Integrated Common Sense Learning and Planning in POMDPs

Brendan Juba*
Harvard University
bjuba@alum.mit.edu
October 14, 2013

Abstract

We describe an efficient algorithm for constructing decision tree policies for goal-oriented factored POMDPs based on “common sense” learned from example traces. More precisely, when (1) there is a CNF that is almost always observed satisfied on the traces of the POMDP, capturing a sufficient approximation of its dynamics and (2) for a decision tree policy of bounded complexity, there exist small-space resolution proofs that a given goal is achieved on each branch using the aforementioned CNF “common sense rules,” then our algorithm (provably) succeeds at finding a policy for the given goal. Such a CNF always exists for noisy STRIPS domains, for example.

1 Introduction

A central problem in Artificial Intelligence, first considered by McCarthy [21] concerns how to enable a machine to autonomously operate in an environment. The dominant approach to this problem – with some exceptions discussed later – has been to first build a model of the environment, and then use the model representation to generate plans. While these models were originally built by hand, modern advances across AI have been powered by the substitution of learning for hand-crafted knowledge representations, and we will likewise here consider the problem of learning an environment in order to support planning. Although such separation of concerns is generally considered good engineering practice, such a decomposition of the problem mediated by an explicit intermediate representation may be intractable even when the overall problem has efficient algorithms: Khardon and Roth [17] famously presented an efficient algorithm for an NP-hard reasoning task over a learned DNF representation, for example. And, learning models of interactive environments directly is indeed usually infeasible (if, e.g., they include DFAs [15]). Although it is feasible to learn environments that are generated by simple models, the problem becomes much harder when we are trying to utilize incomplete state information, when simple models do not perfectly capture the environment’s dynamics, and when the state space is large—indeed, any one of these alone poses a challenge that current work seeks to address, but for most interesting applications, we must confront all three. In this work, we propose an approach to formulating plans using integrated learning and reasoning that provably simultaneously addresses all three challenges.

*Supported by ONR grant number N000141210358.
We will aim to learn the dynamics (and observation model) of a factored Partially Observed Markov Decision Process (POMDP) with partial but noiseless observations, a standard partial-information environment model in which the states are described by a vector of \( n \) attributes, sufficiently well to achieve a variety of goals on the same POMDP. A naïve approach might directly estimate the dynamics of (“belief”) distributions over the \( 2^n \) states produced by the model, but notice that the dynamics can require \( 2^{\Omega(2^n)} \) bits to represent. We will need some further assumptions. First, we do not attempt to address the exploration problem: we avoid the exploration problem by fixing an exploration policy and only seeking to plan using actions it explores with non-negligible probability.

Following the spirit of McCarthy’s approach, we propose to focus on cases where an agent is given a goal predicate (a DNF formula here) as input and, based on “common sense knowledge” that can be learned from experience with the POMDP, there exists a proof that the goal is satisfied on certain traces of interaction between the agent’s policy and the POMDP; technically, the proofs serve to provide evaluations of the agent’s performance under partial information since our “reward function” is given as input separately from the domain examples. In particular, we will use treelike resolution proofs of bounded \textit{Strahler numbers} (aka pebble number or clause space, cf. Section 2.2.1).

We propose here to use algorithms for reasoning over such common sense knowledge learned from partial information, as described by Juba [12] for most known tractable proof systems, to encapsulate the learning of the POMDP’s dynamics in planning. Indeed, we reduce the planning problem to answering such queries, establishing that such solutions to the integrated learning and reasoning problem are sufficient to solve our POMDP planning problem. This is analogous to Kautz and Selman’s SATPLAN [13], but for stochastic domains, with richer policy representations, and with the domain encoding learned from the partially observed traces; this learning is provided by the aforementioned algorithm from prior work by Juba, that stands in for the SAT-solver in this analogy. Specifically, we will leverage the integrated learning and reasoning algorithm to find decision tree policies of bounded Strahler number for achieving the given goal in the POMDP (when they exist). This is again a natural restriction: it encompasses bounded-fault tolerant policies [10], for example. Our reduction searches through policies of a constant Strahler number bound in polynomial time by adapting an algorithm of Ehrenfeucht and Haussler [5] from supervised learning to also search over actions and invoke the reasoning algorithm on an appropriate query to detect suitable leaves. (We can also search through all decision trees of a given size in \textit{quasipolynomial} time, i.e., time \( 2^{\text{polylog } n} \).) We give an analysis of the overall algorithm showing in particular that only polynomially many traces are needed to learn the POMDP well enough to find plans.

As promised, we note that this system finds plans even when the domain’s dynamics are only partially, approximately captured by a small CNF formula (and hence, by the premises in a small resolution proof). For example, logical encodings of planning problems typically use “frame axioms” that assert that nothing changes unless it is the effect of an action. In a real world setting, these axioms are not strictly true, but such rules still provide a useful approximation. It is therefore crucial that we can learn to utilize such imperfect logical encodings. We will more generally be able to learn to plan in noisy versions of STRIPS instances [8] (a standard test domain of interest) which are approximated well but imperfectly by their noiseless versions. Also, as a consequence of our adoption of \textit{PAC-Semantics} [33] for our logic, we will also be able to soundly incorporate explicitly specified background knowledge into our algorithms, even when this knowledge may likewise only hold in an approximate sense.
2 Preliminaries

2.1 Factored POMDPs

Informally, a POMDP is a state-based model of an environment for an agent that evolves probabilistically and only provides the agent with partial information about its states.

**Definition 1 (POMDP)** A Partially Observed Markov Decision Process (POMDP) is given by collections of distributions on a state space \( S \) and an observation space \( O \) as follows: there is an action set \( A \) such that for each action \( a \in A \) and \( s \in S \), there is a distribution \( D_{(a,s)} \) over \( S \), and for each \( s \in S \) there is a distribution \( D_s \) over \( O \).

For any fixed sequence of actions \( a^{(1)}, a^{(2)}, \ldots \) from \( A \), the distributions \( \{D_{(a,s)}\}_{(a,s) \in A \times S} \) give rise to a (non-stationary) Markov process on \( S \): given an initial state \( s^{(1)} \), \( s^{(2)} \) is drawn from \( D_{(a^{(2)}, s^{(1)})} \), and generally thereafter, \( s^{(i+1)} \) is drawn from \( D_{(a^{(i)}, s^{(i)})} \). For a sequence of states so generated, the distributions \( \{D_s\}_{s \in S} \) now give rise to the POMDP distribution over observations:

The \( i \)th observation \( o^{(i)} \) is drawn from \( D_{s^{(i)}} \). We normally think of the agent choosing \( a^{(i)} \) on the basis of the history of interaction with the environment somehow, i.e., with knowledge of \( o^{(1)}, \ldots, o^{(i)} \) and \( a^{(1)}, \ldots, a^{(i-1)} \). We refer to the agent’s strategy for choosing such actions as a *policy*. One normally allows some kind of a reward (or loss) function over the states \( S \) to quantify how “good” or “bad” an agent’s policy for acting in the environment is (we will elaborate on this shortly).

We will not use the most general definition of a POMDP; we will instead use the following natural special case featuring most of the key features of a POMDP. We will assume that the state space is factored, that is, described by \( n \) propositional variables (”fluents,” in the usual language of planning), taking \( S = \{0, 1\}^n \) (we will continue to denote the indices of these propositional variables in \( S \) by subscripts, hence our use of superscripts to denote the sequence of states). We choose to use propositional representations because they are sufficient to capture the only cases of first-order representations (to our knowledge) that have tractable learning and inference algorithms—that is, following (e.g.) Valiant [33], we could consider relational representations of small arity and a polynomial size universe of objects, and take the atomic formulas obtained by various bindings of our relations over these objects as our propositional variables. But, the propositional case is surely simpler and suffices for the current work. Thus, in particular, we will fix a (possibly polynomially related to \( n \)) horizon bound \( T \), and for each \( t \)th step (of a candidate plan) and \( i \)th component of the state of the POMDP, we will have a separate propositional variable \( s^{(i)}_i \).

When reasoning about plans for the POMDP, we will naturally likewise use a propositional representation: for each possible action \( \alpha \) and action taken by the agent \( a^{(t)} \), we will have a vector of propositional variables \( a^{(t)}_\alpha \) encoding \( [a^{(t)} = \alpha] \). Naturally, these propositional variables should be mutually exclusive. Typically, we will add a list of clauses to our background knowledge (”KB”) for every pair of actions \( \alpha \neq \alpha' \) asserting that only one is taken, i.e., \( -a^{(t)}_\alpha \vee -a^{(t)}_{\alpha'} \).

The observations will then have the form of state vectors with some of their entries masked. More precisely:

**Definition 2 (Partial states)** A partial state \( \rho \) is an element of \( \{0, 1, *\}^n \). We say that a partial state \( \rho \) is consistent with a state \( s \in \{0, 1\}^n \) if whenever \( \rho_i \neq * \), \( \rho_i = s_i \).

The observations will consist of partial states produced by a fixed masking process applied to the current state of the POMDP.
Definition 3 (Masking process) A mask is a function \( m : \{0,1\}^n \to \{0,1,*\}^n \), with the property that for any \( s \in \{0,1\}^n \), \( m(s) \) is consistent with \( s \). A masking process \( M \) is a mask-valued random variable (i.e., a random function).

Notice, which attributes are hidden can depend arbitrarily on the underlying state.

It will be helpful for us to distinguish between the partial state as an observation and as a state of knowledge. In particular, our decision-tree policies will produce actions based on the three-valued observation vectors. Supposing we denote our observation in the \( t \)th step by \( o(t) \), We will encode \( [o(t)_i = b] \) for \( b \in \{0,1,*\} \) as variables \( x(t)_i,b \), and add the clauses \( x(t)_i,0 \lor x(t)_i,1 \lor x(t)_i,* \) and \( \neg x(t)_i \lor \neg x(t)_{i,b'} \) for all \( i \) and \( b \neq b' \) to the KB (thus asserting that \( x \) takes exactly one of the three values).

In contrast to the usual set-up, our reward function is not fixed with the POMDP. It will be specified to the algorithm by a goal predicate \( G \) given by a DNF over the propositional state variables. Along these lines, an agent’s policy is considered good if it manages to (quickly) reach a state in which \( G \) is satisfied. (As opposed to demanding a policy that maintains a high average reward over time.)

2.1.1 STRIPS domains

A class of simple domains were first introduced for the STRIPS planner of Fikes and Nilsson [8]. Such domains are sufficiently general to provide a variety of examples that serve as excellent illustrations. In a (propositionalized) STRIPS domain, the actions are described by lists of literals capturing the action’s preconditions and effects. The intention is that the preconditions must be satisfied at time \( t \) in order for the action to be available to the agent. Once the action is taken, the state of the environment is updated so that at time \( t + 1 \), the literals capturing the effects are all satisfied. All other literals are unchanged. We will take the convention that if the preconditions of an action are not satisfied at time \( t \) but the agent tries to “take the action at time \( t \),” the action fails—as opposed to being unassertable, as traditionally required; this will be useful in partial information settings where the agent may not be certain whether or not the preconditions hold. Finally, the goals are given by a conjunction of literals.

The encoding of the action rules into clauses is then very natural. Supposing the action \( \alpha \) has preconditions \( p_1, \ldots, p_k \) and effects \( e_1, \ldots, e_\ell \), for each time \( t \), we have for each effect \( e_i \) a clause \( [p_1(t) \land \cdots \land p_k(t) \land a_\alpha(t)] \Rightarrow e_i(t+1) \). The convention that the attributes of the state are only changed by actions is encoded by “frame axioms” like so. For each literal \( \ell \) over some state of the environment and each action \( \alpha_1 \ldots, \alpha_k \) that has an effect that falsifies \( \ell \), i.e., has some \( e_i = \neg \ell \), we have a clause \( [\ell(t) \land \neg a_{\alpha_1(t)} \land \cdots \land \neg a_{\alpha_k(t)}] \Rightarrow \ell(t+1) \).

Note that our DNF goals generalize the STRIPS-style “conjunctive goals.” Likewise, although the actions are not necessarily captured by STRIPS actions, and the evolution of our environment is generally not deterministic, such examples are helpful to keep in mind as examples of POMDPs for which the dynamics can be concisely described by a CNF.

2.1.2 An Example

We can illustrate (STRIPS) rules, POMDPs, and their relationship with an example of a simple domain based on the Gripper problem from the first International Planning Competition (IPC 1998). Informally, the problem captures an environment consisting of two rooms, which may
contain several balls. The agent is a robot that can pick up and put down the balls and carry them between rooms. The goal is usually to carry all of the balls to a given room.

The states. We can describe the environment (POMDP) with a vector of Boolean attributes as follows: we will index the rooms by \( r \in \{1, 2\} \), and the balls by \( b \in B \) (for some other set \( B \)). We will let the variables \( x_r \) indicate whether the agent is in room \( r \), which are mutually exclusive; the variables \( x_{r,b} \) indicate whether ball \( b \) is in room \( r \), and for each fixed \( b \) the \( x_{r,b} \) are mutually exclusive; and the variables \( x_b \) indicate whether the agent is holding ball \( b \), and these are also mutually exclusive. The agent’s actions are \( a_r \), meaning “go to room \( r \)” ; \( a_b \), meaning “pick up ball \( b \)” ; and \( a_0 \), meaning “drop.”

The rules. The state variables and actions are related by the following clauses, and these are the STRIPS-style rules that will approximate the dynamics of the POMDP. First, if the agent executes action \( a_r \), then the \( x_r \) is set to 1 and for \( r' \neq r \), \( x_{r'} \) is set to 0, that is for each time index \( t \), we have rules \( -a_r(t) \lor x_r(t+1) \) and \( -a_r(t) \lor -x_{r'}(t+1) \). In the usual language of STRIPS, \( x_r(t+1) \) and \( -x_{r'}(t+1) \) are the effects of \( a_r \) at \( t \). Furthermore, if the agent is holding ball \( b \), then \( x_{b,r} \) is also set to 1 and \( x_{b,r'} \) is set to 0. These correspond to clauses \( -x_{b,r}(t) \lor -a_r(t) \lor x_{b,r'}(t+1) \) and \( -x_{b,r}(t) \lor -a_r(t) \lor -x_{b,r'}(t+1) \) (for \( r' \neq r \)). These are conditional effects of \( a_r(t) \) with the precondition \( x_{b,r}(t) \). Now, if the agent is in room \( r \) and ball \( b \) is also in room \( r \), then the agent may pick up \( b \) with its gripper; when this happens, the agent also drops any other balls it was holding. This is captured by clauses \( -x_{b,r}(t) \lor -x_{r}(t) \lor -a_b(t) \lor x_b(t+1) \) for each \( r \) and \( b \), and clauses \( -a_b(t) \lor -x_{b'}(t+1) \) for \( b' \neq b \). Here \( x_{b,r}(t) \) and \( x_{r,b}(t) \) are again the preconditions for (conditional effects of) the action (they must jointly hold for some room \( r \), or at the end the agent fails to hold \( b \)). Finally, if the agent executes \( a_0 \), it simply ceases to hold any balls at all, that is, \( -a_0(t) \lor -x_b(t+1) \) for all \( b \). Now, we also require frame axioms indicating that the state does not change unless it is the effect of one of these actions as described above. Actually, these are elegantly stated using negation-as-failure, as we describe in Section 4, so we will not list out the classical form of these rules here.

The actual POMDP and validity of the STRIPS-style rules. As we hinted at above, the deterministic STRIPS-style rules will only approximate the true environment, which is actually a Markov decision process. In this Markov decision process, we will model an agent whose actions are not flawlessly performed: suppose that the pick-up actions fail 1% of the time, and that when carrying a ball from one room to another, the agent has a 1% chance of dropping the ball in each of the rooms (for a 2% chance overall of ending up in the destination without holding the ball). That is, in states where the pick-up action \( a_b \) should have set \( x_b \) to 1, the agent only enters such a state with probability .99, and enters a state where \( x_b \) is (still) 0 instead with probability .01. Likewise, in states where \( x_b \) is 1, \( x_r \) is 1, and the agent takes action \( a_{r'} \), then with probability .98 the agent enters the state where \( x_b \) is still 1, \( x_{r,b} \) and \( x_r \) are 0, and \( x_{r',b} \) and \( x_{r'} \) are 1; but, with probability .01, \( x_b \) switches to 0, and with probability .01, not only does \( x_b \) become 0, but moreover \( x_{r,b} \) is still 1 and \( x_{r',b} \) is still 0 (that is, the ball didn’t get carried). We could also further envision an environment in which the balls may switch rooms without our agent’s intervention, because they were moved by some other agent for example. We will assume that the other effects (in particular,
the room-switching effect of \( a_r \) occur with probability 1, and thus the corresponding rules are actually “1-valid” in the language of PAC-Semantics that we will discuss next; the rules for \( a_r \) described above turn out to be “.99-valid” except for the frame axiom that asserts that \( x_b \) should remain 1 (unless \( a_0 \) or \( a_{\nu'} \) are executed), which is only “.98-valid.”

In a POMDP, the agent does not observe the complete states of these underlying Markov decision processes in general. In our case, partial states are generated by the following simple masking process: when \( x_r \) is 1 and \( x_{r,b} \) is not 1, the variables \( x_{r',b} \) for \( r' \neq r \) are masked (always set to \( * \) in the partial states appearing in the example traces and the partial states provided to the agent’s policy when it is executed). Intuitively, this means that the agent does not “see” the balls in room \( r' \), but may know that a ball \( b \) is in room \( r \) and not \( r' \). In this simple environment, we will assume that all of the other attributes remain unmasked. Therefore, all of the rules that do not mention the \( x_{r,b} \) variables are always “witnessed true” (i.e., they always have a satisfied literal—we will define this formally later); it turns out that the rules that do mention \( x_{r,b} \) – namely, the rules capturing the effects of \( a_r \) and \( a_b \) – also each have a satisfied literal and are therefore “witnessed true” with probability .99. Notice, \( x_{r,b} \) (in the effects of \( a_r \) and \( a_{\nu'} \)) is only not “witnessed true” when both \( \neg x_r \) and \( \neg x_b \) are 0, and in this case, after \( a_r \), \( x_{r,b} \) and \( \neg x_{r',b} \) are both guaranteed to be witnessed true unless the ball was dropped in room \( r' \), which occurs with probability .01. Similarly, in the rules describing the effect of \( a_b \), either \( \neg x_r \) or \( \neg x_{b,r} \) is witnessed true (if the ball is absent), or else \( x_b \) is witnessed true unless the action fails, which occurs with probability .01. Since these STRIPS-style rules are all “witnessed true” with high probability, it will turn out that they are therefore guaranteed to be learnable in pursuit of policy construction, as we will see.

2.2 PAC-Semantics

Valiant [33] introduced PAC-Semantics in order to capture the property satisfied by the outputs of PAC-learning algorithms when formulated in a logic. It is proposed to capture certain kinds of learned “common sense” knowledge in general [34].

**Definition 4 (\((1 - \epsilon)-valid\))** Given a distribution \( D \), we say that a relation \( R \) is \((1 - \epsilon)-valid\) w.r.t. \( D \) if \( \Pr_{x \in D}[R(x) = 1] \geq 1 - \epsilon \).

Naturally, our relations will be given by formulas of propositional logic (i.e., formed by the usual Boolean connectives over propositional variables) over the literals describing the observations over the \( n \) state attributes and the agent’s actions for each time step \( t = 1, \ldots, T \) over a trace of a policy in the POMDP.

Of course, since learning is impossible when all entries of our examples are hidden by a masking process, we must restrict our attention to settings where it is possible to learn something about \( D \). We will consider formulas that can be evaluated in the straightforward way from the partial states with high probability—it is easy to see that only such clauses are known to be true under \( D \), and thus can possibly be learned from random examples.\(^2\)

**Definition 5 (Witnessed and testable CNFs)** We say that a CNF \( \varphi \) is witnessed to evaluate to true on a partial state \( \rho \) if every clause of \( \varphi \) contains a literal \( \ell \) such that \( \ell(\rho) = 1 \). If \( \varphi \) is

\(^2\)NB: these simple rules will only serve as the “base case” for reasoning, since our algorithms will further carry out reasoning based on them. Thus, in some sense, we also will “learn” knowledge that can be derived (easily enough) from these empirically observed rules.
witnessed true with probability at least \( p \) on partial states from \( M(D) \), we say that \( \varphi \) is \( p \)-testable with respect to \( M(D) \).

As we will only be able to learn the rules governing the POMDP’s dynamics when they are witnessed true with high probability, it is important to note when rules are witnessed. In particular, consider the clause encoding a STRIPS action rule. We assume that the literals capturing the action are always observed. Thus, the rule is witnessed if either the effect of the action is observed or if some unsatisfied precondition is observed when the action fails. Likewise, a frame axiom expressing that the setting of a fluent \( \ell \) persists at time \( t \) is witnessed when either an action is taken that changes the fluent \( \ell \), \( \ell(t-1) \) is observed to be false, or when \( \ell(t) \) is observed to be true.

2.2.1 A measure of tree complexity

The notion of “complexity” of a tree (either treelike resolution proof or decision tree) that we use is:

**Definition 6 (Strahler number)** The Strahler number (aka rank or pebble number) of the nodes of a rooted binary tree are inductively defined as follows: all leaves have Strahler number one, nodes with two children of equal Strahler number \( s \) have Strahler number \( s + 1 \), and nodes otherwise have Strahler number equal to the maximum of the Strahler numbers of their children. Finally, the Strahler number of the tree is the Strahler number of its root.

A significant property is that derivations of Strahler number 2 correspond precisely to derivations using the well-known unit propagation rule, which in turn naturally simulates chaining with, e.g., Horn rules. More generally, we note that the Strahler number of a treelike resolution proof is always one less than the clause space of the proof, as defined by Esteban and Torán [6]: that is, the number of clauses that need to be remembered simultaneously to carry out the corresponding derivation. Indeed, if we consider derivations using a constant number of “registers” in which the registers’ contents are deleted or overwritten upon their use in a derivation, then we obtain an alternative characterization of treelike proofs with bounded Strahler number.

These characterizations of bounded Strahler number proofs help clarify when such simple derivations exist. For example, plans (given by a set of action literals) in STRIPS domains can be proved to achieve their goals by chaining the action and initial state literals through the relevant clauses, and hence have derivations of Strahler number 2.

We also briefly remark that Ansótegui et al. [2] report experiments indicating that “industrial” instances of SAT that are easy in practice actually have (cf. the connection between modern SAT-solvers and resolution refutations [3]) resolution (sub-)refutations with Strahler numbers that are significantly lower than those of random formulas of the same size and clause density. They propose that the property of having bounded Strahler number derivations might be what distinguishes instances that are easy in practice from those that are not.

2.2.2 Algorithms for integrated learning and reasoning in PAC-Semantics

We now recall some efficient (but incomplete) algorithms for verifying the \((1 - \epsilon)\)-validity of a query DNF using a sample of partial assignments from our masked background distribution. Specifically, we recall an algorithm of Juba [12] for verifying the \((1 - \epsilon)\)-validity of a formula when a Strahler-s treelike resolution proof of the formula from a \((1 - \epsilon)\)-testable CNF (and the KB, given as a CNF) exists.
Theorem 7 (cf. Theorem 13 of [12]) Let a KB CNF $\Phi$ and DNF query $\varphi$ be given, and suppose that partial assignments are drawn from a masking process for an underlying distribution $D'$ and are given together with weights $w$ such that for the underlying example $x$, $w = D(x)/D'(x)$, with $w \leq W$; suppose further that for $\gamma > 0$ either

1. There exists some CNF $\psi$ such that $\psi$ is $(1 - \epsilon + \gamma)$-testable under $M(D)$ and there is a Strahler-tree refutation of $\Phi \land \neg \varphi \land \psi$ or else
2. $[\Phi \Rightarrow \varphi]$ is at most $(1 - \epsilon - \gamma)$-valid w.r.t. $D$

Then, there an algorithm running in time $O\left(\frac{W^2}{\gamma^2} (|\Phi| + |\varphi|) n^2(s-1) \log \frac{1}{\delta}\right)$ that distinguishes these cases with probability $1 - \delta$ when given $\varphi$, $\Phi$, $\epsilon$, $\gamma$, and a sample of $O\left(\frac{W^2}{\gamma^2} \log \frac{1}{\delta}\right)$ partial assignments.

The cited result of Juba [12] is for a sample of assignments from $D$ directly without weights. Given that the algorithm simply computes an empirical estimate of an expectation over examples from $M(D)$, by reweighting each examples from $M(D')$ by $w$, we obtain an unbiased estimate over $M(D)$ (at a cost of needing to scale $\gamma$ by $W$ to obtain our guarantee, yielding the quoted change to the sample complexity and running time).

One natural strategy to produce plans by reasoning under PAC-Semantics would be to try to prove that a sequence of actions $\alpha(1), \ldots, \alpha(T)$ results in the satisfaction of the goal predicate $G$ in $s(T)$; if $G$ is a DNF, then the formula $[\alpha(1) \land \cdots \land \alpha(T)] \Rightarrow G(s(T))$ is also a DNF, so the algorithm promised by Theorem 7 can be used. We will develop such an approach.

2.3 Decision Tree Policies

The approach as described in the previous section – i.e., guaranteeing that the goal is satisfied given only the sequence of actions – is oblivious to the state of the environment, and hence only captures "conformant" planning. Such an approach side-steps issues of partial information when it is possible, but of course such plans with high probability of success are unlikely to exist in most situations. On the other hand, introducing the entire sequence of observations (as a conjunction) is problematic as the space of possible observations is large, and under many natural environments it is unlikely that we would ever encounter a consistent sequence of observations even twice. It would then be infeasible to collect a large enough sample to learn the dynamics sufficiently well for such an approach to be effective. A natural alternative that incorporates some knowledge about the state of the environment at a "coarser" level is to use a policy that is computed by a (small) decision tree.

Definition 8 (Decision tree policy) A decision tree policy is a rooted tree of degree $\leq 2$ in which nodes of degree-2 are labeled by an attribute and an observed value for the attribute from the set of observable values $\{0, 1, *\}$, and one edge to a child node is labeled true, and the other false, and other nodes may be labeled with actions with the interpretation that given an observation $a \in \{0, 1, *\}^n$, the agent takes the action (possibly, "no action") labeling the next such node on the branch starting from the previous action (or the root, if no such action exists) on which the true edge at each intermediate node is taken precisely when the attributes take the values on given node’s label.

If at most $T$ action nodes appear on any path, then we say that the policy is a $T$-horizon decision tree.

A small decision tree policy is "coarser" in the sense that the actions of the policy only depend on the observed state (or lack of observation) of the attributes examined on a branch of the tree, which
in general must contain far less than all of the attributes. We note that when the decision tree has polynomial size (in $n$), it has a polynomial number of leaves with polynomial length branches—and hence, if on each branch labeled by the literals $\ell_1, \ldots, \ell_k$, the formula $[\ell_1 \land \cdots \land \ell_k] \Rightarrow G$ is $(1 - \epsilon)$-valid, then by a union bound, the policy will achieve $G$ with probability at least $1 - \epsilon'$ for some $\epsilon'$ polynomially related to $\epsilon$.

### 2.3.1 Strahler-s decision tree policies

As hinted at in the previous section, we will also consider the Strahler number as a measure of decision tree complexity. This will guarantee the existence of polynomial-time algorithms when the policies are “simple,” and will guarantee the existence of quasipolynomial-time algorithms for general decision trees. Much as with resolution proofs, the Strahler number of decision tree policies has a natural interpretation as follows. First, observe that “decision trees” of Strahler number 1 (i.e., a single branch) are precisely conformant plans. More generally, a policy of Strahler number $s$ can be viewed as a decision list over Strahler number $s-1$ policies. That is, we have a hierarchically structured family of policies in which a $s$th level policy selects from among the members of level $s-1$ by testing a series of literals. (The policy may also take some actions amidst these tests, before selecting a lower-level policy to invoke.) We remark that, by testing the effects of each action after taking it, this captures a class of partial-information $s$-fault tolerant plans (cf. Jensen et al. [10]), where each fault is signaled by the first literal that did not end up satisfied. We note that Strahler-s policies can capture all such plans. We briefly remark that Strahler-s decision trees and the more popular OBDDs are of incomparable strength, depending on the variable ordering.

### 2.3.2 Example decision tree policies

Returning to our running example based on the Gripper domain, we now describe a simple Strahler-2 decision tree policy for the goal $x_{r,b}$, i.e., carrying ball $b$ to room $r$, that only fails on any given branch with probability at most $.01$. The policy first executes $a_r$, and branches on $x_{r,b}$; if $x_{r,b} = 1$, the agent terminates the execution (and the goal is satisfied w.p. 1 on this branch). Otherwise, the agent executes $a_{r'}$ for the other room $r'$, and executes $a_b$. The agent then executes $a_r$, and branches on $x_{r,b}$. Again, if it is 1, the agent terminates and the goal is satisfied w.p. 1. Otherwise, the agent again executes $a_{r'}$, $a_b$, and $a_r$, and finally terminates. Notice, this final branch is only taken with probability $.01 + .01 \cdot .99 = .0199$, and conditioned on it having been taken, it only fails with probability $.0199$ again, so the overall probability of the branch ending in failure is $(.0199)^2 < .01$ (it is actually less than 0.04%), and it turns out that this is the total failure probability of this policy. The policy can be seen to have Strahler number 2 since at each test, (at least) one branch terminates with no further tests.

We note that although a union bound over the failure probabilities of the two STRIPS-style rules only gives .98-validity for the conclusion that the ball should be in room $r$ after the action sequence $a_b, a_r$, and the actual validity is .9801 since these failures are independent. (This is still too high for the policy to safely terminate after executing it just once.) The more complex rule

$$[a_{r'}^{(t)} \land a_b^{(t+1)} \land a_r^{(t+2)} \land \neg x_{r,b}^{(t+3)} \land a_{r'}^{(t+3)} \land a_b^{(t+4)} \land a_r^{(t+5)}] \Rightarrow x_{r,b}^{(t+6)}$$

on the other hand, actually not only has validity $1 - (.0199)^2$ (it is satisfied if either $x_{r,b}^{(t+3)}$ is 1 or $x_{r,b}^{(t+6)}$ is 1), but is also witnessed true with that probability in our environment (conditioned on that
action sequence). Since this same conjunction of literals is asserted on the aforementioned branch of the policy, there is a Strahler-2 (chaining) derivation of the goal, $x_{r,b}$, using this witnessed rule.

**Iteration and Strahler numbers.** It should be evident that even if the failure probabilities were much higher than 1% – say it was $p^k$ – by repeating the action sequence $a_{r'}, a_b, a_r$ $k$ times, we can drive the probability of failure overall down to $p^k$; so, we can reach a desired $\epsilon$ failure probability in $3 \log_{p\epsilon} k$ steps (assuming the time horizon $T$ is sufficiently large), and that such a policy still has Strahler number 2: it simply has a longer list of tests.

One change to the POMDP that would require a policy with a larger Strahler number would be if the number of rooms increased from 2 to $R$ (since then it does not suffice from testing room $r$ to determine that the ball is in the other room; more tests are needed). But, in this case there is a Strahler-3 decision tree policy: consider a decision list that iterates over the $R$ rooms by invoking $a_{r'}$ for each room $r'$, then branching on $x_{b,r'}$, moving on if $x_{b,r'}$ is not 1, but otherwise executing the Strahler-2 policy above for the room $r'$ where $x_{b,r'}$ was 1. By our decision list characterization of the Strahler number, this is a Strahler-3 decision tree policy. By an essentially similar argument, it achieves the goal with similar reliability.

**The role of reasoning.** Back in the two-room environment, a more complicated goal is $x_{r,b} \land x_{r',b'}$ (where $r \neq r'$ and $b \neq b'$), that is, placing ball $b$ in room $r$ and ball $r'$ in room $b'$. There is a natural Strahler-3 policy for this goal: the agent executes the Strahler-2 policy for $x_{r,b}$ until it would terminate, it then executes $a_0$ (dropping $b$ in room $r$) and invokes the Strahler-2 policy for $x_{r',b'}$. (Since at every branch, at least one branch leads to a Strahler-2 policy, the overall policy is again guaranteed to have Strahler number 3.) It is easily verified that for this policy, we can bound the probability that each branch is taken and leads to failure by $2 \cdot (.0199)^2 - (.0199)^4$ which is less than 0.08%, and the overall probability of the policy failing is likewise less than 0.08%.

The verification of this goal is *not* trivial, however, since $x_{r,b} \land x_{r',b'}$ are never simultaneously unmasked. In the absence of any further knowledge about the environment, this would pose a problem. But, by providing the agent with explicit frame axioms as we describe in Section 4, we will see how the agent can still identify a successful policy. As we noted above, for every branch of the initial policy for $r$ and $b$, we have a proof that $x_{r,b}$ is satisfied. Now, after $a_0$ is executed, $x_b$ is set to 0, and then none of the actions taken in the policy for $r'$ and $b'$ would ever falsify $x_{r,b}$. Therefore, the frame axioms for the literal $x_{r,b}$ can be invoked – and these are 1-valid here – and so by chaining $x_{r,b}$ across the subsequent steps, we obtain that $x_{r,b}$ is satisfied at the final step of the plan. Together with the derivation of $x_{r',b'}$ on the branch in question (variously described above for the simple one-ball policy) this allows us to refute $\neg x_{r,b} \lor \neg x_{r',b'}$ with a chaining proof, and hence we have a Strahler-2 derivation of $x_{r,b} \land x_{r',b'}$.

## 2.4 Typical Actions

Our objective is to enable the agent to utilize knowledge that it may learn from its experiences. This means that the agent reasons about knowledge drawn from its particular experiences, which consist of a fixed collection of traces. We suppose that the agent’s histories are generated by some arbitrary policy $\Pi^*$, and are given together with the probabilities that $\Pi^*$ chose its actions on each step. These probabilities, although a nonstandard addition, are usually easy to generate (or at least estimate) while an algorithm computes a policy, and provide a convenient way to re-use the traces to estimate the quality of alternative policies (an observation due to Precup et al. [27]).
a fixed sample, we can only hope to evaluate actions that \( \Pi^* \) takes with probability polynomially related to the size of the sample, which are then explored reasonably well. Conceptually, the policy \( \Pi^* \) will serve as a reference point that defines “typical” actions in the POMDP, and we will only expect the agent to understand the dynamics of the POMDP so far as it concerns traces that have some non-negligible weight under \( \Pi^* \). Such settings reasonably capture everyday cases where the agent has been taught relevant sub-policies, or where the relevant plan is merely very simple.

More formally, we will only hope for the agent to choose a policy consisting of sequences of actions \( \alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(t)} \) such that \( \Pi^* \) takes the sequence of actions from the initial state of the POMDP with probability at least \( \mu \) for some \( \mu > 0 \). If \( \Pi^* \) is a random walk policy and the space of possible actions is small, then every short sequence of actions will be “typical” and we will expect the agent to be able to find any suitable short plan in such a case. We then reach the following definition of a typical sequence of actions:

**Definition 9 (\( \mu \)-typical)** If \( \Pi^* \) produces the sequence of actions \( (\alpha^{(1)}, \ldots, \alpha^{(t)}) \) with probability at least \( \mu \) in a POMDP, then we say that the sequence of actions is \( \mu \)-typical for \( \Pi^* \) in the POMDP.

We remark that the distribution over the initial states of the POMDP could be taken to be a “typical state” initial distribution such as the long-run state distribution under \( \Pi^* \).

### 3 Learning and Planning

Algorithms for simultaneously learning and reasoning in PAC-Semantics (as captured in Theorem 7) turn out to provide a bridge between learning dynamics from examples and logic-based approaches to planning. Intuitively, Theorem 7 describes an algorithm that can verify when we have found a good branch of a policy by finding a proof (using knowledge learned from example traces) that the branch achieves the goal sufficiently well. Our algorithm is then based on the work of Ehrenfeucht and Haussler [5] on learning decision trees with low Strahler number: once we can verify that individual branches are suitable, their analysis shows that the entire policy can be found efficiently.

**Theorem 10 (Finding decision tree policies)** There is an algorithm that, when given a time horizon \( T \), Strahler number bound \( s \), \( \mu, \gamma, \delta, \epsilon > 0 \), KB CNF \( \Phi \), goal DNF \( G \) and action set \( A \) for an \( n \) attribute POMDP, runs in time \( O(m|\Phi|(nT)^{4(s-1)}) \) using at most \( m = O(\frac{1}{\mu^2 \gamma^2} (nT + \log |A| + \log \frac{1}{\delta})) \) weighted traces of an arbitrary policy \( \Pi^* \). Suppose there is a Strahler-s \( T \)-horizon decision tree policy \( \Pi \) taking \( \mu \)-typical actions with respect to \( \Pi^* \) such that for each branch, there is a Strahler-s treelike resolution proof of “this branch is taken \( \Rightarrow G \)” from the KB and some CNF \( \psi \) that is \( (1 - \epsilon + \gamma) \)-testable over traces of the POMDP with \( \Pi \). Then with probability \( 1 - \delta \), the algorithm finds a Strahler-s \( T \)-horizon decision tree policy that achieves \( G \) in the POMDP with probability \( 1 - O((nT)^{s-1}(\epsilon + \gamma)) \).

We require the following combinatorial lemma:

**Lemma 11**

1. The number of branches \( k \) in a \( T \)-horizon decision tree policy over \( \{x_1, \ldots, x_n\} \) of Strahler number \( s \) with \( n \geq s \geq 1 \) is \( 2^{s-1} \leq k \leq (3enT/(s-1))^{(s-1)} \) where \( e \) is the base of the natural logarithm.
2. At most \( 3^{3nT} \frac{T}{\mu} \) distinct sets of literals and actions label branches of any \( T \)-horizon decision tree policy over \( \{x_1, \ldots, x_n\} \) with \( \mu \)-typical actions.
Proof: Note that our $T$-horizon decision tree policy can be written as a decision tree over $3nT$ variables, $x_{i,b}^{(t)}$ for $i = 1, \ldots, n$, $b \in \{0, 1, *\}$, and $t = 1, \ldots, T$; therefore, the first part follows immediately from the first part of Lemma 1 of [5]. For the second part, we simply note that there are at most $\frac{1}{\mu}$ possible maximal sequences of $\mu$-typical actions (i.e., that are not prefixes of one another). These sequences have at most $T$ prefixes each, and for each of the $3nT$ variables, the branch may either omit the variable or check that the variable is true or false. The bound is now immediate. 

Proof: (of Theorem 10) The algorithm, given as Algorithm 1, recursively searches for a sub policy in which each leaf, with probability at least $(1 - \epsilon - \gamma)$, either achieves $G$ or is not reached. Intuitively, we are filtering our samples so that we obtain samples from a policy $\tilde{\Pi}^*$ that simulates $\Pi^*$ until $\Pi^*$ takes an action such that the action sequence would have overall probability less than $\mu/4$, and then takes a distinct “abort” action instead. Note that $t$-step traces in which $\tilde{\Pi}^*$ does not abort are precisely those with $\prod_{i=1}^t w_i \leq 4/\mu$. At each node, our learning and reasoning algorithm is invoked on the example traces to test if either $G$ is satisfied or that the branch is not taken with a $1 - \epsilon$ fraction of the $m$ examples (with examples off the branch removed from the sample since they are guaranteed to satisfy our target condition). If so, we have found a suitable leaf.

In $m \geq \frac{8}{\mu^2} (3nT \ln 3 + \ln \frac{4TA\mu}{\mu} + \ln \frac{3}{\delta})$ example traces, the fraction of times a sequence of actions appears approximates its probability under $\tilde{\Pi}^*$ to within an additive $\mu/4$ with probability $1 - \delta/(3|A|M)$ where $M$ is the number of distinct events corresponding to taking branches appearing in $T$-horizon decision tree policies labeled by $\mu/4$-typical actions, as given in part 2 of Lemma 11. In particular, since there are at most $|A|M$ action sequences that are either $\mu/4$-typical or not $\mu/4$ typical and minimally so, this guarantees that with probability $1 - \delta/3$, all of the action sequences which appear in a $\mu/2$-fraction of the traces are at least $\mu/4$-typical under $\tilde{\Pi}^*$, and that the $(3/4)\mu$-typical actions under $\tilde{\Pi}^*$ all appear in at least a $\mu/2$-fraction of the traces. In particular, since all of the $\mu$-typical action sequences of $\Pi^*$ are at least $(3/4)\mu$-typical under $\tilde{\Pi}^*$, all of these appear.

Soundness. We first note that as a consequence of our filtering the examples to be consistent with the branch $\Pi$, the weights we compute for our queries are

$$\prod_{i=1}^t w_i = \prod_{i=1}^t \frac{I[a_{\alpha(i)}^{(i)} | \Pi \text{ followed for } 1, \ldots, i - 1]}{Pr_{\tilde{\Pi}^*}[a_{\alpha(i)}^{(i)} | \Pi \text{ followed for } 1, \ldots, i - 1]} = \frac{Pr_{\Pi}[\Pi \text{ followed for } 1, \ldots, t]}{Pr_{\tilde{\Pi}^*}[\Pi \text{ followed for } 1, \ldots, t]}$$

where these weights are bounded by $4/\mu$. Therefore $m$ examples are sufficient to bound the probability of queries being satisfied over traces of $\Pi$ by an additive $\gamma$ with probability $1 - \frac{\delta}{35|A|}$. By a union bound over the $M$ possible queries, every query is answered correctly w.p. $1 - \delta/3$.

As Theorem 7 guarantees that for each branch the formula asserting that either the leaf is not reached or the policy succeeds is $(1 - \epsilon - \gamma)$-valid, the probability that the policy fails on each branch is at most $\epsilon + \gamma$. So, when a decision tree policy is returned, a union bound over the $O((nT)^{s-1})$ branches (by part 1 of Lemma 11) yields the claimed performance. So the algorithm only fails by returning a bad policy when some query is answered incorrectly, which occurs w.p. at most $\delta/3$.

Completeness. We already saw that the algorithm considers all sequences of $\mu$-typical actions under $\Pi^*$ with probability $1 - \delta/3$. Now, if the algorithm does not find a policy in which a branch
with lower Strahler number takes no action, then if there is a policy with the given Strahler number bound, it must take some action at the next step, and otherwise, (i.e., if a sub-policy was found) then there is no harm in asserting the opposite setting of \( x_{t,b}^{(l)} \), since adding this literal to the formula for the branches of the target policy only increases the probability of the branch not being taken (and thus improves the success probability for that branch). In particular, the original Strahler-\( s \) tree is still a candidate. Theorem 7 also guarantees that for the quoted number of examples, all of the branches of the target policy will be accepted w.p. at least \( 1 - \frac{\delta}{3} \) as needed. Thus overall the algorithm succeeds at returning a good policy when it exists w.p. \( 1 - \delta \).

**Time complexity.** Finally, the work per recursive call of our algorithm involves partitioning the examples and running \texttt{PAC-refute} on the sample, which is linear in the sample size \(|R|\). So, given \( W \cdot m \) work per node with \(|R| = m\), the running time bound for the algorithm of \( O(W \cdot m (nT)^{2(s-1)}) \) is easily established by considering the following recurrence with boundary conditions \( f(0,m,0) = W \cdot m \) over \( n' = (3n + 1)T \), counting the possible remaining literals plus one action for each time step: the bound is a solution to the recurrence

\[
f(n', m, s) \leq f(n' - 1, m, s) + 2n' f(n' - 1, m, s - 1) + W \cdot m.
\]

The running time of the algorithm also satisfies this recurrence since for such a \( f \) linear in \( m \), if we take \( m_\alpha = |R_\alpha| \) in Algorithm 1 \( \sum_{\alpha \in A} f(n' - 1, m_\alpha, s) \leq f(n' - 1, m, s) \), and indeed our algorithm only iterates over actions (in which it partitions the example set among the recursive calls) if it does not make a Strahler-\( s \) recursive call while searching for a branch over the literals. Thus the running time satisfies the given bound satisfying the recurrence. We now merely observe that the work by the learning and reasoning algorithm per node is \( W \cdot m = O(|\Phi|(nT)^{2(s-1)}m) \), giving the claimed time bound.

It follows from Theorem 10 that we can find general decision tree policies taking \( \mu \)-typical actions in quasipolynomial time since if a policy has \( B \) branches, Lemma 11 guarantees it has Strahler number at most \( s = \log B + 1 \).

### 4 Extension to Non-monotonic Reasoning

One weakness of the result of the previous section in learning STRIPS domains is that the frame axioms can only be learned under partial information when they are not needed: in order for them to be witnessed, the corresponding attributes of the POMDP must be observed. At the same time, we cannot supply generic frame axioms to the algorithm since the standard formulation of the frame axioms depends on the preconditions and effects of the actions, which are what we hope the agent to learn in each domain. Generic non-monotonic formulations of the frame axioms do exist, however: the clause \([\ell(t) \land \neg\ell(t+1)] \Rightarrow \ell(t+1)\), where \( \neg\ell \) roughly means “\( \ell \) cannot be proved,” nicely captures the frame axiom whenever we have learned the effects and preconditions of the actions. In this section, we will describe a semantics for \( \neg\ell \) that is suitable for these purposes with respect to Strahler-\( s \) resolution and can be evaluated in polynomial time; we can then extend the algorithm of Theorem 7 to answer queries containing such literals, and thus we can add these generic frame axioms to the KB when invoking Algorithm 1.

We will use a variant of the Well-Founded Semantics \([35]\) (WFS) for logic programs. More specifically, we will modify the iterated quotient definition of Przymusinski \([29]\) to obtain, for
each $s$, an analogue of WFS for Strahler-$s$ resolution. In particular, since Strahler-2 resolution corresponds precisely to chaining, our generalization for Strahler-2 resolution coincides with WFS over all (appropriate) directed versions of our original, undirected set of clauses.

**Definition 12 (Quotient operator)** For any CNF $\varphi$ over literals possibly using $\sim$ and any pair of sets of literals $(L_+, L_-)$, the quotient of $\varphi$ modulo $(L_+, L_-)$, denoted $\varphi/(L_+, L_-)$, is the CNF obtained by substituting 1 for occurrences of $\sim \ell$ s.t. $\ell \in L_-$, substituting 0 for occurrences of $\sim \ell$ s.t. $\ell \in L_+$, substituting other occurrences of $\sim \ell$ with occurrences of a corresponding new variable $\ell$, and simplifying the resulting formula by deleting satisfied clauses and falsified literals.

The intuition behind the quotient is that the sets $L_+$ and $L_-$ consists of the literals that we know, respectively, can and can’t be proved from $\varphi$. We will see how to obtain a pair $(L_+, L_-)$ conservatively respecting this intuition for any formula.

**Definition 13 (Least partial state)** The Strahler-$s$ least partial state $LPS_s$ of a CNF $\varphi$ is a pair of sets of literals $(L_+, L_-)$ (not over the $\ell$ variables) s.t. $\ell \in L_+$ iff $\ell$ has a Strahler-$s$ proof from $\varphi$ and $\ell \in L_-$ iff for the CNF $\varphi'$ in which all occurrences of the $\ell$ variables have been deleted, there is no Strahler-$s$ proof of $\ell$ from $\varphi'$.

This definition is analogous to a construction of the (unique) “least partial models” for normal logic programs from [28], in which the name has a more meaningful interpretation in terms of the partial models of logic programs. The name “least” partial state comes from considerations like the following. The pair of sets of literals $(L_+, L_-)$ is a partial state in the sense that all literals $\ell \in L_+$ under a $\sim$ are set to 1, $\ell \in L_-$ under $\sim$ are set to 0, and otherwise $\sim \ell$ is set to 1/2. We suppose we are trying to (pointwise) minimize the truth values of the literals of $\varphi$ while respecting the Strahler-$s$ conclusions that can be derived from it, possibly given some further settings of the $\sim$ literals. That is, the existence of a Strahler-$s$ derivation of $\ell$ requires that the $\ell$ be true in the partial state, and if for some substitution of the different occurrences of $\sim \ell$ for truth values there exists a Strahler-$s$ derivation of $\ell$, then that prevents setting $\ell$ to false. Then we assign each literal the least possible value subject to these constraints.

We will finally obtain our “well-founded” Strahler-$s$ semantics for negation-as-failure by taking the least defined fixed-point of the quotient and least partial state operators: the quotient incorporates the settings for $\sim$ literals that have been derived, and the least partial state operator obtains the consequences for Strahler-$s$ provability. That is, it can be shown (moreover) that literals are only marked as “provable” (placed in $L_+$) or “unprovable” (placed in $L_-$) precisely when they are provable or unprovable, respectively, in every partial state that is a fixed-point under the composition of the quotient and least partial state operators. Much as in the original conception of the Well-Founded Semantics in [35], this definition also conforms to the desirable intuition that literals are only marked as “provable” or “unprovable” if this can be ultimately derived from $\varphi$ itself (with no additional knowledge or assumptions about what else is provable), ruling out consistent but circular negation-as-failure settings.

**Proposition 14 (Semantics of Strahler-s NAF)** For any CNF $\varphi$ over literals possibly using $\sim$, the sequence of pairs of sets of literals $(L_+^{(0)}, L_-^{(0)}) = (\emptyset, \emptyset), (L_+^{(i+1)}, L_-^{(i+1)}) = LPS_s(\varphi/(L_+^{(i)}, L_-^{(i)}))$ converges to a pair of sets of literals $(L_+^*, L_-^*)$ s.t. for every literal $\ell$ (without $\sim$),

1. if there is a Strahler-$s$ proof of $\ell$ from $\varphi/(L_+^*, L_-^*)$, then $\ell \in L_+^*$

14
2. if $\ell \in L^\times_-$ then there is no Strahler-s proof of $\ell$ from $\varphi/(L^+_s, L^+_s)$

Furthermore, there is an algorithm that computes $(L^+_s, L^+_s)$ given $\varphi$ in time $O(|\varphi| n^{2s})$

**Proof:** Convergence will follow from the observation that $L^+_i \subseteq L^+_i$ and $L^-_i \subseteq L^-_i$. For $i = 0$, the statement is trivial.

To see $L^+_i \subseteq L^+_i$ for $i \geq 1$, consider the proof of $\ell$ from $\varphi/(L^+_i, L^+_i)$ witnessing $\ell \in L^+_i$. By induction on the structure of the proof of $\ell$, we construct a new proof as follows: if this clause was obtained by a cut rule on one of the $\tilde{\ell}$ variables and $\ell' \in L^+_i \cup L^-_i$, in which case, one of the clauses survives in $\varphi/(L^+_i, L^+_i)$ with the occurrence of $\tilde{\ell}$ eliminated, and hence the clause obtained by weaking the clause of $\varphi/(L^+_i, L^+_i)$ from which $\tilde{\ell}$ was eliminated; otherwise, by our induction hypothesis, we can derive subclauses of the two clauses involved in this final step from $\varphi/(L^+_i, L^+_i)$. (The base case is trivial.)

As for $L^-_i \subseteq L^-_i$ for $i \geq 1$, we assume inductively that $L^+_i \subseteq L^+_i$ and $L^-_{i-1} \subseteq L^-_{i-1}$. We note that when $\ell \notin L^-_{i+1}$, there is a proof of $\ell$ from $(\varphi/(L^+_i, L^+_i))'$, which is constructed from $\varphi$ ultimately by deleting clauses that either contain $\sim \ell'$ for $\ell' \in L^+_i$ or $\neg(\sim \ell')$ for $\ell' \in L^-_i$, and eliminating the rest of the $\tilde{\ell}$ literals from the remaining clauses. Thus, $(\varphi/(L^+_i, L^+_i))'$ is a subformula of $(\varphi/(L^+_i, L^+_i))'$ since $L^+_i \subseteq L^+_i$ and $L^-_{i-1} \subseteq L^-_{i-1}$ by our inductive hypothesis. Therefore, the same proof establishes $\ell \notin L^+_i$, completing the inductive step.

Since we add at least one literal on each iteration until convergence, if we compute $(L^+_s, L^+_s)$ using the iterative definition, the algorithm terminates in at most $n$ iterations. Since each iteration can be computed by at most $2n$ applications of the Strahler-s proof search algorithm on a formula of size at most $|\varphi|$ (which runs in time $O(|\varphi| n^{2(s-1)})$, the running time bound follows.

For our fixed-point, we have a pair of sets of literals $(L^+_s, L^+_s)$ such that $(L^+_s, L^+_s) = LPS_s(\varphi/(L^+_s, L^+_s))$, and hence the two claimed properties are immediate from the definition of the Strahler-s least partial state. ■

Naturally, the interpretation of $\varphi$ is given by the classical propositional formula $\varphi/(L^+_s, L^+_s)$. In particular, we can now extend our PAC-Semantics to incorporate negation-as-failure like so:

**Definition 15** We say that a formula $\varphi$ (possibly using $\sim$) is $(1 - \epsilon)$-valid w.r.t. a distribution $D$ and masking process $M$ if, for $m$ drawn from $M$, $x$ drawn from $D$, and the formula $\psi$ consisting of the conjunction of literals satisfied on $m(x)$, putting $(L^+_s, L^+_s)$ equal to the fixed point obtained from $\varphi \land \psi$, the probability (over $m$ and $x$) that $x$ satisfies $\varphi/(L^+_s, L^+_s)$ is at least $1 - \epsilon$.

We can answer queries in this extended PAC-Semantics by an easy modification of the algorithm underlying Theorem 7 wherein, for each example, we first compute $(L^+_s, L^+_s)$ for the KB $\Phi$, and then for a query $\varphi$, check for a refutation of $\varphi \land (\Phi/(L^+_s, L^+_s))$. Such an algorithm simply replaces the existing subroutine in Algorithm 1. For $s \geq 2$, we can then verify that the generic frame axioms $[\ell(t) \land \sim(\ell(t+1))] \Rightarrow \ell(t+1)$ are $(1 - \epsilon - \nu)$-valid under the Strahler-s NAF semantics in a noisy STRIPS instance with noise rate $\nu$ and $(1 - \epsilon)$-testable actions. We are therefore free to include them in the KB at a small cost in validity. Crucially, it may also be verified that they allow the values of hidden state attributes to be propagated forward, at least when they are the only clauses in the KB containing the state attributes.
5 Relationship to Other Work

Our ability to learn descriptions of the dynamics that are simple but imperfect already distinguishes our work from some others that attempt to learn explicit descriptions of an environment’s rules from examples, e.g., [25, 1]. Others attempt to construct a model that accounts for the imperfection [9, 30, 26, 36, 18, 23], but cannot provide a formal analysis. This lack of theoretical grounding seems inherent due to the difficulties posed by agnostic learning; moreover, the negative results of Michael [22] speak to the advantages of attempting to answer queries without producing an explicit action model. Recent work on PSRs [4] only establishes consistency, not fast convergence under any clear conditions.

Our setting is related to a setting considered previously by Khardon [16], apprenticeship learning [31] and reductions to classification [19]: These works aim to learn a policy using example traces, even for sophisticated policy representations. The distinction is that we don’t assume that $\Pi^*$ is our desired policy, just that there exists a good policy that $\Pi^*$ agrees with at least a $\mu$-fraction of the time. (If the number of actions is small, $\Pi^*$ could even be a random walk.)

Reinforcement learning [14, 7, 20] can learn from a poor initial policy such as a random walk, but uses repeated experience with the model, in particular, with an explicitly provided reward function – i.e., goal – at hand, which does not transfer. Learning from exercises [24, 32] is similar, except that the learner is provided sequences of examples that gradually increase in difficulty. The difference with our work is that we don’t assume that the current goal can be reduced to subproblems of “lower difficulty” (but nor do we guarantee success when this is the case). Later, Joshi, Kersting, and Khardon [11] took a similar approach in which example policies (generated either by simple strategies or by other planning algorithms) were used to obtain a fast model-checking policy search algorithm for relational planning in a fully observed setting. Their setting was thus quite different from ours—we are seeking to cope with missing information, as opposed to gaining a speed-up in a full-information setting.

6 Future Directions

One natural direction concerns improving Theorem 10: One of the main insights underlying Theorem 7 is that the main barrier to agnostic learning is finding an explicit representation, and yet Algorithm 1 proceeds by constructing an explicit policy. It is conceivable that the $O((nT)^{s-1})$-factor blow-up in the policy failure probability $\epsilon$ could be eliminated if we could similarly identify a good action on-line, without going so far as to construct an entire policy.

Our set-up of learning policies that take typical actions with respect to a reference policy naturally suggests a bootstrapping approach to learning complex policies; another immediate direction is then to investigate what can be proved learnable by bootstrapping (in contrast to the empirical work of [7]). More generally, our work side-steps the entire problem of exploration of POMDPs; it can be viewed as finding a policy, given that the POMDP has been sufficiently well explored. So, one might try to address the exploration problem in the context of such planning algorithms by showing that a class of environments can be efficiently explored sufficiently well to permit good policies to be found.
Acknowledgements

I am grateful to Leslie Kaelbling for numerous detailed suggestions and criticisms of this work that improved it immensely. This work was also heavily influenced by conversations with Leslie Valiant.

References


Algorithm 1 Find-DT()

PAC-refute(ϕ, R) decides if ϕ \land ψ is at most ϵ-valid given some testable ψ over weighted partial examples in R (taken out of m) (cf. Theorem 7).

Input: DNF goal G, CNF KB Φ, Strahler number s, policy branch Π taking t − 1 actions, horizon bound T, sample of traces with stepwise weights R

Output: A Strahler-s decision tree or ⊥ if Π takes more than T actions then
    return ⊥
end if

R’ ← \{(x_{i=1}^{t-1}(o^{(i)}, a^{(i)}) \times o^{(t)}, \Pi_{i=1}^{t-1}w_i) : ((o^{(i)}, a^{(i)}, w_i)_i^{T}, o^{(T+1)}) \in R, \Pi_{i=1}^{t-1}w_i \leq \frac{4}{\mu}\}

if PAC-Refute(¬G^{(t)} \land Φ \land Π taken, R') then
    return Π.
end if

if s = 1 then
    for all α s.t. #{ρ ∈ R : a^{(t)} = α} ≥ (μ/2)m do
        Π_α ← Π with a^{(t)} = α.
        R_α ← {ρ ∈ R : a^{(t)} = α}
        Π’ ← Find-DT(G, Φ, s, Π_α, T, R_α)
        if Π’ ≠ ⊥ then
            return a^{(t)} = α followed by Π’
        end if
    end for
    return ⊥
end if

for all ℓ s.t. ℓ^{(t)} and ¬ℓ^{(t)} not in Π do
    Π_ℓ ← Π with a final test for ℓ^{(t)}
    Π_1 ← Find-DT(G, Φ, s − 1, Π_ℓ, T, R)
    if Π_1 ≠ ⊥ then
        Π_0 ← Find-DT(G, Φ, s, Π_¬ℓ, T, R)
        if Π_0 ≠ ⊥ then
            return Policy following Π_0 if ℓ^{(t)} = b
        else
            return ⊥
        end if
    end if
end if

for all α s.t. #{ρ ∈ R : a^{(t)} = α} ≥ (μ/2)m do
    Π_α ← Π with a^{(t)} = α
    R_α ← {ρ ∈ R : a^{(t)} = α}
    Π’ ← Find-DT(G, Φ, s, Π_α, T, R_α)
    if Π’ ≠ ⊥ then
        return a^{(t)} = α followed by Π’
    end if
end for

return ⊥