Journal of Machine Learning Research 17 (2016) 1-37 Submitted 12/13; Revised 9/15; Published 6/16

Integrated Common Sense Learning and Planning in POMDPs

Brendan Juba* BJUBA@WUSTL.EDU
Washington University
1 Brookings Dr.
St. Louis, MO 63130 USA

Editor: John Langford

Abstract

We formulate a new variant of the problem of planning in an unknown environment, for which we can provide algorithms with reasonable theoretical guarantees in spite of large state spaces and time horizons, partial observability, and complex dynamics. In this variant, an agent is given a collection of example traces produced by a reference policy, which may, for example, capture the agent's past behavior. The agent is (only) asked to find policies that are supported by regularities in the dynamics that are observable on these example traces. We describe an efficient algorithm that uses such "common sense" knowledge reflected in the example traces to construct decision tree policies for goal-oriented factored POMDPs. More precisely, our algorithm (provably) succeeds at finding a policy for a given input goal 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 the goal is achieved on each branch using the aforementioned CNF capturing the "common sense rules." Such a CNF always exists for noisy STRIPS domains, for example. Our results thus essentially establish that the possession of a suitable *exploration policy* for collecting the necessary examples is the fundamental obstacle to learning to act in such environments.

Keywords: Partially Observed Markov Decision Process, Decision Tree Policies, PAC-Semantics, Noisy STRIPS, Non-monontonic Reasoning

1. Introduction

A central problem in artificial intelligence, first considered by McCarthy (1959) concerns how to enable a machine to autonomously operate in an environment. The dominant approach to this problem 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

*. Work originally performed while the author was affiliated with Harvard University and supported by ONR grant number N000141210358. Preparation of this article was supported by an AFOSR Young Investigator Award.

may be intractable even when the overall problem has efficient algorithms: Khardon and Roth (1997) famously presented an efficient algorithm for an NP-hard reasoning task over a learned DNF representation, for example. And, *both* learning models of interactive environments *and* planning in them directly is indeed usually infeasible if, e.g., one is learning a model that can express DFAs (Kearns and Valiant, 1994), or one is attempting to plan in a model that captures propositional STRIPS domains with a bounded time horizon (Bylander, 1994; Erol et al., 1995). Although it is feasible to learn and plan in environments that are generated by extremely 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.

1.1 Formulation of the Integrated Learning and Planning Problem

Naturally, the task of developing algorithms to address the hard instances of intractable problems is inherently hopeless; as both learning and planning feature many hard instances, we can only hope to guarantee good performance by carefully formulating the theoretical problem to avoid these hopeless instances. Indeed, perhaps the main contribution of this work is the formulation of the integrated learning and planning problem that "factors out" the difficult task of *exploring* the environment: We avoid the exploration problem by assuming that an *exploration policy* has been fixed, and only seeking to plan using actions this reference policy explores with non-negligible probability. Naturally, we will also restrict the policies we consider—this is the analogue of the usual restrictions on the representations we consider in standard machine learning problems. For the concrete families of policies we consider, then, the polynomial-time algorithms we present for learning and planning thus establish that the only inherent difficulty in the task is exploring the environment.

Exploration often captures the essence of the kind of "needle in a haystack" search task that is necessarily slow: Even exploring environments described by very simple rules can force algorithms to suffer a complexity that is linear in the number of environment states, such as "combination lock" environments (Kakade, 2003, Section 8.6). And yet, the formulation of, for example, the standard reinforcement learning task combines the exploration task with the tasks of learning and planning, denying the possibility of any analysis establishing reasonable bounds on the time or number of samples required for most standard environment classes. By shifting our attention to settings where exploration is either solved for us or guaranteed to be possible, we will be able to obtain algorithms of polynomial complexity in the number of attributes describing the states, as opposed to the number of actual states (which is of course exponentially larger). This complexity bound is similar to what is achieved by the relaxation of Kakade and Langford (2002) for fully observed environments with a value function (as opposed to goal satisfaction), but using a different assumption.

We will aim to learn the dynamics (and observation model) of a *factored Partially Observed Markov Decision Process* (POMDP) with partial but noiseless observations. Pre-

cisely, the state space we consider will be described by vectors of n Boolean attributes ("fluents" in the planning literature), and the agent's observations will be of the settings of a *subset* of these attributes. In our model, the agent may have access to some arbitrary *background knowledge* about the environment's dynamics. This knowledge is given to the agent as a collection of Boolean formulas over the state attributes that are assumed to be simultaneously true (with probability close to 1) on sequences of states generated by the environment. The agent also has access to a collection of observation histories that have been generated by the (arbitrary) exploration policy. The agent is then provided with a *goal*, represented by another Boolean formula over the attributes of the environment's state. The agent wishes to choose *actions* that guide the environment to a state satisfying this goal formula. We wish to use the background knowledge and example histories generated by the reference exploration policy to inform these choices of actions in the POMDP in order to achieve this new goal. That is, we are only seeking to learn the dynamics of the POMDP from this background knowledge and these example histories to the extent that it enables the agent to reach a goal state.

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 note that unlike the full-information task, the history of past observations may provide additional information about the current state of a POMDP, and hence it is important to be able to construct a stateful policy. Following the spirit of McCarthy's approach, we propose to focus on cases where based on "common sense knowledge" that can be learned from the exploration policy's observations of the POMDP, there exists a *proof* that a given goal is satisfied on the actual, underlying trajectory generated by the policy and the POMDP.

Technically, the proofs serve to provide evaluations of the agent's performance under partial information: our "reward function" is given as an input formula that may refer to the unobserved portions of the state, separate from the domain examples. In order to determine that the reward function is satisfied when some information is missing, it is inherently necessary that we establish that none of the missing values could prevent the satisfaction of the goal—each setting of the missing data is either inconsistent with some background knowledge we have, or the goal condition is satisfied on it anyway. In particular, we do not assume that we have any model for the distribution for the missing values; these values are missing from the very data we use for learning. Addressing all of the possible values of such missing information is the essence of theorem-proving, and the existence of a simple proof gives us a potential means to efficiently consider all of these possible values for the missing data. In particular, we will assume that the goal is given as a DNF formula, and we will use treelike resolution proofs of bounded *Strahler numbers* (aka pebble number or clause space, cf. Section 2.2.1). This is a class of proofs that, even for Strahler number two, captures the kind of reasoning needed to verify the correctness of STRIPS plans, for example. Moreover, the reasoning problem for any constant Strahler number can furthermore be solved in polynomial time with the exponent depending on the Strahler number, cf. Kullmann (1999).

1.2 Techniques and Results

We propose here to use algorithms for reasoning over such common sense knowledge learned from partial information, as described by Juba (2013) 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 (Kautz and Selman, 1992), 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). A restriction to the space of decision tree policies was first considered for MDPs by Chapman and Kaelbling (1991) and essentially for POMDPs in the work of McCallum (1995, 1996); decision list policies (an example of *bounded* Strahler number decision tree policies) were considered by Khardon (1999). The restriction of the space of policies to more general bounded Strahler number trees is also natural: it encompasses bounded-fault tolerant policies (Jensen et al., 2004), for example. Such policy-space restrictions serve two relatively well-understood purposes: as with restrictions on the class of representations in learning theory, a restriction on the space of policies enables us to statistically distinguish good policies from bad policies (Kearns et al., 2002), and moreover enables us to design algorithms to *find* policies efficiently (e.g., as in the reduction of Mansour, 1999).

Our reduction adapts the algorithm of Ehrenfeucht and Haussler (1989) for supervised learning of decision tree classifiers of bounded Strahler number to search for policies. (Just like Ehrenfeucht and Haussler (1989), we can also search through all decision trees of a given size in *quasipolynomial* time, i.e., time $2^{\text{poly}\log n}$.) This algorithm proceeds by recursively building a tree below a candidate branch, using the ability to test when a branch is consistent with the input examples, terminating successfully when such a consistent branch is found. The key insight of Ehrenfeucht and Haussler was that a Strahler-number restriction enables the algorithm to control the amount of branching performed during this recursive search, by allowing us to search over the smaller (lower Strahler-number) subtrees first.

In our setting, deciding whether or not a branch is "consistent" means deciding whether or not the goal is achieved on the (example) traces that are consistent with the tests and actions selected on the branch. Such a test is thus naturally accomplished by a query to the integrated learning and reasoning algorithm of Juba (2013). The decision tree search algorithm also must be extended to search over *actions*, as opposed to just branches on settings of the observed attributes, which are all that the original decision trees use. We control this search indirectly, by partitioning the set of example traces according to the mutually exclusive choice of next action, and only passing the corresponding subset of the examples to the recursive call; the amount of work performed by the algorithm can be shown to be linear in the number of examples, so it is easy to bound the total work performed by these branches. 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 policies.

This in turn crucially exploits the problem formulation: we only promise to find policies using actions that the reference exploration policy is likely to consider.

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 (Fikes and Nilsson, 1971) (a standard test domain of interest) which are approximated well but imperfectly by their noiseless versions. Also, as a consequence of our adoption of *PAC-Semantics* (Valiant, 2000a) 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.

In order to support some more interesting cases of planning under partial information, we also extend the PAC-Semantics slightly to non-monotonic logics using the Well-Founded Semantics (van Gelder et al., 1991) for negation-as-failure. The Well-Founded Semantics for propositional logics is a relatively clean semantics that features polynomial-time reasoning, and allows us to formulate, for example, "generic" frame axioms as explicit background knowledge; that is, frame axioms that do not depend on the unknown dynamics of the environment. These explicit frame axioms in turn allow the agent to reason about attributes of the environment that are unobserved in the traces collected by the exploration policy. The agent can then, for example, plan to achieve goals that refer to portions of the environment that cannot be simultaneously observed.

2. Preliminaries

Before stating the problem we address, we must describe the relevant models of environments, approximate reasoning, policies, and typical experiences.

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 $D_{(a^{(1)},s^{(1)})}$, 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 ith 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 also fixes 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). For a fixed policy, a sample from the joint distribution over the actions and observations generated by the interaction between the POMDP and the policy is called a *trace* or *history*; a sample from the distribution over the actual underlying states, actions, and observations is a *trajectory.* Naturally, a history can be obtained from a trajectory by dropping the actual states.

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—note that Haussler (1989) shows that fitting first-order expressions even in very simple domains is intractable. By contrast, following Valiant (2000a) for example, 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, as the propositional case is surely simpler and suffices for the current work, we will not consider first-order representations further. Thus, in particular, we will fix a (possibly polynomially related to n) horizon bound T, and for each tth step (of a candidate plan) and ith component of the state of the POMDP, we will have a separate propositional variable $s_i^{(t)}$.

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 ρ *is an element of* $\{0,1,*\}^n$. *We say that a partial state* ρ *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 (Michael, 2010):

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. Masking leads to a kind of perceptual aliasing if the distinguishing attributes in two distinct states are masked; indeed, there may even be propositional variables that the masking process *never* reveals, that have an arbitrary effect on the dynamics (as they correspond to distinct states). In such a case, we cannot learn about the relationship of these missing variables to the larger dynamics of the POMDP using the observations alone. We may have *background knowledge* that mentions these variables though, thereby permitting us to reason about their contents. We shall discuss this further shortly.

In contrast to the most typical set-up, our reward function is *not* fixed with the POMDP, and in particular its value may not be determined by the partially observed example traces. 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).

Noise. This model assumes that there is no noise in the observations themselves. Although as given this does not accurately reflect robotics problems (for example), it is a reasonable model for agents that interact with computer systems, for example a web-crawling agent.[1] In such applications, only a portion of the environment's state, such as a single web-page, single directory, etc. is visible at a time, but perception of this state is not at issue; the actions available to the agent then would correspond to the interface elements (e.g., links, buttons, etc.). In any case, it happens that noise in the observations will only affect our approach by possibly leading us to evaluate the goal predicate incorrectly, and therefore leading the policy to terminate prematurely. Otherwise there is not much difference between noise (stochasticity) in the dynamics—which we *do* consider, cf. Section 2.3.2 for an illustration—and noise in the observations.

2.1.1 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 Π^*. In a fixed sample, we can only hope to evaluate actions that Π^* takes with probability polynomially related to the size of the sample, which are then explored reasonably well; conversely, we can also guarantee that all of the actions that Π^* takes with some minimum probability μ are well-explored in a sample of size polynomially related to $1/\mu$. So, the probability that Π^* explores a sequence of actions approximately characterizes the sequences that are well-represented in a sample of traces of a given size. In particular, since distinct sequences of actions refer to disjoint events, there are at most $1/\mu$ distinct sequences of actions that are taken with probability μ. A lower bound on this probability therefore bounds the number of sequences of actions we have to consider, and so the policy Π^* does all of the work in "exploring" the POMDP.

Conceptually, the policy Π^* 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 Π^*. Such settings reasonably capture everyday cases where the agent has been taught or has learned relevant sub-policies, or where the relevant plan is merely very simple. Some such guarantee seems to be necessary in order to obtain algorithms of reasonable complexity: Kakade (2003, Section 2.5) shows that even very simple fully-observed environments can be hard to learn, depending linearly on the number of possible states, which may be exponential in the number of fluents, or exponentially on the time horizon.

1. Indeed, the original motivation for considering this model arose from the desire to develop devices and distributed/networked software that featured "universal" (or at least broad) compatibility (Goldreich et al., 2012; Juba, 2011).

We further suppose that the agent's histories using Π^* are given together with the probabilities that Π^* chose its actions on each step in the history. These probabilities, although a somewhat 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 via importance weighting, an observation due to Precup et al. (2000, 2001) for POMDPs. (The technique is discussed for full-information MDPs by Sutton and Barto, 1998, Chapter 5.)

Formally now, 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 Π^* takes the sequence of actions from the initial state of the POMDP with probability at least μ for some $\mu > 0$. For example, if Π^* 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. But, Π^* may also describe some more sophisticated policy for exploring the relevant portions of the POMDP, or may be simply the time-average of the various policies used by the agent in the past (in which case such an estimator was considered by Shelton, 2001). 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 Π^*. Regardless of the details of this policy, we then reach the following definition of a typical sequence of actions:

Definition 4 (μ-typical) *If Π^* produces the sequence of actions $(\alpha^{(1)}, \ldots, \alpha^{(t)})$ with probability at least μ in a POMDP, then we say that the sequence of actions is* μ-typical *for Π^* in the POMDP.*

We stress that algorithms that achieve a sample complexity m for learning and planning in the POMDP under most conventional models can be used as an exploration policy for producing traces in which the (significant) actions of the plan are $1/\text{poly}(m)$-typical. Otherwise, the algorithm would not have an adequate estimate of the quality of the traces to produce a reliable plan.

Another definition that similarly evaluates policies with respect to some fixed (reset) distribution was proposed previously by Kakade and Langford (2002) for learning in fully observed MDPs. The assumption is of course quite different, and their setting attempts to optimize a value function (with respect to this reset distribution) rather than achieving a goal. Nevertheless, much like in Kakade and Langford's work, we will be able to avoid an exponential dependence on the time horizon or number of attributes in the time and sample complexity of our algorithm as a result of our definition. A related assumption for POMDPs was introduced by Bagnell et al. (2004), which assumes that the fixed distribution is *close* to that induced by the target policy in statistical distance.

2.1.2 Knowledge Representations and STRIPS Domains

As we noted earlier, the special case of a factored POMDP that we consider here is naturally captured by a propositional logic in which there is a propositional variable for each ith fluent of the POMDP on each tth step of the trajectory. In this way, we can use propositional logic to reason about the actual, underlying state that the POMDP could be in on each step, and thus cope with our missing observations. In general, we will have a collection of propositional formulas (i.e., formed by the usual Boolean connectives over propositional

variables) capturing basic knowledge about our encoding of the POMDP, or specific to the actual POMDP under consideration. The entire collection is referred to as the *"background knowledge"* or *"knowledge base"* (abbreviated as "KB"). We will generally assume that the formulas in the knowledge base hold simultaneously with probability close to 1 over the distribution of sequences of states generated by the reference policy and POMDP.

It will be helpful for us to distinguish between the partial state as an observation and as a state of knowledge. In particular, looking ahead, our decision-tree policies will produce actions based on the three-valued observation vectors, not the actual state—so for example, it is meaningful for the policy to branch on whether or not an attribute's value is observed. Supposing we denote our observation in the tth step by $o^{(t)}$, we will encode $[o_i^{(t)} = b]$ for $b \in \{0, 1, *\}$ as variables $x_{i,b}^{(t)}$; we will add the clauses $x_{i,0}^{(t)} \vee x_{i,1}^{(t)} \vee x_{i,*}^{(t)}$ and $\neg x_{i_b}^{(t)} \vee \neg x_{i,b'}^{(t)}$ for all i and $b \neq b'$ to the background knowledge thus asserting that x_i takes exactly one of the three values.

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

STRIPS domains. A class of simple domains that are sufficiently general to provide a variety of excellent examples of propositional domains were first introduced for the STRIPS planner of Fikes and Nilsson (1971). 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 α has preconditions $p_1, \ldots, p_r$ and effects $e_1, \ldots, e_s$, for each time t, we have for each effect e_i a clause $[p_1^{(t)} \wedge \cdots \wedge p_r^{(t)} \wedge a_\alpha^{(t)}] \Rightarrow e_i^{(t+1)}$. The convention that the attributes of the state are only changed by actions is encoded by *"frame axioms"* like so. For each literal ℓ over some state of the environment and each action $\alpha_1 \ldots, \alpha_k$ that has an effect that falsifies ℓ, i.e., has some $e_i = \neg\ell$, we have a clause $[\ell^{(t)} \wedge \neg a_{\alpha_1}^{(t)} \wedge \cdots \wedge \neg a_{\alpha_k}^{(t)}] \Rightarrow \ell^{(t+1)}$.

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 (and our approach does *not* require it to be so, cf. Section 2.3.2), such examples are helpful to keep in mind as examples of POMDPs for which the dynamics can be concisely described by a CNF. We stress that in spite of our focus on STRIPS-like examples, our approach will extend to the kind of environments that can be expressed by, e.g., grounded PPDDL (Younes and Littman, 2004), provided

that there exists a sufficiently simple policy (for a given goal) that is "fault-tolerant" or otherwise reasonably reliable overall.

2.1.3 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 then 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 $\neg a_r^{(t)} \vee x_r^{(t+1)}$ and $\neg a_r^{(t)} \vee \neg x_{r'}^{(t+1)}$. In the usual language of STRIPS, $x_r^{(t+1)}$ and $\neg x_{r'}^{(t+1)}$ 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 $\neg x_b^{(t)} \vee \neg a_r^{(t)} \vee x_{b,r}^{(t+1)}$ and $\neg x_b^{(t)} \vee \neg a_r^{(t)} \vee \neg x_{b,r'}^{(t+1)}$ (for $r' \neq r$). These are *conditional effects* of $a_r^{(t)}$ with the *precondition* $x_b^{(t)}$.[2] 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 $\neg x_{b,r}^{(t)} \vee \neg x_r^{(t)} \vee \neg a_b^{(t)} \vee x_b^{(t+1)}$ for each r and b, and clauses $\neg a_b^{(t)} \vee \neg x_{b'}^{(t+1)}$ for $b' \neq b$. Here $x_{b,r}^{(t)}$ and $x_r^{(t)}$ 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, $\neg a_0^{(t)} \vee \neg x_b^{(t+1)}$ 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

2. Strictly speaking, conditional effects are not supported by STRIPS, so this is a (standard) extension.

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 (the agent drops the ball upon arrival in room r'), 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 $a_{b'}$ 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 $a_{r'}$) 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 $\neg x_{r,b}$ 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 (2000a) introduced PAC-Semantics in order to capture the property satisfied by the outputs of PAC-learning algorithms when formulated in a logic. For our purposes, PAC-Semantics will capture the extent to which a propositional formula holds on trajectories sampled according to the POMDP and some policy. We remark that Valiant's original motivation for formulating this semantics for a logic was to provide semantics for the representations created in the neuroidal cognitive model (Valiant, 2000b), and it is further proposed to capture certain kinds of learned "common sense" knowledge in general (Valiant, 2006). This roughly means that the agent possesses a large knowledge base without having been explicitly given most of its contents, as in the sense in which McCarthy (1959) used "common sense" when introducing the problem that we aim to address in this work. But, it also

so happens that this semantics features other hallmarks of "common sense" reasoning as identified in the AI literature, such as defaults and nonmonotonicity under conditioning, cf. Valiant (1995, 2000b) and Roth (1995). (We will not dwell on such aspects further in this work.) We note that systems using learning and logical reasoning based on PAC-Semantics have been successfully built and, for example, demonstrated to improve the accuracy with which a missing word in a sentence can be guessed (Michael and Valiant, 2008).

The key definition, capturing the notion of "approximation" in PAC-learning is:

Definition 5 ($(1-\epsilon)$-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 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. The distribution D we consider will generally be the distribution over actual states of the POMDP (and actions of the policy) given by the interaction between some policy and the POMDP, either the reference policy or else some (partial) policy under consideration by the algorithm.

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:

Definition 6 (Witnessed and testable CNFs) *We say that a CNF φ is* witnessed to evaluate to true *on a partial state ρ if every clause of φ contains a literal ℓ such that $\ell(\rho) = 1$. If φ is witnessed true with probability at least p on partial states from some distribution over masked states $M(D)$ (usually given by a masking process M on a distribution D), we say that φ is* p-testable *with respect to $M(D)$.*

With respect to the individual clauses of the CNF, it is easy to see that only clauses that are witnessed to evaluate to true are known to be true in an example, and thus only clauses that are testable under D can possibly be learned from random examples.[3]

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. In our model, 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 ℓ persists at time t is witnessed when either an action is taken that changes the fluent ℓ, $\ell^{(t-1)}$ is observed to be false, or when $\ell^{(t)}$ is observed to be true.

2.2.1 Reasoning

We introduced the notions of propositional knowledge representations and PAC-Semantics in order to capture the process of reasoning about missing information. In this work, we

3. 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.

are seeking algorithms for which we can provide theoretical guarantees of their performance. However, propositional reasoning was the original example of a (co-)NP-complete task (Cook, 1971). We will eventually recall a theorem due to Juba (2013) capturing a situation in which such reasoning is tractable. But, this theorem requires that we restrict the family of reasoning problems under consideration.

Although there are a variety of examples of families of reasoning problems for which algorithms exist, for our purposes the most natural family is described in terms of the *resolution* proof system. This is because resolution is one of the simplest propositional proof systems, and it simultaneously captures the capabilities of the most effective algorithms for reasoning in practice, cf. Beame et al. (2004). Recall, resolution is a proof system that operates on clauses. There is a single rule of inference in resolution called *cut* that allows one to derive new clauses: given a pair of clauses, $x \vee y \vee \cdots$ and $\neg x \vee z \vee \cdots$ that respectively contain an attribute x and its negation, cut allows one to derive the clause obained by taking the OR of the rest of the clauses, $(y \vee \cdots) \vee (z \vee \cdots)$.

Resolution is typically used to prove a DNF formula using a proof by contradiction: since the cut rule is sound, if we can derive the unsatisfiable, empty clause from an initial set of clauses, then the initial set must also have been unsatisfiable. If we are given that our background knowledge is true and we wish to prove that a DNF (say representing our goal) is satisfied, then by taking the negation of that DNF, we obtain a CNF via de Morgan's law; by showing that this CNF cannot be satisfied together with the background knowledge, we establish that the original DNF cannot be falsified, so it must be true.

For example, in our running example of the Gripper problem, suppose there are three rooms r, r', and r'', and two balls, b and b'. Suppose our goal for the moment is simply to not have the balls in the same room. This may be encoded by the DNF

$$[\neg x_{r,b} \wedge x_{r,b'}] \vee [\neg x_{r',b} \wedge x_{r',b'}] \vee [\neg x_{r',b} \wedge x_{r'',b'}].$$

Recall that we supposed that we had background knowledge capturing the fact that each ball is in exactly one room—in particular, for each of b and b', exactly one of $x_{r,b}$, $x_{r',b}$ and $x_{r'',b}$ is true. We would encode this by the clauses $x_{r,b} \vee x_{r',b} \vee x_{r'',b}$ (so at least one is true) and $\neg x_{r,b} \vee \neg x_{r',b}$, $\neg x_{r,b} \vee \neg x_{r'',b}$, and $\neg x_{r',b} \vee \neg x_{r'',b}$ (so at most one is true), and similarly for b'. Now, if we are in room r and we know that the ball b is in room r but the ball b' is not, then there is a resolution proof that the goal is satisfied as follows. We are now seeking to refute the negation of the goal,

$$[x_{r,b} \vee \neg x_{r,b'}] \wedge [x_{r',b} \vee \neg x_{r',b'}] \wedge [x_{r',b} \vee \neg x_{r'',b'}].$$

A resolution proof might proceed as follows: since we know $x_{r,b}$, our background knowledge that b is in at most one room, $\neg x_{r,b} \vee \neg x_{r',b}$ and $\neg x_{r,b} \vee \neg x_{r'',b}$ allows us to derive respectively $\neg x_{r',b}$ and $\neg x_{r'',b}$. In the second and third clauses of the negation of the goal, these allow us to derive $\neg x_{r',b'}$ and $\neg x_{r'',b'}$. But now, our background knowledge that b' is in some room, $x_{r,b'} \vee x_{r',b'} \vee x_{r'',b'}$ allows us to derive $x_{r,b'}$. But, we are given that b' is *not* in room r, i.e., $\neg x_{r,b'}$. So we can use this to arrive at the empty clause, a contradiction.

There is a natural graph associated with a proof: the clauses used in the proof appear as nodes of the graph, and for each derived clause in the proof, there is a directed edge from each of the clauses used in the derivation to the result. The sources of this graph are

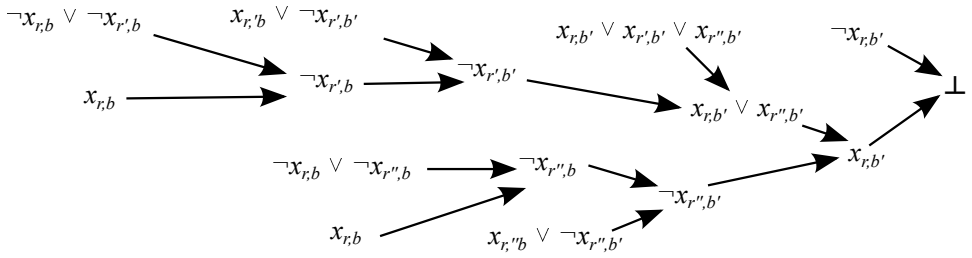

Figure 1: An example (treelike) resolution proof of $[\neg x_{r,b} \wedge x_{r,b'}] \vee [\neg x_{r',b} \wedge x_{r',b'}] \vee [\neg x_{r',b} \wedge x_{r'',b'}]$ from $x_{r,b}$, $\neg x_{r,b'}$, and the background knowledge.

labeled by premises of the proof, and the sink should be labeled by the empty clause. If this graph is a tree (we allow the same premise to label multiple leaves) then the proof is said to be *treelike*. Equivalently, a treelike proof is one that does not re-use intermediate derivations. The proof we just gave is treelike for example, cf. Figure 1. Any proof can be made treelike by simply repeating any intermediate steps; this comes at the price of increasing the number of steps, perhaps by up to a factor of two. Thus, the property of having a *small* treelike resolution proof (in terms of the number of steps) is a significant restriction. This restriction can be exploited to give efficient algorithms that guarantee that they find conclusions whenever such proofs exist. Again, we stress that without *some* such restriction, the problem of propositional reasoning is (co-)NP-complete.

A measure of tree complexity for proofs and policies. The main notion of "complexity" of a tree (either treelike resolution proof or decision tree) that we use is:

Definition 7 (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.*

So, for example, our example treelike resolution proof given in Figure 1 has Strahler number three. Kullmann (1999) was the first to consider resolution proofs of bounded Strahler number, and showed that they can be found in polynomial time (where the polynomial depends on the Strahler number).

A significant property, noted by Ansótegui et al. (2008), 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. In particular, our earlier example essentially followed from applications of the unit propagation rule, and the corresponding Strahler-2 derivation appears in Figure 2. (In general the Strahler-2 resolution proof follows the opposite derivation order from the unit propagation or chaining derivation.)

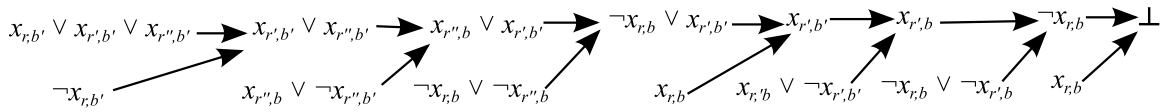

Figure 2: A Strahler-2 resolution proof of $[\neg x_{r,b} \wedge x_{r,b'}] \vee [\neg x_{r',b} \wedge x_{r',b'}] \vee [\neg x_{r',b} \wedge x_{r'',b'}]$ from $x_{r,b}$, $\neg x_{r,b'}$, and the background knowledge.

We also 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 (2001): 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. Thus, it will be helpful to think of this work as giving a family of algorithms, parameterized by the Strahler numbers they use, as fixed Strahler numbers encompass natural syntactic classes of proofs and decision trees. (We will discuss the interpretation of bounded Strahler number decision trees later.)

We also briefly remark that Ansótegui et al. (2008) 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 from Beame et al., 2004) 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.

Complex goals. We note that this environment model and class of proofs encompasses goals defined by arbitrary polynomial-size circuits, and not merely DNFs by a standard transformation: given a circuit defining a goal predicate, conceptually, we replace the POMDP we wish to solve with a new POMDP in which the states are extended with new propositional attributes that are always masked for each of the wires of the circuit, and the clausal (SAT) encoding of the constraints imposed by the circuit's gates is included as explicit background knowledge. Reasoning with resolution proofs of Strahler number two or greater will then use the background knowledge to evaluate the goal circuit: That is, once a setting for the inputs to a circuit has been guessed or fixed in some way, the unit propagation rule (which corresponds to a Strahler-2 proof) fills in the remaining, forced values of the circuit, much as it fills out the effects of a plan in an environment described by STRIPS rules. The propositional formula that we give to the algorithm then ultimately

consists only of the propositional variable encoding the output wire of the circuit, which then indeed indicates whether or not the goal is achieved.

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 (2013) 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 8 (cf. Theorem 13 of Juba 2013) *Let a KB CNF Φ and DNF query φ 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 ψ such that ψ is $(1-\epsilon+\gamma)$-testable under $M(D)$ and there is a Strahler-s treelike refutation of $\Phi \wedge \neg\varphi \wedge \psi$ or else*
2. *$[\Phi \Rightarrow \varphi]$ is at most $(1-\epsilon-\gamma)$-valid w.r.t. D*

Then, there is an algorithm running in time $O(\frac{W^2(|\Phi|+|\varphi|)}{\gamma^2} n^{2(s-1)} \log \frac{1}{\delta})$ (where $|\cdot|$ refers to the representation size in bits) that distinguishes these cases with probability $1-\delta$ when given φ, Φ, ϵ, γ, and a sample of $O(\frac{W^2}{\gamma^2} \log \frac{1}{\delta})$ partial assignments.

The cited result of Juba (2013) 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 example from $M(D')$ by w, we obtain an unbiased estimate over $M(D)$ (at a cost of needing to scale γ by W to obtain our guarantee, yielding the quoted change to the sample complexity and running time).

As suggested earlier, works such as Precup et al. (2001) showed how such weights enable us to use importance weighting to use the traces produced by the exploration policy Π^* to estimate the probability with which a new, candidate policy Π achieves a goal. Since our goal predicate G will be given by a DNF, Theorem 8 allows us to check that a sequence of actions $\alpha^{(1)}, \ldots, \alpha^{(T)}$ results in the satisfaction of the goal predicate G in $s^{(T)}$, and hence test if a sequence of actions leads to the goal being achieved: the formula $[a^{(1)}_{\alpha^{(1)}} \wedge \cdots \wedge a^{(T)}_{\alpha^{(T)}}] \Rightarrow G(s^{(T)})$ is also a DNF, and hence Theorem 8 can be applied. Theorem 8 guarantees that if the algorithm reports that the formula is valid, then with probability $1-\delta$ over the examples, G will be satisfied on $s^{(T)}$ after these actions with probability $1-\epsilon-\gamma$ over the dynamics of the POMDP. At the same time, Theorem 8 guarantees that the algorithm reports that the formula is valid when some appropriate testable CNF exists, under which G can be proved from $a^{(1)}_{\alpha^{(1)}} \wedge \cdots \wedge a^{(T)}_{\alpha^{(T)}}$, e.g., a CNF encoding STRIPS rules for the environment if we are using Strahler-2 treelike resolution as our proof system.

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 (Goldman and Boddy, 1996). 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, as proposed by Chapman and Kaelbling (1991) in the context of fully observed MDPs, and essentially applied to POMDPs in McCallum (1995, 1996).

Definition 9 (Decision tree policy) *A* decision tree policy *is a rooted tree of degree* ≤ 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,*
- *other nodes (of degree zero or one) may be labeled with actions*

with the interpretation that given an observation $o \in \{0, 1, *\}^n$*, 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 some fixed 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 $[\ell_1 \wedge \cdots \wedge \ell_k] \Rightarrow G$ is $(1-\epsilon)$-valid, then by a union bound, the policy will achieve G with probability at least $1 - \epsilon'$ for some ϵ' polynomially related to ϵ. Such a guarantee of $(1-\epsilon)$-validity can, in turn, be provided by the algorithm of Theorem 8 if a proof of $[\ell_1 \wedge \cdots \wedge \ell_k] \Rightarrow G$ (from some witnessed CNF) exists for each branch of the tree.

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, specifying a restriction on the space of policies we consider. For a fixed Strahler number, the trees will indeed have a fixed polynomial bound on their size, and hence we can establish their quality leaf-by-leaf as discussed above. Moreover, a Strahler number bound will enable polynomial-time algorithms to *find* a policy when such "simple," provably good policies exist, 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

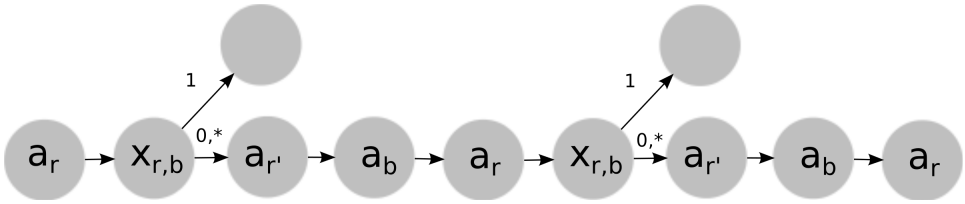

Figure 3: A Strahler-2 decision tree policy for the simple Gripper problem.

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 sth 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., 2004), 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 (Bryant, 1992) are of incomparable strength, depending on the variable ordering.

Alternatively, we can also characterize these policies in terms of the depth of nesting of the conditional branching: in a Strahler-s policy, at each conditional branch, there must be some sequence of branches such that the policy tests at most $s-1$ literals before terminating. Of course, this means that a policy that always tests k attributes to decide what action(s) to take has Strahler number k. Such a policy is a decision tree of size 2^k, and so may be very complex if k is even moderately large.

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 is depicted in Figure 3. It 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 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

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

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.

Stochastic environments, iteration, and Strahler numbers. It should be evident that even if the failure probabilities were much higher than 1%—say it was p—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 ϵ failure probability in $3 \log_p \frac{1}{\epsilon}$ 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. In such a way, even if the environment is highly stochastic and is not well described by STRIPS rules, e.g., a more general PPDDL domain (Younes and Littman, 2004), there may still be rules referring to multiple states (e.g., repetitions of a faulty action) that are witnessed true with high probability and that guarantee that the goal is achieved. We stress that Theorem 8 (and hence our ultimate algorithm) only requires that such a rule exists to (implicitly) make use of it. We do not need to ever explicitly construct the rule, so it does not matter if the rule is complex, and it does not matter if all of the simple (STRIPS) rules are unreliable.

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 $x_{r,b} \wedge x_{r',b'}$ (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 $x_{r,b}$ and $x_{r',b'}$ 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* $\neg x_{r,b} \vee \neg x_{r',b'}$ with a chaining proof, and hence we have a Strahler-2 derivation of $x_{r,b} \wedge x_{r',b'}$.

3. Learning and Planning

Algorithms for simultaneously learning and reasoning in PAC-Semantics (as captured in Theorem 8) turn out to provide a bridge between *learning* dynamics from examples and logic-based approaches to *planning*. Intuitively, Theorem 8 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 (1989) 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 via an algorithm that recursively searches for a decision tree policy that achieves the goal with high probability below a given, candidate branch.

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 Φ, 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 $m = O(\frac{1}{\mu^2\gamma^2}(nT + \log\frac{|A|}{\mu} + \log\frac{1}{\delta}))$ weighted traces of an arbitrary policy Π^*. (Again, $|\Phi|$ refers to the representation size of Φ in bits, while $|A|$ refers to the cardinality of A.)*

Suppose there is a Strahler-s T-horizon decision tree policy Π taking μ-typical actions with respect to Π^* 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 ψ that is $(1 - \epsilon + \gamma)$-testable over traces of the POMDP with Π. 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 stress that our focus is primarily on policies and proofs with small Strahler numbers, say $s = 2$, $s = 4$, etc., so that this algorithm is polynomial-time and obtains a polynomially bounded increase of the failure probability.

We require the following combinatorial lemma:

Lemma 11 *1. The number of branches k in a T-horizon decision tree policy over n fluents 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 n fluents with μ-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

Algorithm 1 Find-DT()

PAC-refute(φ, R) decides if $\varphi \wedge \psi$ is at most ϵ-valid given some testable ψ over weighted partial examples in R (taken out of m) (cf. Theorem 8).
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 $\perp$
if Π takes more than T actions **then**
  return $\perp$
end if
$R^t \leftarrow \{(\times_{i=1}^{t-1}(o^{(i)}, a^{(i)}) \times o^{(t)}, \Pi_{i=1}^{t-1} w_i) : ((o^{(i)}, a^{(i)}, w_i)_{i=1}^{T}, o^{(T+1)}) \in R, \Pi_{i=1}^{t-1} w_i \leq \frac{4}{\mu}\}$
if PAC-Refute($\neg G^{(t)} \wedge \Phi \wedge \Pi$ taken, R^t) **then**
  return Π.
end if
if $s > 1$ **then**
  for all ℓ s.t. $\ell^{(t)}$ and $\neg\ell^{(t)}$ not in Π **do**
    $\Pi_\ell \leftarrow \Pi$ with a final test for $\ell^{(t)}$
    $\Pi_1 \leftarrow$Find-DT$(G, \Phi, s-1, \Pi_\ell, T, R)$
    if $\Pi_1 \neq \perp$ **then**
      $\Pi_0 \leftarrow$Find-DT$(G, \Phi, s, \Pi_{\neg\ell}, T, R)$
      if $\Pi_0 \neq \perp$ **then**
        return Policy following Π_b if $\ell^{(t)} = b$
      else
        return $\perp$
      end if
    end if
  end for
end if
for all α s.t. $\#\{\rho \in R : a^{(t)} = \alpha\} \geq (\mu/2)m$ **do**
  $\Pi_\alpha \leftarrow \Pi$ with $a^{(t)} = \alpha$
  $R_\alpha \leftarrow \{\rho \in R : a^{(t)} = \alpha\}$
  $\Pi' \leftarrow$Find-DT$(G, \Phi, s, \Pi_\alpha, T, R_\alpha)$
  if $\Pi' \neq \perp$ **then**
    return $a^{(t)} = \alpha$ followed by Π'
  end if
end for
return $\perp$

follows immediately from the first part of Lemma 1 of Ehrenfeucht and Haussler (1989). For the second part, we simply note that there are at most $\frac{1}{\mu}$ possible maximal sequences of μ-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. ■

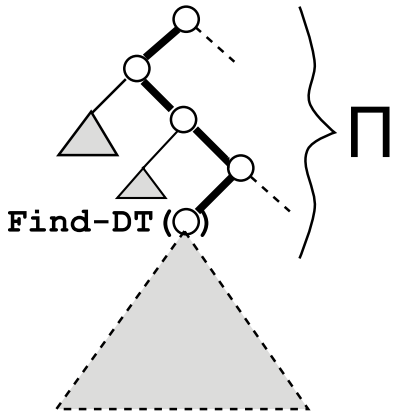

Figure 4: A snapshot of the execution of Algorithm 1: `Find-DT` is invoked at the indicated point at the end of the branch Π. `Find-DT` returns a subtree (shown in dashes) rooted at the end of Π such that the overall policy achieves the input goal G with high probability at all of the leaves of the subtree.

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 (see Figure 4). Intuitively, we are filtering our samples so that we obtain samples from a policy $\tilde{\Pi}^*$ that simulates Π^* until Π^* 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 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 $m \geq \frac{8}{\mu^2}(3nT \ln 3 + \ln \frac{4T|A|}{\mu} + \ln \frac{3}{\delta})$ 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 μ-typical action sequences of Π^* 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 Π, the weights we compute for our queries are

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

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

As Theorem 8 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 μ-typical actions under Π^* 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) there is no harm in asserting the opposite setting of $x^{(t)}_{i,b}$, 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 8 also guarantees that for the quoted number of examples, all of the branches of the target policy will be accepted w.p. at least $1 - \delta/3$ 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 `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 $f(0, m, s) = 0$, $f(n', m, 0) = W \cdot m$ 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 an 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 μ-typical actions in quasipolynomial time since if a policy has B branches, Lemma 11 guarantees it has Strahler number at most $s = \log B + 1$.

3.1 An Illustration of the Algorithm

We return once again to our running example of the simple goal in the Gripper domain to illustrate the behavior of our algorithm; recall that in Section 2.3.2 we described a Strahler-2 decision tree policy for the goal $x_{r,b}$ (carrying the ball b to room r in a two-room environment), that had a Strahler-2 proof from a witnessed CNF that the goal was achieved with probability greater than 99% on each branch (it actually had probability greater than 99.96%). The policy itself appeared in Figure 3.

There is actually a conformant (Strahler-1) policy that achieves the goal with similar probability. Indeed, the policy we discussed in Section 2.3.2 does not need to examine the state of the environment. Our analysis of the plan could be applied about as well to the plan that simply applies the sequence of actions "move to room r', try to pick up the ball, move to room r, drop the ball" twice. In this case, when the plan is successful, the goal $x_{r,b}$ is actually witnessed satisfied on the final step (the attribute is unmasked in room r), and hence is "proved" by a trivial, Strahler-1 proof. Let's suppose that this policy involves a μ-typical sequence of actions for the exploration policy Π^*, and first consider what Algorithm 1 does when invoked on Strahler number $s = 1$.

The algorithm first checks to see if the goal, $x_{r,b}$ can be proved satisfied in the initial environment configuration with probability 99% using `PAC-refute`. Let's suppose not. Then, since the algorithm is invoked on Strahler number $s = 1$, it skips the first loop that searches for literals to branch on, and moves directly to a search over actions. It partitions the examples according to the first action taken, invoking a recursive call along a branch taking each possible action that is empirically at least $\mu/2$-typical. Let us consider the recursive call on a branch that starts with the action, "move to room r'" that we know may lead to a successful policy. This branch consists of a literal asserting $a^{(1)}_{r'} = 1$, and the recursive call only receives the subset of traces in which this action was taken. Again, `PAC-refute` should determine that $x^{(2)}_{r,b}$ is not provable on these traces. Now, the algorithm searches over actions at $t = 2$; since we are only considering traces in which $a^{(1)}_{r'} = 1$, strictly fewer actions produce a sequence (following $a_{r'}$ at $t = 1$) that is empirically $\mu/2$-typical; perhaps few of these traces try to move to room r' again, or few traces try to move immediately back to room r. The algorithm then does not make a recursive call for those actions. Since we have supposed that the overall policy is μ-typical, the action "pick up ball b" in particular will still be considered, so we consider the recursive call on this branch that now asserts $a^{(1)}_{r'} = 1$ and $a^{(2)}_{b} = 1$, and only considers the traces satisfying these conditions. On this call, we should still not find that $x^{(3)}_{r,b}$ is provable, and so we will recursively consider the sequences that begin with $a^{(1)}_{r'}$ and $a^{(2)}_{b}$, and are empirically $\mu/2$-typical; a_r should be an option. Now, on this call, `PAC-refute` should find that $x^{(4)}_{r,b}$

is at least 98%-valid. Unfortunately, this is not high enough, so the algorithm continues; but, we know that on the branch $a^{(1)}_{r'}$, $a^{(2)}_{b}$, $a^{(3)}_{r}$, $a^{(4)}_{0}$, $a^{(5)}_{r'}$, $a^{(6)}_{b}$, $a^{(7)}_{r}$, then in more than 99% of the traces, $x^{(8)}_{r,b} = 1$, so this branch will be returned. Of course, there is no guarantee that this policy will be the one that is found. For example, if the sequence of actions that starts with dropping any balls, and then moving to room r, and then continuing with the above sequence is empirically $\mu/2$-typical under Π^* for some reason, then it may be that the algorithm returns a policy $a^{(1)}_{0}, a^{(2)}_{r}, a^{(3)}_{r'}, a^{(4)}_{b}, a^{(5)}_{r}, a^{(6)}_{0}, \ldots$.

It is also instructive to consider the behavior of Algorithm 1 in this example, when invoked with Strahler number $s = 2$. This may find a rather different policy than the one illustrated in Figure 3. As before, we suppose that $x^{(1)}_{r,b}$ is not provable with sufficiently high probability. The $s = 2$ case then searches for tests of a single literal such that Strahler-1 branches suffice to achieve the goal, given that the test is satisfied. Naturally, the goal itself, $x_{r,b}$ is such a test, the leaf of the branch on which $x^{(1)}_{r,b} = 1$ trivially satisfies the goal w.p. 1. So, the algorithm will generally construct such a branch when it considers the literal $x_{r,b}$, and continues recursively searching for a Straher-2 policy on the subtree where $x^{(1)}_{r,b} \in \{0, *\}$. But, on any other literal, we also know that there is a Strahler-1 policy that achieves the goal with probability greater than 99%. For such literals, depending on how many traces are consistent with the sequence of conditions on $x^{(1)}$ that have been imposed thus far, a branch featuring some appropriate Strahler-1 plan will be added. Relatedly, we may also find that some branches are added that use, e.g., the policy we discussed that only achieves the goal with probability 98%, as long as at least 1% of the traces that would end in failure do not reach that branch, so that overall only at most 1% of the traces actually fail on the branch. Finally, some conditions on $x^{(1)}$ may simply be satisfied by fewer than 1% of the traces. Since our algorithm tolerates branches with a 1% failure probability in general, the algorithm will also add such branches to the tree, terminating immediately with leaves.[4] In general, the algorithm continues until either (1) less than 1% of the traces remain consistent with all of the conditions on the traces imposed by the main, Strahler-2 branch, in which case the algorithm will consider the remaining leaf to be adequate or (2) the main, Strahler-2 branch adds a successful sequence of actions, allowing $x^{(t)}_{r,b}$ to be proved to hold with probability 99% (as in the Strahler-1 case described above). Actually, in this example, once the algorithm considers branches for each of the three possible observed settings of any single attribute, every example trace is consistent with one of these three branches, and hence less than 1% (actually, 0%) will remain consistent with the main branch. The algorithm will therefore terminate before moving on to consider a single action on the main, Strahler-2 branch. In any case, depending on the order in which the attributes are considered, the final decision tree policy that is returned may have up to $\sim n$ branches, each either containing some suitable conformant plan or reached by fewer than 1% of the traces (and may not achieve the goal). This possible inclusion of extra branches that are each allowed a 1% failure probability is the reason for the $\sim nT$ increase in the total failure probability in the guarantee provided by Theorem 10.

4. Intuitively, such branches that only lead to failure are undesirable. But, in general sometimes we may need to take a branch with a high conditional failure probability in order to capture a few traces where the policy succeeds.

3.2 Discussion

The complexity of Algorithm 1, as established by Theorem 10 is roughly what one would hope to achieve given the current state of knowledge in learning theory. Our task is very similar to supervised learning in the sense that we are using examples from a *fixed distribution* over trajectories generated by the reference policy Π^* to estimate the probability with which policies achieve the goal. Our sample complexity depends logarithmically on the number of possible branches (that is, linearly in their representation size), and quadratically on $\mu\gamma$. Note that a branch of a decision tree computes a conjunction, i.e., the AND of the literals that lead to that branch; the former quantity is the *VC-dimension* (Vapnik and Chervonenkis, 1971) of the class of conjunctions, which characterizes the number of examples needed to estimate their fit (Vapnik and Chervonenkis, 1971; Blumer et al., 1989; Ehrenfeucht et al., 1989). We are exploiting the fact that the VC-dimension actually characterizes the number of examples needed in order to obtain a *uniform bound* on the estimates of the error for *the entire class of conjunctions*—that is, the set of all possible branches, here. As we have seen (in the algorithm of Ehrenfeucht and Haussler, 1989), one only needs to know which branches have low error in order to identify a good tree. The latter, quadratic dependence on $\mu\gamma$ arises because we are trying to estimate the probability that a sequence of actions (that only is taken with probability μ) fails up to an accuracy of γ; since we are essentially conditioning on an event of probability $\sim \mu$, this means that we are essentially seeking to estimate the probability that the sequence of actions leads to failure up to an accuracy of $\mu\gamma$ in the original distribution. It is well-known that this latter task requires $\sim 1/(\mu\gamma)^2$ samples. In any case, we stress that if we had not separated out the task of exploration, a $\sim |A| \cdot 2^n$ or 2^T lower bound on the sample complexity holds for even very simple environments (Kakade, 2003, Section 2.5).

Likewise, the time complexity for finding a Strahler-s policy is comparable to the time taken by the algorithm of Ehrenfeucht and Haussler (1989) for finding decision tree classifiers, which remains the fastest known algorithm for this task. Our running time is slightly greater since we need to use theorem-proving techniques to determine whether or not a branch of our policies are suitable, and we test each branch individually. It is an interesting question whether or not, for example, some intermediate results can be cached across invocations of our theorem-proving algorithms, possibly leading to a reduced total running time.

The main quantitative downside of this algorithm is the potential $\sim (nT)^{s-1}$ increase of the failure probability of the policy found by the algorithm over the best-possible policy. Unfortunately, this is roughly in accordance with what we hope to achieve with the current state of the art. We again draw an analogy to supervised classification, in which even very simple tasks seem to lead to a similarly large increase in the error in the related "agnostic" noise model. The trouble is that the task of finding representations that optimize the error appears to be intractable under various plausible assumptions, even for very simple classes of representations such as halfspaces (Shalev-Shwartz et al., 2011) or even disjunctions (Kearns et al., 1994; Daniely et al., 2014; Daniely and Shalev-Shwartz, 2014). We still do not know how tight an approximation of the optimal error can be achieved, but an approximation factor that depends polynomially on the number of attributes reflects the current state of the art: consider for example, the relatively simple task of learning a disjunctive classifier (an

OR of the attributes). The best known algorithm for this task in the agnostic noise model suffers an increase of $\sim n^{1/3}$ in its error probability when there are n attributes (Awasthi et al., 2010).

Thus, in practice, this algorithm is only useful in situations when a highly fault-tolerant policy exists. As we noted in Section 2.3.2, such policies may exist, even if the environment is highly stochastic. Our algorithm is relatively tolerant of a long time horizon, and by repeating actions, it may be possible to decrease the failure probability of the overall plan exponentially with T (whereas the dependence of the overhead on T only grows linearly). This depends on repetitions of such actions being "typical" under Π^*, of course, but if Π^* is produced by (for example) solving other, smaller instances of planning problems, such sequences of repeated actions in a stochastic environment may well be typical.

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 $[\ell^{(t)} \wedge \sim(\neg\ell^{(t+1)})] \Rightarrow \ell^{(t+1)}$, where $\sim \ell$ roughly means "ℓ 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 $\sim\ell$ 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 8 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* (van Gelder et al., 1991) (WFS) for logic programs. More specifically, we will modify the iterated quotient definition of Przymusinski (1994) 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 φ over literals possibly using $\sim$ and any pair of sets of literals (L_+, L_-), the* quotient of φ 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 $\tilde{\ell}$, and simplifying the resulting formula by deleting satisfied clauses and falsified literals.*

The intuition behind the quotient is that the sets L_+ and L_- consist of the literals that we know, respectively, can and can't be proved from φ. 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 φ is a pair of sets of literals (L_+, L_-) (*not *over the $\tilde{\ell}$ variables) s.t. $\ell \in L_+$ iff ℓ has a*

Strahler-s proof from φ and $\ell \in L_-$ iff for the CNF φ' in which all occurrences of the $\tilde{\ell}$ variables have been deleted, there is no Strahler-s proof of ℓ from φ'.

This definition is analogous to a construction of the (unique) "least partial models" for normal logic programs from Przymusinski (1991), 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 φ 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 ℓ requires that the ℓ be true in the partial state, and if for *some* substitution of the different occurrences of $\sim$ literals for truth values there exists a Strahler-s derivation of ℓ, then that prevents setting ℓ 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 van Gelder et al. (1991), this definition also conforms to the desirable intuition that literals are only marked as "provable" or "unprovable" if this can be ultimately derived from φ 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 φ over literals possibly using $\sim$, the sequence of pairs of sets of literals $(L_+^{(0)}, L_-^{(0)}) = (\varnothing, \varnothing)$, $(L_+^{(i+1)}, L_-^{(i+1)}) = LPS_s(\varphi/(L_+^{(i)}, L_-^{(i)}))$ converges to a pair of sets of literals (L_+^*, L_-^*) s.t. for every literal ℓ (without $\sim$),*

1. *if there is a Strahler-s proof of ℓ from $\varphi/(L_+^*, L_-^*)$, then $\ell \in L_+^*$*
2. *if $\ell \in L_-^*$ then there is no Strahler-s proof of ℓ from $\varphi/(L_+^*, L_-^*)$*

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

Proof Convergence will follow from the observation that $L_+^{(i)} \subseteq L_+^{(i+1)}$ and $L_-^{(i)} \subseteq L_-^{(i+1)}$. For $i = 0$, the statement is trivial.

To see $L_+^{(i)} \subseteq L_+^{(i+1)}$ for $i \geq 1$, consider the proof of ℓ from $\varphi/(L_+^{(i-1)}, L_-^{(i-1)})$ witnessing $\ell \in L_+^{(i)}$. By induction on the structure of the proof of ℓ, 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 may be obtained by weakening 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 $L_-^{(i)} \subseteq L_-^{(i+1)}$ for $i \geq 1$, we assume inductively that $L_+^{(i-1)} \subseteq L_+^{(i)}$ and $L_-^{(i-1)} \subseteq L_-^{(i)}$. We note that when $\ell \notin L_-^{(i+1)}$, there is a proof of ℓ from $(\varphi/(L_+^{(i)}, L_-^{(i)}))'$, which is constructed from φ 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 $(\varphi/(L_+^{(i-1)}, L_-^{(i-1)}))'$ since $L_+^{(i-1)} \subseteq L_+^{(i)}$ and $L_-^{(i-1)} \subseteq L_-^{(i)}$ 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_+^*, L_-^*) 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_+^*, L_-^*) such that $(L_+^*, L_-^*) = LPS_s(\varphi/(L_+^*, L_-^*))$, and hence the two claimed properties are immediate from the definition of the Strahler-s least partial state. ■

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

Definition 15 *We say that a formula φ (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 ψ consisting of the conjunction of literals satisfied on $m(x)$, putting (L_+^*, L_-^*) equal to the fixed point obtained from $\varphi \wedge \psi$, the probability (over m and x) that x satisfies $\varphi/(L_+^*, L_-^*)$ is at least $1 - \epsilon$.*

We can answer queries in this extended PAC-Semantics by an easy modification of the algorithm underlying Theorem 8 wherein, for each example, we first compute (L_+^*, L_-^*) for the KB Φ, and then for a query φ, check for a refutation of $\varphi \wedge (\Phi/(L_+^*, L_-^*))$. Such an algorithm simply replaces the existing subroutine in Algorithm 1. For $s \geq 2$, we can then verify that the generic frame axioms $[\ell^{(t)} \wedge \sim(\neg\ell^{(t+1)})] \Rightarrow \ell^{(t+1)}$ are $(1 - \epsilon - \nu)$-valid under the Strahler-s NAF semantics in a noisy STRIPS instance with noise rate ν 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., (Otero, 2005; Amir and Chang, 2008). Others attempt to construct a model that accounts for the imperfection (García-Martínez and Borrajo, 2000; Schmill et al., 2000; Pasula et al., 2007; Yoon and Kambhampati, 2007;

Lang and Toussaint, 2009; Mourão et al., 2012), 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 (2011) speak to the advantages of attempting to answer queries without producing an explicit action model. Work on applying logic programming to learning for planning (Thon et al., 2009) features algorithms with theoretical grounding for several tasks, but similarly encounters the structure learning problem when learning rule sets, and indicates that it (together with dealing with hidden information) remains a frontier problem for that approach. Work on Predictive State Representations by Boots et al. (2011) only establishes consistency, not fast convergence under any clear conditions—some kind of assumption on the condition number of the matrices seems essential, but neither the details nor the significance of this requirement seem to be understood.

The use of a limited class of policies to approximate optimal behavior has a long history in the reinforcement learning literature, see Sutton and Barto (1998, Chapter 8) for a review; in particular, techniques that used neural nets or other kinds of linear approximators to choose a good action are examples of such techniques. Chapman and Kaelbling (1991) seem to be the first to limit their policies to decision trees. The Utile Suffix Memory of McCallum (1995) similarly builds a coarsened approximation of the state space itself by classifying histories in partial information environments. Both of these works viewed the decision trees as dynamically refined estimates of the environments in which (approximately) optimal actions may be selected directly, rather than simply as policy representations to optimize; this latter view emerged only somewhat later (Kearns et al., 2002; Meuleau et al., 1999). We note that McCallum (1995) did not consider factored state and observation spaces, and so Utile Suffix Memory differs from our decision trees in that it does not branch along the propositional factors of the observations as we use here; this was later considered in the U-Tree algorithm of McCallum (1996). In any case, both Utile Suffix Memory and U-Tree are still viewed as classifying a partial trace, indexed by offsets from the current observation (e.g., branching on the observation from t steps prior) by a choice of action. Conceptually, this is more like a reactive policy, but applied to a longer history; by contrast, "states" of our decision tree policies are captured by a node of the tree, so the behavior of the policy is "non-uniform" with respect to time. The representation is still thus not equivalent once complexity measures such as the size come into consideration; our objective is also different, in that we merely seek for our trees to reach a goal state, whereas McCallum's policies are in the usual utility-maximization framework in which one seeks a high discounted utility over time. The algorithm we use thus also ends up being rather different from those proposed by McCallum. Finally, McCallum focuses strictly on empirical evaluation of his strategies, whereas our focus is strictly on a theoretical evaluation.

Our setting is related to a setting considered previously by Khardon (1999), apprenticeship learning (Syed and Schapire, 2010) and reductions to classification (Langford and Zadrozny, 2005): These works aim to learn a policy using example traces, even for sophisticated policy representations. The distinction is that we don't assume that Π^* is our desired policy, just that there exists a good policy that Π^* agrees with at least a μ-fraction of the time. (If the number of actions is small, Π^* could even be a random walk.)

Reinforcement learning techniques such as Kearns et al. (2002) can learn from a poor initial policy such as a random walk. But, these works often face the exploration problem by using repeated experience with the environment model, in particular with an explicitly

provided reward function – i.e., goal – at hand to guide their search for a policy. Largely as a consequence of the inherent difficulties of exploration in general environments, the theoretical analyses provided by such works exhibit an undesirable dependence on either the size of the state space or, in the particular case of Kearns et al., on the time horizon. (We note that a discount factor of $1 - 1/T$ is roughly equivalent to a time horizon of T.) Along somewhat similar lines, work by Fern et al. (2006) and Lazaric et al. (2010) consider learning from a poor initial policy in a full-information model by using a reduction to classification; the full-information setting is quite different in that the history of observations is no longer relevant (and so there is no need to consider a stateful policy like our decision trees). Theoretically, Fern et al. showed how to solve the MDP if the best action at each state is substantially better than the second best, and if the resulting deterministic optimal policy for a fixed horizon is expressed by an efficiently learnable class. Naturally, these assumptions may easily be violated, in particular if the actions in a plan can be reordered. Lazaric et al. extended the analysis of Fern et al., but assume that a policy minimizing the (nonzero, in general) loss can be found somehow. It is not clear what kind of policies Lazaric et al. have in mind, but in Boolean classification, this is essentially the problem of agnostic learning (Kearns et al., 1994); recent evidence suggests that this is intractable for all but the simplest kinds of classifiers (Daniely et al., 2014; Daniely and Shalev-Shwartz, 2014).[5]

Work by Walsh (2010) touches on all of the above areas; he constructs complete relational domain models with a full theoretical analysis, but under the assumption of constant arity expressions and numbers of effects (and/or in an apprenticeship learning setup). The assumptions of bounded arity mean that these relational expressions can be expressed by moderate size propositional representations, so the main trade-off between Walsh's work and ours is that Walsh produces explicit rules unlike us, but at the cost of exponential scaling when these arities and numbers of possible effects is large.

Learning from exercises (Natarajan, 1989; Tadepalli, 2008) 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 et al. (2010) 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.

We noted that the use of importance sampling to enable off-policy learning in POMDPs was first considered by Precup et al. (2000, 2001), building on earlier work on off-policy learning in MDPs (Sutton and Barto, 1998, Chapter 5). Precup et al. used a second-moment bound in their analysis of importance sampling since in general, the likelihood ratios may be large. Indeed, the issue that generally plagues importance sampling is that this variance may be high (Shelton, 2001; Peshkin and Shelton, 2002), and a variety of techniques (e.g., Hachiya et al., 2009, 2011) have been proposed to control the variance in more general settings. This problem actually does not arise in our setting, as a consequence

5. And, *approximate* agnostic learning generally incurs an increase in the loss of a similar magnitude as we obtain in Theorem 10—cf., the best known polynomial time algorithm for learning disjunctive classifiers on n attributes increases the loss by a factor of $n^{1/3}$ (Awasthi et al., 2010).

of our exclusive focus on μ-typical sequences of actions under the sampling distribution, which thus translates into an absolute $1/\mu$ bound on the likelihood ratio. We note that some other works (Uchibe and Doya, 2004; Wawrzyński, 2009) have simply capped the likelihood ratios in an importance sampling computation; here, we achieve such a bound by *disregarding* policies that have a large likelihood ratio, instead of altering it. Our analysis of importance sampling then follows a standard learning theory paradigm, most similar to that of Peshkin and Mukherjee (2001), except that the bound on the likelihood ratios allows us to get away with the use of Hoeffding's inequality rather than Bernstein's inequality (which considers the variance).[6]

6. Future Directions

One natural direction concerns improving Theorem 10: One of the main insights underlying Theorem 8 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 ϵ 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 Fern et al., 2006). In a related direction, Ross and Bagnell (2012) essentially showed that a penalty for atypicality similar to what our algorithm suffers can be eliminated in the standard (discounted, fixed cost-function, full-information) MDP setting by iteratively constructing a policy, collecting new data, retraining the policy, and repeating. In this way, the training data is shaped so that the actions of the policy become highly "typical" and well-estimated. It is likely that a similar strategy will also work in our partial information setting, although it may rely on the goal remaining fixed during this process.

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.

Acknowledgments

I am grateful to Leslie Kaelbling for numerous detailed suggestions and criticisms of this work that improved it immensely. I likewise thank Mithun Chakraborty and my anonymous reviewers for their detailed comments and suggestions. This work was also heavily influenced by conversations with Leslie Valiant.

6. For off-policy evaluation in practice, it may be vastly preferable to apply a Bernstein-like inequality using an empirical estimate of the variance instead of Hoeffding's inequality (Thomas et al., 2015). I thank a reviewer for bringing this to my attention.

References

Eyal Amir and Allen Chang. Learning partially observable deterministic action models. *JAIR*, 33:349–402, 2008.

Carlos Ansótegui, María Luisa Bonet, Jordi Levy, and Felip Manyá. Measuring the hardness of SAT instances. In *Proc. 23rd AAAI*, pages 222–228, 2008.

Pranjal Awasthi, Avrim Blum, and Or Sheffet. Improved guarantees for agnostic learning of disjunctions. In *Proc. 23rd COLT*, 2010.

J. Andrew Bagnell, Sham Kakade, Andrew Ng, and Jeff Schneider. Policy search by dynamic programming. In S. Thrun, L. K. Saul, and B. Schőlkopf, editors, *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, pages 831–838. MIT Press, Cambridge, MA, 2004.

Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *JAIR*, 22:319–351, 2004.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

Byron Boots, Sajid M. Siddiqi, and Geoffrey J. Gordon. Closing the learning-planning loop with predictive state representations. *Int. J. Robotics Res.*, 30(7):954–966, 2011.

Randy E. Bryant. Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):293–218, 1992.

Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69:165–204, 1994.

David Chapman and Leslie Pack Kaelbling. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proc. 12th IJCAI*, pages 726–731, 1991.

Stephen A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd STOC*, pages 151–158, 1971.

Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limtations on learning DNF's. To appear in *29th COLT*, 2016. Preprint version: arXiv:1404.3378

Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Proc. 46th STOC*, pages 441–448, 2014.

Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Inf. Comp.*, 82(3):231–246, 1989.

Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Inf. Comp.*, 82:247–261, 1989.

Kutluhan Erol, Dana S. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 76(1–2):75–88, 1995.

Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Inf. Comp.*, 171(1): 84–97, 2001.

Alan Fern, Sungyook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias: solving relational Markov decision processes. *JAIR*, 25:85–118, 2006.

Richard E. Fikes and Nils J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

Ramón García-Martínez and Daniel Borrajo. An integrated approach of learning, planning, and execution. *J. Intelligent and Robotic Sys.*, 29(1):47–78, 2000.

Robert P. Goldman and Mark S. Boddy. Expressive planning and explicit knowledge. In *Proc. 3rd AIPS*, pages 110–117, 1996.

Oded Goldreich, Brendan Juba, and Madhu Sudan. A theory of goal-oriented communication. *J. ACM*, 59(2):8:1–8:65, 2012.

Hirotaka Hachiya, Takayuki Akiyama, Masashi Sugiyama, and Jan Peters. Adaptive importance sampling for value function approximation in off-policy reinforcement learning. *Neural Networks*, 22(10):1399–1410, 2009.

Hirotaka Hachiya, Jan Peters, and Masashi Sugiyama. Reward-weighted regression with sample reuse for direct policy search in reinforcement learning. *Neural Networks*, 23(11): 2798–2832, 2011.

David Haussler. Learning conjunctive concepts in structural domains. *Mach. Learn.*, 4(1): 7–40, 1989.

Rune M. Jensen, Manuela M. Veloso, and Randal E. Bryant. Fault tolerant planning: Toward probabilistic uncertainty models in symbolic non-deterministic planning. In *Proc. 14th ICAPS*, pages 335–344, 2004.

Saket Joshi, Kristian Kersting, and Roni Khardon. Self-taught decision theoretic planning with first order decision diagrams. In *Proc. 20th ICAPS*, pages 89–96, 2010.

Brendan Juba. *Universal Semantic Communication*. Springer, Berlin, 2011.

Brendan Juba. Implicit learning of common sense for reasoning. In *Proc. 23rd IJCAI*, pages 939–946, 2013.

Sham M. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, 2003.

Sham M. Kakade and John Langford. Approximately optimal reinforcement learning. In *Proc. 19th ICML*, pages 267–274, 2002.

Henry Kautz and Bart Selman. Planning as satisfiability. In *Proc. 10th ECAI*, pages 359–363, 1992.

Michael Kearns and Leslie Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM*, 41:67–95, 1994.

Michael Kearns, Yishay Mansour, and Andrew Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Mach. Learn.*, 49(2):193–208, 2002.

Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Towards efficient agnostic learning. *Mach. Learn.*, 17(2-3):115–141, 1994.

Roni Khardon. Learning to take actions. *Mach. Learn.*, 35(1):57–90, 1999.

Roni Khardon and Dan Roth. Learning to reason. *J. ACM*, 44(5):697–725, 1997.

Oliver Kullmann. Investigating a general hierarchy of polynomially decidable classes of CNF's based on short tree-like resolution proofs. Technical Report TR99-041, ECCC, 1999.

Tobias Lang and Marc Toussaint. Approximate inference for planning in stochastic relational worlds. In *Proc. 26th ICML*, pages 585–592, 2009.

John Langford and Bianca Zadrozny. Relating reinforcement learning performance to classification performance. In *Proc. 22nd ICML*, pages 473–480, 2005.

Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Analysis of classification-based policy iteration algorithms. In *Proc. 27th ICML*, 2010.

Yishay Mansour. Reinforcement learning and mistake bounded algorithms. In *Proc. 12th COLT*, pages 183–192, 1999.

Andrew Kachites McCallum. Learning to use selective attention and short-term memory in sequential tasks. In *From animals to animals 4: proceedings of the fourth international conference on simulation of adaptive behavior*, pages 315–325. MIT Press, Cambridge, MA, 1996.

R. Andrew McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *Proc. 12th ICML*, pages 387–395, 1995.

John McCarthy. Programs with common sense. In *Teddington Conf. on the Mechanization of Thought Processes*, pages 756–791, 1959. Available at http://www-formal.stanford.edu/jmc/mcc59.html.

Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proc. 15th UAI*, pages 417–426, 1999.

Loizos Michael. Partial observability and learnability. *Artificial Intelligence*, 174(11):639–669, 2010.

Loizos Michael. Causal learnability. In *Proc. 22nd IJCAI*, pages 1014–1020, 2011.

Loizos Michael and Leslie G. Valiant. A first experimental demonstration of massive knowledge infusion. In *Proc. 11th KR*, pages 378–389, 2008.

Kira Mourão, Luke Zettlemoyer, Ronald P. A. Petrick, and Mark Steedman. Learning STRIPS operators from noisy and incomplete observations. In *Proc. 28th UAI*, pages 614–623, 2012.

Balas K. Natarajan. On learning from exercises. In *Proc. 2nd COLT*, pages 72–87, 1989.

Ramon P. Otero. Induction of the indirect effects of actions by monotonic methods. In *Proc. ILP 2005*, volume 3625 of *LNAI*, pages 279–294. Springer, 2005.

Hanna M. Pasula, Luke S. Zettlemoyer, and Leslie Pack Kaelbling. Learning symbolic models of stochastic domains. *JAIR*, 29:309–352, 2007.

Leonid Peshkin and Sayan Mukherjee. Bounds on sample size for policy evaluation in Markov environments. In *Proc. COLT/EuroCOLT 2001*, volume 2111 of *LNAI*, pages 616–629. Springer, 2001.

Leonid Peshkin and Christopher R. Shelton. Learning from scarce experience. In *Proc. 19th ICML*, pages 498–505, 2002.

Doina Precup, Richard S. Sutton, and Sanjoy Dasgupta. Eligibility traces for off-policy policy evaluation. In *Proc. 17th ICML*, pages 759–766, 2000.

Doina Precup, Richard S. Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *Proc. 18th ICML*, pages 417–424, 2001.

Teodor C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Comput.*, 9:401–424, 1991.

Teodor C. Przymusinski. Well-founded and stationary models of logic programs. *Ann. Mathematics and Artificial Intelligence*, 12(3):141–187, 1994.

Stéphane Ross and J. Andrew Bagnell. Agnostic system identification for model-based reinforcement learning. In *Proc. 29th ICML*, pages 1703–1710, 2012.

Dan Roth. Learning to reason: the non-monotonic case. In *Proc. 14th IJCAI*, volume 2, pages 1178–1184, 1995.

Matthew D. Schmill, Tim Oates, and Paul R. Cohen. Learning planning operators in real-world, partially observable environments. In *Proc. 5th AIPS*, pages 246–253, 2000.

Shai Shalev-Shwartz, Ohad Shamir, and Karthik Sridharan. Learning kernel based half-spaces with the 0-1 loss. *SIAM J. Comput.*, 40(6):1623–1646, 2011.

Christopher Shelton. Policy improvement for POMDPs using normalized importance sampling. In *Proc. 17th UAI*, pages 496–503, 2001.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

Umar Syed and Robert E. Schapire. A reduction from apprenticeship learning to classification. In *Proc. 23rd NIPS*, pages 2253–2261, 2010.

Prasad Tadepalli. Learning to solve problems from exercises. *Computational Intelligence*, 24(4):257–291, 2008.

Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence off-policy evaluation. In *Proc. 29th AAAI*, pages 3000–3006, 2015.

Ingo Thon, Bernd Gutmann, Martijn van Otterlo, Niels Landwehr, and Luc De Raedt. From non-deterministic to probabilistic planning with the help of statistical relational learning. In *ICAPS 2009 - Proc. Workshop on Planning and Learning*, pages 22–30, 2009.

Eiji Uchibe and Kenji Doya. Competitive-cooperative-concurrent reinforcement learning with importance sampling. In *Proc. Intl. Conf. Simulation of Adaptive Behavior*, pages 287–296, 2004.

Leslie G. Valiant. Rationality. In *Proc. 8th COLT*, pages 3–14, 1995.

Leslie G. Valiant. Robust logics. *Artificial Intelligence*, 117:231–253, 2000a.

Leslie G. Valiant. A neuroidal architecture for cognitive computation. *J. ACM*, 47(5): 854–882, 2000b.

Leslie G. Valiant. Knowledge infusion. In *Proc. 21st AAAI*, pages 1546–1551, 2006.

Allen van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.

Vladimr Vapnik and Alexei Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2): 264–280, 1971.

Thomas J. Walsh. *Efficient Learning of Relational Models for Sequential Decision Making*. PhD thesis, Rutgers University, 2010.

Paweł Wawrzyński. Real-time reinforcement learning by sequential actor-critics and experience replay. *Neural Networks*, 22:1484–1497, 2009.

Sungyook Yoon and Subbarao Kambhampati. Towards model-lite planning: A proposal for learning & planning with incomplete domain models. In *Proc. ICAPS Workshop on AI Planning and Learning*, 2007.

Håkan L. S. Younes and Michael L. Littman. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, Carnegie Mellon University, 2004.