Learning Planning Domains from Non-redundant Fully-Observed Traces: Theoretical Foundations and Complexity Analysis

Pascal Bachor[1], **Gregor Behnke**[2]

[1]Albert-Ludwigs-Universität Freiburg
[2]Universiteit van Amsterdam
bachorp@cs.uni-freiburg.de, g.behnke@uva.nl

Abstract

Domain learning is the task of finding an action model that can explain given observed plan executions, so-called traces. It allows us to automate the identification of actions' preconditions and effects instead of relying on hand-modeled expert knowledge. While previous research has put forth various techniques and covers multiple planning formalisms, the theoretical foundations of domain learning are still in their infancy.

We investigate the most basic setting, that is grounded classical planning without negative preconditions or conditional effects with full observability of the state variables. The given traces are assumed to be justified in the sense that either no single action or no set of actions can be removed without violating correctness of the plan. Furthermore, we might be given additional constraints in the form of a propositional logical formula. We show the consequences of these assumptions for the computational complexity of identifying a satisfactory planning domain.

Introduction

Automated planning is concerned with finding out how to act in a given environment by means of computation. A *domain*, or *action model*, specifies available actions along with the effects they have on the current world state and the conditions under which they can be applied. To be able to utilize planning algorithms, we require an action model. Modeling actions, however, can be difficult and expensive, as it usually requires the knowledge and labor of domain experts. This drawback, known as the *knowledge acquisition problem*, has been identified as a bottleneck for the successful application of planning systems (Wang 1995; Benson 1996; Kambhampati 2007; Gregory and Lindsay 2016).

Domain learning is concerned with automatically inferring the preconditions and effects of actions from a given set of observations. These observations usually take the form of *traces*, i.e. plans together with information about the state before and after each action execution (Jimenez et al. 2012; Cresswell and Gregory 2011; Arora et al. 2018). In this work, we consider only the fully-observable case in which traces contain all state information. If the traces are (partially) unobservable (see e.g. (Yang, Wu, and Jiang 2007; Aineto, Celorrio, and Onaindia 2019)), domain learning can be more complicated as we might have to infer the existence of certain unobserved state features.

In most cases, domain learning algorithms are provided only with positive examples, i.e. traces that are supposed to be valid in the domain to be learned. The possible inferences when requiring only the validity of the given traces are limited. Therefore, usually either further assumptions on the given input are made or there is some other bias toward certain models. In this work, we will presuppose that given traces are non-redundant (we say *justified*) in the sense that either no single action or no set of actions can be skipped without violating the validity of the trace. We will then analyze the problem of deciding whether there exists a domain in which the given traces are valid and justified. As far as we are aware, these particular assumptions have not yet been studied in the context of domain learning. Furthermore, an analysis of the problem's computational complexity (in this or any other formulation) has never been conducted. We aim at building a firm theoretical foundation that facilitates the formal definition and analysis of varied domain learning scenarios in the future.

Problem Definition

We adopt the popular classical planning formalism *STRIPS* (Fikes and Nilsson 1971). The environment *state* will be modeled using a set of *variables* (or *state variables*) that can either be contained (true) or missing (false) in a state. Actions are characterized by their *preconditions*, variables that must be true in–, *deletions*, variables that will be removed from–, and *additions*, variables that will be added to a state in which the action is executed. A *domain* is the definition of actions' preconditions and effects. Note that we do not allow negative preconditions or any other non-atomic preconditions or effects.

Definition 1. $\delta = (\mathrm{pre}, \mathrm{del}, \mathrm{add})$*, where* $\mathrm{pre} = \delta_{\mathbf{pre}}$*,* $\mathrm{del} = \delta_{\mathbf{del}}$*, and* $\mathrm{add} = \delta_{\mathbf{add}}$ *are relations between* <u>*actions*</u> *and* <u>*variables*</u>*, is a* <u>*domain*</u> *if* $(\mathrm{pre} \cup \mathrm{del}) \cap \mathrm{add} = \emptyset$*.*

We let $\mathcal{A}(\delta) := \{\, \mathtt{a} \mid \exists x.(\mathtt{a}, x) \in \delta_{\mathbf{add}} \,\}$ *and* $\mathcal{V}(\delta) := \{\, x \mid \exists \mathtt{a}.(\mathtt{a}, x) \in \delta_{\mathbf{add}} \,\}$*.*

A relation rel between actions and variables is a set of pairs $(\mathtt{a}, x)$ such that $\mathtt{a}$ is an action and x is a variable. We let $\mathrm{rel}(\mathtt{a}) := \{\, x \mid (\mathtt{a}, x) \in \mathrm{rel} \,\}$ and $\mathrm{rel}^{-1}(x) := \{\, \mathtt{a} \mid (\mathtt{a}, x) \in$

	a	b	c	a	g
x	●	●		●	●
y		●	●		
z			●	●	●

Figure 1: The trace $\tau = (\pi, \nu)$, where $\pi = (\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{a}, \mathtt{g})$ and $\nu = (\{x\}, \{x, y\}, \{y, z\}, \{x, z\}, \{x, z\})$.

rel$\}$. The condition $(\text{pre} \cup \text{del}) \cap \text{add} = \emptyset$ rules out contradictory or redundant effects. Namely, both deleting and adding a variable and adding a variable that is necessarily already present. This is particularly important in the context of domain learning because redundant effects can never be observed in a given trace.

Definition 2. *A state is a set of variables. The application of the action* a *to the state* ν *in the domain* $\delta = (\text{pre}, \text{del}, \text{add})$ *is*

$$\text{app}(\delta, \nu, \mathtt{a}) := \begin{cases} (\nu \setminus \text{del}(\mathtt{a})) \cup \text{add}(\mathtt{a}), & \text{pre}(\mathtt{a}) \subseteq \nu \\ \bot & \textit{otherwise}. \end{cases}$$

Planning

The planning problem is usually posed as follows. Given a domain, find a sequence of actions (a *plan*) that, when starting in a given *initial state*, yields a state that contains all variables in a given set of *goal variables*. Such plan is said to be *successful*.

In this work, instead of an initial state, we use an optional initialization action i that adds the appropriate variables; and instead of goal variables we use a goal action g that has the appropriate variables as preconditions. We will assume that when the first action of a plan is executed, all variables are false and that the final action of a successful plan is the goal action. Mocking initial state and goal variables using dedicated actions is by no means a weakening of the definition and reduces the technical overhead considerably, especially in the context of this paper. This technique has been used before, e.g. by Yang, Wu, and Jiang (2007).

Definition 3. *A plan* $\pi = (\pi_i)_i$ *is a sequence of actions. A trace* $\tau = (\pi, \nu)$ *is a plan* $\pi = \tau_{\boldsymbol{\pi}}$ *along with a sequence of states* $\nu = \tau_{\boldsymbol{\nu}}$ *such that* $|\tau| := |\pi| = |\nu|$.

We let $\mathcal{A}(\pi) := \bigcup_i \{\pi_i\}$ *and we let* $(\pi, \nu)^\frown(\pi', \nu') := (\pi^\frown\pi', \nu^\frown\nu')$.

Note that $(s_i)_i$ is shorthand for $(s_i)_{i=0}^{|s|-1}$ and $s_{-i} := s_{|s|-i}$, where $|s|$ denotes the length of the sequence s. $s^\frown t := (s_0, \dots, s_{-1}, t_0, \dots, t_{-1})$ denotes the concatenation of s and t. An example trace τ is shown in Figure 1. Our intended semantics for traces is that the action π_{i+1} is applied to the state ν_i with ν_{i+1} as the resulting state. In τ, for example, we see that applying the action c to the state $\{x, y\}$ results in the state $\{y, z\}$.

Definition 4. *The trace of the plan* π *in the domain* δ*, denoted* $T(\delta, \pi)$*, is the unique trace* (π, ν) *such that*

$$\nu_0 = \text{app}(\delta, \emptyset, \pi_0)$$
$$\nu_{i+1} = \text{app}(\delta, \nu_i, \pi_{i+1})$$

if it exists, else $\bot$*. The set of valid plans in the domain* δ *is*

$$\Pi(\delta) := \{\pi \mid T(\delta, \pi) \neq \bot\}.$$

We can now formally define the planning problem.

Definition 5 (Planning)**.** *Given a domain* δ *and a* goal action g*, find a plan* $\pi \in \Pi(\delta)$ *such that* $\pi_{-1} = \text{g}$.

The problem is PSPACE-complete in general and NP-complete when limited to plans of length at most k, where k is polynomial in the size of the input (Bylander 1994).

Plan Justification

We are usually interested in finding a plan that is not only successful in reaching the goal but also does so quickly and without detours. We will formally define two such additional qualities, namely well-justification and perfect justification. We adopt both notions from Fink and Yang (Fink 1992; Fink and Yang 1992, 1993).

Definition 6. $s = (s_i)_i$ *is a subsequence of* $t = (t_i)_i$ *if* $s = (t_{i_j})_j$ *for some* $(i_j)_j$ *with* $i_j < i_{j+1}$*.* s *is a proper subsequence of* t *if* s *is a subsequence of* t *and* $s \neq t$*.*

For example, $(1, 1)$ and $(2, 3, 1)$ are proper subsequences of $t = (1, 2, 3, 2, 1, 3)$, but $(3, 1, 2)$ is not a subsequence of t.

Definition 7.

(i) $\pi' \prec \pi$ *if* $\pi' = (\pi_0, \dots, \pi_{i-1}, \pi_{i+1}, \dots, \pi_{-1})$ *for some* $i < |\pi| - 1$.

(ii) $\pi' \lll \pi$ *if* π' *is a proper subsequence of* π *such that* $\pi'_{-1} = \pi_{-1}$.

A plan $\pi' \prec \pi$ contains all but a single action of π. A plan $\pi' \lll \pi$ contains all but arbitrarily many actions of π. In both cases the final action π_{-1} must not be missing.

Definition 8. *A plan* $\pi \in \Pi(\delta)$ *is well-justified (perfectly justified) in the domain* δ *if there exists no* $\pi' \in \Pi(\delta)$ *such that* $\pi' \prec \pi$ *(*$\pi' \lll \pi$*).*

Remark. *If* $\pi' \prec \pi$*, then* $\pi' \lll \pi$ *and therefore perfect justification implies well-justification.*

For a plan π, the number of plans $\pi' \prec \pi$ is at most linear in $|\pi|$, but the number of plans $\pi' \lll \pi$ can be exponential in $|\pi|$. In fact, we can efficiently test a plan for well-justification, but perfect justification cannot be checked efficiently unless P = NP (Fink and Yang 1992).

Lemma 1. *Given a plan* π *and a domain* δ*, deciding whether* π *is well-justified in* δ *is in* P. $\square$

Lemma 2. *Given a plan* π *and a domain* δ*, deciding whether* π *is perfectly justified in* δ *is* coNP-complete. $\square$

Domain Learning

The objective of domain learning is to find a domain that *explains* a given (set of) trace(s). Given a trace $\tau = (\pi, \nu)$, a domain δ explains (or *is consistent with*) τ if the state sequence ν results from executing π in δ. In our proofs, we restrict ourselves to learning from a single trace. We will, however, show that with regard to the computational complexity this does not weaken the results (Lemma 8).

Definition 9. *The set of domains that are consistent with the trace τ is $\Delta(\tau) := \{ \delta \mid T(\delta, \tau_{\boldsymbol{\pi}}) = \tau \}$. A trace τ is* <u>*consistent*</u> *if $\Delta(\tau) \neq \emptyset$.*

Based on the semantics of action application (Definition 4), we can specify necessary and sufficient conditions for a domain to be consistent with a trace. Namely, that (i) the preconditions of an action must hold in the state to which they are applied and (ii) deletions (additions) to a state in the trace must correspond to a delete (add) effect of the appropriate action and vice versa. These properties are formulated in the following alternative definition of consistency.

Lemma 3. $(\mathrm{pre}, \mathrm{del}, \mathrm{add}) \in \Delta((\pi, \nu))$ *if and only if*

$$\mathrm{pre}(\pi_0) \quad = \emptyset \tag{1}$$

$$\mathrm{del}(\pi_0) \quad \subseteq \nu_0^{\complement} \tag{2}$$

$$\mathrm{add}(\pi_0) \quad = \nu_0 \tag{3}$$

and

$$\mathrm{pre}(\pi_{i+1}) \subseteq \nu_i \tag{4}$$

$$\nu_i \setminus \nu_{i+1} \subseteq \mathrm{del}(\pi_{i+1}) \subseteq \nu_{i+1}^{\complement} \tag{5}$$

$$\nu_{i+1} \setminus \nu_i \subseteq \mathrm{add}(\pi_{i+1}) \subseteq \nu_{i+1}, \tag{6}$$

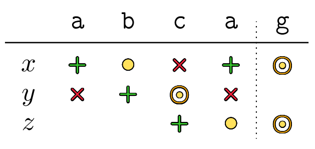
where $X^{\complement}$ denotes the complement of X, i.e. $X \subseteq Y^{\complement} \iff X \cap Y = \emptyset$. □

As an example, let us consider our trace as shown in Figure 2. From the trace we can infer that $\mathrm{add}(\mathtt{a}) = \{ x \}$ and $\mathrm{del}(\mathtt{a}) = \{ y \}$. This follows from the fact that the action a, when it first occurs, can be observed to add exactly x and, when it occurs for the second time, can be observed to delete y and not delete z. This corresponds to Propositions (3) and (5) of Lemma 3. Similarly, we can infer that $y \in \mathrm{add}(\mathtt{b})$, $x \in \mathrm{del}(\mathtt{c})$, and $z \in \mathrm{add}(\mathtt{c})$. Whether or not, for example, b deletes z or c adds y can, however, not be observed. Regarding preconditions, we only have the upper bounds from Propositions (1) and (4) and cannot make any positive inferences. Under the assumption that the trace is well-justified, however, we can infer that $\mathrm{pre}(\mathtt{g}) = \{ x, z \}$. This follows from the fact that if $x \notin \mathrm{pre}(\mathtt{g})$, $(\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{g}) \prec \pi$ is a valid plan and if $z \notin \mathrm{pre}(\mathtt{g})$, $(\mathtt{a}, \mathtt{b}, \mathtt{a}, \mathtt{g}) \prec \pi$ is a valid plan. Furthermore, we can infer that y must be a precondition of c and x must be a precondition of b or c.

Therefore, if we want to learn more than is implied simply by the semantics of action application, in particular any preconditions, we might want to learn from justified traces. We will define the domain learning problem accordingly.

Definition 10 (Domain Learning)**.** *Given a trace τ, find a domain $\delta \in \Delta(\tau)$ in which τ (that means $\tau_{\boldsymbol{\pi}}$) is well-justified (perfectly justified).*

Lin and Bercher (2021) study the complexity of the related problem *domain modification*. There, we are given an almost correct domain δ and a plan π that is not consistent with δ, i.e. $\pi \notin \Pi(\delta)$. We are then searching for a domain δ' consistent with π having as few differences to δ as possible. They do not consider justification.



	a	b	c	a	g
x	+	●	×	+	◎
y	×	+	◎	×	
z			+	●	◎

Figure 2: Example trace τ with observed additions + and deletions × and preconditions ◎ necessary for the justification of the trace. If additionally $x \in \mathrm{pre}(\mathtt{b}) \cup \mathrm{pre}(\mathtt{c})$, the trace is both well-justified and perfectly justified.

Preliminary Results

Next, we introduce a few constructions that are essential to the subsequent proofs. First, we define a notion of relative restrictiveness for domains that will allow us to discard most candidate domains when searching for a domain in which a given trace is justified. Second, we define an encoding that allows us to enforce preconditions and effects in a trace. Third, we define a trace combination procedure that allows us to work with multiple traces instead of a single one.

The Domain Restrictiveness Order

We introduce the order relation $\leq$ for domains. If $\delta \leq \delta'$, we say δ is *more restrictive than* δ'.

Definition 11. *Let δ and δ' be domains. We let $\delta \leq \delta'$ if $\delta_{\mathbf{pre}} \supseteq \delta'_{\mathbf{pre}}$, $\delta_{\mathbf{del}} \supseteq \delta'_{\mathbf{del}}$, and $\delta_{\mathbf{add}} \subseteq \delta'_{\mathbf{add}}$. For a given trace τ we let $\leq_\tau := (\leq, \Delta(\tau))$.*

Note that $\delta \leq \delta'$ holds iff δ has at least the preconditions and delete effects– and at most the add effects of δ'.

Lemma 4. *Both $\leq$ and $\leq_\tau$ form a lattice.*

A *lattice* is a relation that is reflexive, antisymmetric, and transitive (i.e. a *partial order*) such that any two elements have a unique infimum and a unique supremum.

Proof. Because $\subseteq$ ($\supseteq$) forms a lattice with infimum $\cap$ ($\cup$) and supremum $\cup$ ($\cap$) and $\leq$ is isomorphic to the product lattice $\supseteq \times \supseteq \times \subseteq$, $\leq$ forms a lattice with

$$\inf\{ \delta, \delta' \} = (\delta_{\mathbf{pre}} \cup \delta'_{\mathbf{pre}}, \delta_{\mathbf{del}} \cup \delta'_{\mathbf{del}}, \delta_{\mathbf{add}} \cap \delta'_{\mathbf{add}})$$

$$\sup\{ \delta, \delta' \} = (\delta_{\mathbf{pre}} \cap \delta'_{\mathbf{pre}}, \delta_{\mathbf{del}} \cap \delta'_{\mathbf{del}}, \delta_{\mathbf{add}} \cup \delta'_{\mathbf{add}}).$$

Note that $\delta \in \Delta(\tau)$ if and only if a set of equations of the form $X \subseteq v \subseteq Y$, where v is a set of state variables of δ, is fulfilled (Lemma 3). Furthermore,

$$X \subseteq v, v' \subseteq Y \implies X \subseteq v \cap v' \subseteq v \cup v' \subseteq Y$$

and therefore

$$\delta, \delta' \in \Delta(\tau) \implies \inf\{ \delta, \delta' \}, \sup\{ \delta, \delta' \} \in \Delta(\tau). \quad \square$$

The following result reveals the strength of the lattice structure of the domain space: the more restrictive a domain, the fewer plans it admits.

Lemma 5. *If $\delta \leq \delta'$, then $\Pi(\delta) \subseteq \Pi(\delta')$.*

Proof. We show that $\pi \in \Pi(\delta)$ implies $\pi \in \Pi(\delta')$ via induction over the length ℓ of the plan π.

Induction hypothesis:

(i) $(\pi_0, \ldots, \pi_{\ell-1}) \in \Pi(\delta')$
(ii) $\nu'_\ell \supseteq \nu_\ell$, where $\nu = T(\delta, \pi)_{\boldsymbol{\nu}}$ and $\nu' = T(\delta', \pi)_{\boldsymbol{\nu}}$

	a	b	c	a	g
x	+	◎	⊗	+	◎
y	×	+	◎	×	×
z		×	+	○	◎

Figure 3: τ in $\min \Delta(\tau)$.

$\underline{\ell = 0}$:

$$\begin{aligned} &(\,) \in \Pi(\delta') \\ &\nu'_0 = \delta'_{\mathbf{add}}(\pi_0) \supseteq \delta_{\mathbf{add}}(\pi_0) = \nu_0 \end{aligned}$$

$\underline{\ell = i+1}$:

$$\begin{aligned} \nu'_{i+1} &= (\nu'_i \setminus \delta'_{\mathbf{del}}(\pi_{i+1})) \cup \delta'_{\mathbf{add}}(\pi_{i+1}) \\ &\supseteq (\nu_i \setminus \delta_{\mathbf{del}}(\pi_{i+1})) \cup \delta_{\mathbf{add}}(\pi_{i+1}) \\ &= \nu_{i+1} \supseteq \delta_{\mathbf{pre}}(\pi_{i+1}) \supseteq \delta'_{\mathbf{pre}}(\pi_{i+1}) \end{aligned}$$

From $(\pi_0, \ldots, \pi_{i-1}) \in \Pi(\delta')$ and $\delta'_{\mathbf{pre}}(\pi_{i+1}) \subseteq \nu'_{i+1}$ follows $(\pi_0, \ldots, \pi_i) \in \Pi(\delta')$. □

The fewer plans a domain admits, the fewer counterexamples to the well- or perfect justification there can be. We can take advantage of this fact by considering only such domains that restrict the valid plans as much as possible.

Definition 12. *Let Δ be a set of domains. $\delta \in \Delta$ is a most restrictive domain of Δ, denoted $\delta \in \underline{\min \Delta}$, if δ is $\leq$-minimal in Δ.*

For convenience, we let $\min \Delta(\tau)$ denote *the* most restrictive domain of $\Delta(\tau)$ that does not use any actions or variables that do not occur in τ. If τ is consistent, $\min \Delta(\tau)$ exists and can be constructed efficiently.

We sketch the most restrictive domain of our example trace in Figure 3. Consult Figure 4 for an overview of the symbols we use in such diagrams throughout this paper.

Lemma 6. *If π is well-justified (perfectly justified) in any $\delta \in \Delta(\tau)$, then π is well-justified (perfectly justified) in $\min \Delta(\tau)$.*

Proof. $\Pi(\min \Delta(\tau)) \subseteq \Pi(\delta)$ (Lemma 5) and therefore any counterexample to the well-justification (perfect justification) of π in $\min \Delta(\tau)$ is a counterexample to the well-justification (perfect justification) of π in δ. □

Note that the notion of *safe action model* from Stern and Juba corresponds to our notion of relative restrictiveness $\leq_\tau$ and their $F(\Pi_\mathcal{T})$ or M_{SGAM} corresponds to our most restrictive domain in an SAS$^+$ setting (Stern and Juba 2017; Juba, Le, and Stern 2021).

Encoding a Domain in a Trace

We show that for a given domain δ we can construct a trace $\mathtt{t}(\delta)$ such that any action $\mathtt{a}$ adds (deletes) a variable x in δ if and only if $\mathtt{a}$ adds (deletes) x in any domain consistent with $\mathtt{t}(\delta)$. Furthermore, if x is not a precondition of $\mathtt{a}$ in δ, it won't be a precondition of $\mathtt{a}$ in any domain consistent with $\mathtt{t}(\delta)$. Note that the presence of preconditions, opposed to the their absence, cannot be observed and therefore cannot be

	The variable is			
	present	*a precondition*	*deleted*	*added*
○	yes	no	no	no
◎	yes	yes	no	no
×	no	no	yes	no
⊗	no	yes	yes	no
+	yes	no	no	yes

Figure 4: Legend.

	$\mathtt{p_a}$	$\mathtt{a}$	$\mathtt{q_a}$	$\mathtt{a}$
$\mathcal{V}(\delta) \setminus (\mathrm{pre}(\mathtt{a}) \cup \mathrm{del}(\mathtt{a}) \cup \mathrm{add}(\mathtt{a}))$			●	○
$\mathrm{add}(\mathtt{a})$		+		+
$\mathrm{pre}(\mathtt{a}) \setminus \mathrm{del}(\mathtt{a})$	●	◎	●	◎
$\mathrm{del}(\mathtt{a}) \setminus \mathrm{pre}(\mathtt{a})$		×	●	×
$\mathrm{pre}(\mathtt{a}) \cap \mathrm{del}(\mathtt{a})$	●	⊗	●	⊗
$p_\mathtt{a}$	+	○	⊗	
$q_\mathtt{a}$			+	○

Figure 5: The trace $\tau^\mathtt{a}$ as in the proof of Lemma 7. We depict one representative for each of the five possible classes of variables of δ as well as the auxiliary variables $p_\mathtt{a}$ and $q_\mathtt{a}$.

encoded in a trace. We can, however, enforce them at least for the most restrictive domain in which there are as many preconditions as possible.

Lemma 7. *For any domain δ there exists a trace $\underline{\mathtt{t}(\delta)}$ such that for any $\mathtt{a} \in \mathcal{A}(\delta)$, $x \in \mathcal{V}(\delta)$, and $\delta' \in \Delta(\mathtt{t}(\delta))$ it holds*

(i) $(\mathtt{a}, x) \in \delta'_{\mathbf{del}} \iff (\mathtt{a}, x) \in \delta_{\mathbf{del}}$
$(\mathtt{a}, x) \in \delta'_{\mathbf{add}} \iff (\mathtt{a}, x) \in \delta_{\mathbf{add}}$

(ii) $(\mathtt{a}, x) \notin \delta_{\mathbf{pre}} \implies (\mathtt{a}, x) \notin \delta'_{\mathbf{pre}}$
$(\mathtt{a}, x) \in \delta_{\mathbf{pre}} \implies (\mathtt{a}, x) \in \min \Delta(\mathtt{t}(\delta))_{\mathbf{pre}}$

(iii) $\mathtt{t}(\delta)$ *is perfectly justified in* $\min \Delta(\mathtt{t}(\delta))$.

Proof. Besides the actions and variables from δ, $\mathtt{t}(\delta)$ uses the action $\mathtt{g}$, as well as for each $\mathtt{a} \in \mathcal{A}(\delta)$ the actions $\mathtt{p_a}$ and $\mathtt{q_a}$ and variables $p_\mathtt{a}$ and $q_\mathtt{a}$. The construction is sketched in Figure 5. Let $\mathtt{t}(\delta) = \left(\frown_{\mathtt{a} \in \mathcal{A}(\delta)} \tau^\mathtt{a}\right) \frown ((\mathtt{g}, \emptyset))$, where

$$\begin{aligned} \tau^\mathtt{a} &= (\pi^\mathtt{a}, \nu^\mathtt{a}) = ((\mathtt{p_a}, \mathtt{a}, \mathtt{q_a}, \mathtt{a}), (\nu^\mathtt{a}_0, \nu^\mathtt{a}_1, \nu^\mathtt{a}_2, \nu^\mathtt{a}_3)) \\ \nu^\mathtt{a}_0 &= \delta_{\mathbf{pre}}(\mathtt{a}) \cup \{ p_\mathtt{a} \} \\ \nu^\mathtt{a}_1 &= \mathrm{app}(\delta, \mathtt{a}, \nu^\mathtt{a}_0) \\ \nu^\mathtt{a}_2 &= \delta_{\mathbf{pre}}(\mathtt{a}) \cup (\mathcal{V}(\delta) \setminus \delta_{\mathbf{add}}(\mathtt{a})) \cup \{ q_\mathtt{a} \} \\ \nu^\mathtt{a}_3 &= \mathrm{app}(\delta, \mathtt{a}, \nu^\mathtt{a}_2). \end{aligned}$$

Let $\delta' \in \Delta(\mathtt{t}(\delta))$, $\mathtt{a} \in \mathcal{A}(\delta)$, and $x \in \mathcal{V}(\delta)$.

(i) If x is a deletion of $\mathtt{a}$ in δ, then $x \in \nu^\mathtt{a}_2 \setminus \nu^\mathtt{a}_3$ witnesses that x is a deletion of $\mathtt{a}$ in δ'. If x is not a deletion of $\mathtt{a}$ in δ, then $x \in \nu^\mathtt{a}_3$ witnesses that x is not a deletion of $\mathtt{a}$ in δ'. If x is an addition of $\mathtt{a}$ in δ, then $x \in \nu^\mathtt{a}_1 \setminus \nu^\mathtt{a}_0$ witnesses that x is an addition of $\mathtt{a}$ in δ'. If x is not an addition of $\mathtt{a}$ in δ, then $x \notin \nu^\mathtt{a}_1$ witnesses that x is not an addition of $\mathtt{a}$ in δ'. These statements correspond to propositions (5) and (6) of Lemma 3.

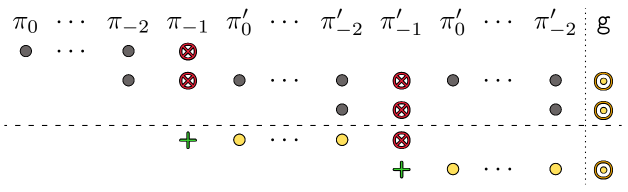

π_0 ⋯ π_{-2} π_{-1} π'_0 ⋯ π'_{-2} π'_{-1} π'_0 ⋯ π'_{-2} g

Figure 6: Sketch of the combination of two traces as in Lemma 8. The lower part depicts the variables x and x'.

(ii) If x is not a precondition of $\mathtt{a}$ in δ, then $x \notin \nu_0^{\mathtt{a}}$ witnesses that x is not a precondition of $\mathtt{a}$ in δ'. This corresponds to proposition (4) of Lemma 3. If x is a precondition of $\mathtt{a}$ in δ, then $x \in \nu_0^{\mathtt{a}}, \nu_2^{\mathtt{a}}$ such that x is always present when $\mathtt{a}$ is executed and therefore x is a precondition of $\mathtt{a}$ in $\min \Delta(\mathtt{t}(\delta))$ by $\leq$-minimality.

(iii) Let $\pi' \lll \mathtt{t}(\delta)_{\boldsymbol{\pi}}$ and let $(a_i)_i$ be the enumeration of $\mathcal{A}(\delta)$. Note that no a_i modifies a p_{a_j} or q_{a_j}. In $\min \Delta(\mathtt{t}(\delta))$, g requires the variable $q_{a_{-1}}$ and therefore $q_{a_{-1}} \in \mathcal{A}(\pi')$. Similarly, $\mathtt{q}_{a_i}$ requires the execution of $\mathtt{p}_{a_i}$ and $\mathtt{p}_{a_{i+1}}$ requires the execution of q_{a_i}. It follows $\mathtt{p}_{a_i}, \mathtt{q}_{a_i} \in \mathcal{A}(\pi')$ for all i. Furthermore, $\mathrm{pre}(\mathrm{g}) \supseteq \mathrm{add}(a_{-1})$, $\mathrm{pre}(\mathtt{q}_{a_i}) \supseteq \mathrm{add}(a_i)$, and $\mathrm{pre}(\mathtt{p}_{a_{i+1}}) \supseteq \mathrm{add}(a_i)$. It follows $\pi' = \mathtt{t}(\delta)_{\boldsymbol{\pi}}$. □

Trace Combination

As the final preliminary result, we show that, regarding the computational complexity, there is no difference between learning from a single trace and learning from multiple traces. For given traces τ and τ' we construct a trace $\tau \wedge \tau'$ that combines τ and τ' in the sense that $\tau \wedge \tau'$ is justified if and only if both τ and τ' are justified.

Lemma 8. *For any two traces τ, τ' there exists a trace $\underline{\tau \wedge \tau'}$ such that the following are equivalent.*

(i) There exists a domain $\delta \in \Delta(\tau \wedge \tau')$ in which $\tau \wedge \tau'$ is well-justified (perfectly justified).

(ii) There exists a domain $\delta \in \Delta(\tau) \cap \Delta(\tau')$ in which both τ and τ' are well-justified (perfectly justified).

Proof. Let $\tau = (\pi, \nu)$ and $\tau' = (\pi', \nu')$ and w.l.o.g. let $\pi_{-1} \neq \pi'_{-1}$. Besides the actions and variables from τ and τ', $\tau \wedge \tau'$ uses the action g and the variables x and x'.

$\tau \wedge \tau'$ is constructed from (1) a copy of τ except that the last state is replaced by $\{ x \}$, followed by (2) a copy of τ' except that the last state is replaced by $\{ x' \}$ and all other states additionally contain the variable x, followed by (3) another copy of τ' except that the last action is replaced by g and all other states additionally contain the variable x'. The construction is shown in Figure 6.

Utilizing Lemma 6, we can restrict ourselves to the domains $\min \Delta(\tau \wedge \tau')$ and $\min(\Delta(\tau) \cap \Delta(\tau'))$.

$\boldsymbol{(i) \rightarrow (ii)}$: A counterexample to the well-justification (perfect justification) of τ or τ' induces a counterexample to the well-justification (perfect justification) of $\tau \wedge \tau'$ as we can simply leave out the same actions in $\tau \wedge \tau'$.

$\boldsymbol{(ii) \rightarrow (i)}$: Note that $\tau \wedge \tau'$ is constructed in a way that ensures that no actions other than π_{-1} and π'_{-1} can modify x or x'. Let π'' be a counterexample to the well-justification (perfect justification) of $\tau \wedge \tau'$. π'' must contain π'_{-1} as $x' \in \mathrm{pre}(\mathrm{g})$ and $\mathrm{add}^{-1}(x') = \{ \pi'_{-1} \}$. Then, it must also contain π_{-1} as $x \in \mathrm{pre}(\pi'_{-1})$ and $\mathrm{add}^{-1}(x) = \{ \pi_{-1} \}$. Now the first, the second, or the third component, as delimited by π_{-1}, π'_{-1}, and g, of π'' induces a counterexample to the well-justification (perfect justification) of τ or τ'. □

With the operation $\wedge$ we can combine an arbitrary number of traces. And, if we use a simple heuristic, we can avoid an exponential blowup of the constructed trace. A consequence of this is that domain learning with a single trace is polynomially equivalent to domain learning with multiple traces.

Definition 13 (Domain Learning – Multiple Traces)**.** *Given a set of traces $T = \{ \tau_i \}_i$, find a domain $\delta \in \bigcap_i \Delta(\tau_i)$ such that all τ_i are well-justified (perfectly justified) in δ.*

Lemma 9. *Given a set of traces T, we can construct $\bigwedge T$ in polynomial time.*

Proof. The trick is to duplicate the smaller of two traces when combining. The number of additional variables is linear in $|T|$ and the construction requires no expensive operations. We show that the length of $\bigwedge T = \bigwedge_i \tau_i$ is at most twice the sum of the lengths of the τ_i using induction.

$$\left| \bigwedge_{i=1}^{k+1} \tau_i \right| = \left| \bigwedge_{i=1}^{k} \tau_i \right| + |\tau_{k+1}| + \min\{ \left| \bigwedge_{i=1}^{k} \tau_i \right|, |\tau_{k+1}| \}$$
$$\leq 2 \sum_{i=1}^{k} |\tau_i| + 2 \cdot |\tau_{k+1}| = 2 \sum_{i=1}^{k+1} |\tau_i| \,. \qquad \square$$

Computational Complexity

Well-equipped with the results from the previous section, we can now work out the main results of this paper. We will show that the complexity of domain learning is P for well-justification and coNP-C for perfect justification and therefore equivalent to that of checking plan justification (Lemmata 1 and 2). We also show that imposing additional constraints in the form of a propositional logical formula on the domain to be found increases the complexity to $\mathrm{NP}^{\mathfrak{C}}$, where $\mathfrak{C}$ is the complexity of the basic domain learning problem.

Theorem 1. *Given a trace τ, deciding whether there exists a domain $\delta \in \Delta(\tau)$ in which τ is well-justified is in* P.

Proof. It suffices to check whether τ is well-justified in $\min \Delta(\tau)$ (Lemma 6). We can iterate over all potential counterexamples $\pi' \prec \tau_{\boldsymbol{\pi}}$ and check if $\pi' \in \Pi(\min \Delta(\tau))$ in polynomial time. □

For the next result, we adapt a construction originally introduced by Fink and Yang (1992).

Theorem 2. *Given a trace τ, deciding whether there exists a domain in which τ is perfectly justified is* coNP-complete.

Proof. Membership: It suffices to check whether τ is perfectly justified in $\min \Delta(\tau)$ (Lemma 6). We can guess a potential counterexample $\pi' \lll \tau_{\boldsymbol{\pi}}$ and check whether $\pi' \in \Pi(\min \Delta(\tau))$ in polynomial time.

Hardness: Given a propositional formula $\varphi(\overline{y})$ in conjunctive normal form, we construct a domain δ and a plan π such that φ is satisfiable if and only if π is not perfectly justified. Then there exists a domain in which $\mathtt{t}(\delta) \wedge T(\delta, \pi)$ is perfectly justified if and only if φ is unsatisfiable (Lemmata 7 and 8).

For constructing δ and π, we use the actions $\mathtt{i}$, $\mathtt{r}$, and $\mathtt{g}$ and state variables i^*, and g. For every propositional variable number j we use the action $\mathtt{y}_j$ and the state variables $y_j^\perp$ and $y_j^\top$. For every literal number d in clause number k, we use the actions $\mathtt{l}_{k,d}$ and the variables c_k and $l_{k,d}$. Let

$$\varphi = \bigwedge_k \bigvee_d \ell_{k,d}, \text{ where } \ell_{k,d} \in \{\, y_j, \neg y_j \,\}_j$$

$$\pi = (\mathtt{i})^\frown \left(\bigfrown_j (\mathtt{y}_j) \right)^\frown (\mathtt{r})^\frown \left(\bigfrown_k \bigfrown_d (\mathtt{l}_{k,d}) \right)^\frown (\mathtt{g})$$

and let δ be such that

$\mathrm{pre}(\mathtt{i}) = \emptyset$	$\mathrm{del}(\mathtt{i}) = \emptyset$	(below)
$\mathrm{pre}(\mathtt{y}_j) = \emptyset$	$\mathrm{del}(\mathtt{y}_j) = \{\, y_j^\perp \,\}$	$\mathrm{add}(\mathtt{y}_j) = \{\, y_j^\top \,\}$
$\mathrm{pre}(\mathtt{r}) = \{\, y_j^\top \,\}_j$	$\mathrm{del}(\mathtt{r}) = \{\, l_{k,d} \,\}_{k,d}$	$\mathrm{add}(\mathtt{r}) = \{\, y_j^\perp \,\}_j$
(below)	$\mathrm{del}(\mathtt{l}_{k,d}) = \emptyset$	$\mathrm{add}(\mathtt{l}_{k,d}) = \{\, l_{k,d}, c_k \,\}$
(below)	$\mathrm{del}(\mathtt{g}) = \emptyset$	$\mathrm{add}(\mathtt{g}) = \emptyset$

and

$$\mathrm{add}(\mathtt{i}) = \{\, i^* \,\} \cup \{\, l_{k,d} \,\}_{k,d} \cup \{\, y_j^\perp \,\}_j$$

$$\mathrm{pre}(\mathtt{l}_{k,d}) = \begin{cases} \{\, y_j^\top \,\}, & \ell_{k,d} = y_j \\ \{\, y_j^\perp \,\}, & \ell_{k,d} = \neg y_j \end{cases}$$

$$\mathrm{pre}(\mathtt{g}) = \{\, i^* \,\} \cup \{\, l_{k,d} \,\}_{k,d} \cup \{\, c_k \,\}_k.$$

Claim: If φ is satisfiable, then π is not perfectly justified in δ.

Let $\gamma : \{\, y_j \,\}_j \to \{\, \perp, \top \,\}$ be such that $\gamma \models \varphi$ and let

$$Y = \{\, (\mathtt{y}_j) \mid \gamma(y_j) = \top \,\}_j$$

$$L = \{\, (\mathtt{l}_{k,d}) \mid \gamma \models \ell_{k,d} \,\}_{k,d}$$

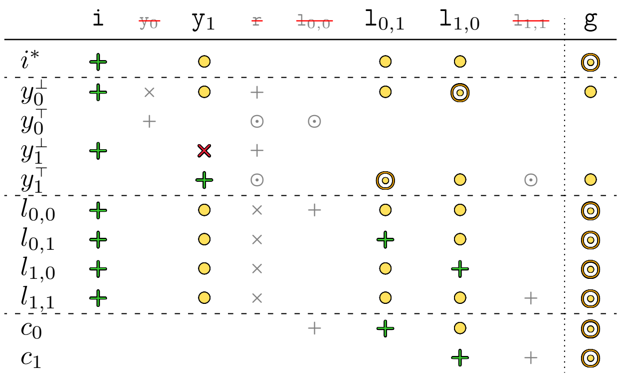
$$\pi' = (\mathtt{i})^\frown (\bigfrown Y)^\frown (\bigfrown L)^\frown (\mathtt{g}).$$

For each clause number k there is a d such that $\gamma \models \ell_{k,d}$ and therefore $(\mathtt{l}_{k,d}) \in L$. Let j be such that $\ell_{k,d} \in \{\, y_j, \neg y_j \,\}$. It holds $\ell_{k,d} = y_j \iff (\mathtt{y}_j) \in Y$ such that $\mathtt{l}_{k,d}$ is executable as it occurs in π'. Therefore, $\mathtt{g}$ is also executable as it occurs in π' and π' is a counterexample to the perfect justification of π in δ.

Claim: If π is not perfectly justified in δ, then φ is satisfiable.

Let $\pi' \lll \pi$ be such that $\pi' \in \Pi(\delta)$. We start by showing that $\mathtt{i} \in \mathcal{A}(\pi')$ and $\mathtt{r} \notin \mathcal{A}(\pi')$. First, $\mathtt{i} \in \mathcal{A}(\pi')$ follows from $i^* \in \mathrm{pre}(\mathtt{g})$ and $\mathrm{add}^{-1}(i^*) = \{\, \mathtt{i} \,\}$. Second, suppose π' contained $\mathtt{r}$. Then π' also contains all $\mathtt{y}_j$ as $y_j^\top \in \mathrm{pre}(\mathtt{r})$ and $\mathrm{add}^{-1}(y_j^\top) = \{\, \mathtt{y}_j \,\}$. Furthermore, π' contains all $\mathtt{l}_{k,d}$ as $l_{k,d} \in \mathrm{pre}(\mathtt{g})$, $\mathrm{add}^{-1}(l_{k,d}) = \{\, \mathtt{i}, \mathtt{l}_{k,d} \,\}$ and $l_{k,d} \in \mathrm{del}(\mathtt{r})$. It follows $\pi' = \pi$, which contradicts $\pi' \lll \pi$.

From $\mathtt{i} \in \mathcal{A}(\pi')$ and $\mathtt{r} \notin \mathcal{A}(\pi')$ follows that $\pi' = (\mathtt{i})^\frown (\bigfrown Y)^\frown (\bigfrown L)^\frown (\mathtt{g})$ for some $Y \subseteq \{\, (\mathtt{y}_j) \,\}_j$ and $L \subseteq \{\, (\mathtt{l}_{k,d}) \,\}_{k,d}$.



	$\mathtt{i}$	~~$\mathtt{y}_0$~~	$\mathtt{y}_1$	~~$\mathtt{r}$~~	~~$\mathtt{l}_{0,0}$~~	$\mathtt{l}_{0,1}$	$\mathtt{l}_{1,0}$	~~$\mathtt{l}_{1,1}$~~	$\mathtt{g}$
i^*	✚		●			●	●		◎
$y_0^\perp$	✚	×	●	+		●	◎		●
$y_0^\top$		+		⊙	⊙				
$y_1^\perp$	✚		✗	+					
$y_1^\top$			✚	⊙		◎	●	⊙	●
$l_{0,0}$	✚		●	×	+	●	●		◎
$l_{0,1}$	✚		●	×		✚	●		◎
$l_{1,0}$	✚		●	×		●	✚		◎
$l_{1,1}$	✚		●	×		●	●	+	◎
c_0					+	✚	●		◎
c_1							✚	+	◎

Figure 7: Counterexample to the perfect justification of $T(\delta, \pi)$ corresponding to $\varphi = (y_0 \vee y_1) \wedge (\neg y_0 \vee y_1)$ as in the proof of Theorem 2.

We now let $\gamma : \{\, y_j \,\}_j \to \{\, \perp, \top \,\}$ be such that

$$\gamma(y_j) = \begin{cases} \top, & (\mathtt{y}_j) \in Y \\ \perp & \text{otherwise.} \end{cases}$$

For each clause number k, because $c_k \in \mathrm{pre}(\mathtt{g})$, there is a d such that $(\mathtt{l}_{k,d}) \in L$. Let j be such that $\ell_{k,d} \in \{\, y_j, \neg y_j \,\}$. It holds $(\mathtt{y}_j) \in Y \iff \ell_{k,d} = y_j \iff \gamma(y_j) = \top$ and therefore $\gamma \models \ell_{k,d}$. It follows $\gamma \models \varphi$. $\square$

Domain Learning with Expert Knowledge

Realistically, a domain learning process does not rely solely on the provided traces. The learner might also have access to knowledge that does not come from given traces but from a domain expert. We want to investigate how such additional knowledge can affect the complexity of domain learning.

In order to capture potentially complex constraints, we utilize propositional logical formulas. The atoms of these formulas shall be of the form $x \in \mathrm{rel}(\mathtt{a})$, where x is a state variable, $\mathtt{a}$ is an action, and $\mathrm{rel} \in \{\, \mathrm{pre}, \mathrm{del}, \mathrm{add} \,\}$. We say $\delta \models \psi$ if the formula ψ is valid in the domain δ. This way, we can, for example, model the expert knowledge that the action $\mathtt{a}$ has either x or y as a precondition: $(x \in \mathrm{pre}(\mathtt{a}) \vee y \in \mathrm{pre}(\mathtt{a})) \wedge \neg(x \in \mathrm{pre}(\mathtt{a}) \wedge y \in \mathrm{pre}(\mathtt{a}))$.

Domain learning with expert knowledge is related to both domain modification (Lin and Bercher 2021) and *model reconciliation* (Sreedharan, Bercher, and Kambhampati 2022), which has the same setting as domain modification but additionally requires that the given plan is optimal in the modified domain. The constraint that the domain to be found can have at most k modifications from the base domain can also be expressed as a propositional logical formula. k-bounded model reconciliation is Σ_2^{P}-complete; their results are, however, orthogonal to ours as we assume full observability, while in their setting only the initial and the goal state can be observed.

First of all, we see that constraining the domain to satisfy a monomial, i.e. a conjunction of atoms and negated atoms, does not change the computational complexity.

Lemma 10. *Given a trace τ and a monomial ψ, deciding whether there exists a domain $\delta \models \psi$ in which τ is well-justified (perfectly justified) is in* P *(*coNP-complete*).*

Proof. We can construct a $\leq$-minimal domain that is consistent with both ψ and τ in polynomial time (if it exists). It suffices to check well-justification (perfect justification) in such domain. Hardness is implied by Theorems 1 and 2. $\square$

When allowing arbitrary formulas, however, the complexity increases. In fact, we will show that the resulting complexity is $\mathrm{NP}^{\mathfrak{C}}$, where $\mathfrak{C}$ is the complexity of the basic domain learning problem.

Theorem 3. *Given a trace τ and a formula ψ, deciding whether there exists a domain $\delta \models \psi$ in which τ is well-justified is* NP-complete.

Proof. Membership: Guess an interpretation of ψ and apply Lemma 10.

Hardness: Let $\tau = ((\,), (\,))$. This trace is well-justified in any domain. We can transform the atoms x_i of a given propositional logical formula φ, into atoms of the form $x_i \in \mathrm{add}(\mathtt{a})$. Then the resulting formula ψ is satisfiable if and only if φ is satisfiable and if and only if there exists a domain $\delta \models \psi$ (in which τ is well-justified). $\square$

Theorem 4. *Given a trace τ and a formula ψ, deciding whether there exists a domain $\delta \models \psi$ in which τ is perfectly justified is Σ_2^{P}-complete.*

Proof. Membership: As in the proof of Theorem 3.

Hardness: Given a propositional formula $\varphi(\overline{x}; \overline{y})$ in conjunctive normal form, we construct a formula ψ and a trace τ such that there exists an assignment $\chi : \{\, x_i \,\}_i \to \{\, \bot, \top \,\}$ such that $\varphi[\chi]$ is unsatisfiable if and only if τ is perfectly justified in some $\delta \models \psi$.

We use the actions and variables as in the proof of Theorem 2 and additionally the action $\mathtt{x}$ and the variable x^* as well as for every x-variable number i the variables $x_i^\bot$ and $x_i^\top$. Let

$$\varphi = \bigwedge_k \bigvee_d \ell_{k,d} \text{ with } \ell_{k,d} \in \{\, y_j, \neg y_j \,\}_j \cup \{\, x_i, \neg x_i \,\}_i$$

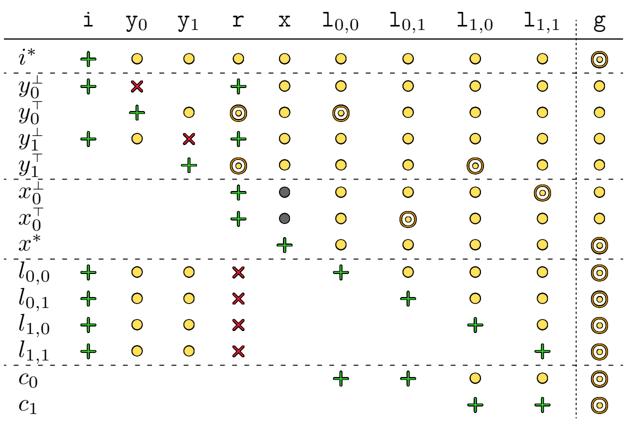
$$\pi = (\mathtt{i})^\frown \left(\underset{j}{\frown} (\mathtt{y}_j) \right)^\frown (\mathtt{r}, \mathtt{x})^\frown \left(\underset{k}{\frown} \underset{d}{\frown} (\mathtt{l}_{k,d}) \right)^\frown (\mathtt{g}).$$

Let our domain be as in the proof of Theorem 2 except that $\mathtt{r}$ now also adds the variables $\{\, x_i^\bot, x_i^\top \,\}_i$, $\mathtt{x}$ adds the variable x^*, and $\mathtt{g}$ has x^* as a precondition. Additionally, we constrain $\mathtt{x}$ to add either one of $x_i^\bot$ or $x_i^\top$ for all i. We encode the domain and constraint in the formula ψ.

Note that the trace τ of π in our domain does not depend on which $x_i^\bot$ or $x_i^\top$ are added by $\mathtt{x}$. These additions cannot be observed. See Figure 8 for an example trace.

Claim: There exists $\chi : \{\, x_i \,\}_i \to \{\, \bot, \top \,\}$ such that $\varphi[\chi]$ is not satisfiable if and only if there exists a domain $\delta \models \psi$ in which π is perfectly justified.

In any $\delta \models \psi$, the add effects of $\mathtt{x}$ correspond to a truth assignment χ of the x-variables. Any counterexample to the perfect justification of π in such δ must be of the form $\pi' = (\mathtt{i})^\frown (\frown Y)^\frown (\mathtt{x})^\frown (\frown L)^\frown (\mathtt{g})$ for some $Y \subseteq \{\, (\mathtt{y}_j) \,\}_j$, $L \subseteq \{\, (\mathtt{l}_{k,d}) \,\}_{k,d}$ such that π' witnesses the satisfiability of $\varphi[\chi]$ and vice versa. The validity of $\exists \overline{x}. \neg \exists \overline{y}. \varphi[\overline{x}, \overline{y}]$ is therefore equivalent to the existence of a domain $\delta \models \psi$ in which π is perfectly justified. $\square$



	$\mathtt{i}$	$\mathtt{y}_0$	$\mathtt{y}_1$	$\mathtt{r}$	$\mathtt{x}$	$\mathtt{l}_{0,0}$	$\mathtt{l}_{0,1}$	$\mathtt{l}_{1,0}$	$\mathtt{l}_{1,1}$	$\mathtt{g}$
i^*	+	○	○	○	○	○	○	○	○	◎
$y_0^\bot$	+	×		+	○	○	○	○	○	○
$y_0^\top$		+	○	◎	○	◎	○	○	○	○
$y_1^\bot$	+	○	×	+	○	○	○	○	○	○
$y_1^\top$			+	◎	○	○	○	◎	○	○
$x_0^\bot$				+	●	○	○	○	◎	○
$x_0^\top$				+	●	○	◎	○	○	○
x^*					+	○	○	○	○	◎
$l_{0,0}$	+	○	○	×		+	○	○	○	◎
$l_{0,1}$	+	○	○	×			+	○	○	◎
$l_{1,0}$	+	○	○	×				+	○	◎
$l_{1,1}$	+	○	○	×					+	◎
c_0						+	+	○	○	◎
c_1								+	+	◎

Figure 8: The trace of π corresponding to $\varphi = (y_0 \vee x_0) \wedge (y_1 \vee \neg x_0)$ as in the proof of Theorem 4. This trace is not perfectly justified in any domain $\delta \models \psi$.

Conclusion

We gave a formal definition of domain learning and presented a theoretical analysis. Finding a domain that is consistent with a given trace is simple, but the possible inferences are limited if the given traces are not assumed to be justified. Regarding the problem of deciding whether there exists a domain such that a given trace is justified, we showed that it suffices to consider the most restrictive domain, which maximizes preconditions and deletions and minimizes additions. For well-justification, the problem has been shown to be in P. For perfect justification, however, the problem is coNP-complete. Additional expert knowledge encoded in a propositional logical formula can increase the complexity to NP-C and Σ_2^{P}, respectively.

Beyond the formulation as in this paper, one can find many research directions and open questions. One could consider stronger notions than well- or perfect justification, e.g. by presupposing *optimality*, i.e. that there exists no shorter plan that reaches the goal. Verifying optimality is, as perfect justification, coNP-complete (Lin et al. 2024). In a preliminary analysis we obtain the same complexities for domain learning under optimality as with perfect justification. Instead of changing the assumptions on the given traces one could also impose additional constraints on the domain to be found such as limiting the number of preconditions or effects. Ultimately, we can also change the underlying planning problem to obtain a more expressive formalism. For example, we could allow conditional effects or parametrized actions. This could also help to align our analysis with the settings found elsewhere in the literature. We plan to investigate partially observable and unobservable states next.

References

Aineto, D.; Celorrio, S. J.; and Onaindia, E. 2019. Learning action models with minimal observability. *Artificial Intelligence*, 275: 104–137.

Arora, A.; Fiorino, H.; Pellier, D.; Métivier, M.; and Pesty, S. 2018. A Review of Learning Planning Action Models. *The Knowledge Engineering Review*, 33e20: 1–25.

Benson, S. 1996. *Learning Action Models for Reactive Autonomous Agents*. Ph.D. thesis, Stanford University.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69(1-2): 165–204.

Cresswell, S.; and Gregory, P. 2011. Generalised Domain Model Acquisition from Action Traces. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 42–49.

Fikes, R.; and Nilsson, N. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3–4): 189–208.

Fink, E. 1992. *Justified Plans and Ordered Hierarchies*. Master's thesis, University of Waterloo.

Fink, E.; and Yang, Q. 1992. Formalizing Plan Justifications. In *Proceedings of the 9th Canadian Conference on Artificial Intelligence*, 9–14.

Fink, E.; and Yang, Q. 1993. A Spectrum of Plan Justifications. In *Proceedings of the AAAI 1993 Spring Symposium*, 23–33.

Gregory, P.; and Lindsay, A. 2016. Domain Model Acquisition in Domains with Action Costs. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 149–157.

Jimenez, S.; De La Rosa, T.; Fernandez, S.; Fernandez, F.; and Borrajo, D. 2012. A Review of Machine Learning for Automated Planning. *The Knowledge Engineering Review*, 27(4): 433–467.

Juba, B.; Le, H. S.; and Stern, R. 2021. Safe Learning of Lifted Action Models. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning (KR 2021)*, 379–389.

Kambhampati, S. 2007. Model-lite Planning for the Web Age Masses: The Challenges of Planning with Incomplete and Evolving Domain Models. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007)*, 1601–1604.

Lin, S.; and Bercher, P. 2021. Change the World - How Hard Can that Be? On the Computational Complexity of Fixing Planning Models. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 4152–4159.

Lin, S.; Olz, C.; Helmert, M.; and Bercher, P. 2024. On the Computational Complexity of Plan Verification, (Bounded) Plan-Optimality Verification, and Bounded Plan Existence. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI 2024).*

Sreedharan, S.; Bercher, P.; and Kambhampati, S. 2022. On the Computational Complexity of Model Reconciliations. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI 2022)*, 4657–4664.

Stern, R.; and Juba, B. 2017. Efficient, Safe, and Probably Approximately Complete Learning of Action Models. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2017)*, 4405–4411.

Wang, X. 1995. Learning by Observation and Practice: An Incremental Approach for Planning Operator Acquisition. In *Proceedings of the 12th International Conference on Machine Learning (ICML 1995)*, 549–557.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence*, 171(2-3): 107–143.