conditional effects over hidden propositions, while others do not. Some domains allow for many observations while for others information concerning the hidden propositions must be concluded from a single observation.

The domain sizes that we experiment with are currently unsolvable using all state-of-the-art generic online contingent planners [26,7,4,1]. 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 regresses the observed value, conjoins it with the initial belief, 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,

Algorithm 2: Regression operations.

IsGoal($\bar{a}_t, F_t$)
  Input: $\bar{a}_t$ – an action sequence of length t, F_t – known facts at t
  if *ConsistentWith*($\neg G, \bar{a}_t, F_t$) **then**
    return false
  else
    return true

IsApplicable($a, \bar{a}_t, F_t$)
  Input: a, an action, $\bar{a}_t$ – an action sequence of length t, F_t – known facts at t
  if *ConsistentWith*($\neg pre(a), \bar{a}_t, F_t$) **then**
    return false
  else
    return true

ConsistentWith($\phi, \bar{a}_t, F_t$)
  Input: ϕ: a formula, $\bar{a}_t$: an action sequence of length t,
    F_t: known facts at t
  $\phi_t \leftarrow \phi$
  for $i = t$ *down to 1* **do**
    if *IsTrue*(ϕ, F_t) **then**
      return true
    if *IsFalse*(ϕ, F_t) **then**
      return false
    $\phi_{t-1} \leftarrow rg^*_{a_t}(\phi_t)$
  return ConsistentWithInitialBelief(ϕ_0)

ConsistentWithInitialBelief(ϕ)
  Input: ϕ, a formula
  Convert ϕ into CNF form
  Apply unit propagation to ϕ
  $s \leftarrow MiniSAT(b_I \wedge \phi)$
  return $s \neq null$

Table 1
Benchmark properties. We report the number of propositions (P), the number of hidden propositions (H), the number of directly observable hidden propositions (O), the number of actions (A), the number of conditional effects containing hidden variables (C), and the number of clauses in the initial belief representation (b_I).

Domain	P	H	O	A	C	$\lvert b_I \rvert$
Battleship $n \times n$	$3n^2$	$2n^2$	n^2	n^2	0	$3(n-1)^2$
Minesweeper $n \times m$	$10nm + k$	$10nm$	$8nm$	k	0	$nm \cdot (\sum_{i=0}^{8} \binom{i}{8})$
Wumpus $n \times n$	$6n^2 + 2n$	$6n^2$	$6n^2$	$3n^2$	0	$3n^2 + 1$
Localize n	$n + 6$	$n + 4$	4	9	$4n$	1
RockSample $n \times m$	$4n^2 + m$	m	1	$5n^2 + m$	$m \cdot n^2$	m
MasterMind n, m	$2nm$	$nm + 2n$	$2n$	$nm + 2n$	$2\sum_{i=0}^{m} \binom{i}{m}$	$n + m$

to be comparable to previous experiments. For every problem, we run 25 iterations, varying on the hidden initial state, and hence the observations, and report the average time per step in milliseconds.

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. On the other hand, if, e.g., cells $3, 2$ and $3, 3$ both contain a ship, then the updated belief following the observations will conclude that cells $4, 2$ and $4, 3$ cannot contain a ship, and we will not fire on them. We must thus check for a set of cells whether they were hit, or possibly contain a ship. For this, we use only the cached facts in the heuristic computation, avoiding additional regression queries. Thus, the heuristic uses sound but incomplete information for making decisions, but it is very fast to compute. Our heuristic is less effective than the heuristic implemented by BG. For example, we require about 50 shots to solve Battleship 5×5 while BG require about 39 shots. On MineSweeper, we evaluate on a set of specially designed boards[5] where only a single action is applicable at each step, and no heuristic is needed. Hence, on this domain, the measured time purely estimates belief queries. In Wumpus, our heuristic maintains a grid map of the domain, marking all locations that are known to be safe. We repeatedly direct the agent towards the closest safe place from which additional information concerning nearby possibly safe cells can be sensed, until the treasure has been found.

[5] Created by Blai Bonet.

Table 2
Comparing decision time (ms) of regression and beam tracking.

Domain	Regression	Beams
Battleship 10×10	0.87	0.057
Battleship 20×20	2.5	0.074
Battleship 30×30	6.0	0.085
Battleship 40×40	9.8	0.095
Minesweeper 8×8	24	8.3
Minesweeper 16×16	85	12
Minesweeper 16×30	270	11
Large Wumpus 20×20	0.27	2.4
Large Wumpus 30×30	0.46	4.7
Large Wumpus 40×40	0.72	2.8
Large Wumpus 50×50	0.86	13

Table 3
Regression time (ms) for benchmark domains with conditional effects over hidden propositions.

Domain	Regression
Localize 20	3.1
Localize 30	9.5
Localize 40	25
Localize 50	32
RockSample 8×8	0.21
RockSample 16×16	0.99
RockSample 32×32	2.7
MasterMind 6c, 4p	1.2
MasterMind 8c, 4p	1.5
MasterMind 10c, 6p	23

We compare regression to Beam Tracking [5,6], which is a more advanced, approximate implementation of the ideas behind FBT. This method maintains for each proposition p that appears in a precondition, the goal, or an observation, a belief over the causally relevant propositions to p, called beams. When updating the belief, pairs of beams that share propositions are joined iteratively until convergence. This method is sound but incomplete.

We chose to compare to CBT/FBT for two reasons: First, it is the current state of the art in terms of its ability to scale-up to large domain sizes. Second, it has the best worst-case bounds on run-time due to its analysis in terms of problem width. In fact, a key point of our theoretical analysis is to show that regression enjoys similar properties. Thus, when the problem width is bounded by a constant or is even logarithmic in the input size, these are the only two methods that have polynomial run-time guarantees (contingent on a convenient representation of the initial state).

7.3. Comparing regression to BG beam tracking

We first compare our implementation of regression to the published implementation of beam tracking.[6] This implementation is non-generic, allowing execution only over three domains: Battleship, Wumpus, and Minesweeper, where it performs very well. Also, its implementation makes use of a manually designed multi-valued variable representation. As we have explained above (Section 5), the use of multi-valued variables impacts the width of the problem, and thus the worst-case performance of the methods. Furthermore, it is reasonable to believe that it lends additional practical efficiency, compared to the 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 32 GB RAM. Regression is implemented in C# while beam tracking uses Cygwin.

Nevertheless, as shown in Table 2 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 BG, 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 BG, regression is faster than beam tracking. In addition, as shown in Table 3, over domains not supported by the current BG code, regression-based belief tracking scales very well to domain sizes that cannot be handled by any other method.

[6] https://github.com/bonetblai/belief-tracking.

Table 4
Comparing action execution time (ms) of regression and our non-optimized implementation of beam tracking for propositional domains.

Domain	Regression		Beams	
	All actions	Observation actions	All actions	Observation actions
Wumpus 10-4-4	4.08	5.98	20.25	24.17
Wumpus 10-8-8	4.91	7.14	22.5	27.03
Wumpus 10-16-16	4.66	6.81	21.22	25.37
Battleship 6	1.53	1.47	53.63	49.15
Battleship 8	1.46	1.37	116.42	110.95
Battleship 10	1.65	1.53	204.41	198.47
Battleship 12	1.95	1.83	328.83	322.43
RockSample 8-4	0.17	0.55	0.09	0.1
RockSample 8-6	0.17	0.51	0.14	0.11
RockSample 8-8	0.21	0.58	0.66	0.56
RockSample 8-10	0.28	0.85	4.23	3.53
RockSample 8-12	0.3	0.93	52.93	42.15

7.4. Comparing regression to a non-optimized implementation of beam tracking

To further understand how regression compares to FBT and CBT,[7] we implemented a simple version of beam tracking into our own framework, which allows us to experiment with domains specified in a PDDL-like language, using the same planning heuristics and basic operations as our regression method. We implement beam tracking [5], where separated beliefs are maintained over the causally relevant propositions to any goal, precondition, or observation. In our implementation the separated beliefs are maintained by a simple DNF representation, that is, an explicit enumeration of all the possible assignments of truth values to the causally relevant propositions. As such, the local beliefs can be represented only if the number of relevant propositions (the causal width) is relatively low. In our implementation, once the number of relevant propositions grows beyond 15, enumerating the possible assignments becomes too time consuming to experiment with.

When updating the belief, complete joins of all the possible assignments of the neighboring beliefs that share a variable in common becomes infeasible, and we therefore implement the iterative method that joins only pairs of local beliefs until convergence. We implemented an optimized join operation, removing first identical states, and searching for propositions that have only a single valid value in some belief, before running the actual join operation. In all the domains that we experiment with, only a handful of iterations were required before convergence. The implementation uses the same caching mechanism as our regression method, allowing us to use the above belief mechanism only for propositions whose value is currently unknown.

Conforming with the definitions of Bonet and Geffner [6], the initial belief is specified only over single literals, avoiding complex constraints over sets of literals. Such constraints are introduced by the conditional effects of a special *init* action. As such, the implementation of these domains differs from the domains used in the experiments in the previous section, and runtime may vary. Furthermore, such an implementation is less convenient for regression, which benefits from a CNF definition of the initial belief instead of additional conditional effects that must be regressed. On the other hand, we did not implement the invariant constraints of Bonet and Geffner, which may reduce runtime for beam tracking.

Table 4 compares the average time for executing a single action using regression and our implementation of beam tracking. When executing an action using beam tracking, we perform a belief update, including the iterative joins. When executing an action using regression we validate that the preconditions hold using one regression query, then, in case of an observation action, execute a second regression query to join the regressed observation to the initial belief state. As such, we also report in Table 4 the execution time of observation actions. In all experiments we run both methods together over the exact same sequence of actions and observations, and we report the average over 25 runs.

There are some interesting observations in Table 4. First, in all these domains, the complete regression is much faster than the incomplete beam tracking over the larger domains. This may be attributed to the simplistic and inefficient DNF belief representation.

Different domains exhibit different properties. The Wumpus domain has pits and wumpuses that need to be avoided, while searching for the gold that is hidden on a map. Wumpus *n*-*m*-*k* has a board size of $n \times n$ with m wumpuses and k pits hidden on the board. Wumpus is perhaps the simplest domain here, with a relatively low number of causally relevant variables (4 at most), and hence scaling up does not require significant additional effort from both methods. In this domain, beam tracking is about 4 times slower than regression.

In Battleship n, where n is the board size, the causally relevant propositions represent cells in the immediate vicinity of the tracked cell. In this domain, as the number of cells grows quadratically with the board size, the belief update becomes more difficult. We also tried to add additional constraints over cells with a distance of 2 from the tracked cell, resulting in

[7] A more detailed explanation of FBT and CBT is given in the next section.

Table 5
Comparing observation regression time (ms) of regression with and without caching of known literals.

Domain	No caching	Caching
Localize 20	50.84	4.8
Localize 30	223.7	10.9
Localize 40	712.7	39.7
Localize 50	1151.7	64.6

25 relevant literals, but our DNF belief representation could not handle this many propositions, although regression had no trouble scaling up to 25 relevant propositions.

RockSample is arguably the most interesting domain of this set, as it contains additional conditional effects aside from the *init* action. In RockSample *n*-*m*, a rover patrols an $n \times n$ board containing m rocks whose location is known. The rover can activate a medium range sensor, capturing the existence of "good" rocks containing an interesting mineral up to 2 cells away. The rover can then observe whether a green light was lit on the sensor, meaning that at least one good rock is in range. Thus, all rocks are causally relevant to the green light observation. On the other hand, during a regression of a green light observation, only rocks in range of the current cells are relevant to the observation. Indeed, this domain demonstrates the advantage of regression over beam tracking, when the contextual width, which is the number of relevant propositions given a specific sequence of actions and observations, is much lower than the true causal width of the problem.

7.5. Regression and caching

In Section 7.1 we suggest an improvement to regression, caching at each time step the set of literals whose value is known at that time step. Then, future regression queries can be simplified using this cached information, resulting in smaller formulas. We experiment over the Localize domain, containing many conditional effects, and thus, potentially large regressed formulas, to study the effect of caching on regression queries.

We measure the average time over 25 runs, with and without caching, for regressing observations. Table 5 shows the substantial benefits that the caching of known facts provides in this case. In domains with less conditional effects, such as Battleship and Wumpus, there is almost no effect to the caching of learned facts.

Another interesting effect of caching learned facts is with respect to the action–observation sequence length. As more facts are learned, the formulas can be reduced to a smaller size. Therefore, regression queries produce larger formulas in the beginning of the action–observation sequence, and much smaller formulas later on. Fig. 2 shows the size of the formula (size of the syntax tree of the formula) and the width of the formula (number of propositions) as the length of the action–observation sequence grows, for observation regressions in the Localize domain. We report the maximal number for each time step over 25 executions.

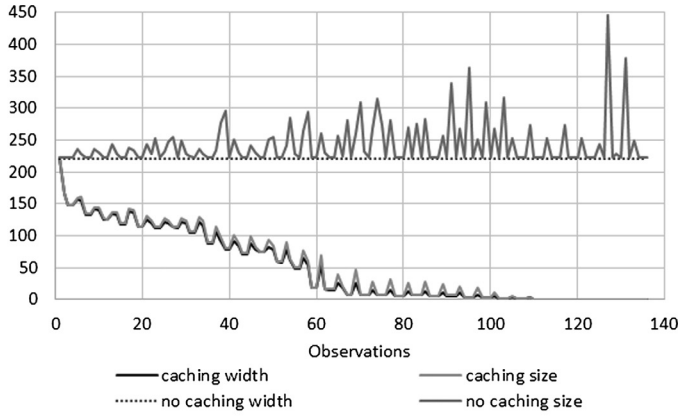
As can be seen, with no caching, the width of the formula is constant, including all the unknown propositions in this domain. With caching, the width rapidly becomes significantly lower. The size of the formula can be significantly larger than the width. When no caching is allowed, there are many cases where the regression queries result in very large formulas. With caching, the size of the formula is always of the same order of magnitude as the width. These experiments clearly show the value of caching learned facts.

8. Related work

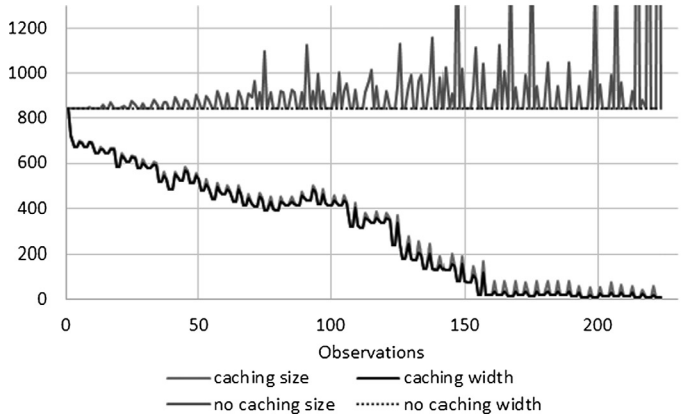
We now review related work, starting with previous usage of regression for various applications, then regression in planning, and finally similar methods for belief tracking, and their comparison to our work.

Regression was introduced into planing as a method for generalizing backwards search and goes back at least to [28]. Goal-regression refers to the computation of the weakest pre-image or weakest precondition of some goal condition under a given operator. That is, the weakest condition that if satisfied *before* the operator is applied ensures that the goal will hold *after* the operator is applied. Nowadays, regression-based planning refers to planning backwards from the goal using goal-regression. Our focus is not on regression-based planners, but on the goal regression technique as a method for answering queries about an implicitly represented belief state in planning under uncertainty with partial observability.

Much work on goal regression techniques was done in the context of the situation calculus [20,21]. Regression is a natural technique in situation calculus because situations are defined in terms of an initial situation and a sequence of applied actions, using the $do(a, s)$ relation. This is exactly the implicit state representation used in this paper. Consequently, many of the techniques explored in this paper have their background in situation calculus. The very nature of the regression formula stems from the successor-state axiom of the situation calculus. The situation calculus uses $poss(a, s)$ to denote that a is executable in situation s. As in our work, this aspect of the action is not propagated in the regression formula [20,21]. More specifically, the issue of the regression of knowledge producing actions, i.e., actions with observations, was discussed in Scherl and Levesque [23], and the technique of conditioning the regression formula on the observation condition ($\omega_{a,o}$ in our case, and $\psi(s)$ in situation calculus) was considered in earlier work on diagnosis with situation calculus [18].



(a) Localize 20



(b) Localize 40

Fig. 2. Comparing the formula size and width, with and without caching.

However, the situation calculus is a first-order formalism that is much more expressive than the propositional formalism used in most planners. As such, it must deal with the issue of ramification, stemming from the ability to formulate various state axioms, and with quantification. And in the context of the work on reasoning about knowledge producing actions [23], the formulas developed use a modal logic of knowledge. Our work focuses specifically on the simple propositional language that is used in most current planners, and even more specifically, on belief tracking, leading to a thinner and simpler formalism. We extend Rintanen's work [22] on regression to handle observations, paying special attention to the computational complexity of the reasoning algorithms, and leading to theoretically appealing and empirically validated results.

Explicit belief tracking refers to the maintenance of an explicit or symbolic representation of the set of possible states of the world. Enumerating the set of possible states is not a practical option – their number is worst-case exponential in the number of state variables, and in practice, it often grows quickly. 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 [3], prime-implicates, CNFs, and DNFs [27]. Unfortunately, symbolic representations also have an exponential worst-case description, and some methods can require worst-case exponential work per 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.

One way to address the exponential worst-case description is to try to focus on maintaining and/or computing relevant information only. For planning algorithms we can focus on a narrower scope, which requires only answering specific queries concerning the set of possible states. CFF [15] exploited this observation within a planner that searches over belief states. It uses an implicit representation of the belief state, avoiding the explicit update of belief states. CFF maintains a copy of the state variables for every time point, together with constraints over the values of these variables. This produces a SAT encoding similar to earlier work on SAT-based solutions to planning under uncertainty [12]. The representation is updated with each action and grows linearly with the number of actions executed, and hence has low-order polynomial space complexity. However, answering a query regarding the current value of a variable requires solving an UNSAT problem is co-NP hard, although seems to work quite well in practice. As information is obtained, the representation and constraints concerning earlier stages can be simplified. Our regression-based method takes this lazy approach to the extreme, avoiding

the maintenance and update required by CFF's belief representation altogether. In a previous paper [8] we showed that CFF's belief tracking method scales worse than regression.

Closely related to CFF's method, and very similar in many aspects to our own work, is Shahaf and Amir's work on logical filtering with circuits [25]. In that work, the authors show how to maintain a circuit representation of the current belief state, whose size is polynomial in the set of propositional fluents and the length of the action–observation sequence. The circuit construction is closely related to the regression formula developed in this paper. More precisely, the circuit could be thought of as incrementally updating a regression formula for every literal in a forward manner. Initially, a circuit corresponding to the initial state formula is constructed. For each primitive proposition p, a sub-circuit that "explains" the conditions under which p is true is maintained. Initially, p is true iff it is true in the initial state. After each action and observation, the circuit is updated. The explanation for p being true following an action is based on Equation (3.2), that is, if p was true before the action was executed and was not destroyed by the action, or if p is an effect of the action. This new explanation replaces the old explanation for p (i.e., in the one that explains p in the state that preceded the execution of the last action). Thus, one does not explicitly maintain explanations for intermediate states, although, in fact, a node that corresponds to the explanation of every primitive proposition at every state in the sequence will always be maintained in the circuit. If an observation p is made, then the initial state formula is conjoined with p's explanation.

If a standard formula is used, the new formula that "explains" p could be twice as large as the old formula, and hence the formula grows exponentially. However, when a circuit representation is used, identical substructures require only a single copy, plus multiple pointers to this copy, ensuring a poly-sized representation. Although constructed forward and incrementally, the circuit represents, compactly, the same formula that we obtain via regression, with observations conjoined to the initial belief state, and without the intermediate simplifications we perform on that formula. Hence, the essential differences between Shahaf and Amir's method and ours are as follows: they use a compact representation for the regression formula, constructing and updating this formula for every primitive proposition at every step. This formula is maintained in the form of a circuit that need not replicate identical sub-structure. A new explanation is generated following each action using a forward update of the formula. When querying, they feed the entire formula into a SAT solver for circuits. As such, their approach is very similar to CFF, aside from the different formula structure. As opposed to CFF and the Shahaf and Amir, we take a lazy approach, in which very little information (i.e., our cache) is maintained, and a regression formula is computed on the fly when needed. This formula is simplified during its backwards construction using the cache and previous observations.

In many respects, Shahaf and Amir's technique is quite similar to CFF's belief maintenance approach. CFF explicitly maintains a proposition for every fluent and every time point, which corresponds to the pointers used within the circuit. Both structures implicitly represent the belief state using a formula, that is maintained and updated as actions are executed. Both representations require only low-order polynomial space to maintain, and this is achieved by not replicating identical sub-structures – in the case of circuits these are sub-circuits, and in the case of the SAT encoding, these are identical intermediate variables. In both cases, verifying the validity of some condition requires a validity check (with respect to the current circuit or formula), which is Co-NP hard. It is claimed that the circuit representation, with its explicit sub-structures, can be exploited for added efficiency, and Shahaf and Amir's experimental results indicate that it is better than what they refer to as "unrolling", which appears to be similar to the SAT-based encoding. It is an open question whether similar width-based guarantees can be provided for the circuit validity checks. Given the close similarity to our regression formula with regressed observation (Section 6.1), it is likely to face similar issues regarding the impact of regressed observations on the structure of the initial state formula.

More recently, Bonet and Geffner further exploited the idea of focusing on relevant information. They maintain the beliefs over each variable p separately, by maintaining the possible set of assignments of relevant variables to p [5,6]. In this scheme, a variable may appear in multiple belief spaces, corresponding to different variables, as one variable may be relevant to multiple variables and the "belief-state" associated with each variable is exponential in the number of variables relevant to it. In problems where the number of relevant variables is large, this can still be too large to store and update.

To address this problem, BG suggest maintaining beliefs only over the causally relevant variables, ignoring the evidentially relevant ones. To compensate for the lack of evidentially relevant variables, they maintain causally relevant beliefs over observed variables as well. Thus, their second algorithm, *causal belief tracking* (CBT) uses more factored beliefs, but of a smaller width – the causal width of the problem.

To pass information between different factored beliefs that share similar variables, CBT uses join operations. But while the size of the factored beliefs is reduced in CBT, the complete join operation over all beliefs that share a variable may still be infeasible. BG therefore suggest to run iterative join operations of pairs of beliefs that share a variable, until convergence. They also suggest to leverage invariant constraints over the problem to add power to the iterative join operations. This version of CBT, which they call *beam tracking* is sound but incomplete, yet scales up to much larger problems than previous methods. An extensive empirical evaluation of this method with the regression method was presented in the previous section.

9. Conclusion

In this paper we discussed 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 evaluated the use of regression empirically, showing that it scales up very well on many current contingent benchmark domains.

Regression over actions naturally enjoys a focus on relevant variables only, and over observations, too, if their regression is conjoined to the initial state. 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 [17] or FBT.

The success of approximate FBT and CBT techniques points to potentially interesting line of future work focusing on approximate regression methods. For example, by possibly weakening the regression formula in some cases and maintaining a simple syntactic form, one might be able to reduce its computational cost in practice, at the price of some loss of completeness. In addition, it may be possible to combine ideas from CBT and regression to obtain a hybrid method, where the information maintained by CBT would be used to simplify the regression formula, or even to refrain from using it when the condition is already implied by the maintained belief state, while regression will ensure the completeness of the approach.

Acknowledgements

This work was supported by ISF Grant 933/13, by the Lynn and William Frankel Center for Computer Science, and by the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive Robotics Center of Ben-Gurion University of the Negev.

References

[1] A. Albore, H. Palacios, H. Geffner, A translation-based approach to contingent planning, in: IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11–17, 2009, 2009, pp. 1623–1628.
[2] E. Amir, S.J. Russell, Logical filtering, in: IJCAI'03, 2003, pp. 75–82.
[3] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, P. Traverso, MBP: a model based planner, in: IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information, 2001.
[4] B. Bonet, H. Geffner, Planning under partial observability by classical replanning: theory and experiments, in: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16–22, 2011, 2011, pp. 1936–1941.
[5] B. Bonet, H. Geffner, Width and complexity of belief tracking in non-deterministic conformant and contingent planning, in: AAAI, 2012.
[6] B. Bonet, H. Geffner, Belief tracking for planning with sensing: width, complexity and approximation, J. Artif. Intell. Res. 50 (2014) 923–970.
[7] R.I. Brafman, G. Shani, A multi-path compilation approach to contingent planning, in: AAAI'12, 2012.
[8] R.I. Brafman, G. Shani, Replanning in domains with partial information and sensing actions, J. Artif. Intell. Res. 45 (2012) 565–600.
[9] A. Darwiche, P. Marquis, A knowledge compilation map, J. Artif. Intell. Res. 17 (2002) 229–264.
[10] E.W. Dijkstra, A Discipline of Programming, vol. 1, Prentice-Hall, Englewood Cliffs, 1976.
[11] N. Eén, N. Sörensson, An extensible SAT-solver, in: Theory and Applications of Satisfiability Testing, Springer, 2003, pp. 502–518.
[12] P. Ferraris, E. Giunchiglia, Planning as satisfiability in nondeterministic domains, in: AAAI'00, 2000, pp. 748–753.
[13] A. Herzig, J. Lang, P. Marquis, Action representation and partially observable planning in epistemic logic, in: IJCAI'03, 2003, pp. 1067–1072.
[14] C.A.R. Hoare, An axiomatic basis for computer programming, Commun. ACM 12 (10) (1969) 576–580.
[15] J. Hoffmann, R.I. Brafman, Conformant planning via heuristic forward search: a new approach, Artif. Intell. 170 (6) (2006) 507–541.
[16] H. Levesque, F. Pirri, R. Reiter, Foundations for the situation calculus, Linköp. Electron. Artic. Comput. Inf. Sci. 3 (18) (1998).
[17] Y. Lin, M.J. Druzdzel, Computational advantages of relevance reasoning in Bayesian belief networks, in: UAI'97, 1997, pp. 342–350.
[18] S. McIlraith, R. Scherl, What sensing tells us: towards a formal theory of testing for dynamical systems, in: AAAI'00, 2000.
[19] H. Palacios, H. Geffner, Compiling uncertainty away in conformant planning problems with bounded width, J. Artif. Intell. Res. 35 (2009) 623–675.
[20] R. Reiter, The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression, in: Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of J. McCarthy, 1991.
[21] R. Reiter, Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, MIT Press, 2001.
[22] J. Rintanen, Regression for classical and nondeterministic planning, in: ECAI'08, 2008, pp. 568–572.
[23] R. Scherl, H. Levesque, The frame problem and knowledge producing actions, in: AAAI'93, 1993, pp. 689–695.
[24] R. Scherl, T.C. Son, C. Baral, State-based regression with sensing and knowledge, Int. J. Softw. Inform. 3 (1) (2009) 3–29.
[25] D. Shahaf, E. Amir, Logical circuit filtering, in: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6–12, 2007, 2007, pp. 2611–2618.
[26] G. Shani, R.I. Brafman, Replanning in domains with partial information and sensing actions, in: IJCAI'11, 2011, pp. 2021–2026.
[27] S.T. To, E. Pontelli, T.C. Son, On the effectiveness of CNF and DNF representations in contingent planning, in: IJCAI'11, 2011, pp. 2033–2038.
[28] R. Waldinger, Achieving several goals simultaneously, in: E. Elcock, D. Michie (Eds.), Machine Intelligence, vol. 8, Ellis Horwood, Edinburgh, Scotland, 1977, pp. 94–136.