Guarantees for Sound Abstractions for Generalized Planning

Blai Bonet[1,2*], Raquel Fuentetaja[2], Yolanda E-Martín[2] and Daniel Borrajo[2]
[1]Universidad Simón Bolívar, Venezuela
[2]Universidad Carlos III de Madrid, Spain
bonet@usb.ve, {rfuentet,yescuder}@inf.uc3m.es, dborrajo@ia.uc3m.es

Abstract

Generalized planning is about finding plans that solve collections of planning instances, often infinite collections, rather than single instances. Recently it has been shown how to reduce the planning problem for generalized planning to the planning problem for a qualitative numerical problem; the latter being a reformulation that simultaneously captures all the instances in the collection. An important thread of research thus consists in finding such reformulations, or abstractions, automatically. A recent proposal learns the abstractions inductively from a finite and small sample of transitions from instances in the collection. However, as in all inductive processes, the learned abstraction is not guaranteed to be correct for the whole collection. In this work we address this limitation by performing an analysis of the abstraction with respect to the collection, and show how to obtain formal guarantees for generalization. These guarantees, in the form of first-order formulas, may be used to 1) define subcollections of instances on which the abstraction is guaranteed to be sound, 2) obtain necessary conditions for generalization under certain assumptions, and 3) do automated synthesis of complex invariants for planning problems. Our framework is general, it can be extended or combined with other approaches, and it has applications that go beyond generalized planning.

1 Introduction

Generalized planning is about finding plans that solve a whole collection of instances of planning problems rather than finding a plan for a single instance as in classical planning [Srivastava *et al.*, 2008; Hu and De Giacomo, 2011; Srivastava *et al.*, 2011b; Belle and Levesque, 2016; Segovia *et al.*, 2016]. In its simplest form, the instances in the collection share a common pool of actions and observable features [Hu and De Giacomo, 2011; Bonet *et al.*, 2017], yet other formulations consider relational domains where the actions and features in the instances result of grounding a collection of actions and atom schemas with different sets of objects [Boutilier *et al.*, 2001; Wang *et al.*, 2008; Srivastava *et al.*, 2011a; van Otterlo, 2012].

A recent proposal for handling relational domains casts the problem of generalized planning as the problem of solving a single abstraction, or reformulation, that captures all the instances in the collection [Bonet and Geffner, 2018]. This abstraction however involves qualitative numerical features, in addition to the standard boolean features, that are defined in terms of the objects in the states and their relationships. The actions in the abstraction tell how the features change their values when actions are applied. Qualitative rather than exact numerical features are used to avoid undecidability issues [Helmert, 2002]. The change for such features is only *qualitative* as they only specify whether the numerical feature increases, decreases, or remain unchanged. Under such effects, the problem of solving the abstraction, and hence the generalized planning problem, can be reduced to the problem of solving a single fully observable *non-deterministic* (FOND) problem [Geffner and Bonet, 2013].

This formulation of generalized planning is appealing as it leverages the existing FOND planners to solve, in one shot, a complete (often infinite) class of problems, but it requires the right set of features and the right abstraction. Bonet *et al.* [2019a] learn the abstraction *inductively* from a small sample of transitions from instances in the collection. The abstraction is guaranteed to generalize when the sample is sufficiently general and diverse, but, as far as we know, there have been no attempts to automatically check whether the learned abstraction is sound for the collection.

In this work we bridge this gap by providing a general framework for the synthesis of guarantees for generalization. The guarantees are in the form of first-order formulas that provide sufficient conditions for generalization: every instance whose reachable states satisfy the formulas is guaranteed to be handled correctly by the abstraction. We only address the synthesis of such formulas and defer to future work the problem of verifying whether the formulas are satisfied on the reachable states of a given instance. Nonetheless, the automatically synthesized formulas have a rich and complex structure, and they often express novel and interesting invariants on well-known benchmarks. For example, in Blocksworld, the classical problem of moving blocks with a

[*]On sabbatical leave from Universidad Simón Bolívar.

gripper, one such formula says that every tower must end in a clear block, a formula that thus forbids the existence of "circular towers"; we are not aware of any other approach for invariant synthesis that is able to produce such a formula.

Our contributions are the following: 1) a crisp theoretical foundation for the synthesis of formulas only using as input the relational planning domain and the abstraction, 2) the obtained formulas define subcollections of instances that are guaranteed to be handled correctly by the abstraction, 3) under additional assumptions, necessary conditions for generalization are obtained, and 4) the synthesis also provides candidates for invariants that would then need to be verified.

The paper is organized as follows. The next section provides background on the feature-based account for generalized planning. First-order structures and abstractions are discussed in Sect. 3. The framework for generalization and the synthesis algorithm are given in Sect. 4 and 5. Sect. 6, discusses necessary conditions and the synthesis of invariants. The paper concludes with examples and a discussion.

2 Background

2.1 Collections of Instances

We consider collections $\mathcal{Q}$ of *grounded* STRIPS instances $P = (F, A, I, G)$ where F is a set of atoms (propositions), A is a set of actions, and $I \subseteq F$ and $G \subseteq F$ describe the initial and goal states of P. It is assumed that all instances in $\mathcal{Q}$ result from grounding a *common domain* $\mathcal{D}$ with a set of objects, particular to each instance, and descriptions of the initial and goal situations. As it is standard, $\mathcal{D}$ specifies the constant and predicate symbols that define the propositions via the grounding process, and it also contains lifted action schemas that generate the set A of grounded actions. $\mathcal{Q}(\mathcal{D})$ denotes the class of all grounded instances for domain $\mathcal{D}$. Hence, $\mathcal{Q} \subseteq \mathcal{Q}(\mathcal{D})$ as all instances in $\mathcal{Q}$ come from $\mathcal{D}$.

2.2 Abstractions

The boolean and numerical features are used to build *uniform abstractions* for the instances in $\mathcal{Q}$. Such instances, although sharing a common relational domain, may differ substantially in the number of actions, objects, and observables.

A boolean feature ϕ for $\mathcal{Q}$ is a function that maps each instance $P \in \mathcal{Q}$ and state s for P (reachable from the initial state of P) into a truth value $\phi(P, s) \in \{\top, \bot\}$. A numerical feature is a function ϕ that maps P and s into a non-negative integer $\phi(P, s)$. When P or s are clear from context we may simplify notation. The set of features for $\mathcal{Q}$ is denoted by F. For boolean features f, an F-literal is either f or $\neg f$, while for numerical features n, an F-literal is $n > 0$ or $n = 0$.

An abstraction for $\mathcal{Q}$ is a tuple $\bar{Q} = (F, A_F, I_F, G_F)$ where F is a set of features, A_F is a set of abstract actions, and I_F and G_F describe the abstract initial and goal states in terms of the features. An abstract action $\bar{a}$ is a pair $\langle \text{Pre}; \text{Eff} \rangle$ where Pre is a collection of F-literals, and Eff is a collection of effects for F. Effects for boolean features are denoted by F-literals, while effects for numerical features n correspond to increments or decrements denoted by $n\!\uparrow$ or $n\!\downarrow$ respectively. The items I_F and G_F denote *consistent* sets of F-literals. It is assumed that the effects of actions and G_F are consistent sets of literals, and that I_F is maximal consistent.[1]

The pair (I_F, G_F) of initial and goal states in the abstraction $\bar{Q} = (F, A_F, I_F, G_F)$ complies with $\mathcal{Q}$ when (I_F, G_F) complies with each instance P in $\mathcal{Q}$. The pair (I_F, G_F) complies with the instance P when the initial state of P is consistent with I_F, and if s is a state in P that is consistent with G_F, then s is a goal state for P. A state s in P is consistent with I_F (resp. G_F) iff $\bar{s} \cup I_F$ (resp. $\bar{s} \cup G_F$) is consistent, where $\bar{s}$ denotes the *boolean valuation* of F on s; i.e., $\bar{s} = \{f : f(s){=}\top\} \cup \{\neg f : f(s){=}\bot\} \cup \{n{=}0 : n(s){=}0\} \cup \{n{>}0 : n(s){>}0\}$. If the pair (I_F, G_F) for the abstraction $\bar{Q}$ complies with P, we write $\bar{Q} \sim P$.

Following Bonet and Geffner [2018], an abstraction $\bar{Q}$ is sound for $\mathcal{Q}$ if it complies with $\mathcal{Q}$ and each action $\bar{a}$ in A_F is sound (for $\mathcal{Q}$). An abstract action $\bar{a} = \langle \text{Pre}; \text{Eff} \rangle$ is sound iff for each instance P in $\mathcal{Q}$ and reachable state s in P where Pre holds in $\bar{s}$, $\bar{a}$ represents at least one action a from P in s. The abstract action $\bar{a}$ represents the action a in the state s iff 1) the preconditions of a and $\bar{a}$ both hold in s and $\bar{s}$ respectively, and 2) the effects of a and $\bar{a}$ over F are similar; namely,

a) for any boolean feature f in F, if f changes from true to false (resp. false to true) in the transition $s \rightsquigarrow res(a, s)$ (where $res(a, s)$ is the state that results of applying a in s), then $\neg f \in$ Eff (resp. $f \in$ Eff),

b) for any boolean feature f in F, if f (resp. $\neg f$) is in Eff, then f is true (resp. false) in $res(a, s)$, and

c) for each numerical feature n in F, $n\downarrow$ (resp. $n\uparrow$) appears in Eff if and only if $n(P, res(a, s)) < n(P, s)$ (resp. $n(P, s) < n(P, res(a, s))$).

We write $\bar{a} \sim_{P,s} a$ to denote that the abstract action $\bar{a}$ represents the action a in the (reachable) state s of P. In such a case, we also say that a instantiates $\bar{a}$ in s. When there is no confusion about P, we simplify notation to $\bar{a} \sim_s a$.

Soundness links plans for $\bar{Q}$ with generalized plans: if $\bar{\pi}$ is a plan that solves an abstraction $\bar{Q}$ that is *sound* for $\mathcal{Q}$ and P is an instance in $\mathcal{Q}$, then *any* execution $(a_0, a_1, \ldots)$ spawned by $\bar{\pi}$ on P reaches a goal state for P. The execution $(a_0, a_1, \ldots)$ is spawned by $\bar{\pi}$ on P iff 1) a_i instantiates $\bar{\pi}(\bar{s}_i)$ in s_i, for $i \geq 0$, 2) $\bar{s}_i$ is the boolean valuation of s_i, for $i \geq 0$, 3) $s_{i+1} = res(a_i, s_i)$, for $i \geq 0$, and 4) s_0 is the initial state of P.

Example. Consider the collection $\mathcal{Q}_{clear}$ with all Blocksworld instances with goal $clear(\mathsf{A})$ where A is a fixed block. The domain $\mathcal{D}_{clear}$ has no explicit gripper, contains a single constant A, and has two action schemas: Newtower(x, y) to move block x from block y to the table, and Move(x, y, z) to move block x from block y onto block z. An abstraction for $\mathcal{Q}_{clear}$ is $\bar{Q}_{clear} = (F, A_F, I_F, G_F)$ where $F = \{n\}$ and n is the feature that counts the number of blocks above A, $A_F = \{\bar{a}\}$ where $\bar{a} = \langle n{>}0; n\!\downarrow \rangle$, $I_F = \{n{>}0\}$, and $G_F = \{n{=}0\}$. It is easy to check that $\bar{Q}_{clear}$ is sound and solved by the plan $\bar{\pi}_{clear}$ that executes $\bar{a}$ whenever $n{>}0$. An action Newtower(x, y) or Move(x, y, z) that "removes"

[1] A set S of F-literals is consistent if for any boolean feature f, S excludes either f or $\neg f$, and for any numerical feature n, S excludes either $n > 0$ or $n = 0$. S is maximal consistent if it is consistent, and $S \cup \{L\}$ is not consistent for any F-literal $L \notin S$.

a block from above A in state s is an action that instantiates $\bar{a}$ in s. Notice that $\mathcal{Q}_{clear} \neq \mathcal{Q}(\mathcal{D}_{clear})$ since, for example, $\mathcal{Q}(\mathcal{D}_{clear})$ contains instances that have "circular towers". $\square$

2.3 Inductive Learning and Concepts

Bonet *et al.* [2019a] show how an abstraction can be learned from a sample of transitions and a collection $\mathcal{F}$ of candidate features. In their approach, each feature in $\mathcal{F}$ is associated with a concept C that is obtained from a set of atomic concepts, and a concept grammar [Baader *et al.*, 2003].[2]

In general, a concept for $\mathcal{Q}$ may be thought of as a function C that maps instances P in $\mathcal{Q}$ and states s in P into sets $C(P, s)$ of *tuples of objects*. Concepts define features: boolean features f_C that denote whether $C(P, s)$ is non-empty, and numerical features n_C that denote the cardinality of $C(P, s)$. The concepts by Bonet *et al.* are limited to denotations that are subsets of objects rather than object tuples.

3 First-Order Abstractions

We deal with formulas in first-order logic that are built from a signature $\sigma = \sigma(\mathcal{D})$ given by the relational domain $\mathcal{D}$. The constants defined in $\mathcal{D}$ appear as constant symbols in σ, and the predicates defined in $\mathcal{D}$ appear as relational symbols of corresponding arity in $\mathcal{D}$. The signature also contains binary relations p^+ and p^* for the binary predicates p in $\mathcal{D}$. As usual, $\mathcal{L}(\sigma)$ denotes the class of well-formed formulas over σ.

First-order formulas are interpreted over first-order structures, also called interpretations. We are only interested in structures that are associated with states. A state s provides the universe $\mathcal{U}_s$ of objects and the interpretations for the constant and relational symbols in $\mathcal{D}$. The interpretations for p^+ and p^*, for the binary predicates p, are provided by the transitive and reflexive-transitive closure of the interpretation of p provided by s. We write $\varphi(\bar{x})$ to denote a formula whose free variables are among those in $\bar{x}$. If $\varphi(\bar{x})$ is a formula, s is a state in P, and $\bar{u}$ is a tuple of objects in $\mathcal{U}_s$ of dimension $|\bar{x}|$, $s \vDash \varphi(\bar{u})$ denotes that the interpretation provided by s satisfies φ when the variables in $\bar{x}$ are interpreted by the corresponding objects in $\bar{u}$.

For a concept C characterized by $\Psi_C(\bar{x})$, the extension of C for s in P is $C^s = C(P, s) = \{\bar{u} : s \vDash \Psi_C(\bar{u})\}$. We assume that all features correspond to concepts whose characteristic functions are first-order definable:

Definition 1 (First-Order Abstraction)**.** *Let $\mathcal{D}$ be a planning domain and let $\sigma = \sigma(\mathcal{D})$ be the signature for $\mathcal{D}$. A concept C is (first-order) $\mathcal{D}$-definable if $\Psi_C(\bar{x})$ belongs to $\mathcal{L}(\sigma)$. A feature f is $\mathcal{D}$-definable if f is given by a concept C that is $\mathcal{D}$-definable. An abstraction $\bar{Q} = (F, A_F, I_F, G_F)$ is a first-order abstraction for $\mathcal{D}$ if each feature f in F is $\mathcal{D}$-definable.*

When $\mathcal{D}$ is clear from the context, we just say that $\bar{Q}$ is a first-order abstraction without mentioning $\mathcal{D}$. The applicability of an abstract action $\bar{a}$ in a first-order abstraction $\bar{Q}$ on a state s can be decided with a first-order formula Pre$(\bar{a})$.

Example. The abstraction $\bar{Q}_{clear}$ is a first-order abstraction because $F = \{n\}$ and n is the cardinality of the concept C given by $\Psi_C(x) = \exists y(on(x, y) \wedge on^*(y, \mathsf{A}))$. However, C is also given by $\Psi'_C(x) = on^+(x, \mathsf{A})$. As usual, both representations may yield different results although being *logically equivalent*; more about this below. $\square$

4 Conditions for Generalization

Let $\bar{Q} = (F, A_F, I_F, Q_F)$ be a first-order abstraction for $\mathcal{D}$. We look for conditions to establish the soundness of $\bar{Q}$ for a generalized problem $\mathcal{Q} \subseteq \mathcal{Q}(\mathcal{D})$. In particular, we aim for conditions of the form $\mathcal{G} = \{\Phi_{\bar{a}} : \bar{a} \in A_F\}$ where $\Phi_{\bar{a}} = \exists\bar{z}\big(\bigvee_i \Psi_{\bar{a}}^{a_i}(\bar{z})\big)$ is associated with the abstract action $\bar{a}$ and satisfies the following:

- a_i is an *action schema* in $\mathcal{D}$,
- $\bar{z}$ is a tuple of variables that represent the parameters of the action schemas in $\mathcal{D}$ (these are existentially quantified on the objects of the given state s in problem P), and
- if $\bar{o}$ is a tuple of objects of dimension $|\bar{z}|$ such that $s \vDash$ Pre$(\bar{a}) \wedge \Psi_{\bar{a}}^{a_i}(\bar{o})$, where s is a reachable state in problem $P \in \mathcal{Q}(\mathcal{D})$, then the *ground action* $a_i(\bar{o})$ instantiates the abstract action $\bar{a}$ in the state s (i.e., $\bar{a} \sim_s a_i(\bar{o})$).

The idea is that $\Psi_{\bar{a}}^{a_i}(\bar{o})$ suffices to establish $\bar{a} \sim_s a_i(\bar{o})$ directly from Pre$(\bar{a})$ and the *(lifted) domain $\mathcal{D}$* without using any other information about the reachability of state s (e.g., invariant information for reachable states). On the other hand, such formulas would be "accompanied" by *assumed conditions* Pre$(\bar{a}) \Rightarrow \exists\bar{z}\big(\bigvee_i \Psi_{\bar{a}}^{a_i}(\bar{z})\big)$ on the *reachable states* that together with the above properties provide the guarantee:

Definition 2 (Guarantee)**.** *Let $\mathcal{D}$ be a planning domain and let $\bar{Q} = (F, A_F, I_F, G_F)$ be a first-order abstraction. A* guarantee *for $\bar{Q}$ is a set $\mathcal{G} = \{\Phi_{\bar{a}} = \exists\bar{z}\big(\bigvee_i \Psi_{\bar{a}}^{a_i}(\bar{z})\big) : \bar{a} \in A_F\}$ of formulas for each abstract action $\bar{a}$ in $\bar{Q}$. The guarantee is valid in instance $P \in \mathcal{Q}(\mathcal{D})$ iff for each state $s \in P$ (reachable or not) and tuple $\bar{o}$ of objects in P, if $s \vDash Pre(\bar{a}) \wedge \Psi_{\bar{a}}^{a_i}(\bar{o})$ then $\bar{a} \sim_s a_i(\bar{o})$. The guarantee $\mathcal{G}$ is valid for $\mathcal{D}$ iff it is valid for each problem P in $\mathcal{Q}(\mathcal{D})$.*

Theorem 3 (Soundness)**.** *Let $\mathcal{D}$ be a planning domain, let $\bar{Q} = (F, A_F, I_F, I_F, G_F)$ be a first-order abstraction, and let $\mathcal{G} = \{\Phi_{\bar{a}} = \exists\bar{z}\big(\bigvee_i \Psi_{\bar{a}}^{a_i}(\bar{z})\big) : \bar{a} \in A_F\}$ be a guarantee for $\mathcal{D}$. If $\mathcal{G}$ is valid, then $\bar{Q}$ is a sound abstraction for the generalized problem $\mathcal{Q} = \{P \in \mathcal{Q}(\mathcal{D}) : \bar{Q} \sim P$ and $Pre(\bar{a}) \Rightarrow \Phi_{\bar{a}}$ holds in the reachable states in $P\}$.*

Proof. Let P be a problem in $\mathcal{Q}$, let s be a reachable state in P, and let $\bar{a}$ be an abstract action that is applicable in $\bar{s}$. Since $\bar{Q} \sim P$, we only need to show $\bar{a} \sim_s a$ for some action a. By definition of $\mathcal{Q}$, $s \vDash \Phi_{\bar{a}}$ where $\Phi_{\bar{a}} = \exists\bar{z}\big(\bigvee_i \Psi_{\bar{a}}^{a_i}(\bar{z})\big)$. Hence, there is a tuple $\bar{o}$ of objects such that $s \vDash$ Pre$(\bar{a}) \wedge \Psi_{\bar{a}}^{a_i}(\bar{o})$ for some schema a_i in $\mathcal{D}$. Then, by Definition 2, $\bar{a} \sim_s a_i(\bar{o})$. $\square$

5 Synthesis

For a feature f defined by concept C we need to track its value along transitions $s \rightsquigarrow res(a(\bar{o}), s)$. Let $\Psi_C^a(\bar{z}, \bar{x})$ be a formula that defines at state s the extension of C in the state $res(a(\bar{o}), s)$ that results of applying $a(\bar{o})$ in s; i.e.,

$$s \vDash \Psi_C^a(\bar{o}, \bar{u}) \quad \text{iff} \quad \bar{u} \in C^{res(a(\bar{o}), s)}$$

[2]We do not consider the distance features [Bonet *et al.*, 2019a] as it is not clear how to express them in first-order logic.

where $\bar{u}$ is a tuple of objects. For example, a boolean feature f defined by C goes from true to false in $s \rightsquigarrow res(a(\bar{o}), s)$ iff $C^s \neq \emptyset$ and $C^{res(a(\bar{o}),s)} = \emptyset$ iff $s \vDash \exists \bar{x}(\Psi_C(\bar{x}))$ and $s' \vDash \neg\exists\bar{x}(\Psi_C(\bar{x}))$ iff $s \vDash \exists\bar{x}(\Psi_C(\bar{x})) \wedge \forall\bar{x}(\neg\Psi^a_C(\bar{o}, \bar{x}))$.

Since the concept C may be defined in terms of relations p^* or p^+ that denote the transitive closure of p, and that transitive closure is not first-order definable [Vardi, 1982], it is not always possible to track in first-order logic the change of denotation for p^* or p^+ after an action changes the denotation of p. Hence, we settle for a "logical approximation" of $\Psi^a_C(\bar{z}, \bar{x})$ in terms of necessary and sufficient conditions:

$$S^a_C(\bar{z}, \bar{x}) \Rightarrow \Psi^a_C(\bar{z}, \bar{x}) \Rightarrow N^a_C(\bar{z}, \bar{x}) .$$

A *base for synthesis* provides approximations for all the atoms in the language $\mathcal{L}(\sigma)$:

Definition 4 (Base for Synthesis). *A base for synthesis for domain $\mathcal{D}$ is a set $\mathcal{T}$ that contains formulas $\mathcal{T}_X(a, p)(\bar{z}, \bar{x})$ for $X \in \{N, S\}$, action schemas $a(\bar{z}) \in \mathcal{D}$, and predicates $p \in \mathcal{D}$ of arity $|\bar{x}|$. It also contains formulas $\mathcal{T}_X(a, p^c)(\bar{z}, x, y)$ for $X \in \{N, S\}$, action schemas $a(\bar{z}) \in \mathcal{D}$, binary predicates $p \in \mathcal{D}$, and at least one of $c = *$ or $c = +$. These formulas should provide necessary and sufficient conditions as follows. For any problem P in $\mathcal{Q}(\mathcal{D})$, state s in P, tuple $\bar{o}$ such that $a(\bar{o})$ is applicable at s, tuple $\bar{u}$, objects u and v, and $c \in \{*, +\}$:*

$$\begin{aligned} s \vDash \mathcal{T}_S(a, p)(\bar{o}, \bar{u}) &\Rightarrow \bar{u} \in C_p^{res(a(\bar{o}),s)} \\ &\Rightarrow s \vDash \mathcal{T}_N(a, p)(\bar{o}, \bar{u}) , \\ s \vDash \mathcal{T}_S(a, p^c)(\bar{o}, u, v) &\Rightarrow (u, v) \in C_{p^c}^{res(a(\bar{o}),s)} \\ &\Rightarrow s \vDash \mathcal{T}_N(a, p^c)(\bar{o}, u, v) \end{aligned}$$

where $C^s_p = \{\bar{u} : s \vDash p(\bar{u})\}$ and $C^s_{p^c} = \{(u, v) : s \vDash p^c(u, v)\}$ are the concepts associated with p and p^c respectively.

The approximation for the atoms in $\mathcal{L}(\sigma)$ that is provided by the base $\mathcal{T}$ is *lifted* over all first-order formulas. Indeed, the following structural induction gives necessary and sufficient conditions $N^a_\varphi(\bar{z}, \bar{x})$ and $S^a_\varphi(\bar{z}, \bar{x})$ for any concept C defined in terms of formula φ. For $X \in \{N, S\}$:

- $X^a_p(\bar{z}, \bar{x}) = \mathcal{T}_X(a, p)(\bar{z}, \bar{x})$ where p is a predicate of arity $|\bar{x}|$, or $p = q^c$ for some binary predicate q, and $c \in \{*, +\}$,
- $X^a_{\varphi \circ \psi}(\bar{z}, \bar{x}) = X^a_\varphi(\bar{z}, \bar{x}) \circ X^a_\psi(\bar{z}, \bar{x})$ where $\circ \in \{\wedge, \vee\}$,
- $N^a_{\neg\varphi}(\bar{z}, \bar{x}) = \neg S^a_\varphi(\bar{z}, \bar{x})$ and $S^a_{\neg\varphi}(\bar{z}, \bar{x}) = \neg N^a_\varphi(\bar{z}, \bar{x})$, and
- $X^a_{Qy\varphi}(\bar{z}, \bar{x}) = QyX^a_\varphi(\bar{z}, \bar{x}, y)$ where $Q \in \{\exists, \forall\}$.

The base provides approximations for either p^* or p^+, or both. In the former case, this is enough since one of the closures can be expressed in terms of the other; e.g., $on^+(x, y) \equiv \exists z(on(x, z) \wedge on^*(z, y)))$.

Below we propose a general base for synthesis of formulas. With this base, the formulas N^a_φ and S^a_φ are identical except when φ contains a transitive closure. Hence, except for such φ, both formulas are necessary and sufficient.

Theorem 5 (Lift). *Let $\mathcal{T}$ be a base for synthesis for domain $\mathcal{D}$, let $a = a(\bar{z})$ be an schema in $\mathcal{D}$, and let $\varphi(\bar{z}, \bar{x})$ be a first-order formula in $\mathcal{L}(\sigma(\mathcal{D}))$. Then, for any instance P in $\mathcal{Q}(\mathcal{D})$, state s for P, and tuples $\bar{o}$ and $\bar{u}$ of objects in P:*

$$s \vDash S^a_\varphi(\bar{o}, \bar{u}) \Rightarrow res(a(\bar{o}), s) \vDash \varphi(\bar{o}, \bar{u}) \Rightarrow s \vDash N^a_\varphi(\bar{o}, \bar{u}) .$$

As noted earlier, tracking the change of boolean features f defined by concepts C is easy since f is true or false at s iff C^s is non-empty or empty respectively. Tracking the qualitative numerical changes is more challenging, however. For example, f increases in the transition $s \rightsquigarrow s'$ iff $|C^s| < |C^{s'}|$. This condition is difficult to capture because the extension of C may increase size by the result of a small change, as simple as one new element entering the set, or by a large change involving many elements. The case of local, small, changes is common and easy to define:

Definition 6 (Monotonicity). *Let $\mathcal{D}$ be a domain and let P be an instance in $\mathcal{Q}(\mathcal{D})$. A concept C for $\mathcal{Q}(\mathcal{D})$ is* monotone *in P if for every* reachable *state s in P, and action $a(\bar{o})$ that is applicable in s, either $C(P, s) \subseteq C(P, s')$, $C(P, s) \supseteq C(P, s')$, or $C(P, s) = C(P, s')$ for $s' = res(a(\bar{o}), s)$. A first-order abstraction $\bar{Q}$ is monotone for P if each feature f in $\bar{Q}$ is defined by a concept C that is $\mathcal{D}$-definable and monotone.*

Necessary and sufficient conditions for the change of value of monotone features f along transitions $s \rightsquigarrow res(a(\bar{z}), s)$, for action schema $a(\bar{z})$, are provided by the formulas:

$$\begin{aligned} X^{inc}_C(\bar{z}) &= \exists\bar{x}(\neg\Psi \wedge X^a_\Psi) , \\ X^{dec}_C(\bar{z}) &= \exists\bar{x}(\Psi \wedge \neg\widehat{X}^a_\Psi) , \\ X^{eq}_C(\bar{z}) &= \forall\bar{x}(\Psi \Rightarrow X^a_\Psi) \wedge \forall\bar{x}(\widehat{X}^a_\Psi \Rightarrow \Psi) , \\ X^{true}_C(\bar{z}) &= \exists\bar{x}(X^a_\Psi(\bar{z}, \bar{x})) , \\ X^{false}_C(\bar{z}) &= \forall\bar{x}(\neg\widehat{X}^a_\Psi(\bar{z}, \bar{x})) \end{aligned}$$

where C is the concept that defines f, $X \in \{N, S\}$ denotes a necessary or sufficient condition, Ψ and X^a_Ψ denote $\Psi_C(\bar{x})$ and $X^a_{\Psi_C}(\bar{z}, \bar{x})$ respectively, and $\widehat{N}^a_\Psi = S^a_\Psi$ and $\widehat{S}^a_\Psi = N^a_\Psi$.

For example, $S^{dec}_C(\bar{z}) = \exists\bar{x}(\Psi \wedge \neg N^a_\Psi)$. If $s \vDash S^{dec}_C(\bar{o})$, there is object tuple $\bar{u}$ such that $s \vDash \Psi(\bar{u}) \wedge \neg N^a_\Psi(\bar{o}, \bar{u})$; i.e., $\bar{u} \in C^s$ and $\bar{u} \notin C^{res(a(\bar{o}),s)}$. For monotone features, the only possibility is $C^{res(a(\bar{o}),s)} \subsetneq C^s$ which means that the feature *decreases* in the transition $s \rightsquigarrow res(a(\bar{o}), s)$.

For obtaining sufficient conditions for general features, the first two formulas from above are strengthen as

$$\begin{aligned} S^{inc}_C(\bar{z}) &= \forall\bar{x}(\Psi \Rightarrow S^a_\Psi) \wedge \exists\bar{x}(\neg\Psi \wedge S^a_\Psi) , \\ S^{dec}_C(\bar{z}) &= \forall\bar{x}(N^a_\Psi \Rightarrow \Psi) \wedge \exists\bar{x}(\Psi \wedge \neg N^a_\Psi) \end{aligned}$$

where the added conjunct enforces that the feature defined by the concept C is indeed monotone. For the remaining cases, the formulas for sufficiency correspond to those above.

Lemma 7. *Let C be a concept characterized by formula $\Psi_C(\bar{x})$, and let s be a state in $P \in \mathcal{Q}(\mathcal{D})$ on which the action $a(\bar{o})$ is applicable. Then,*

$$\begin{aligned} s \vDash S^{inc}_C(\bar{o}) &\Rightarrow C^s \subsetneq C^{s'} \Rightarrow |C^s| < |C^{s'}| , \\ s \vDash S^{dec}_C(\bar{o}) &\Rightarrow C^s \supsetneq C^{s'} \Rightarrow |C^s| > |C^{s'}| , \\ s \vDash S^{eq}_C(\bar{o}) &\Rightarrow C^s = C^{s'} \Rightarrow |C^s| = |C^{s'}| , \\ s \vDash S^{true}_C(\bar{o}) &\Rightarrow C^{s'} \neq \emptyset \Rightarrow |C^{s'}| > 0 , \\ s \vDash S^{false}_C(\bar{o}) &\Rightarrow C^{s'} = \emptyset \Rightarrow |C^{s'}| = 0 \end{aligned}$$

where $s' = res(a(\bar{o}), s)$ is the result of applying $a(o)$ in s.

Reference	Formula	
$\mathcal{B}_X(a,p)(\bar z,\bar x)$	$[\![p(\bar x) \in \text{Post}]\!] \vee (p(\bar x) \wedge [\![\neg p(\bar x) \notin \text{Post}]\!])$	
$\mathcal{B}_N(a,p^*)(\bar z,x,y)$	$p^*(x,y) \vee \exists uv\big([\![p(u,v) \in \text{Post}]\!] \wedge p^*(x,u) \wedge p^*(v,y)\big)$	(action adds at most 1 p-atom)
	$p^*(x,y) \vee \exists uv\big([\![p(u,v) \in \text{Post}]\!] \wedge (p^*(x,u) \vee p^*(v,y))\big)$	(action adds 2 or more p-atoms)
$\mathcal{B}_S(a,p^*)(\bar z,x,y)$	$(x=y) \vee \big(p^*(x,y) \wedge \forall uv([\![\neg p(u,v) \in \text{Post}]\!] \Rightarrow u=v)\big)$	

Table 1: General base $\mathcal{B}$ for synthesis of any domain $\mathcal{D}$. Post$(a(\bar z))$ is abbreviated by Post. $X \in \{N,S\}$. There are two versions of the necessary condition for p^*; one for actions that add at most one atom $p(u,v)$, and the other for actions that add two or more atoms of this form. The first version uses a conjunction, $p^*(x,u) \wedge p^*(v,y)$, while the second version replaces it with a disjunction.

We have expressed how the value of individual features changes in transitions. Before providing the complete synthesis, we need to express the value of preconditions of abstract actions, and how the actions affect the different features.

Preconditions of abstract actions $\bar a$ on features $f = f_C$ are expressed by Pre$(\bar a)_C = \top$ if there is no precondition on f, Pre$(\bar a)_C = \exists \bar x(\Psi_C(\bar x))$ if f is boolean (resp. numeric) and Pre$(\bar a)$ contains f (resp. $f > 0$), and Pre$(\bar a)_C = \forall \bar x(\neg\Psi_C(\bar x))$ if f is boolean (resp. numeric) and Pre$(\bar a)$ contains $\neg f$ (resp. $f = 0$).

On the other hand, $\bar a$ partitions the set of features according to their type and the effects of $\bar a$ on them:

$$\begin{aligned}
\Delta^{inc}_{\bar a} &= \{n \in F : n \text{ is numeric and } n\!\uparrow\, \in \text{Eff}(\bar a)\}\,,\\
\Delta^{dec}_{\bar a} &= \{n \in F : n \text{ is numeric and } n\!\downarrow\, \in \text{Eff}(\bar a)\}\,,\\
\Delta^{eq}_{\bar a} &= \{f \in F : f \text{ is not affected by } \bar a\}\,,\\
\Delta^{true}_{\bar a} &= \{f \in F : f \text{ is boolean and } f \in \text{Eff}(\bar a)\}\,,\\
\Delta^{false}_{\bar a} &= \{f \in F : f \text{ is boolean and } \neg f \in \text{Eff}(\bar a)\}\,.
\end{aligned}$$

Definition 8 (Synthesis)**.** *Let $\mathcal{T}$ be a base for synthesis for domain $\mathcal{D}$, and let $\bar Q = (F, A_F, I_F, G_F)$ be a first-order abstraction for $\mathcal{D}$. For abstract action $\bar a$ in A_F and schema $a(\bar z)$ in $\mathcal{D}$, we define the formula $\Psi^a_{\bar a}(\bar z)$ as*

$$\textit{Pre}(a(\bar z)) \;\wedge \bigwedge_{\varphi\in Chg}\;\bigwedge_{f_C\in\Delta^{\varphi}_{\bar a}} \textit{Pre}(\bar a)_C \wedge S^{\varphi}_C(\bar z)$$

where Chg=$\{inc, dec, eq, true, false\}$. The guarantee for $\bar Q$ is $\mathcal{G}(\mathcal{T},\bar Q) = \{\Phi_{\bar a} = \exists \bar z\big(\bigvee_{a\in\mathcal{D}} \Psi^a_{\bar a}(\bar z)\big) : \bar a \in A_F\}$.

Theorem 9 (Main)**.** *Let $\mathcal{T}$ be a base for synthesis for domain $\mathcal{D}$, and let $\bar Q = (F, A_F, I_F, G_F)$ be a first-order abstraction. Then, $\mathcal{G}(\mathcal{T},\bar Q)$ is a* valid guarantee *for $\mathcal{D}$ (cf. Definition 2).*

We cannot yet provide a complete example because the synthesis requires the conditions for the atoms in the language that are given by the base for synthesis. We now provide one such base, and apply it to the running example.

5.1 A General Base for Synthesis

The synthesis framework is parametrized by the base. Trivial, non-informative, bases are easy to obtain: it is enough to define sufficient and necessary conditions as $\bot$ and $\top$ respectively for each atom in the language. We provide a simple, general, and non-trivial base that can be used with any domain $\mathcal{D}$. The conditions provided by two different bases, or by the same base for different but logically equivalent formulas, do not need to be logically equivalent.

Table 1 shows a template for obtaining bases $\mathcal{B}(\mathcal{D})$ for any domain $\mathcal{D}$. No formula in the template involves the predicate p^+; i.e., all such predicates have been replaced by equivalent formulas involving p^*. (Alternatively, we may define a base that only resolves p^+ and assumes that no formula contains p^*.) Two versions for the necessary condition for p^* are provided: one when the action a adds at most one atom $p(u,v)$, and the other when a adds two or more such atoms.

The formulas in Table 1 involve "bracket expressions" that instantiate to first-order formulas. For schema $a(\bar z)$ and tuple $\bar x$, a bracket expression reduces to either to a logical constant $\top$ or $\bot$, or to an expression involving equality over the variables in $\bar z$ and $\bar x$, and the constant symbols in $\mathcal{D}$. For example, $[\![\neg on(x,y) \notin \text{Post}]\!]$ reduces to $xy \neq z_1z_2$ for the action Newtower(z_1,z_2) since this action removes only $on(z_1,z_2)$.

Theorem 10 (General Base)**.** *Let $\mathcal{D}$ be a planning domain. The set $\mathcal{B}(\mathcal{D})$ is a base for synthesis for domain $\mathcal{D}$.*

Corollary 11. *Let $\mathcal{D}$ be a domain and let $\bar Q = (F, A_F, I_F, G_F)$ be a first-order abstraction for $\mathcal{D}$. The guarantee $\mathcal{G}(\mathcal{B}(\mathcal{D}),\bar Q)$ is valid for $\mathcal{D}$ and, hence, $\bar Q$ is a* sound abstraction *for the generalized problem $\mathcal{Q} = \{P \in \mathcal{Q}(\mathcal{D}) : \bar Q \sim P$ and Pre$(\bar a) \Rightarrow \Phi_{\bar a}$ holds in the reachable states in $P\}$.*

Example. The abstraction $\bar Q_{clear} = (F, A_F, I_F, G_F)$ has a single feature n=n_C for $\Psi_C(x)$=$\exists y(on(x,y) \wedge on^*(y,\mathsf{A}))$. $\mathcal{D}_{clear}$ has two schemas a_1 = Newtower(z_1,z_2) and a_2 = Move(z_3,z_4,z_5). The condition $S^{dec}_C(z_1,z_2)$ for a_1 is equivalent (after simplification) to

$$on(z_1,z_2) \wedge on^*(z_2,\mathsf{A}) \wedge \forall y(on(z_1,y)\wedge on^*(y,\mathsf{A}) \Rightarrow y{=}z_2).$$

The formula $\Psi^{a_1}_{\bar a}(z_1,z_2)$ is this formula conjoined with $clear(z_1)$. For action a_2, $S^{dec}_C(z_3,z_4,z_5)$ is

$$\begin{aligned}&on^+(z_3,\mathsf{A}) \wedge \neg on^*(z_5,\mathsf{A}) \wedge\\ &\qquad \forall y(on(z_3,y) \wedge on^*(y,\mathsf{A}) \Rightarrow y = z_4)\,.\end{aligned}$$

The formula $\Psi^{a_2}_{\bar a}(z_1,z_2)$ is this formula conjoined with $clear(z_3)$, $on(z_3,z_4)$, and $clear(z_5)$. The guarantee for $\bar a$ is $\Phi_{\bar a} = \exists \bar z\big(\Psi^{a_1}_{\bar a}(z_1,z_2) \vee \Psi^{a_2}_{\bar a}(z_3,z_4,z_5)\big)$.

By Corollary 11, $\mathcal{Q}_{clear}$ is sound for instances with goal $clear(\mathsf{A})$ and *reachable states* that satisfy $\exists x(on^+(x,\mathsf{A})) \Rightarrow \Phi_{\bar a}$. Namely, if there is a block above A, then either there are blocks z_1 and z_2 such that z_1 is clear and on z_2, z_2 is A or above it, and z_2 mediates any "path of blocks" from z_1 to A, or there are blocks z_3, z_4 and z_5 such that z_3 is clear, on z_4, and above A, z_5 is clear and not equal to A or above it, and z_4

mediates any path from z_3 to A. This formula indeed holds in all "real instances" of Blocksworld. $\bar{Q}_{clear}$ is then sound for all of them, and the policy $\bar{\pi}_{clear}$ that solves the abstraction is a generalized plan for $\mathcal{Q}_{clear}$. A full derivation appears in the extended version of this paper [Bonet *et al.*, 2019b]. $\square$

6 Necessary Conditions and Invariants

The conditions provided by $\mathcal{G}$ are only sufficient; i.e., no conclusion about an abstraction $\bar{Q}$ for instance P can be drawn if some reachable state s in P satisfies Pre$(\bar{a})$ but not $\Phi_{\bar{a}}$. For reasoning in such cases, one needs *necessary* rather than sufficient conditions for soundness.

For obtaining necessary conditions, we need to assume that the features in the abstraction are indeed monotone; i.e., their value changes in a local manner along the transitions in any instance P in $\mathcal{Q}(\mathcal{D})$. For example, under monotonicity, f decreases in $s \rightsquigarrow res(a(\bar{o}), s)$ iff $C^{res(a(\bar{o}),s)} \subsetneq C^s$ iff

$$s \vDash \exists\bar{x}(\Psi_C(\bar{x}) \wedge \neg\Psi^a_C(\bar{o},\bar{x}))$$

(where C is the concept that defines f) only if

$$s \vDash \exists\bar{x}(\Psi_C(\bar{x}) \wedge \neg S^a_C(\bar{o},\bar{x}))\,.$$

We obtain necessary conditions similarly as before:

Theorem 12 (Necessary Conditions)**.** *Let $\mathcal{D}$ be a domain, let $\bar{Q}$ be a first-order abstraction for $\mathcal{D}$, and let P be an instance in $\mathcal{Q}(\mathcal{D})$ such that $\bar{Q}$ is* sound and monotone *for P. If s is a reachable state in P, then*

$$s \;\vDash\; Pre(\bar{a}) \Rightarrow \exists\bar{z}\big(\textstyle\bigvee_i \Gamma^{a_i}_{\bar{a}}(\bar{z})\big)$$

where $\Gamma^{a_i}_{\bar{a}}(\bar{z})$ is the formula

$$Pre(a_i(\bar{z})) \;\wedge \bigwedge_{\varphi\in\{inc,dec,eq,\top,\bot\}} \;\bigwedge_{f_C\in\Delta^{\varphi}_{\bar{a}}} N^{\varphi}_C(\bar{z})\,.$$

Necessary conditions are useful for showing that $\bar{Q}$ is not sound for instance P: it is enough to find a reachable state where the condition does not hold. On the other hand, we cannot infer that $\mathcal{Q}$ is sound for P only if the necessary condition does hold in P.

If the abstraction $\bar{Q}$ is sound for $\mathcal{Q}$, every instance P in $\mathcal{Q}$ for which $\bar{Q}$ is monotone must satisfy the necessary conditions. Such conditions, that by definition hold in all reachable states, are *state invariants* in P. Therefore, sufficient and necessary conditions for the soundness of an abstraction $\bar{Q}$ can be regarded as candidates for invariants, which may then be verified by a theorem prover in order to avoid an explicit enumeration of reachable states [Slind and Norrish, 2008].

7 Other Examples

In this section we present two additional examples. One for a gripper problem with an arbitrary number of balls and grippers, and the other about connectivity in directed graphs.

7.1 Gripper

We consider a domain $\mathcal{D}$ for Gripper with constants A (destination) and B (origin) for the rooms, objects l and r for the grippers, and objects b_i for the different balls. The predicates are $at(r)$ and $in(b,r)$ for the position of the robot and balls in rooms, $ca(b,g)$ to indicate when ball b is held by gripper g, and $fr(g)$ to indicate that gripper g is not holding any ball. The action schemas are $a_1 = $ Move(r_1,r_2) for moving the robot, and $a_2 = $ Pick(b,g,r) and $a_3 = $ Drop(b,g,r) for picking and dropping balls in rooms using specific grippers.

Bonet *et al.* [2019a] learn an abstraction that is made of a boolean feature X, and numerical features B, C, and G:

- $X = \{r : at(r) \wedge r{=}\mathsf{A}\}$ tells whether the robot is in A,
- $B = \{b : \exists r(in(b,r) \wedge r{\neq}\mathsf{A})\}$ counts the balls in B,
- $C = \{b : \exists g(ca(b,g))\}$ counts the balls being held, and
- $G = \{g : fr(g)\}$ counts the free grippers.

The abstract actions in abstraction $\bar{Q}_{gripper}$ are:

- pick $= \langle\neg X, B>0, G>0; B{\downarrow}, G{\downarrow}, C{\uparrow}\rangle$,
- drop $= \langle X, C>0; C{\downarrow}, G{\uparrow}\rangle$,
- go1 $= \langle\neg X, B=0, C>0, G>0; X\rangle$,
- go2 $= \langle\neg X, C>0, G=0; X\rangle$,
- leave $= \langle X, C=0, G>0; \neg X\rangle$.

Both, go1 and go2, move the robot from A to B. Go1 moves the robot that still has room to pick more balls only when there are no more balls to be picked at B; go2 moves the robot when it cannot hold any more balls. The formulas $\Psi^{a_i}_{\bar{a}}(\bar{z})$ are $\bot$ except for (conditions in Pre(a_i) removed to fit space):

$$\begin{aligned}
\Psi^{a_2}_{\text{pick}} &= \forall x[\neg ca(b,x)] \wedge r \neq \mathsf{A}\,,\\
\Psi^{a_3}_{\text{drop}} &= at(A) \wedge \neg fr(g) \wedge (r{=}\mathsf{A} \vee \exists x[in(b,x) \wedge x{\neq}\mathsf{A}])\,,\\
\Psi^{a_1}_{\text{go1}} &= \exists xy[ca(x,y)] \wedge \exists x[fr(x)] \wedge\\
&\qquad \forall xy[in(x,y) \Rightarrow y = \mathsf{A}] \wedge r_1 \neq \mathsf{A} \wedge r_2 = \mathsf{A}\,,\\
\Psi^{a_1}_{\text{go2}} &= \exists xy[ca(x,y)] \wedge \forall x[\neg fr(x)] \wedge r_1 \neq \mathsf{A} \wedge r_2 = \mathsf{A}\,,\\
\Psi^{a_1}_{\text{leave}} &= \forall xy[\neg ca(x,y)] \wedge \exists x[fr(x)] \wedge r_1 = \mathsf{A} \wedge r_2 \neq \mathsf{A}\,.
\end{aligned}$$

For example, $\Psi^{a_1}_{\text{pick}} = \Psi^{a_3}_{\text{pick}} = \bot$ means that the abstract pick action cannot be instantiated by any ground instance of Move(r_1,r_2) or Drop(b,g,r): the first changes the feature X that is not affected by pick, and the second increases G in contradiction with the effect $G{\downarrow}$.

On the other hand, $\Psi^{a_2}_{\text{pick}} = \forall x[\neg ca(b,x)] \wedge r \neq \mathsf{A}$ means that the ground action Pick(b,g,r) instantiates the pick action when $r \neq \mathsf{A}$, otherwise the effect $B{\downarrow}$ is not achieved, and when the ball b is not being held by any gripper x, otherwise $C{\uparrow}$ is not met. $\Psi^{a_2}_{\text{pick}}$ is logically implied at reachable states by the preconditions of Pick(b,g,B) and the mutex information that is polynomially computable. Indeed, the preconditions are $at(\mathsf{B})$, $in(b,\mathsf{B})$ and $fr(g)$, while the mutex invariants include $\neg in(b,r) \vee \neg ca(b,g)$ for any b, r, and g. Actually, we can show that the mutex information is enough to show Pre$(\bar{a}) \Rightarrow \exists\bar{z}(\bigwedge_i \Psi^{a_i}_{\bar{a}})$ for all the actions $\bar{a}$ in the abstraction. Hence, the abstraction is sound for any instance of Gripper.

7.2 Connectivity in Graphs

We now consider a graph problem that involves the connectivity of two designated vertices $\mathtt{s}$ and $\mathtt{t}$. The domain $\mathcal{D}$ has constants $\mathtt{s}$ and $\mathtt{t}$, a single binary predicate $E(x,y)$, and a single action schema $\mathrm{Link}(x, y)$ that adds $E(x, y)$ and has no precondition. The exact form of the initial situation or goal is not relevant in the following discussion.

The abstraction $\bar{Q}$ defines two features: a boolean feature $conn$ that is true iff s and t are connected, and a numerical feature n that counts the total number of edges in the graph. These features are defined with the concepts:

$$\begin{aligned} conn &= \{(x, y) : E^*(x, y) \land x = \mathsf{s} \land y = \mathsf{t}\}\,, \\ n &= \{(x, y) : E(x, y)\}\,. \end{aligned}$$

There is a single abstract action $\bar{a} = \langle; n\uparrow\rangle$. Since $conn$ exists as a feature in $\bar{Q}$ and $\bar{a}$ does not affect it, no instantiation of $\bar{a}$ may modify the st-connectivity of the graph. The guarantee $\Phi_{\bar{a}}$ for $\bar{a}$ is then

$$\exists z_1 z_2 \big[\neg E(z_1, z_2) \land (E^*(\mathsf{s}, z_1) \land E^*(z_2, \mathsf{t}) \Rightarrow E^*(\mathsf{s}, \mathsf{t}))\big]\,.$$

That is, a sufficient condition for $\mathrm{Link}(z_1, z_2)$ to instantiate $\bar{a}$, and thus increase n and leave $conn$ intact, is that there should be no edge between z_1 and z_2, and there should be a path $\mathsf{s} \rightsquigarrow \mathsf{t}$ if there are paths $\mathsf{s} \rightsquigarrow z_1$ and $z_2 \rightsquigarrow \mathsf{t}$. The first condition entails that the number of edges in the graph indeed increases after applying $\mathrm{Link}(z_1, z_2)$, while the second entails that the truth value of $conn$ does not change with the application of the action: either it was true and remains true, or it was false and remains false.

On other hand, the synthesis of the necessary condition yields $\neg E(z_1, z_2)$ which is quite weak. The reason is that the formula $\mathcal{B}_S(a, p^*)$ in Table 1 is not strong enough. A better necessary condition is obtained when the following term is added to the disjunction in $\mathcal{B}_S(a, p^*)$:

$$\begin{aligned} &\forall uv [\![\neg p(u, v) \notin \mathrm{Post}]\!] \land \\ &\qquad \exists uv ([\![p(u, v) \in \mathrm{Post}]\!] \land p^*(x, u) \land p^*(v, y))\,. \end{aligned}$$

This term says that the action removes no edge from the graph, and adds one edge (u, v) for existing paths $x \rightsquigarrow u$ and $v \rightsquigarrow y$. Clearly, if this condition is met, the graph has a path $x \rightsquigarrow y$ after the application of the action. The resulting base thus remains valid for any domain $\mathcal{D}$ and, in some cases, it provides tighter conditions. Indeed, with this amendment, the necessary condition becomes equal to the sufficient condition $\Phi_{\bar{a}}$.

Finally, observe that these conditions are not invariants. The reason is that the abstraction is *not* sound since there are configurations (states) in which no edge can be added to the graph without altering the st-connectivity. In such states, the abstract action $\bar{a}$ is still applicable, as it does not have any precondition, but no $\mathrm{Link}(z_1, z_2)$ action instantiates it.

8 Discussion and Future Work

Abstractions for generalized planning can be inductively obtained from small samples of transitions from one or several instances of planning problems. Although there are no guarantees on the soundness of the abstractions, we have shown that the abstractions contain usable information about the intended planning instances. Indeed, by analyzing the abstraction with respect to the planning domain, we have shown how to obtain formulas that capture some of the assumptions being made by the abstraction. These assumptions can be either sufficient or necessary. A sufficient condition is a guarantee for generalization. Necessary conditions may be used to show that an instance is not captured by the abstraction.

Sufficient conditions may also be used to improve the search for instantiations of an abstract action $\bar{a}$ that is applicable in a state s in instance P. In the worst case, one needs to iterate over every grounded action $a(\bar{o})$ to find an instantiation of $\bar{a}$. However, if P satisfies the sufficient condition, it is enough to find a tuple of objects $\bar{o}$ such that $s \vDash \Phi_{\bar{a}}(\bar{o})$, something that may be easier to find than to iterate over the set of grounded actions.

The information provided by invariants has been exploited in planning for different purposes and it is essential in some planning paradigms. The automatic synthesis of invariants is a computationally hard problem, and many of the existing techniques are based on a generate and test approach that often yields an incomplete set of invariants [Fox and Long, 1998; Gerevini and Schubert, 1998; Rintanen, 2000; Rintanen, 2008; Helmert, 2009]. We have shown how to use the synthesis of guarantees for abstractions as a method for generating candidates for invariants. Hence, the computation of abstractions for generalized planning may be relevant even when the focus is to learn invariants rather than to solve generalized planning problems. As seen in the examples, some of these invariants are quite complex and out-of-reach for state-of-the-art methods for invariant synthesis.

There are clear directions for future work. First, come up with better bases for translation, based on either p^+ or p^*, and understand better the strengths and weaknesses of different bases. Second, even though we are able to handle concepts that are more general than those generated from concept grammars, we are not yet able to accommodate distance features as defined by Bonet *et al.* [2019a]. Finally, it may be the time to bring theorem provers into the pipeline of generalized planning: from samples we obtain abstractions $\bar{Q}$ by an inductive process, a solution $\bar{\pi}$ for $\bar{Q}$ is computed with a FOND planner, and an instance P is then assured to be solved by $\bar{\pi}$ if P satisfies the guarantee for $\bar{Q}$. The latter may be automated with the help of theorem provers.

Acknowledgments

B. Bonet is supported by a Banco Santander – UC3M Chair of Excellence Award. This work is partially funded by grants TIN2017-88476-C2-2-R and RTC-2017-6753-4 of the Spanish Ministerio de Economía, Industria y Competitividad.

References

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The description logic handbook: Theory, implementation and applications*. Cambridge University Press, 2003.

[Belle and Levesque, 2016] Vaishak Belle and Hector J. Levesque. Foundations for generalized planning in unbounded stochastic domains. In *Proc. 15th Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 380–389, 2016.

[Bonet and Geffner, 2018] Blai Bonet and Hector Geffner. Features, projections, and representation change for generalized planning. In *Proc. 27th Int. Joint Conf. on Artificial Intelligence*, pages 4667–4673, 2018.

[Bonet *et al.*, 2017] Blai Bonet, Giuseppe De Giacomo, Hector Geffner, and Sasha Rubin. Generalized planning: Nondeterministic abstractions and trajectory constraints. In *Proc. 26th Int. Joint Conf. on Artificial Intelligence*, pages 873–879, 2017.

[Bonet *et al.*, 2019a] Blai Bonet, Guillem Francès, and Hector Geffner. Learning features and abstract actions for computing generalized plans. In *Proc. 33rd AAAI Conf. on Artificial Intelligence*, 2019.

[Bonet *et al.*, 2019b] Blai Bonet, Raquel Fuentetaja, Yolanda E-Martin, and Daneil Borrajo. Guarantees for sound abstractions for generalized planning (extended paper). *arXiv e-prints*, page arXiv:1905.12071, 2019.

[Boutilier *et al.*, 2001] Craig Boutilier, Ray Reiter, and Bob Price. Symbolic dynamic programming for first-order MDPs. In *Proc. 17th Int. Joint Conf. on Artificial Intelligence*, pages 690–700, 2001.

[Fox and Long, 1998] Maria Fox and Derek Long. The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research*, 9:317–371, 1998.

[Geffner and Bonet, 2013] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[Gerevini and Schubert, 1998] Alfonso Gerevini and Lenhart Schubert. Inferring state constraints for domain-independent planning. In *Proc. 15th Nat. Conf. on Artificial Intelligence*, pages 905–912, 1998.

[Helmert, 2002] Malte Helmert. Decidability and undecidability results for planning with numerical state variables. In *Proc. 6th Int. Conf. on Artificial Intelligence Planning Systems*, pages 44–53, 2002.

[Helmert, 2009] Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173:503–535, 2009.

[Hu and De Giacomo, 2011] Yuxiao Hu and Giuseppe De Giacomo. Generalized planning: Synthesizing plans that work for multiple environments. In *Proc. 22nd Int. Joint Conf. on Artificial Intelligence*, pages 918–923, 2011.

[van Otterlo, 2012] Martijn van Otterlo. Solving relational and first-order logical markov decision processes: A survey. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning*, pages 253–292. Springer, 2012.

[Rintanen, 2000] Jussi Rintanen. An iterative algorithm for synthesizing invariants. In *Proc. 17th Nat. Conf. on Artificial Intelligence*, pages 806–811, 2000.

[Rintanen, 2008] Jussi Rintanen. Regression for classical and nondeterministic planning. In *Proc. 18th European Conf. on Artificial Intelligence*, pages 568–572, 2008.

[Segovia *et al.*, 2016] Javier Segovia, Sergio Jiménez, and Anders Jonsson. Generalized planning with procedural domain control knowledge. In *Proc. 26nd Int. Conf. on Automated Planning and Scheduling*, pages 285–293, 2016.

[Slind and Norrish, 2008] Konrad Slind and Michael Norrish. A brief overview of HOL4. In *Proc. 21st Int. Conf. on Theorem Proving in Higher Order Logics (TPHOLs)*, volume LNCS 5170, pages 28–32, 2008.

[Srivastava *et al.*, 2008] Siddarth Srivastava, Neil Immerman, and Shlomo Zilberstein. Learning generalized plans using abstract counting. In *Proc. 23rd AAAI Conf. on Artificial Intelligence*, pages 991–997, 2008.

[Srivastava *et al.*, 2011a] Siddarth Srivastava, Neil Immerman, and Shlomo Zilberstein. A new representation and associated algorithms for generalized planning. *Artificial Intelligence*, 175(2):615–647, 2011.

[Srivastava *et al.*, 2011b] Siddarth Srivastava, Shlomo Zilberstein, Neil Immerman, and Hector Geffner. Qualitative numeric planning. In *Proc. 25th AAAI Conf. on Artificial Intelligence*, pages 1010–1016, 2011.

[Vardi, 1982] Moshe Y. Vardi. The complexity of relational query languages. In *Proc. 14th Annual ACM Symp. on Theory of Computing (STOC)*, pages 137–146, 1982.

[Wang *et al.*, 2008] Chenggang Wang, Saket Joshi, and Roni Khardon. First order decision diagrams for relational MDPs. *Journal of Artificial Intelligence Research*, 31:431–472, 2008.