Extracting Mutual Exclusion Invariants from Lifted Temporal Planning Domains

Sara Bernardini[a], Fabio Fagnani[b], David E. Smith[c]

[a]*Department of Computer Science, Royal Holloway University of London,*
Egham, Surrey, TW20 0EX, UK
[b]*Department of Mathematical Sciences (DISMA), Politecnico di Torino*
Corso Duca degli Abruzzi, 24, 10129 Torino, Italy
[c]*Intelligent Systems Division, NASA Ames Research Center, Moffett Field, CA 94035*

Abstract

We present a technique for automatically extracting mutual exclusion invariants from temporal planning instances. It first identifies a set of invariant templates by inspecting the lifted representation of the domain and then checks these templates against properties that assure invariance. Our technique builds on other approaches to invariant synthesis presented in the literature, but departs from their limited focus on instantaneous actions by addressing temporal domains. To deal with time, we formulate invariance conditions that account for the entire temporal structure of the actions and the possible concurrent interactions between them. As a result, we construct a more comprehensive technique than previous methods, which is able to find not only invariants for temporal domains, but also a broader set of invariants for sequential domains. Our experimental results provide evidence that identifying a broader set of invariants results in the generation of fewer multi-valued state variables. We show that, in turn, this reduction in the number of variables reflects positively on the performance of a number of temporal planners that use a variable/value representation.

Keywords: Automated Planning, Temporal Planning, Mutual Exclusion Invariants, Automatic Domain Analysis

1. Introduction

This paper presents a technique for synthesising mutual exclusion invariants from temporal planning domains expressed in PDDL2.1 (Fox and Long, 2003). A mutual exclusion invariant over a set of ground atoms means that at most one atom in the set is true at any given moment. Mutual exclusion invariants can be expressed as multi-valued state variables by adding a special "none of the above" value so that, at all moments, precisely one value holds. For instance, consider the *Floortile* domain from the 8th International Planning Competition (IPC'14 - see Appendix A). A mutual exclusion invariant for this domain states that two ground atoms that indicate the position of a robot can never be true at the same time. Intuitively, this means that

Email addresses: `sara.bernardini@kcl.ac.uk` (Sara Bernardini), `fabio.fagnani@polito.it` (Fabio Fagnani), `david.smith@nasa.gov` (David E. Smith)

a robot cannot be at two different locations simultaneously. To give a concrete case, consider a planning problem for the *Floortile* domain with one robot `r1` and three locations, `t1`, `t2` and `t3`. We can create a state variable that indicates the position of `r1` with a domain of three values: `robotAt_r1_t1`, `robotAt_r1_t2` and `robotAt_r1_t3`.

Although a number of approaches to invariant synthesis have been proposed so far (Gerevini and Schubert, 2000; Rintanen, 2000, 2008; Fox and Long, 1998; Helmert, 2009), they are limited in scope because they deal with sequencial domains only. Recently, Rintanen (2014) has proposed a technique for temporal domains, but this technique does not scale to complex problems because it requires grounding the domain. We address both limitations. We find invariants for temporal domains by applying an algorithm that works at the lifted level of the representation and, in consequence, is very efficient and scales to large instances.

Our invariant synthesis builds on Helmert (2009), which presents a technique to translate the non-temporal subset of PDDL2.2 (Edelkamp and Hoffmann, 2004) into the Finite Domain Representation (FDR), a multi-valued planning task formalism used by Fast Downward (Helmert, 2006). Since finding invariants for temporal tasks is much more complex than for tasks with instantaneous actions, a simple generalisation of Helmert's technique to temporal settings does not work. In the temporal case, simultaneity and interference between concurrent actions can occur, hence our algorithm cannot check actions individually against the invariance conditions, but needs to consider the entire set of actions and their possible intertwinements over time. In capturing the temporal case, we formulate invariance conditions that take into account the entire structure of the action schemas as well as the possible interactions between them. As a result, we construct a significantly more comprehensive technique that is able to find not only invariants for temporal domains, but also a broader set of invariants for sequencial domains.

We describe our approach in two major steps. First, we provide a general theory at the ground level and offer results that insure invariance under two types of properties: safety conditions for individual instantaneous and durative actions as well as collective conditions that prevent dangerous intertwinements between durative actions. Then, we lift these results at the level of schemas so that all checks needed for verifying invariance can be performed at this higher level, without the need of grounding the domain. Complexity of such checks are of linear or low polynomial order in terms of the number of schemas and literals appearing in the domain.

1.1. Motivations

Automated planning is a well-established field of artificial intelligence and, over more than fifty years since its appearance, several paradigms have emerged. One fundamental difference between these paradigms is whether time is treated implicitly or explicitly. While classical planning focuses on the causal evolution of the world, temporal planning is concerned with the temporal properties of the world. In classical planning, actions are considered to be instantaneous, whereas in temporal planning actions have durations and can be executed concurrently. Another important difference between planning paradigms relates to whether the world is modelled by adopting a Boolean propositional representation or a representation based on multi-valued state variables. The majority of the work in planning has been devoted to classical planning with domains expressed through propositional languages, and in particular PDDL (McDermott, 2000) and its successors (Fox and Long, 2003), the language of the IPC. However, in parallel with the development of classical propositional planning, a number of temporal planning systems have been proposed for coping with practical problems, especially space mission operations (Frank and Jónsson, 2003; Chien et al., 2000; Ghallab and Laruelle, 1994; Muscettola, 1994; Fratini

	Propositional	Variable/Value
Classical	HSP (Bonet and Geffner, 2001) FF (Hoffmann and Nebel, 2001) YAHSP (Vidal, 2004)	FD (Helmert, 2006) LAMA (Richter and Westphal, 2010)
Temporal	LPG (Gerevini et al., 2006) POPF (Coles et al., 2010)	TFD (Eyerich et al., 2009) EUROPA2 (Frank and Jónsson, 2003)

Table 1: Examples of planners and their classification based on whether they treat time explicitly or implicitly and whether they use a Boolean propositional representation or a multi-valued state variable representation.

et al., 2008). Typically, these systems use variable/value representations. Table 1 shows a classification of several well-known planners based on these different characteristics.

Recently, a few techniques have been proposed for translating propositional representations into variable/value representations (Helmert, 2006; Bernardini and Smith, 2008b; Rintanen, 2014). A central task of all these techniques is the generation of state variables from propositions and actions. The basic procedure to do this (which we use as the baseline in our experiments) relies on generating one state variable with two values, true and false, for each proposition. Naturally, such translation produces no performance advantage. A more sophisticated strategy, which produces compact and optimised encodings, rests on extracting mutual exclusion invariants from propositional domains and using such invariants to generate multi-valued state variables. This is the focus of our work.

These translation techniques are important as they allow fair testing of planners developed for variable/value representations on PDDL benchmarks (which are propositional). The practical issue is that planners that permit variable/value representation need this feature to be well-exploited and perform competitively. Since translation between the two different representations can be cheaply automated, there is no reason to reject providing the richer representations to those planners that accept them (if the translation was expensive, one might reasonably argue about the fairness of this process). In consequence, these translating techniques are extremely useful for comparing alternative planning paradigms and for promoting cross-fertilisation of ideas between different planning communities, which is our primary motivation.

However, the importance of these translation techniques goes beyond the engineering of a bridge between different input languages. In transforming propositional representations into state variable representations, they generate new domain knowledge, where *new* means *accessible* in this context. Effectively, these techniques turn into internal mini theorem provers since, rather than merely translating, they firstly selectively explore the deductive closure of the original theory to find theorems that permit optimising the representation and secondly execute those optimisations. We will show that the cost of performing these optimisations is worth it because it is very fast and can be amortised over many problems.

Mutual exclusion invariants are also beneficial in pruning the search space for search methods such as symbolic techniques based on SAT (Kautz and Selman, 1999; Huang et al., 2010) and backward chaining search (Blum and Furst, 1997). In addition, as invariants help to reduce the number of variables required to encode a domain, they are used in planning systems based on binary decision diagrams (BDDs) (Edelkamp and Helmert, 2001), constraint programming (Do and Kambhampati, 2001), causal graph heuristics (Helmert, 2006) and pattern databases (Haslum et al., 2007).

Finally, from a knowledge engineering perspective, the invariant synthesis presented in this paper can be used as a powerful tool for debugging temporal planning domains. We do not focus on this specifically in this paper, but presents a case study in Example 19. As shown in Cushing et al. (2007), several temporal planning tasks developed for the various IPC editions are bugged with the consequence that the planners take a long time to solve them, when they actually manage to do so. As invariants capture intuitive properties of the physical systems described in the domains, it is easy for a domain expert to identify modelling mistakes by inspecting them. Discrepancies between the invariants found by the automatic technique and those that the expert expects to see for a given domain indicate that the domain does not encode the physical system correctly. In consequence, the expert can revise the domain and repair it. For example, considering the *Rover* domain, we expect that a `store` could be either full or empty at any time point. However, the invariant synthesis does not produce an invariant with the atoms `full` and `empty`. It can be shown that this is because the action `drop` is not properly modelled. Our technique not only alerts the expert that the system is not properly modelled, but also refers the expert to the action that is not encoded correctly. This is a useful feature to fix modelling errors quickly and safely.

1.2. Contributions of the paper

In brief, the contributions of this paper are the following.

From a theoretical point of view:

- We give the first formal account of a mutual exclusion invariant synthesis for temporal domains that works at the lifted level of the representation. Our presentation of this topic is rigorous and comprehensive and our theory is general.
- Our technique is based on inferring general properties of the state space by studying the structure of the action schemas and the lifted relations in the domain, without the need to ground it. This is generally a hard task. Our theoretical framework is sophisticated, but it results in practical tools that have high efficiency and low computational cost.

From a practical point of view:

- We provide a tool for optimising the generation of state variables from propositions and actions (both instantaneous and durative). This results in compact encodings that benefit the performance of planners, as we will show in our experimental results (see Section 10).
- We offer a technique that can be used as a debugging tool for temporal planning domains. As this type of domains are particularly challenging to encode, especially when large and complex, a rigorous debugging process is crucial in producing correct representations of the systems under consideration (see Example 19).

1.3. Organisation of the paper

This paper is organised as follows. After presenting PDDL2.1, our input language, in Section 2, we formally introduce the notion of invariance in Section 3.

Sections 4, 5 and 6 are devoted to a detailed analysis of actions at the ground level. In particular, Section 4 focuses on instantaneous actions: the fundamental concept of *strong safety* is introduced and analysed and a first sufficient result for invariance, Corollary 23, is established. Section 5 analyses *sequences* of actions and, in particular, durative actions (seen as a sequence of three instantaneous actions) for which two new concepts of safety are formulated and investigated: *individual* and *simple safety*. Our main technical results are presented in Section 6 and consist of Theorems 51 and 53 and Corollary 58: these results ensure invariance under milder safety requirements on the durative actions than Corollary 23. This is obtained by adding requirements that prevent the intertwinement of durative actions that are not strongly safe.

Sections 7 and 8 *lift* the concepts and results obtained in the previous sections to the level of action *schemas*. In particular, Section 7 deals with the problem of lifting the concept of strong safety for instantaneous schemas, while Section 8 considers durative action schemas and presents the lifted version of our main results, Corollaries 92, 97 and 98.

These results are the basic ingredients of our algorithm to find invariants, which we present in Section 9. Section 10 reports an extensive experimental evaluation of our approach against the domains of all the temporal IPCs. Sections 11 and 12 conclude the paper with a description of related works and closing remarks. There are four appendices, A – D, that contain the specifications of the planning domains used as the running examples in the paper.

2. Canonical Form of Planning Tasks

In this work, we consider planning instances that are expressed in PDDL2.1 (Fox and Long, 2003). However, before applying our algorithm to find invariants, we manipulate the domain to enforce a regular structure in the specification of the action schemas. In Section 2.1, we give a detailed account of this *canonical* form that we use and then, in Section 2.2, we describe how such a form can be obtained starting from a domain expressed in PDDL2.1.

2.1. PDDL Canonical Form

Let us consider a first order language $\mathcal{L}$ with a denumerable set of individual *variable* symbols $\mathcal{V} = \{v_1, v_2, \ldots\}$ and a signature that has a denumerable set of *constants* $\mathcal{K} = \{k_1, k_2, \ldots\}$ and a denumerable set of *relation* symbols $\mathcal{Z}$, where each $r \in \mathcal{Z}$ is associated with a positive integer called *arity* and indicated as *arity*(r).

Given the language $\mathcal{L}$, a planning *instance* is a tuple $\mathcal{I} = (\mathcal{D}, \mathcal{P})$, where $\mathcal{D}$ is a planning *domain* and $\mathcal{P}$ a planning *problem*. The domain $\mathcal{D} = (\mathcal{R}, \mathcal{A}^{\mathrm{i}}, \mathcal{A}^{\mathrm{d}})$ is a 3-tuple consisting of finite sets of relation symbols $\mathcal{R} \subseteq \mathcal{Z}$, instantaneous and durative actions. The problem $\mathcal{P} = (O, \mathit{Init}, \mathcal{G})$ is a triple consisting of the objects $O \subseteq \mathcal{K}$, the initial logical state and the logical goal specification.

The *ground atoms* of the planning instance, *Atms*, are the (finitely many) atomic formulas formed by applying the relations in $\mathcal{R}$ to the objects in O (respecting arities). A *logical state* is any subset of *Atms*. Considering a logical state s, we denote with s^c its complement: $s^c = \mathit{Atms} \setminus s$. $\mathcal{S} = 2^{\mathit{Atms}}$ denotes the set of logical states. The initial logical state can be chosen arbitrarily: $\mathit{Init} \in \mathcal{S}$. A logical goal specification is any choice of a desired set of final logical states: $\mathcal{G} \subseteq \mathcal{S}$. It is typical to restrict to considering goals only of the form $\mathcal{G} = \{S \mid C \subseteq S \in \mathcal{S}\}$ for some conjunction of atoms $C \subseteq \mathit{Atms}$.

A *state* is a tuple in $\mathbb{R} \times \mathcal{S}$, where the first value is the *time* of the state and the second value (*logical state*) is a subset of *Atms*. The initial state for the planning instance $\mathcal{I}$ is, implicitly, of the form (t_0, Init) where t_0, the beginning of the plan execution time, will always take the value $t_0 = 0$ in this paper, which is also the convention used in the IPC benchmarks[1].

The set $\mathcal{A}^i$ is a collection of *instantaneous* action schemas. An instantaneous action schema α is composed of the following sets:

- $V_\alpha \subseteq \mathcal{V}$, the distinct schema's *variables*;
- Pre^+_α, the *positive preconditions*;
- Pre^-_α, the *negative preconditions*;
- Eff^+_α, the *add effects*;
- Eff^-_α, the *delete effects*.

If the schema α is clear in context we drop the subscript. For convenience, we also define:

- $\mathit{Pre}_\alpha = \mathit{Pre}^+_\alpha \cup \mathit{Pre}^-_\alpha$;
- $\mathit{Eff}_\alpha = \mathit{Eff}^+_\alpha \cup \mathit{Eff}^-_\alpha$

Preconditions and effects are sets of formulas l of the form: $(\forall v_1, \ldots, v_k : q)$ where:

- q is an atomic formula: $q = r(v'_1, \ldots, v'_n)$ with $r \in \mathcal{R}$ and $\mathit{arity}(r) = n \geq k$;
- $\{v_1, \ldots, v_k\} \subseteq \{v'_1, \ldots, v'_n\} \subseteq \mathcal{V}$ are the quantified variables in l;
- $\{v'_1, \ldots, v'_n\} \setminus \{v_1, \ldots, v_k\} \subseteq V_\alpha$ are the schema's variables in l.

The universal quantification can be trivial (i.e. quantification over zero variables) and, in this case, it is omitted. Note that we do not allow repeated arguments in the specification of a schema. The representation is untyped.

Given a formula l of the form $(\forall v_1, \ldots, v_k : q)$, we indicate the sets of the positions of its free and quantified variables (starting with zero and in the order they appear in q) respectively as $\mathrm{VarF}[l]$ and $\mathrm{VarQ}[l]$. For example, if $l = (\forall x, z : f(x, y, z, k))$, we have: $\mathrm{VarQ}[l] = \{0, 2\}$ and $\mathrm{VarF}[l] = \{1, 3\}$. Let $\mathrm{Rel}[l] = \langle r/k \rangle$ denote the relation symbol r of arity k that appears in the atomic formula q. In our example, $\mathrm{Rel}[l] = f/4$. Considering a position $i \in \mathrm{VarF}[l]$, we indicate the corresponding variable as $\mathit{Var}[i, l]$. For instance, $\mathit{Var}[2, l] = z$.

The set $\mathcal{A}^d$ is a collection of *durative* action schemas. A durative action schema $D\alpha$ is a triple of instantaneous action schemas $D\alpha = (\alpha^{\mathrm{st}}, \alpha^{\mathrm{inv}}, \alpha^{\mathrm{end}})$ with a common set of variables $V_{D\alpha}$ (i.e. $V_{D\alpha} = V_{\alpha^{\mathrm{st}}} = V_{\alpha^{\mathrm{inv}}} = V_{\alpha^{\mathrm{end}}}$) and with α^{inv} having no effects (i.e. $\mathit{Eff}_{\alpha^{\mathrm{inv}}} = \emptyset$). We indicate as $\{D\alpha\}$ the set $\{\alpha^{\mathrm{st}}, \alpha^{\mathrm{inv}}, \alpha^{\mathrm{end}}\}$.

We call $\mathcal{A}$ the set of all the instantaneous action schemas in the domain, including those induced by durative actions: $\mathcal{A} = \mathcal{A}^i \cup \bigcup_{D\alpha \in \mathcal{A}^d} \{D\alpha\}$. Consider any two action schemas α_1 and α_2 in $\mathcal{A}$ such that it does not exist a durative action $D\alpha$ with both α_1 and α_2 in $\{D\alpha\}$. We assume that the variables of α_1 and α_2 are disjoint sets[2], i.e. $V_{\alpha_1} \cap V_{\alpha_2} = \emptyset$.

[1] We write t_0 to emphasise that the choice of a starting time is theoretically unimportant.

[2] Our implementation forces this assumption true by performing a preprocessing step that appropriately renames all variables of unrelated action schemas apart.

Given an action schema $\alpha \in \mathcal{A}^i$ with variables V_α, consider a *grounding* function $gr : V_\alpha \to O$ that maps the schema's variables in α to the problem's objects O. The function gr induces a function on the formulas in α as follows. Take a formula l that appears in α. We call $\widetilde{gr}(l)$ the formula that is obtained from l by substituting the schema's variables in l with objects in O according to gr. We call $gr(l)$ the set of ground atoms obtained from $\widetilde{gr}(l)$ by substituting each quantified variable in l with objects in O in all possible ways. Note that, when there are no quantified variables, $\{\widetilde{gr}(l)\} = gr(l)$ and they are singletons. For a set L containing formulas $l_1, \ldots, l_n$, we call $\widetilde{gr}(L) = \{\widetilde{gr}(l_1), \ldots, \widetilde{gr}(l_n)\}$ and $gr(L) = gr(l_1) \cup \ldots \cup gr(l_n)$. We call $\widetilde{gr}(\alpha)$ the action schema obtained from α by grounding each formula l that appears in α according to gr and $gr(\alpha)$ the ground action that is obtained from $\widetilde{gr}(\alpha)$ by replacing the quantified variables with the set of ground atoms formed by substituting objects in O for the quantified variables in all possible ways.

Considering a durative action schema $D\alpha \in \mathcal{A}^d$ and a grounding function gr, the ground) durative action $gr(D\alpha)$ is obtained by applying gr to the instantaneous fragments of $D\alpha$, i.e. $gr(D\alpha) = (gr(\alpha^{\text{st}}), gr(\alpha^{\text{inv}}), gr(\alpha^{\text{end}}))$. Note that we cannot apply different grounding functions to different parts of a durative action schema.

We indicate the positive and negative preconditions of an instantaneous ground action a as $Pre_a^\pm$ and its add and delete effects as $Eff_a^\pm$. We also put $Pre_a = Pre_a^+ \cup Pre_a^-$ and $Eff_a = Eff_a^+ \cup Eff_a^-$. Ground actions obtained by grounding different action schema are always assumed to be distinct even in the case they might have the same preconditions and effects. In particular, two different durative actions have always distinct start and the end fragments. Such fragments are also distinct from any other instantaneous action. We call $\mathcal{GA}^i$ and $\mathcal{GA}^d$, respectively, the set of instantaneous and durative ground actions. Finally, we call $\mathcal{GA}$ the set of all ground actions in $\mathcal{I}$ (obtained from grounding all schemas in $\mathcal{A}$).

For the sake of simplicity, from now on we will call a ground action simply an *action*, while at the lifted level we will use the term *action schema*. Moreover, the term instantaneous will be dropped, whenever this does not cause any ambiguity, assuming that actions and action schemas without the appellative durative are always instantaneous.

An action a is *applicable* in a logical state s if $Pre_a^+ \subseteq s$ and $Pre_a^- \cap s = \emptyset$. We denote by S_a the set of states on which a is applicable. The *result* of applying a in s is the state s' such that $s' = (s \setminus Eff_a^-) \cup Eff_a^+$. We call ξ this transition function: $s' = \xi(s, a)$.

The transition function ξ can be generalised to a finite set of actions A to be executed simultaneously: $s' = \xi(s, A)$. However, in order to handle simultaneous actions, we need to introduce the so-called *no moving targets* rule: no two actions can simultaneously make use of a value if one of the two is accessing the value to update it. The value is a *moving target* for the other action to access. This rule avoids conflicting effects, but also applies to the preconditions of an action: no concurrent actions can affect the parts of the state relevant to the precondition tests of other actions in the set (regardless of whether those effects might be harmful or not). In formula, two actions a and b are *non-interfering* if:

$$Pre_a \cap Eff_b = Pre_b \cap Eff_a = \emptyset$$

$$Eff_a^+ \cap Eff_b^- = Eff_b^+ \cap Eff_a^- = \emptyset$$

If two actions are not non-interfering, they are *mutex*.

In this work, whenever we consider a set of simultaneous actions A, we implicitly assume that the component actions are pairwise non-interfering. Moreover, we define

$$Pre_A^{\pm} = \bigcup_{a\in A} Pre_a^{\pm}, \quad Eff_A^{\pm} = \bigcup_{a\in A} Eff_a^{\pm} \tag{1}$$

We say that A is applicable in a state s if each component $a \in A$ is applicable in S. The set of states on which A is applicable is thus $\mathcal{S}_A = \{s \mid Pre_A^+ \subseteq s, Pre_A^- \subseteq s^c\}$. Given $s \in \mathcal{S}_A$, the transition function $s' = \xi(s, A)$ is defined as follows:

$$s' = (s \setminus Eff_A^-) \cup Eff_A^+$$

We say that A is *executable* if $\mathcal{S}_A \neq \emptyset$ or, equivalently, if

$$Pre_A^+ \cap Pre_A^- = \emptyset \tag{2}$$

Set of actions that are not executable do not play any role in our analysis as they will never appear in executable plans (see below for the exact definition). For this reason, in our analysis, we always restrict to consider executable sets of actions. This implies, in particular, that each single action a that we consider satisfies the condition $Pre_a^+ \cap Pre_a^- = \emptyset$.

The following result shows that the application of a set of actions can always be serialised.

Proposition 1 (Serialisability)**.** *Consider a set of actions A pairwise non-interfering and a logical state $s \in \mathcal{S}_A$. Let $\sigma : \{1, \dots, n\} \to A$ be any permutation of A and consider the sequence of states recursively defined as $s_0 = s$ and $s_k = \xi(s_{k-1}, \sigma(k))$ for $k = 1, \dots, n$. Then,*

(i) Each $\sigma(k)$ is applicable in s_{k-1} (so each s_k is well-defined): $\mathrm{Pre}^+_{\sigma(k)} \subseteq s_{k-1}$ *and* $\mathrm{Pre}^-_{\sigma(k)} \cap s_{k-1} = \emptyset$.

(ii) The final logical state coincides with the state that is obtained by applying the set A, namely $s_n = \xi(s_0, A)$.

Proof. The action $\sigma(1)$ is applicable in s_0 by definition. Assuming that $\sigma(j)$ is applicable in s_{j-1} for $j = 1, \dots, k$, we now show that $\sigma(k+1)$ is applicable in s_k. Note that from the definition of transition function ξ for single actions $s_k = (s_0 \setminus \bigcup_{j=1}^{k} Eff^-_{\sigma(j)}) \cup \bigcup_{j=1}^{k} Eff^+_{\sigma(j)}$. Since $Pre^+_{\sigma(k+1)} \subseteq s_0$ and $Pre^-_{\sigma(k+1)} \cap s_0 = \emptyset$ by assumption and $\sigma(k+1)$ is not interfering with $\sigma(1), \sigma(2), \dots, \sigma(k)$, we have that $Pre^+_{\sigma(k+1)} \subseteq s_k$ and $Pre^-_{\sigma(k+1)} \cap s_k = \emptyset$. Also, note that: $s_n = (s_0 \setminus \bigcup_{j=1}^{n} Eff^-_{\sigma(j)}) \cup \bigcup_{j=1}^{n} Eff^+_{\sigma(j)} = \xi(s, A)$. □

An *instantaneous timed action* has the following syntactic form: (t, a), where t is a positive rational number in floating point syntax and a is an action. A *durative timed action* has the following syntactic form: $(t, Da[t'])$, where t is a rational valued time point, Da is a durative action and t' is a non-negative rational-valued duration. It is possible for multiple timed actions to be given the same time stamp, indicating that they should be executed concurrently.

Given a planning instance $\mathcal{I}$, a *plan* Π consists of a finite set of (instantaneous and durative) timed actions.

The *simple plan* π induced by a plan Π is the set of instantaneous timed actions, as described below:

(i) $(t, a) \in \pi$ for each $(t, a) \in \Pi$, where a is an action;

(ii) $(t, a^{\mathrm{st}}) \in \pi$ and $(t + t', a^{\mathrm{end}}) \in \pi$ for all pairs $(t, Da[t']) \in \Pi$, where Da is a durative action;

(iii) $((t_i + t_{i+1})/2, a^{\mathrm{inv}}) \in \pi$ for each pair $(t, Da[t']) \in \Pi$ and for each i such that $t \leq t_i < (t + t')$, where t_i and t_{i+1} are in the happening sequence for Π.

A plan Π and its corresponding induced simple plan π is *admissible* if concurrent instantaneous actions are non-interfering between each other and actions happening inside a durative action $Da = (a^{\mathrm{st}}, a^{\mathrm{inv}}, a^{\mathrm{end}})$ are non-interfering with the action a^{inv}. More precisely, if

- $(t, a), (t, b) \in \pi$ imply that a and b are non-interfering.
- $(t, Da[t']) \in \Pi$ and $(s, b) \in \pi$ for some time $s \in (t, t + t')$ imply that a^{inv} and b are non-interfering.

The *happening time sequence* $\{t_i\}_{i=0,\ldots,\bar{k}}$ for a plan π is: $\{t_0\} \cup \{t \mid (t, a) \in \pi\}$. The *happening* at time t of the plan π is defined as $A_t = \{a \in \mathcal{GA} \mid (t, a) \in \pi\}$. Clearly, $A_t \neq \emptyset$ iff t is in the happening time sequence. The sequence of action sets $A_\pi = (A_{t_0}, \ldots, A_{t_{\bar{k}}})$ is called the *happening sequence* of π.

An admissible simple plan π for a planning instance $\mathcal{I}$ is *executable* if, given its happening time sequence $\{t_i\}_{i=0\ldots\bar{k}}$, there is a sequence of logical states $\{s_i\}_{i=0\ldots\bar{k}}$ such that $s_0 = \mathit{Init}$ and for each $i = 0, \ldots, k$, s_{i+1} is the result of executing the happening at time t_i in π. Formally, we have that $A_{t_{i+1}}$ is applicable in s_i and $s_{i+1} = \xi(s_i, A_{t_{i+1}})$. The state $s_{\bar{k}}$ is called the *final logical state* produced by π. The sequence of times and states $\{S_i = (t_i, s_i)_{i=0\ldots k}\}$ is called the (unique) *trace* of π, $\mathit{trace}(\pi)$. Two simple plans are said to be *equivalent* if they give rise to the same trace. We call *Plans* all the executable simple plans for $\mathcal{I}$ and we call $\mathcal{S}_{\mathrm{r}}$ the union of all the logical states that appear in the traces associated with the plans in *Plans*: $\mathcal{S}_{\mathrm{r}} = \{s \mid \pi \in \mathrm{Plans}, (t, s) \in \mathit{trace}(\pi)\}$. Note that $\mathcal{S}_{\mathrm{r}} \subseteq \mathcal{S}$. We call the states in $\mathcal{S}_{\mathrm{r}}$ *reachable states*. Finally, an executable simple plan for a planning instance $\mathcal{I}$ is *valid* if it produces a final state $s_{\bar{k}} \in \mathcal{G}$.

Note that in the passage from the original plan Π to the simple plan π we have formally lost the coupling among the start and end fragments of durative actions. Since in certain cases this information is necessary, we set a definition: a durative action Da is said to happen in π in the time interval $[t, t+t']$ whenever this holds true in the original plan Π, namely when $(t, Da[t']) \in \Pi$. It will also be convenient to make the following assumption. Whenever two durative timed actions $(t_1, Da_1[t'_1])$ $(t_2, Da_2[t''_2])$ either start at the same point $t_1 = t_2$ or end at the same point $t_1 + t'_1 = t_2 + t'_2$, in a plan Π, but have different durations $t'_1 \neq t'_2$ the constituent durative actions are different: $Da_1 \neq Da_2$. This entails no loss of generality as we can always create multiple copies with different labels of the same durative action: they have the same preconditions and effects in each fragment but have a different name. Durative actions of this type are called *equivalent*. Note, finally, that if two equivalent durative actions Da_1 and Da_2 appear in a plan with the same starting point and duration, $(t, Da_1[t']), (t, Da_2[t']) \in \Pi$, we can get rid of one of them and obtain an equivalent simple plan. For this reason, we assume from now on that the plans considered are free from such simultaneous happening of equivalent durative actions.

All concepts and results presented in this paper will not take into consideration the goal $\mathcal{G}$. They will be concerned with the family *Plans* of all executable plans and not just the valid ones. Moreover, it will be convenient to think of *Init* as a parameter taking all possible values in $\mathcal{S}$, as our results will be universally quantified with respect to it. Whenever in this paper we fix an instance $\mathcal{I}$, we think of a family of instances parameterised by all possible *Init* and $\mathcal{G}$.

2.2. From PDDL2.1 to Canonical PDDL

We build the canonical form described above starting from PDDL2.1 instances, which are characterised by numeric and temporal information (Fox and Long, 2003). We do not consider numeric expressions in our canonical form. We could potentially exploit metric information to find additional invariants, but currently we do not do that. Instead, we ignore the numeric expressions in the domain and focus only on its logical and temporal structure. Setting numeric expressions aside has several consequences: we eliminate numeric constraints from the actions' specification, the actions' preconditions and effects do not depend on the duration and actions' durations become the interval $(0, +\infty)$ in the rational numbers. Note that, crucially, the invariants that we find for the domain without numeric constraints are also invariants for the original domain since, in removing them, we are only expanding the set of possible valid plans.

Temporal information are handled in PDDL2.1 by means of *durative* actions. They can be either discretised or continuous, but we focus on discretised durative actions only here. They have a duration field and temporally annotated conditions and effects.

The duration field contains temporal constraints involving terms composed of arithmetic expressions and the dedicated variable *duration*. As already mentioned above, we ignore numeric constraints and consequently the specific durations of the actions, which we substitute with the interval $(0, +\infty)$. Such precise durations are irrelevant to our technique. We care about the possible intertwinement between duration actions, which can be studied without considering the exact durations.

The annotation of a condition makes explicit whether the associated proposition must hold at the *start* of the interval (the point at which the action is applied), the *end* of the interval (the point at which the final effects are asserted) or *over all* the interval (open at both ends) from the start to the end (invariant over the duration of the action). The annotation of an effect makes explicit whether the effect is immediate (it happens at the start of the interval) or delayed (it happens at the end of the interval). No other time points are accessible. Logical changes are considered to be instantaneous and can only happen at the accessible points. To build our canonical form, we transform durative actions into triples of instantaneous actions. We do this in such a way that we do not change the set of plans that can be obtained for any goal specification. Plans with durative actions, in fact, are always assigned a semantics in terms of the semantics of simple plans (Fox and Long, 2003), as explained in the previous section.

Let us see now in more detail how we obtain the PDDL canonical form from PDDL2.1 instances. PDDL2.1 is a typed representation. We compile away types: for each type that occurs in the domain, we introduce a new unary relation with the same name. For example, the *Floortile* domain contains the type robot, tile and color and so we introduce three new unary relations: $\langle \texttt{robot}, 1\rangle$, $\langle \texttt{tile}, 1\rangle$, and $\langle \texttt{color}, 1\rangle$. We use such relations in the specification of the initial state, where we list the objects of the planning instance, and in the specification of the actions. For each typed variable that appears in an action, we specify it without giving its type, but then we introduce a new precondition in the action that associates the variable to its corresponding unary relation. We follow the same procedure described in Helmert (2009), which can be consulted for further details.

In a PDDL2.1 domain, instead of $\mathcal{A}^{\mathrm{i}}$ and $\mathcal{A}^{\mathrm{d}}$, we find a set $\mathcal{A}^{a}$ that contains both instantaneous and durative action schemas, which have the following characteristics. Durative action schemas have temporally annotated conditions and effects, which we indicate as Pre^{px} and Eff^{py}, where $p \in \{+, -\}$, $x \in \{\mathrm{st}, \mathrm{inv}, \mathrm{end}\}$, and $y \in \{\mathrm{st}, \mathrm{end}\}$. For an action schema in $\mathcal{A}^{a}$ (durative or not), the condition formula can be a relation, a negation, a conjunction or disjunction of relations or a quantified formula on relations. The effect formula can be a relation, a negation or a conjunction of relations, a universally quantified formula on relations or a conditional effect formula, which is a tuple formed by a precondition formula and and effect formula. We manipulate the action schemas in $\mathcal{A}^a$ to obtain $\mathcal{A}^i$ and $\mathcal{A}^d$, where each action schema in these sets has the canonical form described in Section 2.1.

First, we eliminate conditional effects and existentially quantified formulae through an operation referred to as *flattening* (see Fox and Long (2003) for details). These features can be eliminated by applying syntactic transformations with the resulting schemas being equivalent to the original ones. This procedure can potentially lead to an exponential blow-up of the task description.

Given a flattened action schema α, we take the formulas (temporally annotated or not) in its conditions and effects and *normalise* them by using the algorithm introduced by Helmert (2009) (we refer the interested reader to this paper for a full description of the normalisation process[3]). After normalisation, all action schema conditions and effects become sets of formulas l of the form $\forall v_1, \ldots, v_k : q$, where q is an atomic formula and the quantification can be trivial. We indicate by Pre^+_α and Eff^+_α the set of positive formulas that appear positive in α and by Pre^-_α and Eff^-_α the set of positive formulas that appear negative in α.

After flattening and normalisation, we transform the durative action schemas in $\mathcal{A}^a$ into triples of instantaneous action schemas. For each durative action $D\alpha \in A^a$, we create two instantaneous action schemas that correspond to the end points of $D\alpha$, α^{st} and α^{end}, and one that corresponds to the invariant conditions that must hold over that duration of $D\alpha$, α^{inv}. More formally, for a durative action schema $D\alpha$, we create α^{st}, α^{inv} and α^{end} as follows:

α^{st}	α^{inv}	α^{end}
$Pre^+_{\alpha^{\text{st}}} = Pre^{+\text{st}}_{D\alpha}$	$Pre^+_{\alpha^{\text{inv}}} = Pre^{+\text{inv}}_{D\alpha}$	$Pre^+_{\alpha^{\text{end}}} = Pre^{+\text{end}}_{D\alpha}$
$Pre^-_{\alpha^{\text{st}}} = Pre^{-\text{st}}_{D\alpha}$	$Pre^-_{\alpha^{\text{inv}}} = Pre^{-\text{inv}}_{D\alpha}$	$Pre^-_{\alpha^{\text{end}}} = Pre^{-\text{end}}_{D\alpha}$
$Eff^+_{\alpha^{\text{st}}} = Eff^{+\text{st}}_{D\alpha}$	$Eff^+_{\alpha^{\text{inv}}} = \emptyset$	$Eff^+_{\alpha^{\text{end}}} = Eff^{+\text{end}}_{D\alpha}$
$Eff^-_{\alpha^{\text{st}}} = Eff^{-\text{st}}_{D\alpha}$	$Eff^-_{\alpha^{\text{inv}}} = \emptyset$	$Eff^-_{\alpha^{\text{end}}} = Eff^{-\text{end}}_{D\alpha}$

Table 2: Transformation of durative action schemas into triples of instantaneous action schemas.

At this point, we are ready to construct $\mathcal{A}^i$ and $\mathcal{A}^d$ from $\mathcal{A}^a$. We add each flattened and normalised instantaneous action in $\mathcal{A}^a$ to $\mathcal{A}^i$. For each durative action $D\alpha \in \mathcal{A}^a$, after applying flattening and normalisation, we create the corresponding tuple $(\alpha^{\text{st}}, \alpha^{\text{inv}}, \alpha^{\text{end}})$ and add it to $\mathcal{A}^d$.

Consider a planning instance $\mathcal{I}$ in canonical form obtained from a PDDL2.1 instance $\mathcal{I}'$ and a valid plan Π for $\mathcal{I}$. Π can be converted into an equivalent valid plan Π' for $\mathcal{I}'$.

2.3. Running Example: the Floortile Domain

We use the *Floortile* domain as our running example. It has been introduced in the IPC'14 and then reused in 2015. The full PDDL2.1 specification is available in Appendix A. The domain describes a set of robots that use different colours to paint patterns in floor tiles. The robots can move around the floor tiles in four directions (up, down, left and right). Robots paint with one

[3] Our normalisation differs from Helmert (2009) only in that we preliminarily eliminate conditional effects by applying the flattening operation before normalisation and we keep universal quantification in the preconditions. We also apply normalisation not only to formulas appearing in instantaneous actions as in Helmert (2009), but also to temporally annotated formulas in durative actions. We normalise the formulas and leave the temporal annotation unchanged.

color at a time, but can change their spray guns to any available color. Robots can only paint the tile that is in above (up) and below (down) them, and once a tile has been painted no robot can stand on it.

We have the following relations in this domain: $\mathcal{R}$ = { `up,down,right,left,robotAt, robotHas,painted,clear,availableColor` }. They have arity two, except for the last two, which have arity one. `availableColor` indicates whether a colour gun is available to be picked by a robot and `up,down,right,left` indicate the respective positions of two tiles. As we will automatically infer via our invariant synthesis (see Example 1), the relation `clear` in this context means not only that a tile is still unpainted, but also that it is not being painted and is unoccupied.

The set of instantaneous action schemas $\mathcal{A}^i$ is empty, while the set of durative action schemas $\mathcal{A}^d$ is: $\mathcal{A}^d$ = {`changeColor,paintUp,paintDown,up,down,right,left` }.

As an example, the durative action schema `paintUp` corresponds to the following triple: (`paintUp`$^{\text{st}}$, `paintUp`$^{\text{inv}}$, `paintUp`$^{\text{end}}$), where the single instantaneous action schemas have the following specifications:

α	$\mathtt{paintUp}^{\text{st}}$	$\mathtt{paintUp}^{\text{inv}}$	$\mathtt{paintUp}^{\text{end}}$
V_α	$\{\mathtt{r},\mathtt{y},\mathtt{x},\mathtt{c}\}$	$\{\mathtt{r},\mathtt{y},\mathtt{x},\mathtt{c}\}$	$\{\mathtt{r},\mathtt{y},\mathtt{x},\mathtt{c}\}$
Pre^+_α	$\{\mathtt{robotAt}(\mathtt{r},\mathtt{x}), \mathtt{clear}(\mathtt{y}),$ $\mathtt{robot}(\mathtt{r}), \mathtt{tile}(\mathtt{y}),$ $\mathtt{tile}(\mathtt{x}), \mathtt{color}(\mathtt{c})\}$	$\{\mathtt{robotHas}(\mathtt{r},\mathtt{c}), \mathtt{up}(\mathtt{y},\mathtt{x})$ $\mathtt{robot}(\mathtt{r}), \mathtt{tile}(\mathtt{y}),$ $\mathtt{tile}(\mathtt{x}), \mathtt{color}(\mathtt{c})\}$	$\{\mathtt{robot}(\mathtt{r}), \mathtt{tile}(\mathtt{y}),$ $\mathtt{tile}(\mathtt{x}), \mathtt{color}(\mathtt{c})\}$
Pre^-_α	$\emptyset$	$\emptyset$	$\emptyset$
Eff^+_α	$\emptyset$	$\emptyset$	$\{\mathtt{painted}(\mathtt{y},\mathtt{c})\}$
Eff^-_α	$\{\mathtt{clear}(\mathtt{y})\}$	$\emptyset$	$\emptyset$

Table 3: Durative action schema `paintUp` seen as a triple of instantaneous action schemas.

Note that the triple of single instantaneous action schemas in canonical form is obtained from the following PDDL2.1 specification:

```
(:durative-action paintUp
  :parameters (?r - robot ?y - tile ?x - tile ?c - color)
  :duration (= ?duration 2)
  :condition (and (over all (robot-has ?r ?c))
                  (at start (robotAt ?r ?x))
                  (over all (up ?y ?x))
                  (at start (clear ?y)))
  :effect (and (at start (not (clear ?y)))
               (at end (painted ?y ?c))))
```

3. Mutual Exclusion Invariants and Templates

In this section, we formally introduce the concept of mutual exclusion invariant and give examples of them.

In the PDDL2.1 language, an *invariant* of a planning instance is a property of the world states such that when it is satisfied in the initial state *Init*, it is satisfied in all reachable states $\mathcal{S}_r$. For example, for the *Floortile* domain, a trivial invariant says that for each object x, if x is a robot, then x is not a tile. Similar invariants hold for each type defined in the original PDDL domain. A more interesting invariant says that, for any two objects x and y, if `up(x,y)` holds, then `down(y,x)` holds too, but `down(x,y)` does not. It is possible to identify several invariants for the *Floortile* domain, ranging from trivial invariants such as those involving type predicates to very complex invariants.

In this paper, we focus on *mutual exclusion* invariants, which state that a set of ground atoms can never be true at the same time. From now on we assume we have fixed a planning instance $\mathcal{I}$ in PDDL canonical form.

Definition 2 (Mutual Exclusion Invariant). *A set of ground atoms $Z \in \mathcal{S}$ is a* mutual exclusion invariant set *when, if at most one element of Z is true in the initial state, then at most one element of Z is true in any reachable state, namely*

$$|Z \cap \mathit{Init}| \leq 1 \;\Rightarrow\; |Z \cap s| \leq 1,\ \forall s \in \mathcal{S}_{\mathrm{r}}$$

We abuse the distinction between the set Z and a formula such as $\bigwedge_{x,y \in Z} \neg x \vee \neg y$ and call Z a mutual exclusion invariant *directly and, for brevity,* invariant.

Example 1 (*Floortile* domain). *A mutual exclusion invariant for this domain states that two ground atoms indicating the position of a robot identified as* `rbt1`*, such as `robotAt(rbt1, tile1)` and* `robotAt(rbt1,tile2)`*, can never be true at the same time. Intuitively, this means that* `rbt1` *cannot be in two different positions simultaneously. Another more complex invariant states that, given a tile* `tile1`*, a robot* `rbt1` *and a colour* `clr1`*, atoms of the form* `clear(tile1)`, `robotAt(rbt1, tile1)` *and `painted(tile1, clr1)` can never be true at the same time. This means that a tile can be in one of four possible states: not yet painted (clear), already painted, occupied by a robot that is painting an adjacent tile or none of the preceding (which can only be because the tile is being painted).*

Although we aim to find sets of mutually exclusive ground atoms, we often work with relations and action schemas to control complexity. A convenient and compact way for indicating several invariant sets at the same time involves using *invariant templates*, which are defined below, after introducing a few preliminary definitions.

Definition 3 (Component). *A* component *c is a tuple $\langle r/k, p\rangle$, where r is a relation symbol in $\mathcal{R}$ of arity $k = \mathit{arity}(r)$ and $p \in \{0, \ldots, k\}$.*

Take a component $c = \langle r/k, p\rangle$ and a set of variables $v_0, \ldots, v_{k-1}$ and consider the atomic formula $m = r(v_0, \ldots, v_{k-1})$. When $p \leq k-1$, the number p in c represents the position of one of the variables of m, which we call the *counted* variable. When $p = k$, there are no counted variables. The set of the *fixed* variables of c is formally defined as $F_c = \{(c, i) \mid i = 0, \ldots, k-1; i \neq p\}$. We define the set of fixed variables of a set of components $C = \{c_1, c_2, \ldots, c_n\}$ as $F_C = \bigcup\limits_{c \in C} F_c$.

Definition 4 (Admissible Partition). *Given a set of components C and corresponding set of fixed variables F_C, an* admissible partition *of F_C is a partition $\mathcal{F}_C = \{G_1, \ldots, G_s\}$ such that $|G_j \cap F_c| = 1$ for each $c \in C$.*

If two elements (c_1, i) and (c_2, j) of F_C belong to the same set of the partition $\mathcal{F}_C$, we use the notation: $(c_1, i) \sim_{\mathcal{F}_C} (c_2, j)$.

Remark 5. *Note that the existence of an admissible partition of F_C implies that all the components in C have the same number of fixed variables, which is also the number of the sets in the partition. In the special case in which the number of fixed variables in each component is equal to one, there is just one admissible (trivial) partition $\mathcal{F}_C = \{F_C\}$.*

Definition 6 (Template)**.** *A* template $\mathcal{T}$ *is a pair $(C, \mathcal{F}_C)$ such that C is a set of components and $\mathcal{F}_C$ is an admissible partition of F_C. We simply write $\mathcal{T} = (C)$ when the partition is trivial, i.e. $\mathcal{F}_C = \{F_C\}$.*

Definition 7 (Template's Instance)**.** *Given objects O and template $\mathcal{T} = (C, \mathcal{F}_C)$, an* instance *$\gamma : F_C \to O$ of $\mathcal{T}$ maps (all the elements of) each part of its partition to the same object, that is $\gamma(c_a, i) = \gamma(c_b, j)$ if and only if $(c_a, i) \sim_{\mathcal{F}_C} (c_b, j)$ for all (c_a, i), $(c_b, j) \in C$*

Definition 8 (Template's Instantiation)**.** *The* instantiation *of $\mathcal{T}$ according to instance γ, $\gamma(\mathcal{T})$, is the set of ground atoms in 2^{Atms} obtained as follows: for each component $c = \langle r/k, p \rangle$ of $\mathcal{T}$, take the relation symbol r, for each element $(c, i) \in F_c$ bind the variable in position i according to $\gamma(c, i)$ and the counted variable in position p to all the objects O. In formula,*

$$\gamma(\mathcal{T}) = \bigcup_{c=\langle r/k,p \rangle \in \mathcal{T}} \left\{ r(x_0, \ldots, x_{k-1}) \,|\, x_p \in O,\ x_i = \gamma(c, i)\, \forall i \neq p \right\} \tag{3}$$

Instances are interesting because they can be used to reason about their (exponentially larger) instantiations without, in fact, constructing those instantiations.

Considering a template $\mathcal{T}$ and an instance γ, if the ground atoms in the instantiation of $\mathcal{T}$ according to γ are mutually exclusive in the initial state *Init* and remain such in any reachable state $s \in \mathcal{S}_\mathrm{r}$, then $\gamma(\mathcal{T})$ is (by definition) a mutual exclusion invariant set. A template with this property for each possible instantiation γ is called an *invariant template*.

Definition 9 (Invariant Template)**.** *A template $\mathcal{T}$ is an* invariant template *if, for each instance γ, the instantiation of $\mathcal{T}$ according to γ is a mutual exclusion invariant set.*

Given an invariant template $\mathcal{T}$, we can create one state variable for each of its instances. The domains of these variables are the corresponding mutual exclusion invariant sets with an additional *null* value, which is used when no element in the mutual exclusion invariant set is true.

Before describing in what situations we can feasibly prove that a template is invariant, we introduce a final concept:

Definition 10 (Template Instance's Weight)**.** *Take an instance γ of template $\mathcal{T}$ with instantiation $\gamma(\mathcal{T})$. The* weight *$w(\gamma, s)$ of γ in state s is the number of ground atoms of its instantiation true in s:*

$$w(\gamma, s) = |\gamma(\mathcal{T}) \cap s| \,.$$

Proposition 11. *A template $\mathcal{T}$ is invariant if and only if, for each instance γ of $\mathcal{T}$ with instantiation $\gamma(\mathcal{T})$, if $w(\gamma, Init) \leq 1$, then $w(\gamma, s) \leq 1$ for each state $s \in \mathcal{S}_\mathrm{r}$.*

Proof. It follows from Definitions 9 and 10. □

Example 2 (*Floortile* domain)**.** *A template for this domain is $\mathcal{T}_{ft} = (\{c_1, c_2, c_3\})$, where:*

- $c_1 = \langle$robotAt$/2, 0\rangle$ *is the first component. It includes the relation* robotAt *that has an arity of two (i.e. the relation* robotAt(robot,tile) *has two variables) and the variable in position zero, i.e.* robot*, is the counted variable. The remaining variable,* tile*, which is in position one, is the fixed variable:* $F_{c_1} = \{(c_1, 1)\}$.

- $c_2 = \langle$painted$/2, 1\rangle$ *is the second component with* $F_{c_2} = \{(c_2, 0)\}$.

- $c_3 = \langle$clear$/1, 1\rangle$ *is the last component with* $F_{c_3} = \{(c_3, 0)\}$.

For this example, because each component has only one fixed argument, there is only one admissible partition - the trivial one - that places all components together:

$$\mathcal{F}_C = \{\{(c_1, 1), (c_2, 0), (c_3, 0)\}\}$$

Assume that we have a problem $\mathcal{P}$ *with two robots* rbt1 *and* rbt2*, three tiles,* tile1*,* tile2 *and* tile3 *and one colour* black*. Consider one possible instance* γ_1 *such that* $\gamma_1(c_1, 1) = \gamma_1(c_2, 0) = \gamma_1(c_3, 0) =$ tile1*. The instantiation of the template* $\mathcal{T}_{ft}$ *according to* γ_1 *is:* $\gamma_1(\mathcal{T}_{ft}) = \{$ clear(tile1), robotAt(rbt1,tile1), *robotAt(rbt2,tile1), painted(tile1,black)*$\}$*. The weight of the instance* γ_1 *in some state* s *is how many of the atoms in* $\gamma_1(\mathcal{T}_{ft})$ *are true in* s*. For example,* $w(\gamma_1, s_0) = 1$ *for the plausible initial state* $s_0 = \{$ *clear(tile1), robotAt(rbt1,tile2), robotAt(rbt2,tile3)*$\}$*, because the intersection of the state* s_0 *and the instantiation* $\gamma_1(\mathcal{T}_{ft})$ *contains just* *clear(tile1).*

We will see that we can actually prove that $\mathcal{T}_{ft}$ *is an invariant, which states that any given tile (e.g.,* tile1 *for the instance/instantiation* γ_1*) satisfies at most one of: (1) clear, (2) painted a colour, or (3) occupied by a robot. Hence, for the problem* $\mathcal{P}$*, we can create three state variables that represent each of the three tiles. The domains of these variables are the three possible configurations of the tiles and the null value. As it happens, there is only one situation in which none of the above-mentioned three values can be true, and that is when some robot in some adjacent tile is painting the tile in question. The special null value of the state variable has that meaning. For example,* $\mathcal{SV}_{tile1} = \{$*robotAt(rbt1,tile1), robotAt(rbt2,tile1), painted (tile1, black), clear(tile1), null*$\}$ *and similarly for* $\mathcal{SV}_{tile2}$ *and* $\mathcal{SV}_{tile3}$*.*

4. Safe Instantaneous Ground Actions

In this and in the following sections, we fix a planning instance $\mathcal{I} = (\mathcal{D}, \mathcal{P})$ recalling that we consider a family of planning instances parameterised by the initial state *Init* and $\mathcal{G}$. We then consider a template $\mathcal{T}$ and discuss the conditions for $\mathcal{T}$ to be invariant. More precisely, we determine a few *sufficient* conditions on the families of instantaneous and durative actions in $\mathcal{D}$ that ensure that if, for some instance γ, $w(\gamma, Init) \leq 1$, then $w(\gamma, s) \leq 1$ for all other reachable states $s \in \mathcal{S}_r$.

4.1. Strong safety

We assume a template $\mathcal{T}$ to be fixed as well as an instance γ. In the following, $A \subseteq \mathcal{GA}$ always denotes a set of pairwise non-interfering actions and is assumed to be executable.

Definition 12 (Strongly safe actions)**.** *The set of actions* A *is* strongly γ-safe *if, for each* $s \in \mathcal{S}_A$ *where* $w(\gamma, s) \leq 1$*, the successor state* $s' = \xi(s, A)$ *also satisfies* $w(\gamma, s') \leq 1$.

The following result shows that strong γ-safety can always be checked at the level of single actions.

Proposition 13. *Let A be a set of actions. Then, A is strongly γ-safe if a is strongly γ-safe for all $a \in A$.*

Proof. Write $A = \{a_1, \dots, a_n\}$ and let $s \in \mathcal{S}_A$. Note that from Proposition 1, the actions in A can be serialised and the successor state $s' = \xi(s, A)$ can be recursively obtained as $s_0 = s$, $s_k = \xi(s_{k-1}, a_k)$, $k = 2, \dots, n$ and $s' = s_n$. By assumption, it follows that $w(\gamma, s_i) \leq 1$ for every i. In particular, $w(\gamma, s') \leq 1$. □

The converse of the above result does not hold. A counterexample will be shown later (see Example 3).

In order to tease apart the property of strong γ-safety, we will need several more formal definitions, which we will give here and in the following subsections. Firstly, consider restricting an action to a particular instantiation and its complement.

Definition 14. *Given an action $a \in \mathcal{GA}$, a_γ and $a_{\neg\gamma}$ are the actions, respectively, specified by*

$$\begin{array}{ll} Pre^{\pm}_{a_\gamma} = Pre^{\pm}_{a} \cap \gamma(\mathcal{T}), & Eff^{\pm}_{a_\gamma} = Eff^{\pm}_{a} \cap \gamma(\mathcal{T}) \\ Pre^{\pm}_{a_{\neg\gamma}} = Pre^{\pm}_{a} \cap \gamma(\mathcal{T})^c, & Eff^{\pm}_{a_{\neg\gamma}} = Eff^{\pm}_{a} \cap \gamma(\mathcal{T})^c \end{array}$$

(where A^c denotes the set complement of A).

Given an action set A, we define the action sets $A_\gamma = \{a_\gamma \,|\, a \in A\}$ and $A_{\neg\gamma} = \{a_{\neg\gamma} \,|\, a \in A\}$.

We also split the states in a similar way: take $s \in \mathcal{S}$, put $s_\gamma = s \cap \gamma(\mathcal{T})$ and $s_{\neg\gamma} = s \cap \gamma(\mathcal{T})^c$.

Remark 15. *For a state s and an action set A, we have that $s \in \mathcal{S}_A$ if and only if $s_\gamma \in \mathcal{S}_{A_\gamma}$ and$s_{\neg\gamma} \in \mathcal{S}_{A_{\neg\gamma}}$ and it holds that:*

$$s' = \xi(s, A) \;\Leftrightarrow\; \left\{ \begin{array}{lcl} s'_\gamma & = & \xi(s_\gamma, A_\gamma) \\ s'_{\neg\gamma} & = & \xi(s_{\neg\gamma}, A_{\neg\gamma}) \end{array} \right. \tag{4}$$

This leads to the following simple but useful result.

Proposition 16. *For a set of actions A, the following conditions are equivalent:*

(i) A is strongly γ-safe;

(ii) A_γ is strongly γ-safe;

(iii) For every $s \in \mathcal{S}_{A_\gamma}$ such that $s \subseteq \gamma(\mathcal{T})$ and $w(\gamma, s) \leq 1$, it holds that the successor state $s' = \xi(s, A_\gamma)$ is such that $w(\gamma, s') \leq 1$.

Proof. (ii)⇒(iii) is trivial and (iii)⇒(i) is an immediate consequence of (4) and of the fact that $w(\gamma, s) = w(\gamma, s_\gamma)$ and $w(\gamma, s') = w(\gamma, s'_\gamma)$.

Finally, (i)⇒(ii) follows from the following argument. Take any $s \in \mathcal{S}_{A_\gamma}$ such that $w(\gamma, s) \leq 1$. Consider $s^* = s_\gamma \cup Pre^{+}_{A_{\neg\gamma}}$. Since $s^*_\gamma = s_\gamma \in \mathcal{S}_{A_\gamma}$ and $s^*_{\neg\gamma} = Pre^{+}_{A_{\neg\gamma}} \in \mathcal{S}_{A_{\neg\gamma}}$, it follows that $s^* \in \mathcal{S}_A$. If we consider the successor states $s' = \xi(s, A_\gamma)$ and $s'^* = \xi(s^*, A_\gamma)$, it follows from (4) that

$$s'_\gamma = \xi(s_\gamma, A_\gamma) = \xi(s^*_\gamma, A_\gamma) = s'^*_\gamma$$

Therefore,

$$w(\gamma, s') = w(\gamma, s'_\gamma) = w(\gamma, s'^*_\gamma) = w(\gamma, s'^*) \leq 1$$

where the last equality follows from the assumption that A is strongly γ-safe. □

4.2. Characterisation of ground actions with respect to strong safety

Based on the structure of the preconditions and effects, we classify actions in four classes and then relate each class to the notion of strong safety.

Definition 17 (Classification of Ground Actions)**.** *A set of actions A is:*

- γ-unreachable *if* $|Pre^+_{A_\gamma}| \geq 2$*;*
- γ-heavy *if* $|Pre^+_{A_\gamma}| \leq 1$ *and* $|Eff^+_{A_\gamma}| \geq 2$*;*
- γ-irrelevant *if* $|Pre^+_{A_\gamma}| \leq 1$ *and* $|Eff^+_{A_\gamma}| = 0$*;*
- γ-relevant *if* $|Pre^+_{A_\gamma}| \leq 1$ *and* $|Eff^+_{A_\gamma}| = 1$.

Each $A \subseteq \mathcal{GA}$ belongs to one and just one of the above four disjoint classes. The following result clarifies their relation with strong safety.

Proposition 18. *Let A be a set of actions. Then,*

1. if A is γ-unreachable or γ-irrelevant, A is strongly γ-safe;

2. if A is γ-heavy, A is not strongly γ-safe.

Proof. If A is γ-unreachable and A is applicable in the state s, it follows that $Pre^+_A \subseteq s$ and thus $w(\gamma, s) \geq |Pre^+_{A_\gamma})| \geq 2$. This shows that the condition $w(\gamma, s) \leq 1$ is never verified and thus A is strongly γ-safe.

If A is γ-irrelevant and A is applicable in the state s, we have that the successor state $s' = \xi(s, a) \subseteq s$. This yields $w(\gamma, s') \leq w(\gamma, s)$. This implies that A is strongly γ-safe.

Suppose that A is γ-heavy and consider the state $s = Pre^+_A$. A is applicable in s (because of assumption (2)) and $w(\gamma, s) = |Pre^+_{A_\gamma}| \leq 1$. After applying A in s, the successor state $s' = \xi(s, A)$ is such that $s' \supseteq Eff^+_{A_\gamma}$ This yields $w(\gamma, s') \geq |Eff^+_\alpha| \geq 2$ and proves that A is not strongly γ-safe. □

The following example shows how the converse of Proposition 13 does not hold.

Example 3. *Consider a template $\mathcal{T}$ and an instance γ such that $\gamma(\mathcal{T}) = \{q, q', q''\}$, where q, q', q'' are distinct ground atoms as well as q'''. Let $A = \{a, a'\}$, where a and a' are actions such that:*

$$Pre^+_a = \{q, q'\},\ Eff_a = \emptyset,\ \ Pre_{a'} = \emptyset,\ Eff^+_{a'} = \{q'', q'''\}$$

Then, a is γ-unreachable and thus strongly γ-safe, while a' is γ-heavy and not strongly γ-safe. However, the set of actions A is γ-unreachable and thus strongly γ-safe.

As the next example shows, γ-relevant action sets may be strongly γ-safe or not.

Example 4. *Consider a template $\mathcal{T}$ and an instance γ such that $\gamma(\mathcal{T}) = \{q, q', q''\}$, where q, q' and q'' are three distinct ground atoms, and a is an action such that $|Pre^+_a| \leq 1$ and $Eff^+_a = \{q\}$. Since $|Eff^+_a \cap \gamma(\mathcal{T})| = 1$, a is γ-relevant.*

- *Suppose that $Pre^+_a = \{q'\}$ and $Eff^-_a = \{q'\}$ as shown in Figure 1, left. In this case, a is strongly γ-safe. In fact, if $s \in \mathcal{S}_a$ is such that $w(\gamma, s) \leq 1$, $q' \in s$ and in consequence $w(\gamma, s) = 1$. Given $s' = \xi(s, a)$, $q' \notin s'$, but $q \in s'$ and therefore $w(\gamma, s') = 1$.*

- *Suppose that $Pre_a = \emptyset$ and $Eff^-_a = \{q'\}$ as shown in Figure 1, right. In this case, a is γ-relevant, but is not strongly γ-safe. In fact, consider the starting state $s = \{q''\}$ so that $w(\gamma, s) = 1$. Since $q'' \notin Eff^-_a$ and $q \in Eff^+_a$, $s' = \xi(s, a)$ is such that $w(\gamma, s') = 2$.*

Proposition 18 classifies all but relevant actions. Let us now consider relevance. We split it in four categories and then analyse the strong safety of each case.

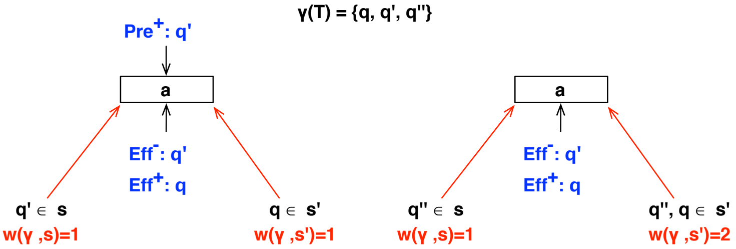


Figure 1: Relevant actions can be either safe (left) or unsafe (right). See Example 3. (Empty action parts are not shown.)

Definition 19 (Classification of Relevant Actions)**.** *A γ-relevant set of actions A has at most one relevant precondition ($|Pre^+_{A_\gamma}| \leq 1$). A is γ-*weighty*, at p, when that is the single relevant precondition: $Pre^+_{A_\gamma} = \{p\}$. A is γ-*weightless *if $|Pre^+_{A_\gamma}| = 0$.*
A γ-weighty at p action set A is either:

- γ-balanced *if the precondition is an effect: $p \in Eff_{A_\gamma}$; or*
- γ-unbalanced *if otherwise: $p \notin Eff_{A_\gamma}$.*

A γ-weightless action set A is either:

- γ-bounded *if the entire instantiation is accessed: $Pre_{A_\gamma} \cup Eff_{A_\gamma} = \gamma(\mathcal{T})$; or*
- γ-unbounded *if otherwise: $Pre_{A_\gamma} \cup Eff_{A_\gamma} \neq \gamma(\mathcal{T})$.*

Every relevant set of actions A belongs to one and just one of the above four disjoint classes. The balanced and unbounded cases are discussed in Example 4. To understand the bounded case better, recall that a relevant action set has just one relevant positive effect l, as in $Eff^+_{A_\gamma} = \{l\}$. When A is bounded, then the rest of the instantiation $\gamma(\mathcal{T}) \setminus \{l\}$ is accessed *negatively*: $\gamma(\mathcal{T}) = Pre_{A_\gamma} \cup Eff_{A_\gamma}$, $Pre^+_{A_\gamma} = \emptyset$ and $Eff^+_{A_\gamma} = \{l\}$ together imply that $\gamma(\mathcal{T}) \setminus \{l\} = Pre^-_A \cup Eff^-_A$. So the weight after executing a bounded set will be exactly one, regardless of what it was before. In other words, bounded is even safer than balanced. Formally:

Proposition 20. *Let A be a γ-relevant set of actions. Then,*

1. *if A is γ-balanced or γ-bounded, A is strongly γ-safe;*
2. *if A is γ-unbalanced or γ-unbounded, A is not strongly γ-safe.*

Proof. We will prove the corresponding property for A_γ making use of Condition (iii) of Proposition 16. We first analyse the case when A is γ-weighty. Let $Pre^+_{A_\gamma} = \{q_1\}$ and $Eff^+_{A_\gamma} = \{q_2\}$. Suppose now that A is γ-balanced and fix a state $s \in \gamma(\mathcal{T})$ such that $w(\gamma, s) \leq 1$ and A_γ is applicable in s. Since $q_1 \subseteq s$, necessarily, $s = \{q_1\}$ and $w(\gamma, s) = 1$. Consider the subsequent state

$s' = \xi(s, A_\gamma)$. If $q_1 = q_2$, we have that $s' = s$ so that $w(\gamma, s') = 1$. If instead $q_1 \in \mathit{Eff}^-_{A_\gamma}$, we have that $s' = (s \setminus \{q_1\}) \cup \{q_2\} = \{q_2\}$ and thus $w(\gamma, s') = 1$. Suppose that A is unbalanced and consider the state $s = \{q_1\}$. The subsequent state $s' = \xi(s, A_\gamma) = \{q_1, q_2\}$ so that $w(\gamma, s') = 2$.

We now consider the the case when A is γ-weightless, i.e. $\mathit{Pre}^+_{A_\gamma} = \emptyset$. Let $\mathit{Eff}^+_{A_\gamma} = \{q_2\}$. Suppose that A is γ-bounded and fix a state $s \in \gamma(\mathcal{T})$ such that $w(\gamma, s) \leq 1$ and A_γ is applicable in s. Since A is γ-relevant, the subsequent state $s' = \xi(s, A_\gamma)$ is such that $w(\gamma, s') \leq w(\gamma, s) + 1$. The only case we need to consider is thus when $w(\gamma, s) = 1$. Suppose that $s = \{q_1\}$. Since, by assumption $\mathit{Pre}_{A_\gamma} \cup \mathit{Eff}_{A_\gamma} = \mathcal{T}(\gamma)$, it follows that $q_1 \in \mathit{Pre}_{A_\gamma} \cup \mathit{Eff}_{A_\gamma}$. We have that $q_1 \notin \mathit{Pre}^-_{A_\gamma}$ (otherwise A_γ would not be applicable on the state s). Therefore, necessarily, either $q_1 \in \mathit{Eff}^+_{A_\gamma}$ or $q_1 \in \mathit{Eff}^-_{A_\gamma}$. In the first case, we have that $q_1 = q_2$ and thus $s' = s = \{q_1\}$. In the second case, $s' = \{q_2\}$. In both cases, $w(\gamma, s') = 1$. Finally, if A is γ-unbounded, we consider any ground atom $q_1 \in \gamma(\mathcal{T}) \setminus (\mathit{Pre}_{A_\gamma} \cup \mathit{Eff}_{A_\gamma})$ and we put $s = \{q_1\}$. We have that A_γ is applicable in q_1 since $\mathit{Pre}^+_{A_\gamma} = \emptyset$ and $q_1 \notin \mathit{Pre}^-_{A_\gamma}$, and $w(\gamma, s) = 1$. Since it also holds that $q_1 \notin \mathit{Eff}^-_{A_\gamma}$, we have that the subsequent state $s' = \xi(s, A_\gamma) = \{q_1, q_2\}$ and $w(\gamma, s') = 2$. □

Putting the Propositions 18 and 20 together, we get the following result:

Theorem 21. *Let A be a set of actions. Then,*

1. *if A is either γ-unreachable, γ-irrelevant, γ-balanced, or γ-bounded, A is strongly γ-safe;*
2. *if A is either γ-heavy, γ-unbalanced, or γ-unbounded, A is not strongly γ-safe.*

Any state in which a γ-heavy or γ-unbalanced action can be executed in will immediately have weight 2 (or more). In contrast, it is possible for a γ-unbounded action to execute without violating the weight bound, and it is conceivable that all reachable states are such that γ-unbounded actions end up being safe. In that sense, Theorem 21 is arguably less of a *complete* classification than it may appear.

A possibly interesting way to approach knowledge generated by a stronger prover would be to augment the descriptions of actions with derived preconditions, i.e. properties that hold in all reachable states and could be freely added as a precondition on all actions. That could convert γ-unbounded actions into γ-balanced or γ-bounded (if negative conditions are added).

Example 5 (*Floortile* domain)**.** *Take the schema α =`paintUp`^{end} with variables $V_\alpha = \{\mathtt{r}, \mathtt{y}, \mathtt{x}, \mathtt{c}\}$ (see Table 3) and grounding gr such that $gr(\mathtt{r})$ =`rbt1`, $gr(\mathtt{y})$ =`tile1`, $gr(\mathtt{x})$ =`tile3`, $gr(\mathtt{c})$ = `red`. Let $c = \langle$`painted`$/2, 1\rangle$ be the component that counts the colour argument of `painted` and $\mathcal{T} = (\{c\}, \{\{(c, 0)\}\})$ be the template on just that component. Consider the safety of $a = gr(\alpha)$ with respect to $\mathcal{T}$. Let $\gamma_1(c, 0)$ =`tile1` be the instance for the tile the robot is painting, $\gamma_2(c, 0)$ =`tile2` be the instance for an unrelated tile, and $\gamma_3(c, 0)$ =`tile3` be the instance for the tile the robot is standing on. Each instantiation is the set of all possible colours per tile.*

The action a is γ_2-irrelevant (as expected), meaning none of $\gamma_2(\mathcal{T})$ are preconditions or effects, so also a is (trivially) strongly γ_2-safe.

While `tile3` is relevant to the action as a whole, a does not access `painted` at `tile3` so a is likewise irrelevant and thus safe with respect to γ_3.

The action a is γ_1-relevant, because it adds `painted(tile1, red)`$\in \gamma_1(\mathcal{T})$. It is not strongly γ_1-safe, because if executed in a state where the tile was already black, the tile would end up being painted both colours: $\xi(\{$`painted(tile1, black)`$\}, a_{\gamma_1}) = \{$`painted(tile1,red)`, `painted(tile1,black)`$\}$. In particular, a is γ_1-unbounded. If we altered it by adding negative preconditions on all the other colours, it would become both bounded and safe.

If we added `clear` *to the template, then instead a would be balanced (and thus safe) at the instantiation for* `tile1`.

4.3. Template safety

This conclusive section presents a definition of strong safety with respect to a template and a first result that expresses a sufficient condition for a template to be invariant.

Definition 22. *For a template $\mathcal{T}$, a set of actions $A \subseteq \mathcal{GA}$ is* strongly safe *if it is strongly γ-*safe *for every instance γ.*

We have the following result:

Corollary 23. *For a template $\mathcal{T}$, $\mathcal{T}$ is invariant if for each $a \in \mathcal{GA}$, a is strongly safe.*

Proof. We proceed as follows. We fix any instance γ of $\mathcal{T}$ and we show that if $w(\gamma, \mathit{Init}) \leq 1$, then $w(\gamma, s) \leq 1$ for every $s \in \mathcal{S}_\mathrm{r}$. This, by Proposition 11, yields the result. By definition of the set of reachable states $\mathcal{S}_\mathrm{r}$, any $s \in \mathcal{S}_\mathrm{r}$ can be obtained by the initial state *Init* by recursively applying a sequence of action sets. Precisely, there exist a sequence of sets (each consisting of pairwise non-interfering actions) $A_1, A_2, \ldots, A_k$ and a sequence of states $\{s_i\}_{i=0\ldots k}$ such that $s_0 = \mathit{Init}$, $s_k = s$ and for each $i = 0, \ldots, k-1$, $s_{i+1} = \xi(s_i, A_{i+1})$. We prove that $w(\gamma, s) \leq 1$ by induction on k. Notice that the case $k = 0$ boils down to $s = s_0 = \mathit{Init}$ and $w(\gamma, \mathit{Init}) \leq 1$ is our standing assumption. Assume it to be true for $k-1$ and let us prove it for k. Notice that, we can write $s = s_k = \xi(s_{k-1}, A_k)$ and the induction assumption implies that $w(\gamma, s_{k-1}) \leq 1$. By assumption every $a \in \mathcal{GA}$ is strongly safe and thus, by Definition 22, strongly γ-safe. Using Proposition 13 we obtain in particular that A_k is strongly γ-safe. Consequently, also $w(\gamma, s_k) \leq 1$. □

The condition expressed in Corollary 23 cannot be inverted in general. Indeed, a template can be invariant even if not all actions are strongly safe. We will see when this happens in the following section.

5. Safe action sequences and safe durative actions

A template can be invariant even if not all actions are strongly safe. This happens for two reasons. On the one hand, since the set of reachable states $\mathcal{S}_\mathrm{r}$ is in general smaller than $\mathcal{S}$, it may be that all the states that are responsible for the lack of strong safety are unreachable, i.e. they are not in $\mathcal{S}_\mathrm{r}$. On the other hand, in domains with durative actions, some instantaneous actions are temporally coupled because they are the start and end fragments of the same durative action. This coupling imposes constraints on the states where the end part can be applied, which might prove helpful to establish that a template is invariant. While in this paper we will not analyse the first case as it would require an analysis of the set of reachable states $\mathcal{S}_\mathrm{r}$, which is infeasible, we now elaborate suitable simple concepts of safety for durative actions, which are weaker than strong safety. These extensions are important in many real-world temporal planning domains. In our experience, these domains often present end-fragments of durative actions that are unsafe as written, often, γ-unbalanced. Nonetheless, every use in an executable plan preserves the weight condition, typically because the associated start-fragments force the weight to zero and the invariant-fragments keep it there. (A γ-unbalanced action is safe if it only executes in weight zero states.) We give a definition of safety for durative actions that captures this case. However, since in a plan a durative action may intertwine with other actions that happen between its start and end points, we need to work out a concept of safety for more general sequence of actions than just durative ones.

Below, we consider general sequences of action sets $\mathbf{A} = (A^1, A^2, \dots, A^n)$. Any simple plan π generates such an object. Indeed, if $\{t_i\}_{i=0,\dots,\bar{k}}$ is the related happening time sequence and A_{t_i} are the relative happenings, we can consider the happening sequence $\mathbf{A}_\pi = (\mathbf{A}_{\mathbf{t_0}}, \dots, \mathbf{A}_{\mathbf{t_{\bar{k}}}})$ that contains all the information on the plan π except the time values at which the various actions happen.

To study the invariance of a template, we break the happening sequence of each plan into subsequences determined by the happenings of durative actions. More precisely, we consider sequences $\mathbf{A} = (A^1, A^2, \dots, A^n)$ where, for some durative action $Da = (a^{\text{st}}, a^{\text{inv}}, a^{\text{end}})$, we have that $a^{\text{st}} \in A^1$ and $a^{\text{end}} \in A^n$. $A^2, \dots, A^{n-1}$, as well as A^1 and A^n, possibly contain other actions that are executed over the duration of Da. However, in the first place, it is convenient to consider general sequences of actions $\mathbf{A} = (A^1, A^2, \dots, A^n)$ without referring to plans or durative actions. Hence, in this section, we first give a definition of safety for $\mathbf{A}$ such that, when $\mathbf{A}$ is executed serially in any executable plan π, if the weight constraint is not violated in the state where the sequence is initially applied, it is not violated in any intermediate step and at the end of the sequence. For single action sets (sequences of length $n = 1$), such concept coincides with the notion of strong safety.

We then consider a slightly stronger notion of safety which is robust to the insertion, between elements of the sequence, of other actions whose positive effects have no intersection with the template. To do this, it is necessary to introduce a number of auxiliary concepts relating to the state dynamics induced by the execution of $\mathbf{A}$. This general theory will be then applied to sequences constructed from durative actions.

We recall our standing assumption that any considered subset of actions consist of pairwise non-interfering actions and is assumed to be executable.

5.1. Safe ground action sequences

For a sequence of action sets $\mathbf{A} = (A^1, A^2, \dots, A^n)$, we denote with $\mathbf{S_A}$ the set of state sequences $(s^0, \dots, s^n) \in \mathcal{S}^{n+1}$ such that

$$s^i = \xi(s^{i-1}, A^i) \text{ and } A^i \text{ is applicable in } s^{i-1} \text{ for all } i \in \{1, \dots, n\}$$

If $(s^0, \dots, s^n) \in \mathbf{S_A}$, we say that $(s^0, \dots, s^n)$ is a state sequence *compatible* with $\mathbf{A}$. Given an instance γ, we also define $\mathbf{S_A}(\gamma) = \{(s^0, \dots, s^n) \in \mathbf{S_A} \,|\, w(\gamma, s^0) \leq 1\}$. We use the following notation for subsequences of $\mathbf{A}$: $\mathbf{A}_h^k = (A^h, A^{h+1}, \dots, A^k)$.

We now fix a template $\mathcal{T}$ and an instance γ and give the following natural definition of safety for a sequence.

Definition 24 (Individually safe actions)**.** *A sequence of action sets* $\mathbf{A} = (A^1, A^2, \dots, A^n)$ *is* individually γ-safe *if for every sequence of states* $(s^0, \dots, s^n) \in \mathbf{S_A}$ *we have that*

$$w(\gamma, s^0) \leq 1 \;\Rightarrow\; w(\gamma, s^i) \leq 1 \; \forall i = 1, \dots, n$$

The invariance of a template can now be expressed in terms of individual safety for the happening sequences.

Proposition 25. *Let $\mathcal{T}$ be a template. Suppose that for every executable simple plan π, the happening sequence $\mathbf{A}_\pi$ is individually γ-safe for every instance γ. Then, $\mathcal{T}$ is invariant.*

Proof. Take any instance γ and assume that $w(\gamma, \mathit{Init}) \leq 1$. We need to show that $w(\gamma, s) \leq 1$ for every $s \in \mathcal{S}_r$. For each $s \in \mathcal{S}_r$, there exists an executable simple plan π having $\mathit{trace}(\pi) = \{S_i = (t_i, s_i)_{i=0,\dots,\bar{k}}\}$ with $s_0 = \mathit{Init}$, $s_{\bar{k}} = s$, and with happening sequence $\mathbf{A}_\pi$. We have that the state sequence $(s_0, \dots, s_{\bar{k}}) \in \mathbf{S}_{\mathbf{A}_\pi}$. Since by assumption $w(\gamma, s_0) \leq 1$ the individual γ-safety of $\mathbf{A}_\pi$ implies that $w(\gamma, s_j) \leq 1$ for every $j = 1, \dots, \bar{k}$. In particular, $w(\gamma, s) \leq 1$.

□

Below are elementary properties of individual γ-safety for subsequences of $\mathbf{A}$.

Proposition 26. *Consider a sequence of action sets $\mathbf{A} = (A^1, A^2, \dots, A^n)$. The following properties hold:*

(i) if, for some k and h such that $k \geq h - 1$, $\mathbf{A}_1^k = (A^1, A^2, \dots, A^k)$ and $\mathbf{A}_h^n = (A^h, \dots, A^n)$ are both individually γ-safe, then also $\mathbf{A}$ is individually γ-safe;

(ii) if $\mathbf{A}$ is individually γ-safe and A^k and A^{k+1} are non-interfering, then $\mathbf{A}' = (A^1, A^2, \dots, A^k \cup A^{k+1}, \dots, A^n)$ is individually γ-safe;

(iii) if $\mathbf{A}$ is individually γ-safe and B^j, for $j = 1, \dots, n$ are action sets such that $\mathit{Eff}_{B^j} = \emptyset$, then, $\mathbf{A}' = (A^1, B^1, A^2, \dots, B^n, A^n)$ is individually γ-safe. If, in addition, A^j and B^j are non-interfering for every $j = 1, \dots, n$, also $\mathbf{A}'' = (A^1 \cup B^1, \dots, A^n \cup B^n)$ is individually γ-safe.

Proof. (i): If $(s^0, s^1, \dots, s^n) \in \mathbf{S}_\mathbf{A}$, we have that

$$(s^0, s^1, \dots, s^k) \in \mathbf{S}_{\mathbf{A}_1^k}, \quad (s^{h-1}, s^1, \dots, s^n) \in \mathbf{S}_{\mathbf{A}_h^n}.$$

Therefore, if $w(\gamma, s^0) \leq 1$, from the fact that $\mathbf{A}_1^k$ is individually γ-safe, it follows that $w(\gamma, s^j) \leq 1$ for every $j = 1, \dots, k$. In particular, being $k \geq h - 1$, we have that $w(\gamma, s^{h-1}) \leq 1$. From the fact that $\mathbf{A}_h^n$ is also individually γ-safe, it now follows that $w(\gamma, s^j) \leq 1$ for every $j = h, \dots, n$. This implies that $w(\gamma, s^j) \leq 1$ for every $j = 1, \dots, n$ and proves the thesis.

(ii): Suppose $(s^0, s^1, \dots, s^{k-1}, s^{k+1}, \dots, s^n) \in \mathbf{S}_{\mathbf{A}'}$ where $s^{k+1} = \xi(A^k \cup A^{k+1}, s^{k-1})$. Put $s^k = \xi(A^k, s^{k-1})$ and note that, by serialisability (see Proposition 1), $s^{k+1} = \xi(A^{k+1}, s^k)$, and therefore $(s^0, s^1, \dots, s^{k-1}, s^k, s^{k+1}, \dots, s^n) \in \mathbf{S}_\mathbf{A}$. This implies that $w(\gamma, s^j) \leq 1$ for every $j = 1, \dots, n$ and proves the thesis.

(iii): If $(s^0, s'^0, s^1, \dots, s'^{n-1}, s^n) \in \mathbf{S}_{\mathbf{A}'}$, then, $s'^{k-1} = s^k$ for every $k = 1, \dots, n$ and $(s^0, s^1, \dots, s^n) \in \mathbf{S}_\mathbf{A}$. Individual γ-safety of $\mathbf{A}$ now yields the thesis. Regarding $\mathbf{A}''$ thesis follows from the fact that $\mathbf{A}'$ is individually γ-safe and previous item (ii). □

The following is a useful consequence of the previous results: it asserts that if individual safety holds locally in a sequence, then it also holds globally.

Corollary 27. *For a sequence of action sets $\mathbf{A} = (A^1, A^2, \dots, A^n)$, the following conditions are equivalent:*

(i) the sequence $\mathbf{A}$ is individually γ-safe;

(ii) for each $j = 1, \dots, n$, there exists a subsequence $\mathbf{A}_{j-r}^{j+s}$, with $r, s \geq 0$, that is individually γ-safe.

Proof. (i)⇒(ii) is trivial and (ii)⇒(i) follows from an iterative use of (i) of Proposition 26. □

Individual safety is a weak property since it is not robust with respect to the insertion of other actions, even when these actions are irrelevant but possess delete effects. This is connected to the fact that, while individual safety has this nice local to global feature illustrated in Corollary 27, it does not possess the opposite feature: subsequences of individually safe sequences may not be individually safe. The following example shows both these phenomena.

Example 6. *Consider a template $\mathcal{T}$ and an instance γ such that $\gamma(\mathcal{T}) = \{q, q'\}$. The set of state sequences compatible with $\mathbf{A} = (a^1, a^2)$ (Figure 2 - top diagram) is: $\mathbf{S_A} = \{(s^0, s^1, s^2) | q \notin s^0, s^1 = s^0 \cup \{q'\}, s^2 = s^1\}$. Note that $q \notin s^0$ because, by hypothesis, a^2 is applicable in s^1 and $s^1 = s^0 \cup \{q'\}$. $\mathbf{A}$ is individually γ-safe since $w(\gamma, s^i) \leq 1$ for every state s^i that appears in $\mathbf{S_A}$. Note that a^1 is γ-unbounded and thus not strongly γ-safe and that $\mathbf{A}_1^1 = (a^1)$ is not individually γ-safe as well.*

Now consider the sequence $\tilde{\mathbf{A}} = (a^1, b, a^2)$ (Figure 2 - bottom diagram) where a γ-irrelevant action b is inserted between a^1 and a^2. The new set of state sequences compatible with $\tilde{\mathbf{A}}$ is: $\mathbf{S}_{\tilde{\mathbf{A}}} = \{(s^0, s^1, s^2, s^3) | s^1 = s^0 \cup \{q'\}, s^2 = s^1 \setminus \{q\}, s^3 = s^2\}$. Note that now q can be in s^0 since it is the action b that ensures the applicability of a^2. If $q \in s^0$, since a^1 adds q' to s^0, $w(\gamma, s^1) = 2$. This new sequence is not individually γ-safe. The insertion of a γ-irrelevant action has failed the individual γ-safety of the sequence $\mathbf{A}$.

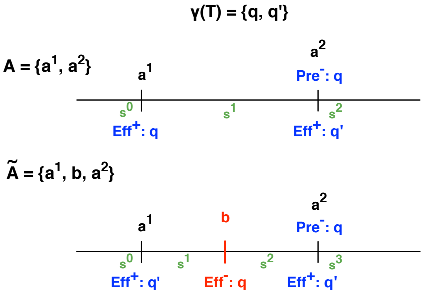


Figure 2: The insertion of the γ-irrelevant action b fails the individual γ-safety of the sequence $\mathbf{A}$. See Example 6.

For proving some of our results, the concept of individual safety is not sufficient. Below we present a stronger definition of safety for an action sequence that is robust with respect to the insertion of irrelevant actions in it. First, we define the simple concepts of executable and reachable sequences.

Definition 28 (Executable and reachable actions)**.** *The sequence $\mathbf{A} = (A^1, A^2, \ldots, A^n)$ is called:*

- executable *if* $\mathbf{S_A} \neq \emptyset$*;*
- γ-(un)reachable *if* $\mathbf{S_A}(\gamma) \neq \emptyset$ *(*$\mathbf{S_A}(\gamma) = \emptyset$*).*

Remark 29. *Note the following chain of implications:*

$$\textit{non-executable} \;\Rightarrow\; \gamma\textit{-unreachable} \;\Rightarrow\; \textit{individually}\,\gamma\textit{-safe}$$

Note that if π is an executable simple plan, γ is an instance and $w(\gamma, Init) \leq 1$, then the happening sequence $\mathbf{A}_\pi$ is γ-reachable. Moreover, every subsequence $\mathbf{A}$ of $\mathbf{A}_\pi$ is executable. If a subsequence $\mathbf{A}$ of $\mathbf{A}_\pi$ is γ-unreachable, the weight will surely exceed 1 at some point of the plan π and thus the template $\mathcal{T}$ will not be invariant.

In the special case of a sequence of length 2, executability and reachability admit very simple characterisations. We report them below as we will need them later. First define, for a generic set of actions A, the *postconditions*:

$$\Gamma^+_A = (Pre^+_A \setminus Eff^-_A) \cup Eff^+_A, \quad \Gamma^-_A = (Pre^-_A \cup Eff^-_A) \setminus Eff^+_A \tag{5}$$

We have the following result:

Proposition 30. *For a sequence of two action sets* $\mathbf{A} = (A^1, A^2)$*, the following conditions are equivalent:*

(i) $\mathbf{A}$ *is executable;*

(ii) $\Gamma^+_{A^1} \cap Pre^-_{A^2} = \emptyset = \Gamma^-_{A^1} \cap Pre^+_{A^2}$.

Proof. (i)⇒(ii): Note that if $(s^0, s^1, s^2) \in \mathbf{S_A}$, it follows that $Pre^+_{A^1} \subseteq s^0$. Since $s^1 = (s^0 \setminus Eff^-_{A^1}) \cup Eff^+_{A^1}$ it follows that $\Gamma^+_{A^1} \subseteq s^1$. Analogously, using the fact that $(Pre^-_{A^1})^c \supseteq s^0$, it follows that $(\Gamma^-_{A^1})^c \supseteq s^1$. Since A^2 must be applicable on s^1 conditions (ii) immediately follow.

(ii)⇒(i): Consider $s^0 = Pre^+_{A^1} \cup (Pre^+_{A^2} \setminus Eff^+_{A^1})$. Straightforward set theoretic computation, using conditions (ii), show that A^1 can be applied on s^0 and that A^2 can be applied on $s^1 = \xi(A^1, s^0)$. This proves (i). □

Proposition 31. *For a sequence of two action sets* $\mathbf{A} = (A^1, A^2)$*, the following conditions are equivalent:*

(i) $\mathbf{A}$ *is* γ*-reachable;*

(ii) $\mathbf{A}$ *is executable and* $|Pre^+_{A^1_\gamma} \cup (Pre^+_{A^2_\gamma} \setminus Eff^+_{A^1_\gamma})| \leq 1$.

Proof. (ii)⇒(i): It follows from the proof of (ii)⇒(i) in Proposition 30 that there exists $(s^0, s^1, s^2) \in \mathbf{S_A}$ with $s^0 = Pre^+_{A^1} \cup (Pre^+_{A^2} \setminus Eff^+_{A^1})$. By the assumption made $w(\gamma, s^0) \leq 1$ and this proves (i).

(i)⇒(ii): it follows from the fact that if $(s^0, s^1, s^2) \in \mathbf{S_A}$, necessarily $Pre^+_{A^1} \cup (Pre^+_{A^2} \setminus Eff^+_{A^1}) \subseteq s^0$. □

The following result shows how executability and γ-reachability are robust with respect to some specific modifications of a sequence, notably, the deletion of actions containing no effects, and serialisation.

Proposition 32. *Consider a sequence* $\mathbf{A} = (A^1, A^2, \dots, A^n)$ *that is executable or* γ*-reachable. Then,*

(i) if $B^j \subseteq A^j$ *are such that* $Eff_{B^j} = \emptyset$ *for every* $j = 1, \dots, n-1$[4]*, then also* $\mathbf{A}' = (A^1 \setminus B^1, A^2 \setminus B^2, \dots, A^n \setminus B^n)$ *is, respectively, executable or* γ*-reachable.*

(ii) if $A^j = A'^j \cup A''^j$ *for some* $j = 1, \dots, n$*, then also* $\mathbf{A}' = (A^1, A^2, \dots, A'^j, A''^j, \dots, A^n)$ *is, respectively, executable or* γ*-reachable.*

[4] B_n is not constrained; removing effects at the end of the sequence does not alter its executability or reachability.

Proof. (i): If $(s^0, s^1, \dots, s^{n-1}, s^n) \in \mathbf{S}_{\mathbf{A}}$, we have that $(s^0, s^1, \dots, s^{n-1}, s'^n) \in \mathbf{S}_{\mathbf{A}'}$ for a suitable state s'^n. Result then follows from the definition of executability and γ-reachability.

(ii): This follows immediately from serialisability (see Proposition 1).

□

Here is our stronger notion of safety:

Definition 33 (Safe actions)**.** *A sequence of action sets* $\mathbf{A} = (A^1, A^2, \dots A^n)$ *is* γ-safe *if it is executable and the subsequences* $\mathbf{A}_1^k$ *are individually* γ*-safe for every* $k = 1, \dots, n$.

Example 6 illustrates why individual safety is too fragile a concept by itself. We would like to infer that $\mathbf{A}$ in Example 6 is *unsafe* for some appropriate notion of safety, but, as the example shows, that concept cannot be individual safety. Definition 33 accomplishes that aim. Indeed, note how the sequence $\mathbf{A} = (a^1, a^2)$ considered in Example 6 is not γ-safe, since (a^1) is not individually γ-safe.

The definition of safety asks for executability — in addition to asking for every prefix to be individually safe — because, without executability, individual safety is a vacuous condition. In particular, we would like to conclude that the happening sequence in Example 7 considered next is *unsafe*, for the reason shown. By having Definition 33 ask for executability we attain that judgment.

Example 7 (Motivating Executability in Safety)**.** *Consider the non-executable sequence* $\mathbf{A} = (a^1, a^2)$ *depicted in Figure 3 - top diagram (q'' is required false by a^2, but it is asserted true by a^1). Consider a template $\mathcal{T}$ and an instance γ such that $\gamma(\mathcal{T}) = \{q, q', q''\}$. As it happens,* $\mathbf{A}$ *is individually safe: vacuously, since it is non-executable. However, this is not really a safe arrangement: we would like to say that inserting γ-irrelevant actions preserves safety, but consider inserting such an action b, deleting q'', as in* $\tilde{\mathbf{A}} = (a^1, b, a^2)$ *and depicted in Figure 3 - bottom diagram. Then the weight bound, which held (vacuously) for* $\mathbf{A}$*, is violated in* $\tilde{\mathbf{A}}$ *(both q and q' end up true), and so we can conclude that individually safe is too weak for our aims. Observe that, by requiring* executability*, the definition of safety rules out this (counter-)example:* $\mathbf{A}$ *is* not *γ-safe, as desired. (Note that, unlike Example 6, here all the prefixes are individually safe.)*

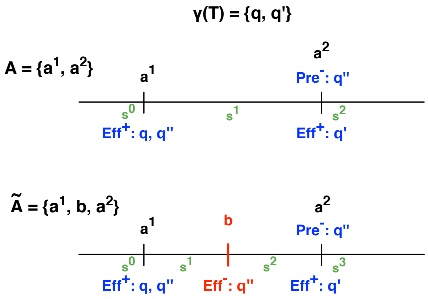


Figure 3: Lack of robustness for individually γ-safe actions. See Example 7.

Remark 34. *If* $\mathbf{A} = (A^1, A^2, \dots A^n)$ *is* γ*-safe, the first action set* A^1 *must necessarily be strongly* γ*-safe. In the other direction, note that if* $\mathbf{A}$ *is executable and every* A^j *for* $j = 1, \dots, n$ *is strongly* γ*-safe then,* $\mathbf{A}$ *is* γ*-safe.*

This motivates the following definition.

Definition 35 (Strongly and weakly safe actions)**.** *A sequence of action sets* $\mathbf{A} = (A^1, A^2, \ldots, A^n)$ *is:*

- strongly γ-safe *if it is executable and every* A^j *for* $j = 1, \ldots, n$ *is strongly* γ*-safe;*
- weakly γ-safe *if it is* γ*-safe but not strongly* γ*-safe.*

With that we are finished with upgrading our notion of *safety*, but there is one last point to consider: having safety alone still does not let us prove anything, because safety only preserves a weight bound, it does not force it true initially. So in general we need to consider sequences that are both γ-safe and γ-reachable. We keep those notions separate, rather than combine them in another definition, for technical reasons: e.g. γ-safe can be shown for an entire planning domain, and then γ-reachable can be checked per problem. The first things that can be said of sequences that are both γ-safe and γ-reachable is that they exclude γ-heavy and γ-unbalanced actions (cf., Propositions 16 and 18).

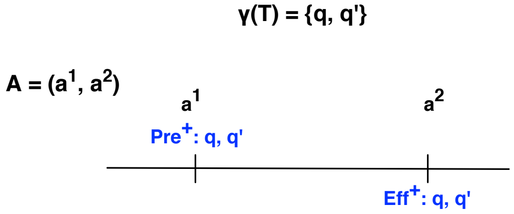


Figure 4: Safety alone does not rule out heavy actions: $\mathbf{A}$ is safe, but a^2 is heavy. See Example 8.

Example 8 (Motivating Reachability in Proposition 36)**.** *Consider the template* $\mathcal{T}$ *and an instance* γ *such that* $\gamma(\mathcal{T}) = \{q, q'\}$*. Now consider the executable, but* γ*-unreachable sequence* $\mathbf{A} = (a^1, a^2)$ *depicted in Figure 4, consisting of a* γ*-unreachable action* a^1 *and a* γ*-heavy action* a^2*. In general, we want to quickly exclude any sequence containing a* γ*-heavy action; but insisting on safety alone, as this example demonstrates, is not enough. Action* a^1 *requires both* q *and* q' *to be true and so the weight has to be 2 initially for* $\mathbf{A}$ *to execute (consequently,* $\mathbf{A}$ *is not* γ*-reachable). Perhaps counterintuitively, for the same reason,* $\mathbf{A}$ *is* γ*-safe: its two prefixes* (a^1) *and* (a^1, a^2) *are both individually safe since all executions begin with the weight bound violated, and then all the implications comprising "safety" just hold vacuously.*

What the example drives at is that, in general, undesirables such as γ-heavy actions can be hidden in nominally safe sequences *only* by failing reachability. Formally, by insisting on safety and reachability together, we can everywhere rule out heavy and unbalanced actions:

Proposition 36. *Let* $\mathbf{A} = (A^1, A^2, \ldots, A^n)$ *be a sequence of action sets. If* $\mathbf{A}$ *is* γ*-safe and* γ*-reachable, then, for every* $j = 1, \ldots, n$*,* A^j *is neither:*

- γ*-heavy, nor*
- γ*-unbalanced.*

Proof. Since $\mathbf{A}$ is γ-reachable, the set $\mathbf{S}_\mathbf{A}(\gamma)$ is not empty. Fix any $(s^0, \ldots, s^n) \in \mathbf{S}_\mathbf{A}(\gamma)$ and note that, since $\mathbf{A}$ is γ-safe, $w(\gamma, s^i) \leq 1$ for every $i = 1, \ldots, n$. For any $j = 1, \ldots, n$, we have that $s^j = (s^{j-1} \setminus \mathit{Eff}^-_{A^j}) \cup \mathit{Eff}^+_{A^j} \supseteq (\mathit{Pre}^+_{A^j} \setminus \mathit{Eff}^-_{A^j}]) \cup \mathit{Eff}^+_{A^j}$. This implies that

$$|(\mathit{Pre}^+_{A^j_\gamma} \setminus \mathit{Eff}^-_{A^j_\gamma}) \cup \mathit{Eff}^+_{A^j_\gamma}| \leq 1 \tag{6}$$

As a consequence, $|\mathit{Eff}^+_{A^j_\gamma}| \leq 1$ that says that A^j can not be γ-heavy. If we now assume that $|\mathit{Eff}^+_{A^j_\gamma}| = 1$, Relation (6) implies that $\mathit{Pre}^+_{A^j_\gamma} \subseteq \mathit{Eff}^+_{A^j_\gamma} \cap \mathit{Eff}^-_{A^j_\gamma}$. Hence A^j can neither be γ-unbalanced. □

In studying the two safety properties for a sequence $\mathbf{A}$ introduced so far, we can essentially restrict ourselves to study the state dynamics on the template instantiation $\gamma(\mathcal{T})$ as we did for strong γ-safety of instantaneous actions (see Remark 15).

Considering the sequence $\mathbf{A} = (A^1, A^2, \ldots A^n)$, we denote by $\mathbf{A}_\gamma = (A^1_\gamma, A^2_\gamma, \ldots A^n_\gamma)$ and $\mathbf{A}_{\neg\gamma} = (A^1_{\neg\gamma}, A^2_{\neg\gamma}, \ldots A^n_{\neg\gamma})$ the corresponding restricted sequences. We have the following result.

Proposition 37. *Given the sequence* $\mathbf{A} = (A^1, A^2, \ldots A^n)$,

1. $\mathbf{A}$ *is executable if and only if* $\mathbf{A}_\gamma$ *and* $\mathbf{A}_{\neg\gamma}$ *are both executable;*

2. $\mathbf{A}$ *is* γ*-reachable if and only if* $\mathbf{A}_\gamma$ *is* γ*-reachable and* $\mathbf{A}_{\neg\gamma}$ *is executable;*

3. $\mathbf{A}$ *is individually* γ*-safe if and only if* $\mathbf{A}_\gamma$ *is individually* γ*-safe.*

4. $\mathbf{A}$ *is* γ*-safe if and only if* $\mathbf{A}_\gamma$ *is* γ*-safe and* $\mathbf{A}_{\neg\gamma}$ *is executable.*

Proof. 1.: It follows from (4) that, for any sequence of states $(s^0, \ldots, s^n) \in \mathbf{S}^{n+1}$, we have that

$$(s^0, \ldots, s^n) \in \mathbf{S}_\mathbf{A} \Leftrightarrow \left\{ \begin{array}{ccc} (s^0_\gamma, \ldots, s^n_\gamma) & \in & \mathbf{S}_{\mathbf{A}_\gamma} \\ (s^0_{\neg\gamma}, \ldots, s^n_{\neg\gamma}) & \in & \mathbf{S}_{\mathbf{A}_{\neg\gamma}} \end{array} \right. \tag{7}$$

This immediately proves the 'only if' implication. On the other hand, if $s' \in \mathbf{S}_{\mathbf{A}_\gamma}$ and $s'' \in \mathbf{S}_{\mathbf{A}_{\neg\gamma}}$, we have that $s'_\gamma \in \mathbf{S}_{\mathbf{A}_\gamma}$ and $s''_{\neg\gamma} \in \mathbf{S}_{\mathbf{A}_{\neg\gamma}}$ and thus $s = s'_\gamma \cup s''_{\neg\gamma} \in \mathbf{S}_\mathbf{A}$ by (7).

2. can be proven analogously to 1. and 3. follows by a straightforward extension of the argument used to prove Proposition 16. Finally, 4. follows from the definition of strong γ-safety and previous items 1. and 3. □

We are now ready to state and prove the following fundamental result, which ensures that the concept of safe sequence is robust to the insertion of irrelevant actions.

Theorem 38. *Consider a* γ*-safe sequence* $\mathbf{A} = (A^1, A^2)$ *and* γ*-irrelevant action sets* $B^1, B^2, \ldots, B^n$. *Then, the sequence* $\tilde{\mathbf{A}} = (A^1, B^1, \ldots, B^n, A^2)$ *is either non executable or* γ*-safe.*

Proof. Consider the sequences restricted on the instantiation $\gamma(\mathcal{T})$ and its complement: $\mathbf{A}_\gamma$, $\mathbf{A}_{\neg\gamma}$ and, respectively, $\tilde{\mathbf{A}}_\gamma$, $\tilde{\mathbf{A}}_{\neg\gamma}$. By virtue of Proposition 37, we have that $\mathbf{A}_\gamma$ is γ-safe and to prove the result it is sufficient to show that $\tilde{\mathbf{A}}_\gamma$ is either non executable or γ-safe.

Assume that $\tilde{\mathbf{A}}_\gamma$ is executable and let $(s^0, s^1, s^2, \ldots, s^{n+1}, s^{n+2}) \in \mathbf{S}_{\tilde{\mathbf{A}}_\gamma}$ be such that $w(\gamma, s^0) \leq 1$. Since $(s^0, s^1) \in \mathbf{S}_{A^1_\gamma}$ and A^1_γ is strongly safe, it follows that $w(\gamma, s^1) \leq 1$. Note now that $s^j = s^{j-1} \setminus \mathit{Eff}^-_{B^{j-1}_\gamma}$ for $j = 2, \ldots, n+1$ and this immediately implies that

$$w(\gamma, s^{n+1}) \leq w(\gamma, s^n) \leq \cdots \leq w(\gamma, s^1) \leq 1$$

What remains to be shown is that also $w(\gamma, s^{n+2}) \leq 1$. To this aim, we introduce the set $F = \cup_{i=1}^{n} \mathit{Eff}^{-}_{B^i_\gamma}$ and we consider the new initial state $\tilde{s}^0 = s^0 \setminus (F \setminus \mathit{Pre}^{+}_{A^1_\gamma})$. Since $\tilde{s}^0 \subseteq s^0$, for sure $\mathit{Pre}^{-}_{A^1_\gamma} \cap \tilde{s}^0 = \emptyset$. Moreover, by construction, we also have that $\mathit{Pre}^{+}_{A^1_\gamma} \subseteq \tilde{s}^0$. Thus A^1_γ can be applied to the state $\tilde{s}^0$ and we obtain the next state:

$$\tilde{s}^1 := (\tilde{s}^0 \setminus \mathit{Eff}^{-}_{A^1_\gamma}) \cup \mathit{Eff}^{+}_{A^1_\gamma}$$

Note that $s^{n+1} = s^1 \setminus F \subseteq \tilde{s}^1$, which implies that $\mathit{Pre}^{+}_{A^2_\gamma} \subseteq \tilde{s}^1$. Also, $\tilde{s}^1 \subseteq s^{n+1} \cup \Gamma^{+}_{A^1_\gamma}$. Since A^2_γ is applicable in the state s^{n+1}, it follows that $\mathit{Pre}^{-}_{A^2_\gamma} \subseteq s^{n+1} = \emptyset$. The fact that $\mathbf{A}_\gamma$, being γ-safe, is executable implies, by Proposition 30, that $\Gamma^{+}_{A^1_\gamma} \cap \mathit{Pre}^{-}_{A^2_\gamma} = \emptyset$. Therefore $\mathit{Pre}^{-}_{A^2_\gamma} \subseteq \tilde{s}^1 = \emptyset$. A^2_γ is thus applicable in the state $\tilde{s}^1$ and there exists $\tilde{s}^2 \in \mathcal{S}$ such that $(\tilde{s}^1, \tilde{s}^2) \in \mathbf{S}_{A^2_\gamma}$. Therefore, we can conclude that $(\tilde{s}^0, \tilde{s}^1, \tilde{s}^2) \in \mathbf{S}_{\mathbf{A}_\gamma}$. Since $\mathbf{A}_\gamma$ is γ-safe and $w(\gamma, \tilde{s}^0) \leq w(\gamma, s^0) \leq 1$, we have that $w(\gamma, \tilde{s}^2) \leq 1$. At the same time, since $s^{n+1} \subseteq \tilde{s}^1$, we also have that $s^{n+2} \subseteq \tilde{s}^2$ and thus $w(\gamma, s^{n+2}) \leq 1$. Proof is complete.

□

Remark 39. *We conjecture that Theorem 38 could be generalised to sequence of actions of length longer than two, but the proof is much more complex. Since this extension is not needed in this paper, we do not present such a proof here.*

For future use, it will be convenient to have a definition of safeness that does not depend on the particular chosen instance.

Definition 40. *A sequence of action sets* $\mathbf{A}$ *is* safe *for a template* $\mathcal{T}$ *if it is* γ*-safe for every instance* γ *of* $\mathcal{T}$*. A sequence of action sets* $\mathbf{A}$ *is* strongly safe *for a template* $\mathcal{T}$ *if it is strongly* γ*-safe for every instance* γ *of* $\mathcal{T}$*. It is* weakly safe *if it is safe but not strongly safe.*

5.2. Safe ground durative actions

We now restrict our attention to durative actions $Da = (a^{\mathrm{st}}, a^{\mathrm{inv}}, a^{\mathrm{end}})$. If we interpret Da as a sequence of three actions, we can consider for it the properties defined for general sequences such as γ-safety and strong γ-safety. Before studying these properties, it is convenient to make some considerations on the way durative actions appear in admissible simple plans. Indeed, several constraints emerge as a direct consequence of the definition of an induced simple plan as well from the concept of admissibility explained in Section 2.1. Consider an admissible simple plan π with happening sequence $\mathbf{A}_\pi = (A_{t_0}, \dots, A_{t_{\bar{k}}})$. If a durative action Da happens in π in the time interval $[t_{i+1}, t_j]$, we have that $a^{\mathrm{st}} \in A_{t_{i+1}}$ and $a^{\mathrm{end}} \in A_{t_j}$. Moreover, $j - i$ is necessarily odd and for every even $h = 2, 4, \dots, j - i - 1$, $A_{t_{i+h}}$ consists of a^{inv} and, possibly, over all conditions of other durative actions happening in the original plan Π simultaneously or intertwined with Da. Finally, for $h = 1, 3, \dots, n - 2$, $A_{t_{i+h}}$ consists of actions that appears in simple admissible plans and that, consequently, inherit such constraints.

Definition 41 (Admissible actions)**.** *A sequence* $\mathbf{A} = (A^1, A^2, \dots, A^n)$ *is:*

- admissible *if there exists an admissible simple plan* π *with happening sequence* $\mathbf{A}_\pi = (A_{t_0}, \dots, A_{t_{\bar{k}}})$ *such that* $(A^1, A^2, \dots, A^n) = (A_{t_{i+1}}, \dots, A_{t_{i+n}})$ *for some* $i = 0, \dots, \bar{k} - n$.
- Da-admissible*, for some durative action Da, if it is admissible and Da happens in the corresponding simple plan* π *as above in* $[t_{i+1}, t_{i+n}]$*. In particular,* $a^{st} \in A^1$ *and* $a^{end} \in A^n$.

The existence of Da-admissible sequences that are executable imposes specific conditions on the durative action Da:

Proposition 42. *Consider a durative action $Da = (a^{st}, a^{inv}, a^{end})$ and assume that there exists an executable Da-admissible sequence. Then, the following conditions are satisfied:*

$$\Gamma^+_{a^{st}} \cap Pre^-_{a^{inv}} = \emptyset, \quad \Gamma^-_{a^{st}} \cap Pre^+_{a^{inv}} = \emptyset \tag{8}$$

$$Pre^+_{a^{end}} \cap Pre^-_{a^{inv}} = \emptyset, \quad Pre^+_{a^{end}} \cap Pre^-_{a^{inv}} = \emptyset \tag{9}$$

where the postconditions $\Gamma^{\pm}$ have been defined in (5).

Proof. Suppose that $\mathbf{A} = (A^1, A^2, \dots, A^n)$ is Da-admissible and executable. Since (A^1, A^2), in particular, is executable, Proposition 30 implies that

$$\Gamma^+_{A^1} \cap Pre^-_{A^2} = \emptyset = \Gamma^-_{A^1} \cap Pre^+_{A^2} \tag{10}$$

Since $a^{\mathrm{st}} \in A^1$ and A^1 consists of pairwise non-interfering actions, we have that

$$\Gamma^+_{a^{\mathrm{st}}} \subseteq \Gamma^+_{A^1}, \quad \Gamma^-_{a^{\mathrm{st}}} \subseteq \Gamma^-_{A^1} \tag{11}$$

Finally, equations (10) and (11), together with the fact that $a^{\mathrm{inv}} \in A^2$, yield conditions (8). Conditions (9) can be similarly proven. □

Durative actions not satisfying any of the conditions expressed in previous result, can be ignored in our analysis. From now on we thus assume that all durative actions satisfy condition (8) and (9) above.

To study the safety of a Da-admissible sequence, we can, in many cases, reduce the analysis of the durative action Da to the analysis of an auxiliary sequence of just two actions $Da_* = (a^{\mathrm{st}}_*, a^{\mathrm{end}}_*)$, where a^{st}_* and a^{end}_* are actions such that:

$$\begin{array}{ll} Eff^{\pm}_{a^{\mathrm{st}}_*} = Eff^{\pm}_{a^{\mathrm{st}}}, & Pre^{\pm}_{a^{\mathrm{st}}_*} = Pre^{\pm}_{a^{\mathrm{st}}} \cup (Pre^{\pm}_{a^{\mathrm{inv}}} \setminus Eff^{\pm}_{a^{\mathrm{st}}}) \\ Eff^{\pm}_{a^{\mathrm{end}}_*} = Eff^{\pm}_{a^{\mathrm{end}}}, & Pre^{\pm}_{a^{\mathrm{end}}_*} = Pre^{\pm}_{a^{\mathrm{end}}} \cup Pre^{\pm}_{a^{\mathrm{inv}}} \end{array}$$

Remark 43. *The executability assumption (2) automatically extends from a^{st} and a^{end} to the auxiliary actions a^{st}_* and a^{end}_*, as a consequence of condition (9).*

The relation between the two sequences Da and Da_* is clarified by the following result. Assume, as always, that a template $\mathcal{T}$ and an instance γ have been fixed.

Proposition 44. *The following facts hold true:*

(i) $(s^0, s^1, s^2) \in \mathbf{S}_{(a^{st}, a^{inv})}$ *if and only if* $s^1 = s^2$ *and* $(s^0, s^1) \in \mathbf{S}_{a^{st}_*}$*;*

(ii) $(s^0, s^1, s^2) \in \mathbf{S}_{(a^{inv}, a^{end})}$ *if and only if* $s^0 = s^1$ *and* $(s^1, s^2) \in \mathbf{S}_{a^{end}_*}$*;*

(iii) $(s^0, s^1, s^2, s^3) \in \mathbf{S}_{Da}$ *if and only if* $s^1 = s^2$ *and* $(s^0, s^1, s^3) \in \mathbf{S}_{Da_*}$*;*

(iv) (a^{st}, a^{inv}) *is individually* γ*-safe if and only if* a^{st}_* *is strongly* γ*-safe;*

(v) (a^{inv}, a^{end}) *is individually* γ*-safe if and only if* a^{end}_* *is strongly* γ*-safe;*

(vi) Da *is individually* γ*-safe if and only if* Da_* *is individually* γ*-safe.*

Proof. (i): Suppose $(s^0, s^1, s^2) \in \mathbf{S}_{(a^{\mathrm{st}}, a^{\mathrm{inv}})}$. Since a^{inv} only contains preconditions, we have that $s^1 = s^2$. Note now that $s^1 = \xi(a^{\mathrm{st}}, s^0) = (s^0 \cup \mathit{Eff}^+_{a^{\mathrm{st}}}) \setminus \mathit{Eff}^-_{a^{\mathrm{st}}}$ must satisfy the conditions $\mathit{Pre}^+_{a^{\mathrm{inv}}} \subseteq s^1 \subseteq (\mathit{Pre}^-_{a^{\mathrm{inv}}})^c$. This yields $\mathit{Pre}^+_{a^{\mathrm{inv}}} \subseteq s^0 \cup \mathit{Eff}^+_{a^{\mathrm{st}}}$ and thus $\mathit{Pre}^+_{a^{\mathrm{inv}}} \setminus \mathit{Eff}^+_{a^{\mathrm{st}}} \subseteq s^0$. Similarly, from $s^0 \setminus \mathit{Eff}^-_{a^{\mathrm{st}}} \subseteq (\mathit{Pre}^-_{a^{\mathrm{inv}}})^c$, we obtain that $s^0 \subseteq (\mathit{Pre}^-_{a^{\mathrm{inv}}} \setminus \mathit{Eff}^-_{a^{\mathrm{st}}})^c$. This implies that also a^{st}_* is applicable on s^0 and $s^1 = \xi(a^{\mathrm{st}}_*, s^0)$ since a^{st} and a^{st}_* have the same effects. If instead $(s^0, s^1) \in \mathbf{S}_{a^{\mathrm{st}}_*}$, we have that a^{st} is applicable on s^0 (since the preconditions of a^{st} are also preconditions of a^{st}_*) and $s^1 = \xi(a^{\mathrm{st}}_*, s^0) = \xi(a^{\mathrm{st}}, s^0)$. (ii) is proven similarly to (i). (iii) follows from (i) and (ii) and, finally, (iv), (v), and (vi) follow, respectively, from (i), (ii), and (iii). □

The next result studies the effect of exchanging the start and end of a durative action *Da* with those of the auxiliary sequence Da_* in a *Da*-admissible sequence.

Proposition 45. *Consider a durative action* $Da = (a^{st}, a^{inv}, a^{end})$ *and a Da-admissible sequence of actions* $\mathbf{A} = (\{a^{st}\}, A^2, \dots, A^{n-1}, \{a^{end}\})$*. Put* $\mathbf{A}_* = (\{a^{st}_*\}, A^2, \dots, A^{n-1}, \{a^{end}_*\})$*. Then* $\mathbf{S}_{\mathbf{A}} = \mathbf{S}_{\mathbf{A}_*}$*. In particular,* $\mathbf{A}$ *is individually* γ*-safe if and only if* $\mathbf{A}_*$ *is individually* γ*-safe.*

Proof. Since $\mathbf{A}_*$ differs from $\mathbf{A}$ only for having more preconditions, it holds that $\mathbf{S}_{\mathbf{A}} \supseteq \mathbf{S}_{\mathbf{A}_*}$. Conversely, suppose $(s^0, \dots, s^n) \in \mathbf{S}_{\mathbf{A}}$. Then, $(s^0, s^1, s^1) \in \mathbf{S}_{(a^{\mathrm{st}}, a^{\mathrm{inv}})}$. Therefore, by (i) of Proposition 44, we have that $(s^0, s^1) \in \mathbf{S}_{a^{\mathrm{st}}_*}$. Similarly, using (ii) of Proposition 44, we obtain that $(s^{n-1}, s^n) \in \mathbf{S}_{a^{\mathrm{end}}_*}$. These two facts together with $(s^1, s^2, \dots, s^{n-1}) \in \mathbf{S}_{(A^2, \dots, A^{n-1})}$, yield $(s^0, \dots, s^n) \in \mathbf{S}_{\mathbf{A}_*}$. □

The last proposition implies that, in analysing the state dynamics in an executable plan, we can replace the start and end of each durative action *Da* with the corresponding ones of the auxiliary sequence Da_*, if such start and end happen isolated from other actions. This is useful for two reasons. On the one hand, there are cases in which Da_* is strongly safe even if *Da* is not. On the other hand, we can directly apply Theorem 38 to Da_* since it is of length 2.

As we shall see later, our sufficient results for the invariance of a template always require safety (strong or simple) of the auxiliary actions $Da_* = (a^{\mathrm{st}}_*, a^{\mathrm{end}}_*)$. The check for strong safety can be done by considering the single components of Da_* and referring back to the analysis that we carried out in previous chapter. Below, we give a full characterisation of simple safety for auxiliary actions.

Note first that if $Da_* = (a^{\mathrm{st}}_*, a^{\mathrm{end}}_*)$ is weakly γ-safe (Definition 35), necessarily Da_* is executable, a^{st}_* is strongly γ-safe and a^{end}_* is not strongly γ-safe. If, besides these three properties, Da_* is γ-unreachable, then Da_* is weakly γ-safe because of Remark 29. If we instead assume that Da_* is weakly γ-safe and γ-reachable, then, because of Proposition 36, we have that a^{end}_* is γ-unbounded. The following result completely characterises simple γ-safety for such actions.

Proposition 46. *Let* $Da_* = (a^{st}_*, a^{end}_*)$ *be a* γ*-reachable sequence such that* a^{st}_* *is strongly* γ*-safe and* a^{end}_* *is* γ*-unbounded. Then,* Da_* *is weakly* γ*-safe if and only if one of the following mutually exclusive conditions are satisfied:*

(a) a^{st}_* *γ-irrelevant,* $|\mathit{Pre}^+_{a^{st}_{*\gamma}}| = 1$, $\mathit{Pre}^+_{a^{st}_{*\gamma}} \subseteq \mathit{Eff}^-_{a^{st}_{*\gamma}}$;

(b) a^{st}_* *γ-irrelevant,* $|\mathit{Pre}^+_{a^{st}_{*\gamma}}| = 1$, $\mathit{Pre}^+_{a^{st}_{*\gamma}} \nsubseteq \mathit{Eff}^-_{a^{st}_{*\gamma}}$, $\mathit{Pre}^+_{a^{st}_{*\gamma}} \subseteq \mathit{Eff}_{a^{end}_{*\gamma}}$;

(c) a^{st}_* *γ-irrelevant,* $|\mathit{Pre}^+_{a^{st}_{*\gamma}}| = 0$, $\mathit{Pre}^-_{a^{st}_{*\gamma}} \cup \mathit{Eff}_{a^{st}_{*\gamma}} \cup \mathit{Eff}_{a^{end}_{*\gamma}} = \gamma(\mathcal{T})$;

(d) a^{st}_* *γ-relevant,* $\mathit{Eff}^+_{a^{st}_{*\gamma}} \subseteq \mathit{Eff}_{a^{end}_{*\gamma}}$.

Proof. Note that Da_* is γ-reachable and a_*^{st} strongly γ-safe. Hence, Da_* is weakly γ-safe if and only if Da_{*_γ} is individually γ-safe. This last fact is equivalent to show that, given any state sequence $(s^0, s^1, s^2) \in \mathbf{S}_{Da_{*\gamma}}$ such that $s^0 \in \gamma(\mathcal{T})$ and $w(\gamma, s^0) \leq 1$, it holds that $w(\gamma, s^i) \leq 1$ for $i = 2$ (since for $i = 1$ that follows from the strong safety of a_*^{st}). Put

$$\mathcal{W}_\gamma = \{s^1 \in \gamma(\mathcal{T}) \,|\, \exists s^0, s^2 \in \gamma(\mathcal{T}),\ w(\gamma, s^0) \leq 1,\ (s^0, s^1, s^2) \in \mathbf{S}_{Da_{*\gamma}}\}$$

We need to show that, for every $s^1 \in \mathcal{W}_\gamma$, we have that $w(\gamma, s^2) \leq 1$, where

$$s^2 = \xi(a_{*\gamma}^{\mathrm{end}}, s^1) = (s^1 \setminus \mathit{Eff}^-_{a_{*\gamma}^{\mathrm{end}}}) \cup \mathit{Eff}^+_{a_{*\gamma}^{\mathrm{end}}}$$

Since a_*^{end} is γ-unbounded, the condition $w(\gamma, s^2) \leq 1$ is equivalent to

$$s^1 \subseteq \mathit{Eff}_{a_{*\gamma}^{\mathrm{end}}} \tag{12}$$

Since a_*^{st} is γ-reachable and strongly γ-safe, it follows from Theorem 18 that is either γ-irrelevant or γ-relevant. If a_*^{st} is γ-irrelevant and $|\mathit{Pre}^+_{a_{*\gamma}^{\mathrm{st}}}| = 1$, we have that $\mathcal{W}_\gamma = \{\mathit{Pre}^+_{a_{*\gamma}^{\mathrm{st}}} \setminus \mathit{Eff}^-_{a_{*\gamma}^{\mathrm{st}}}\}$. Combining with (12), we thus have that in this case Da_* is γ-safe if and only if

$$\mathit{Pre}^+_{a_{*\gamma}^{\mathrm{st}}} \setminus \mathit{Eff}^+_{a_{*\gamma}^{\mathrm{st}}} \subseteq \mathit{Eff}_{a_{*\gamma}^{\mathrm{end}}} \tag{13}$$

This leads to the two possible cases (a) and (b).

Suppose now that a_*^{st} is γ-irrelevant and $|\mathit{Pre}^+_{a_{*\gamma}^{\mathrm{st}}}| = 0$. In this case,

$$\mathcal{W}_\gamma = \{s^1 \subseteq \gamma(\mathcal{T}) \,|\, w(\gamma, s^1) \leq 1,\ \ s^1 \cap (\mathit{Pre}^-_{a_{*\gamma}^{\mathrm{st}}} \cup \mathit{Eff}^-_{a_{*\gamma}^{\mathrm{st}}}) = \emptyset\}$$

Combining with (12), we thus have that in this case Da_* is γ-safe if and only if

$$\mathit{Pre}^-_{a_{*\gamma}^{\mathrm{st}}} \cup \mathit{Eff}^-_{a_{*\gamma}^{\mathrm{st}}} \cup \mathit{Eff}_{a_{*\gamma}^{\mathrm{end}}} = \gamma(\mathcal{T}) \tag{14}$$

This leads to case (c).

Finally, if $a_{*\gamma}^{\mathrm{st}}$ is relevant we have that $\mathcal{W}_\gamma = \{\mathit{Eff}^+_{a_{*\gamma}^{\mathrm{st}}}\}$. Combining again with (12), we obtain that in this case Da_* is γ-safe if and only if condition (d) is verified. □

Remark 47. *If Condition (a) of Proposition 46 holds, this implies that the same conditions needs to be satisfied by a^{st}: $|Pre^+_{a^{st}_\gamma}| = 1$, $Pre^+_{a^{st}_\gamma} \subseteq Eff^-_{a^{st}_\gamma}$.*

Definition 48 (Weakly safe durative actions)**.** *We say that Da_* is* weakly γ-safe of type (x) *where $x \in \{a, b, c, d\}$ if it is γ-reachable, a_*^{st} is strongly γ-safe, a_*^{end} is γ-unbounded, and, finally, Da_* satisfies the condition (x) of Proposition 46.*

Example 9. *Consider a template $\mathcal{T}$ and an instance γ such that $\gamma(\mathcal{T}) = \{q, q'\}$. Figure 5 shows possible instances of actions of types (a)-(d).*

When the start or the end of a durative action Da happen simultaneously with other actions, the reduction of Da to Da_* cannot be performed in general as shown in the following example.

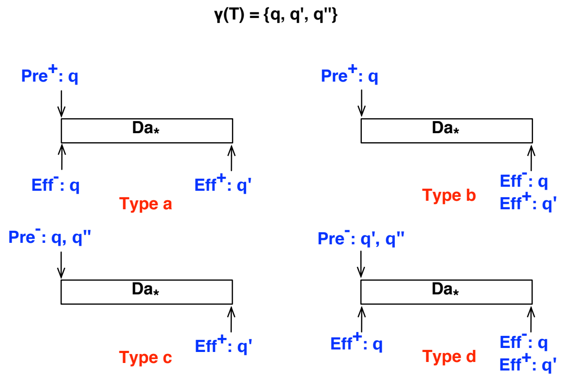


Figure 5: Examples of weakly γ-safe actions of types (a)-(d). See Example 9.

Example 10. *Consider a template $\mathcal{T}$ and an instance γ such that $\gamma(\mathcal{T}) = \{q, q', q''\}$. Figure 6 shows that, when the durative actions Da and Da′ are considered in isolation, both a^{st}_{*} and a'^{st}_{*} are strongly γ-safe since they are γ-unreachable. Since a^{end}_{*} and a'^{end}_{*} are γ-irrelevant, Da_{*} and Da'_{*} are strongly γ-safe and thus (by Proposition 44) Da and Da′ are individually γ-safe. However, if we now consider the case in which Da and Da′ happen simultaneously, giving rise to the sequence $\mathbf{A} = (A^1 = \{a^{st}, a'^{st}\}, A^2 = \{a^{inv}, a'^{inv}\})$, we see that $\mathbf{A}$ is not individually γ-safe. In fact, if we put $s^0 = \{q''\}$ with $w(\gamma, s^0) = 1$, we have that $s^1 = \xi(s^0, A^1) = \{q, q'', q'''\}$ with $w(\gamma, s^1) = 3$, which violates the definition of individual γ-safety.*

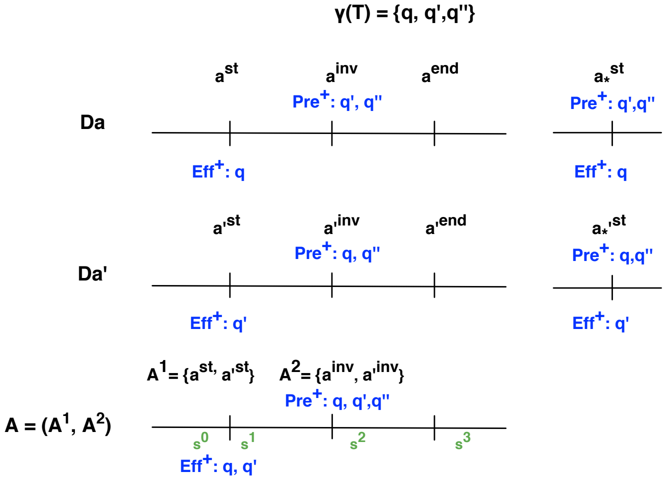


Figure 6: The sequence $\mathbf{A}$ is not individually γ-safe. See Example 10.

Note that, in the previous example, the two durative actions are γ-unreachable. The following result shows that such pathological phenomena can only happen in that case and will be instrumental for the results of the next section.

Proposition 49. *Let Da be a γ-reachable durative action such that a^{st} is not strongly γ-safe, while a_*^{st} is strongly γ-safe. Then,*

(i) a_*^{st} is γ-bounded;

(ii) for every action set A^1 such that $(\{a^{st}\} \cup A^1, a^{inv})$ is executable, $(\{a^{st}\} \cup A^1, a^{inv})$ is individually γ-safe.

Proof. Since Da is γ-reachable, it follows from Proposition 36, that a^{st} must necessarily be γ-unbounded. In particular, this yields $Pre^+_{a^{\mathrm{st}}_\gamma} = \emptyset$. Therefore, $Pre^+_{a^{\mathrm{st}}_{*\gamma}} = Pre^+_{a^{\mathrm{inv}}_\gamma} \setminus Eff^+_{a^{\mathrm{st}}_\gamma}$ cannot have any intersection with $Eff_{a^{\mathrm{st}}_\gamma}$. This says that a^{st}_* cannot be γ-balanced. Since it can neither be γ-unreachable (since Da is γ-reachable), it follows from Corollary 21 that a^{st}_* must be γ-bounded. This proves (i).

Suppose now that the sequence $(\{a^{\mathrm{st}}\} \cup A^1, a^{\mathrm{inv}})$ is executable and let $q \in Eff^+_{A^1_\gamma}$. By (i), it follows that $q \in Eff_{a^{\mathrm{st}}_\gamma} \cup Pre^-_{a^{\mathrm{st}}_{*\gamma}}$. Note that q cannot either belong to $Eff^-_{a^{\mathrm{st}}_\gamma}$ or $Pre^-_{a^{\mathrm{st}}_\gamma}$ since a^{st} and the actions in A^1 must be non-interfering. On the other hand, q cannot belong to $Pre^-_{a^{\mathrm{inv}}_\gamma}$ otherwise the sequence would not be executable. Therefore the only possibility is that $q \in Eff^+_{a^{\mathrm{st}}_\gamma}$. Therefore we have that $Eff^+_{A^1_\gamma} \subseteq Eff^+_{a^{\mathrm{st}}_\gamma}$. Consider now $\tilde{A}^1$ the action set obtained from A^1 by eliminating all positive effects belonging to $\gamma(\mathcal{T})$. We have that $\{a^{\mathrm{st}}\} \cup A^1 = \{a^{\mathrm{st}}\} \cup \tilde{A}^1$. Consider now the sequence $(\tilde{A}^1, a^{\mathrm{st}}, a^{\mathrm{inv}})$ and note that $\tilde{A}^1$ is γ-irrelevant, and $(a^{\mathrm{st}}, a^{\mathrm{inv}})$ is individually γ-safe because of (iv) of Proposition 44. Therefore, by Proposition 26, also $(\tilde{A}^1, a^{\mathrm{st}}, a^{\mathrm{inv}})$ is individually γ-safe, and thus also $(\{a^{\mathrm{st}}\} \cup A^1, a^{\mathrm{inv}})$. □

No similar results hold for the end parts of durative actions as next example shows.

Example 11. *Consider a template $\mathcal{T}$ and an instance γ such that $\gamma(\mathcal{T}) = \{q, q'\}$. When the durative actions Da and Da' (Figure 7) are considered in isolation, both a_*^{end} and a'^{end}_* are strongly γ-safe since they are γ-bounded. Since a_*^{st} and a'^{st}_* are γ-irrelevant, Da_* and Da'_* are strongly γ-safe and thus (by Proposition 44) Da and Da' are individually γ-safe. Moreover, Da and Da' are both γ-reachable. However, if Da and Da' happen simultaneously, giving rise to the sequence $\mathbf{A} = (A^1 = \{a^{inv}, a'^{inv}\}, A^2 = \{a^{end}, a'^{end}\})$, $\mathbf{A}$ is not individually γ-safe. If we put $s^0 = \emptyset$ with $w(\gamma, s^0) = 0$, we have that $s^1 = \xi(s^0, A^1) = \emptyset$ and $s^2 = \xi(s^1, A^2) = \{q, q'\}$ with $w(\gamma, s^2) = 2$, which violates the definition of individual γ-safety.*

6. Conditions for the invariance of a template

Any plan π where all actions are strongly safe, all durative actions are safe and take place in isolation, i.e. with no other actions happening in between them, yields a safe happening sequence $\mathbf{A}_\pi$, as a consequence of Corollary 27. The difficulty, in general, is that durative actions can in principle start or end together and be intertwined with other instantaneous or durative actions. Safety of durative actions must therefore be accompanied by suitable hypotheses guaranteeing that dangerous intertwinements or simultaneous happenings cannot take place in executable plans. In this way, we can work out sufficient conditions for the invariance of a template, which will be useful in analysing concrete examples.

In this section, we present two results that give *sufficient* conditions for the invariance of a template. The first deals with the particular case when all instantaneous actions are strongly safe

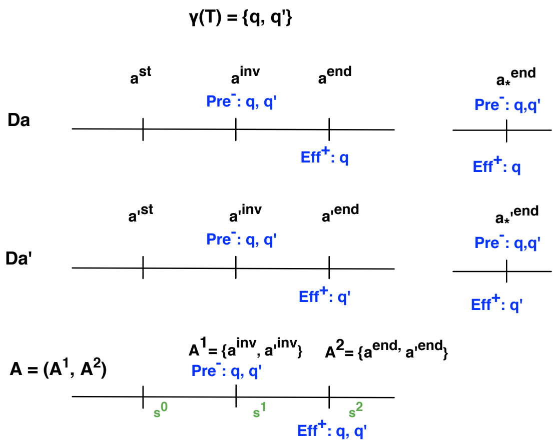


Figure 7: The sequence $\mathbf{A}$ is not individually γ-safe. See Example 11.

and all durative actions Da are such that Da_* is strongly safe. The second result considers a more general case when there are durative actions Da for which Da_* is only weakly safe.

Considering a template $\mathcal{T}$ and an instance γ, we denote by $\mathcal{GA}^d(\mathrm{wk}, \gamma)$ the collection of durative actions that are not strongly γ-safe and with $\mathcal{GA}^{\mathrm{st}}(\mathrm{wk}, \gamma)$ and $\mathcal{GA}^{\mathrm{end}}(\mathrm{wk}, \gamma)$, respectively, the collection of their start and end fragments. The following property prevents the simultaneous end of durative actions that could yield unsafe phenomena.

Definition 50 (Relevant right isolated actions)**.** *For a template $\mathcal{T}$, the set of durative actions $\mathcal{GA}^d$ is said to be* relevant right isolated *if, for every instance γ and for every $Da^1, Da^2 \in \mathcal{GA}^d(wk, \gamma)$, one of the following conditions is satisfied:*

(i) $|Eff^+_{a^{1end}_\gamma} \cup Eff^+_{a^{2end}_\gamma}| \leq 1$*;*

(ii) at least one of the two pairs $\{a^{1end}, a^{2end}\}$, $\{a^{1inv}, a^{2inv}\}$ is either mutex or non executable;

(iii) the sequence $(\{a^{1inv}, a^{2inv}\}, \{a^{1end}, a^{2end}\})$ is γ-unreachable.

Theorem 51. *Consider a template $\mathcal{T}$ and suppose that the set of instantaneous actions $\mathcal{GA}^i$ and that of durative actions $\mathcal{GA}^d$ satisfy the following properties:*

(i) every $a \in \mathcal{GA}^i$ is strongly safe;

(ii) for every instance γ and every $Da \in \mathcal{GA}^d(wk, \gamma)$, Da_* is γ-reachable and strongly γ-safe;

(iii) $\mathcal{GA}^d$ is relevant right isolated.

Then, $\mathcal{T}$ is invariant.

Proof. Fix any executable simple plan π with happening sequence $\mathbf{A}_\pi = (A_{t_0}, \ldots, A_{t_{\bar{k}}})$ and any instance γ. We prove that $\mathbf{A}_\pi$ is individually γ-safe.

We split happenings as follows: $A_{t_i} = A^{\mathrm{st}}_{t_i} \cup A^s_{t_i} \cup A^{\mathrm{end}}_{t_i}$ where

- $A^{\mathrm{st}}_{t_i}$ is either empty or consists in the start fragments in $\mathcal{GA}^d(\mathrm{wk}, \gamma)$;

- $A_{t_i}^{\text{end}}$ is either empty or consists in the ending fragments in $\mathcal{GA}^d(\text{wk}, \gamma)$;
- $A_{t_i}^s = A_t \setminus (A_{t_i}^{\text{st}} \cup A_{t_i}^{\text{end}})$ consists of strongly γ-safe actions (either instantaneous or possibly the starting and ending of durative ones in $\mathcal{GA}^d \setminus \mathcal{GA}^d(\text{wk}, \gamma)$).

Note that if $A_{t_i}^{\text{st}} \neq \emptyset$, it either consists of all strongly γ-safe actions and is thus strongly γ-safe, or there exists a durative action $Da \in \mathcal{GA}^d(\text{wk}, \gamma)$ such that a^{st} is not strongly safe and $a^{\text{st}} \in A_{t_i}^{\text{st}}$. Considering the second case, note that $A_{t_{i+1}}$ simply consists of $\{a^{\text{inv}}\}$ possibly together with other overall fragments of durative actions. The executability of $(A_{t_i}, A_{t_{i+1}})$ yields, by (ii) of Proposition 32, the executability of $(A_{t_i} \setminus A_{t_i}^{\text{st}}, A_{t_i}^{\text{st}}, a^{\text{inv}}, A_{t_{i+1}} \setminus \{a^{\text{inv}}\})$ and thus also of $(A_{t_i}^{\text{st}}, a^{\text{inv}})$. By hypothesis, Da is γ-reachable and a_*^{st} is strongly γ-safe, we can thus apply Proposition 49 and conclude that $(A_{t_i}^{\text{st}}, a^{\text{inv}})$ is individually γ-safe. Using (iii) of Proposition 26, we obtain that $(A_{t_i}^{\text{st}}, A_{t_{i+1}})$ is individually γ-safe. Therefore, in any case, if $A_{t_i}^{\text{st}} \neq \emptyset$, $(A_{t_i}^{\text{st}}, A_{t_{i+1}})$ is individually γ-safe.

Similarly, if $A_{t_i}^{\text{end}} \neq \emptyset$, it either consists of all strongly γ-safe actions and is thus strongly γ-safe, or there exists a durative action $Da \in \mathcal{GA}^d(\text{wk}, \gamma)$ such that a^{end} is not strongly γ-safe and $a^{\text{end}} \in A_{t_i}^{\text{end}}$. Suppose that it exists another durative action $Da' \in \mathcal{GA}^d(\text{wk}, \gamma)$ such that $a'^{\text{end}} \in A_{t_i}^{\text{end}}$ and $\{a^{\text{end}}, a'^{\text{end}}\}$ is γ-heavy. Necessarily, the two pairs $\{a^{\text{inv}}, a'^{\text{inv}}\}$, $\{a^{\text{end}}, a'^{\text{end}}\}$ are composed of non-interfering actions and each of them is executable. Since $\mathcal{GA}^d$ is right relevant isolated, this implies that the sequence $(\{a^{\text{inv}}, a'^{\text{inv}}\}, \{a^{\text{end}}, a'^{\text{end}}\})$ is γ-unreachable. Since $A_{t_{i-1}}$ only consists of actions with no effects, it then follows from Proposition 32 that also the sequence $(A_{t_{i-1}}, A_{t_i}^{\text{end}})$ is γ-unreachable and thus individually γ-safe. The other possibility is that $\mathit{Eff}^+_{A_{t_i\gamma}^{\text{end}}} = \mathit{Eff}^+_{a_\gamma^{\text{end}}}$. Consider in this case $\tilde{A}_{t_i}^{\text{end}}$ to be the action set obtained from $A_{t_i}^{\text{end}} \setminus \{a^{\text{end}}\}$ by eliminating all positive effects belonging to $\gamma(\mathcal{T})$. We have that $A_{t_i}^{\text{end}} = \{a^{\text{end}}\} \cup \tilde{A}_{t_i}^{\text{end}}$. Note now that $(a^{\text{inv}},)$ is individually γ-safe by (v) of Proposition 44. Considering that $A_{t_{i-1}} \setminus \{a^{\text{inv}}\}$ only contains preconditions and $\tilde{A}_{t_i}^{\text{end}}$ is strongly γ-safe, applying, in order, items (iii), (i) and (ii) of Proposition 26, we obtain that $(A_{t_{i-1}}, a^{\text{end}})$, $(A_{t_{i-1}}, a^{\text{end}}, \tilde{A}_{t_i}^{\text{end}})$, and finally $(A_{t_{i-1}}, A_{t_i}^{\text{end}})$ are individually γ-safe.

Note that, for each happening time t_i, there are four possibilities:

- $A_{t_i}^{\text{st}} = \emptyset$, $A_{t_i}^{\text{end}} = \emptyset$: in this case $A_{t_i} = A_{t_i}^s$ is strongly γ-safe by definition;
- $A_{t_i}^{\text{st}} \neq \emptyset$, $A_{t_i}^{\text{end}} = \emptyset$: in this case, since $A_{t_i}^s$ and $(A_{t_i}^{\text{st}}, A_{t_{i+1}})$ are individually γ-safe, using (i) and (ii) of Proposition 26, we obtain that also $(A_{t_i}^s, A_{t_i}^{\text{st}}, A_{t_{i+1}})$ and $(A_{t_i}, A_{t_{i+1}}) = (A_{t_i}^s \cup A_{t_i}^{\text{st}}, A_{t_{i+1}})$ are individually γ-safe.
- $A_{t_i}^{\text{st}} = \emptyset$, $A_{t_i}^{\text{end}} \neq \emptyset$: arguing analogously to the case above we obtain that $(A_{t_{i-1}}, A_{t_i})$ is individually γ-safe.
- $A_{t_i}^{\text{st}} \neq \emptyset$, $A_{t_i}^{\text{end}} \neq \emptyset$: arguing analogously to the case above we obtain that $(A_{t_{i-1}}, A_{t_i}, A_{t_{i+1}})$ is individually γ-safe.

Using Corollary 27 we obtain that $\mathbf{A}_\pi$ is individually γ-safe. □

Note that assumption (iii) in the statement of Theorem 51 is to exclude the simultaneous end of durative actions; if such phenomena can be excluded a-priori, the hypothesis can be removed[5].

[5] It seems more believable to synchronise the beginning of two actions than the ends; we can imagine defining a temporal planning language that a priori excludes simultaneous endings from having any other effect than if sequenced. We can also imagine making the same restriction for beginnings. It seems that PDDL2.1 tries to do so (for both beginnings and endings with its no-moving-targets rule), but in fact PDDL2.1 permits precise synchronisation to have a different result (as seen in Example 10). In any case, in such scenarios, condition (iii) - which is costly to check - can be omitted.

When there are durative actions Da for which Da_* is not strongly γ-safe, further hypotheses are needed in order to guarantee that the template $\mathcal{T}$ is invariant. The main point is that, in this case, not only simultaneity can be harmful, but also any intertwinement between such a durative action and other actions. The following examples show the type of phenomena that can happen and that any theorem extending Theorem 51 needs to prevent.

Example 12. *Consider a template $\mathcal{T}$ and an instance γ such that $\gamma(\mathcal{T}) = \{q, q', q''\}$. Both the durative actions Da and Da′ (Figure 8) are γ-safe. However, they can intertwine in such a way to give rise to a sequence that is not individually γ-safe: $\mathbf{A} = (A^1 = \{a^{st}\}, A^2 = \{a'^{st}\}, A^3 = \{a^{end}\}, A^4 = \{a'^{end}\})$. If we put $s^0 = \{q\}$ with $w(\gamma, s^0) = 1$, we have that $s^4 = \{q', q''\}$ with $w(\gamma, s^4) = 2$.*

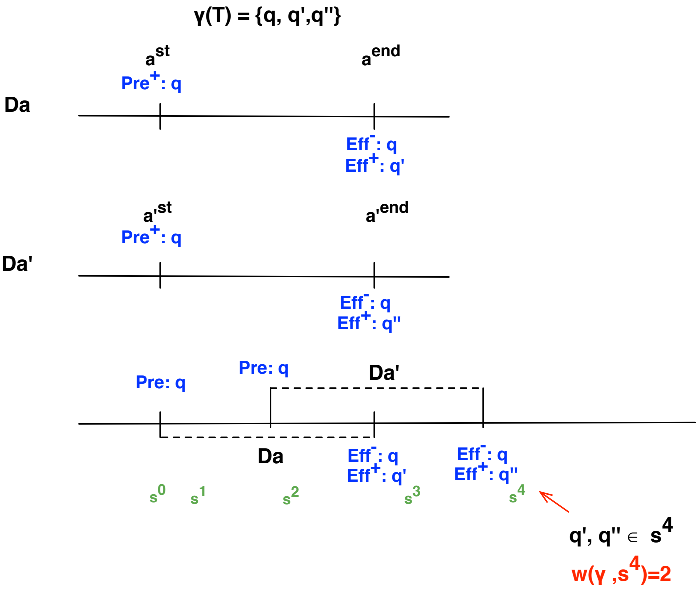


Figure 8: Action schemas Da and Da' can intertwine in such a way to give rise to a sequence that is not individually γ-safe. See Example 12.

The following definition describes a set of durative actions for which such phenomena cannot take place. It consists of three requirements acting, for each instantiation γ, on a subset $\mathcal{GA}^d(\gamma)$ of durative actions containing the potentially dangerous ones $\mathcal{GA}^d(\mathrm{wk}, \gamma)$. The first prevents the simultaneous happening of two start fragments of such durative actions. The second requirement states instead that between the happening of two such start fragments there must be the end of a third durative action in $\mathcal{GA}^d(\gamma)$: this fact, because of the finiteness of the plans, will lead to the impossibility of intertwining between durative actions in the family $\mathcal{GA}^d(\gamma)$. Finally, the third requirement prevents the possibility that other γ-relevant actions might happen in between a durative action in $\mathcal{GA}^d(\gamma)$.

Before stating the exact definition, we set the following notation. Given an instance γ and a $\mathcal{GA}^d(\gamma)$ subset of durative actions, we denote by $\mathcal{GA}^{\mathrm{st}}(\gamma)$ and $\mathcal{GA}^{\mathrm{end}}(\gamma)$, respectively, the set of its start and end fragments.

Definition 52 (Relevant non intertwining actions)**.** *Given a template $\mathcal{T}$, the set of durative actions $\mathcal{GA}^d$ is said to be* relevant non intertwining *if, for every instance γ, we can find a subset of durative actions $\mathcal{GA}^d(\gamma) \supseteq \mathcal{GA}^d(wk, \gamma)$ such that the following property is satisfied. For every $Da \in \mathcal{GA}^d(\gamma)$ and for every γ-reachable Da-admissible sequence of actions*

$$\mathbf{A} = (A^1, A^2, \dots, A^{n-1}, A^n) \tag{15}$$

with $A^1 \subseteq \mathcal{GA}^{st}(\gamma)$, the following conditions are satisfied:

(i) $A^1 = \{a^{st}\}$*;*

(ii) If $b \in A^j \cap \mathcal{GA}^{st}(\gamma)$ for $1 < j < n$, then there exists $b' \in A^{j'} \cap \mathcal{GA}^{end}(\gamma)$ for some $1 < j' \le j$;

(iii) If $A^j \cap (\mathcal{GA}^{st}(\gamma) \cup \mathcal{GA}^{end}(\gamma)) = \emptyset$ for every $j = 2, \dots, n-1$, then each A^j is γ-irrelevant for $j = 2, \dots, n-1$.

Considering, in this definition, subsets of durative actions possibly larger than $\mathcal{GA}^d(\mathrm{wk}, \gamma)$ leads to a more flexible theory and an instance of such a flexibility is later shown in Example 13.

The following is the main technical result of this section: it expresses a sufficient condition for a template to be invariant under the assumptions that instantaneous and durative actions are safe and that the relevant-non-intertwining property holds. Later on, we will look for more easy-to-check conditions that guarantee the relevant non intertwining property.

Theorem 53. *Consider a template $\mathcal{T}$ and suppose that the set of instantaneous actions $\mathcal{GA}^i$ and that of durative actions $\mathcal{GA}^d$ satisfy the following properties:*

(i) every $a \in \mathcal{GA}^i$ is strongly safe;

(ii) for every $Da \in \mathcal{GA}^d$, Da_* is safe;

(iii) the set $\mathcal{GA}^d$ is relevant non intertwining.

Then, $\mathcal{T}$ is invariant.

Proof. Fix any executable simple plan π with happening sequence $\mathbf{A}_\pi = (A_{t_0}, \dots, A_{t_{\bar{k}}})$ and any instance γ. Assume that $w(\gamma, Init) \le 1$. We prove that $\mathbf{A}_\pi$ is individually γ-safe.

Consider the set of durative actions $\mathcal{GA}^d(\gamma)$ as in Definition 52. Suppose that we can prove that if $Da \in \mathcal{GA}^d(\gamma)$ appears in π on the time window $[t_h, t_k]$ (namely, $a^{\mathrm{st}} \in A_{t_h}$ and $a^{\mathrm{end}} \in A_{t_k}$), the corresponding action sequence $\mathbf{A} = (A_{t_h}, \dots, A_{t_k})$ satisfies the following conditions:

(a) for every $i \in (h, k)$, A_{t_i} consists exclusively of γ-irrelevant actions;

(b) for every $i \in [h, k)$, A_{t_i} does not contain actions in $\mathcal{GA}^{\mathrm{st}}(\gamma)$ except $a^{\mathrm{st}} \in A_{t_h}$.

Note that if (b) holds true for every $Da \in \mathcal{GA}^d(\gamma)$, we also have automatically that,

(c) for every $i \in (h, k]$, A_{t_i} does not contain actions in $\mathcal{GA}^{\mathrm{end}}(\gamma)$ except $a^{\mathrm{end}} \in A_{t_k}$.

Assuming this to hold, we now proceed as in the proof of Theorem 51 and we split each happening A_{t_i} in the following way. We put $A_{t_i} = A^{\mathrm{st}}_{t_i} \cup A^{s}_{t_i} \cup A^{\mathrm{end}}_{t_i}$ where:

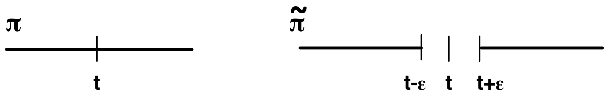
- $A^{\mathrm{st}}_{t_i}$ is either empty or consists in a start fragment in $\mathcal{GA}^{\mathrm{st}}(\gamma)$;

- $A^{\text{end}}_{t_i}$ is either empty or consists in an ending fragment in $\mathcal{G}\mathcal{A}^{\text{end}}(\gamma)$;
- $A^{s}_{t_i} = A_t \setminus (A^{\text{st}}_{t_i} \cup A^{\text{end}}_{t_i})$ only consists of strongly γ-safe actions.

We now consider the new plan $\tilde{\pi}$:

$$\tilde{\pi} = \{(t, a) \in \pi \,|\, a \in A^{s}_{t}\} \cup \{(t - \epsilon, a) \in \pi \,|\, a \in A^{\text{end}}_{t}\} \cup \{(t + \epsilon, a) \in \pi \,|\, a \in A^{\text{st}}_{t}\}$$

where $\epsilon > 0$ is chosen in such a way that $\epsilon < t_{i+1} - t_i$ for every $i = 0, \dots, \bar{k} - 1$.



It follows from Proposition 1 on serializability that plan $\tilde{\pi}$ is also executable. We denote its happening sequence as $\mathbf{A}_{\tilde{\pi}} = (\tilde{A}_{t_0}, \dots, \tilde{A}_{t_{\bar{k}}})$. For the sake of notation simplicity, happening times are denoted as those in π even if in general they differ and form a larger set. Note now that the happening times in $\tilde{\pi}$ can be split into singletons t_i such that $\tilde{A}_{t_i}$ only consists of strongly γ-safe actions, and intervals $[t_{i+1}, t_j]$ such that there exists a durative action $Da \in \mathcal{G}\mathcal{A}^{d}(\gamma)$ happening in that interval. In this case, we have that the subsequence $\mathbf{A} = (\tilde{A}_{t_{i+1}}, \dots, \tilde{A}_{t_j})$ is Da-admissible. Put $\mathbf{A}_* = (\tilde{A}_{t_i} \cup \{a^{\text{inv}}\}, \dots, A_{t_j} \cup \{a^{\text{inv}}\})$. Note that, since $\mathbf{A}$ is executable (as it appears in an executable plan), also $\mathbf{A}_*$ is executable by Proposition 45. Since, by assumption (ii), Da_* is γ-safe, it follows from Theorem 38, that $\mathbf{A}_*$ is also γ-safe. Using again Proposition 45 we finally obtain that $\mathbf{A}$ is individually γ-safe.

We have thus proven that each happening time t_i in the new plan $\tilde{\pi}$ stays inside an individually γ-safe sequence (possibly of length 1). By Corollary 27 this implies that $\mathbf{A}_{\tilde{\pi}}$ is individually γ-safe. A repetitive use of (ii) of Proposition 26 now yields that $\mathbf{A}_{\pi}$ is also individually γ-safe.

We are thus left with proving that every durative action $Da \in \mathcal{G}\mathcal{A}^{d}(\gamma)$ happening in π satisfies properties (a) and (b) stated above. Suppose this is not true and let Da be the first (as starting time) to happen in π (in the time window $[t_h, t_k]$) and to violate either condition (a) or (b). Note that all durative actions in $\mathcal{G}\mathcal{A}^{d}(\gamma)$ happening in π and starting strictly before time t_h, will necessarily end at a time $t \leq t_h$ by the way Da has been chosen. Moreover, all such durative actions will satisfy properties (a) and (b). We can then proceed as before and consider the splitting $A_{t_i} = A^{\text{st}}_{t_i} \cup A^{s}_{t_i} \cup A^{\text{end}}_{t_i}$ for every $i \leq h$ (note that in t_h there could be, in principle, more than one starting actions in $A^{\text{st}}_{t_h}$). Consider now the auxiliary plan $\tilde{\pi}$ constructed exactly like before for $t \leq t_h$ and coinciding with π for $t > t_h$. As before, we denote its happening sequence as $\mathbf{A}_{\tilde{\pi}} = (\tilde{A}_{t_0}, \dots, \tilde{A}_{t_{\bar{k}}})$ using the same notation for the happening times as in π and we assume that $\tilde{A}_{t_h} = A^{\text{st}}_{t_h}$ (this is for simplicity of notation considering that it would be instead $\tilde{A}_{t_h + \epsilon} = A^{\text{st}}_{t_h}$). Note that since π is executable and $w(\gamma, \mathit{Init}) \leq 1$, the sequence $\mathbf{A}_{\pi}$ is γ-reachable and consequently, because of (ii) of Proposition 32, also $\mathbf{A}_{\tilde{\pi}}$ is γ-reachable. Arguing as above, we obtain that $(\tilde{A}_{t_0}, \dots, \tilde{A}_{t_{h-1}})$ is individually γ-safe. If we take any $(\tilde{s}_0, \dots, \tilde{s}_{\bar{k}}) \in \mathbf{S}_{\mathbf{A}_{\tilde{\pi}}}(\gamma)$, we thus have that $w(\gamma, \tilde{s}_{h-1}) \leq 1$. Consider now $\mathbf{A} = (\tilde{A}_{t_h}, \tilde{A}_{t_{h+1}}, \dots, \tilde{A}_{t_k}) = (A^{\text{st}}_{t_h}, A_{t_{h+1}}, \dots, A_{t_k})$ and note that, $(\tilde{s}_{h-1}, \dots, \tilde{s}_k) \in \mathbf{S}_{\mathbf{A}}(\gamma)$ so that $\mathbf{A}$ is γ-reachable. It then follows from the relevant non intertwining property ((i) of Definition 52) that $A^{\text{st}}_{t_h} = \{a^{\text{st}}\}$. Suppose now that property (b) stated above is not satisfied and let $l \in (h, k)$ be the first index such that $A_{t_l} \cap \mathcal{G}\mathcal{A}^{\text{st}}(\gamma) \neq \emptyset$. By (ii) of Definition 52, it follows there must exist a durative action $Da' \in \mathcal{G}\mathcal{A}^{d}(\gamma)$ such that $a'^{\text{end}} \in A_{t_{l'}}$ for some $l' \in (h, l]$. Note that such durative action cannot start, in the plan π and thus also in the plan $\tilde{\pi}$, before time t_h for the way Da was chosen, it cannot either start at time t_h by previous considerations and neither in the interval $(t_h, t_{l'})$ by the way l' has been chosen. This proves that property (b) must be satisfied. Note that this also shows that A_{t_i} does not contain actions in $\mathcal{GA}^{\text{end}}(\gamma)$ for any $i \in (h, k)$ (as the corresponding start fragment cannot happen neither before or after time t_h). Finally, A_{t_i} is γ-irrelevant for every $i \in (h, k)$ because of (iii) of Definition 52. Therefore $\mathbf{A}$ satisfies properties (a) and (b) contrarily to the assumptions made on Da. Proof is now complete. □

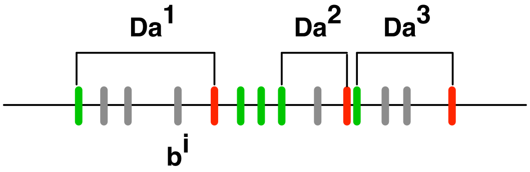


Figure 9: Structure of a plan $\tilde{\pi}$ as constructed in Theorem 53. Strongly safe actions are indicated in green, relevant in red and irrelevant in grey.

The properties that the set of durative actions $\mathcal{GA}^d$ needs to satisfy to be relevant non intertwining, which are expressed in Definition 52, are in general difficult to check as they require considering sequences of actions of possibly any length. Below we give a sufficient condition that guarantees such properties to hold and that only involves pairs of durative and instantaneous actions. It is computationally much simpler and suitable to be later analysed at the lifted level of action schemas.

We start with a property for pairs of actions a, a' that, when verified, prevents the happening of the action a followed by a', in any executable plan, where only irrelevant actions in a certain family happen in between. See (5) to recall the definition of the postconditions Γ_a^+ and Γ_a^- of an action a.

Definition 54 ($M(\gamma)$-unreachable actions)**.** *Consider a template $\mathcal{T}$, an instantiation γ and a subset $M(\gamma) \subseteq \mathcal{GA}$ of γ-irrelevant actions. A pair of actions (a, a') is $M(\gamma)$-*unreachable *if any of the following conditions are satisfied:*

(i) there exists $q \in \Gamma_a^+ \cap Pre_{a'}^-$ such that, for every $a'' \in M(\gamma)$, $q \notin Eff_{a''}^-$;

(ii) there exists $q \in \Gamma_a^- \cap Pre_{a'}^+$ such that, for every $a'' \in M(\gamma)$, $q \notin Eff_{a''}^+$;

(iii) $|Pre_{a_\gamma}^+ \cup (Pre_{a'_\gamma}^+ \setminus Eff_{a_\gamma}^+)| > 1$.

The first condition essentially says that the application of the action a leads to a state containing a ground atom q that needs to be false in order to then apply a' and that there is no action in $M(\gamma)$ that can make this atom false. The second condition is the analogous of the first, but exchanges the role of true and false atoms. Finally, the third condition requires that (a, a') is a γ-unreachable pair, assuming that it is executable. The following result explains the name that we have chosen for these properties.

Proposition 55. *If a pair of actions (a, a') is $M(\gamma)$-*unreachable*, any sequence of actions*

$$\mathbf{A} = (\{a\}, A^2, \dots, A^{n-1}, \{a'\})$$

such that, for $i = 2, \dots, n-1$, $A^i \subseteq M(\gamma)$, is γ-unreachable.

Proof. Assume that, by contradiction, there exists a γ-reachable sequence

$$\mathbf{A} = (\{a\}, A^2, \ldots, A^{n-1}, \{a'\})$$

such that $A^j \subseteq M(\gamma)$ for each j. Consider $(s^0, \ldots, s^n) \in \mathbf{S_A}(\gamma)$.

Suppose condition (i) is satisfied. We have that $q \in s^1$ and, because of the assumption made, it follows that $q \notin \mathit{Eff}^-_{A^j}$ for every $j = 2, \ldots, n-1$. Therefore, $q \in s^{n-1}$. Since $q \in \mathit{Pre}^-_{a'}$, this is a contradiction.

A similar argument can be used if instead condition (ii) is satisfied.

Finally, assume that condition (iii) is satisfied. Note that, since $A^2, \ldots, A^{n-1}$ are γ-irrelevant, $\mathit{Pre}^+_{a_\gamma} \cup (\mathit{Pre}^+_{a'_\gamma} \setminus \mathit{Eff}^+_{a_\gamma}) \subseteq s^0$ and this contradicts the fact that $(s^0, \ldots, s^n) \in \mathbf{S_A}(\gamma)$. □

The following is a stronger version of the relevant non intertwining property that is completely formulated at the level of pairs of actions without referring to sequences. It consists of three points that correspond to the three properties required in Definition 52. It will be convenient to first set a notation. Given an instance γ and a $\mathcal{GA}^d(\gamma)$ subset of durative actions, we denote by $\mathcal{GA}(\gamma)_{\rm irr}$ and $\mathcal{GA}(\gamma)_{\rm rel}$ the set of actions in $\mathcal{GA} \setminus (\mathcal{GA}^{\rm st}(\gamma) \cup \mathcal{GA}^{\rm end}(\gamma))$ that are, respectively, γ-irrelevant and γ-relevant.

Definition 56 (Pairwise relevant non-overlapping actions)**.** *For a template $\mathcal{T}$, the set of durative actions $\mathcal{GA}^d$ is said to be* pairwise relevant non-overlapping *if, for every instance γ, we can find a subset of durative actions $\mathcal{GA}^d(\gamma) \supseteq \mathcal{GA}^d(wk, \gamma)$ such that the following properties are satisfied:*

A for every $Da^1, Da^2 \in \mathcal{GA}^d(\gamma)$, one of the following conditions holds true:

- *(Ai) at least one of the two pairs $\{a^{1st}, a^{2st}\}$, $\{a^{1inv}, a^{2inv}\}$ is either mutex or non executable;*
- *(Aii) the sequence $(\{a^{1st}, a^{2st}\}, \{a^{1inv}, a^{2inv}\})$ is γ-unreachable;*
- *(Aiii) the pairs $\{a^{1inv}, a^{2end}\}$, $\{a^{1end}, a^{2inv}\}$ are mutex and, if Da^1 and Da^2 are not equivalent, the pair $\{a^{1end}, a^{2end}\}$ is either mutex or non executable.*

B for every $Da^1, Da^2 \in \mathcal{GA}^d(\gamma)$, one of the following conditions holds true:

- *(Bi) $\{a^{1inv}, a^{2st}\}$ is mutex;*
- *(Bii) $\{a^{1st}, a^{2st}\}$ is $\mathcal{GA}(\gamma)_{irr}$-unreachable;*
- *(Biii) the pairs $\{a^{1inv}, a^{2end}\}$, $\{a^{1end}, a^{2inv}\}$ are mutex and the pair $\{a^{1end}, a^{2end}\}$ is either mutex or non executable.*

C for every $Da^1 \in \mathcal{GA}^d(\gamma)$, $a^2 \in \mathcal{GA}(\gamma)_{rel}$, one of the following conditions is satisfied:

- *(Ci) $\{a^{1inv}, a^2\}$ is mutex;*
- *(Cii) $\{a^{1st}, a^2\}$ is $\mathcal{GA}(\gamma)_{irr}$-unreachable;*

The next result expresses a sufficient condition for the set $\mathcal{GA}^d$ to be relevant non intertwining.

Proposition 57. *Consider a template $\mathcal{T}$. The set $\mathcal{GA}^d$ is relevant non intertwining if the following conditions are satisfied:*

(i) for every $Da \in \mathcal{GA}^d$, a^{st} is strongly safe;

(ii) $\mathcal{GA}^d$ is pairwise relevant non-overlapping

Proof. For a fixed γ, consider the set of durative actions $\mathcal{GA}^d(\gamma)$ as in Definition 56. Let $Da \in \mathcal{GA}^d(\gamma)$ and consider a γ-reachable Da-admissible sequence

$$\mathbf{A} = (A^1, A^2, \dots, A^{n-1}, A^n)$$

such that $A^1 \subseteq \mathcal{GA}^{\mathrm{st}}(\gamma)$. We first show that property (i) in Definition 52 holds true. Assume, by contradiction, that $Da' \neq Da$ is such that $a'^{\mathrm{st}} \in A^1$. We now show that Da and Da' violate property A in the above definition. Indeed, since $\mathbf{A}$ is admissible, $\{a^{\mathrm{st}}, a'^{\mathrm{st}}\}$ and $\{a^{\mathrm{inv}}, a'^{\mathrm{inv}}\}$ are non-interfering and executable. We now show that $(\{a^{\mathrm{st}}, a'^{\mathrm{st}}\}, \{a^{\mathrm{inv}}, a'^{\mathrm{inv}}\})$ is γ-reachable. Note first that (A^1, A^2) is clearly γ-reachable. By assumption (ii), $\tilde{A}^1 = A^1 \setminus \{a^{\mathrm{st}}, a'^{\mathrm{st}}\}$ only consists of strongly γ-safe actions. Consider $\tilde{\mathbf{A}} = (\tilde{A}_1, \{a^{\mathrm{st}}, a'^{\mathrm{st}}\}, A^2)$. Notice that if $(s^0, s^1, s^2) \in S_{(A^1,A^2)}(\gamma)$, then $(s^0, \tilde{s}^1, s^1, s^2) \in S_{\tilde{\mathbf{A}}}(\gamma)$ for some $\tilde{s}^1$ such that $w(\gamma, \tilde{s}^1) \leq 1$. This implies that $\mathbf{B} = (\{a^{\mathrm{st}}, a'^{\mathrm{st}}\}, A^2)$ is also γ-reachable since $(\tilde{s}^1, s^1, s^2) \in S_{\mathbf{B}}$. Considering that A^2 only consists of invariants, by (i) of by Proposition 32, we have that $(\{a^{\mathrm{st}}, a'^{\mathrm{st}}\}, \{a^{\mathrm{inv}}, a'^{\mathrm{inv}}\})$ is γ-reachable. Finally, note that the end fragment of the durative action Da' can either appear in A^k for some $k < n$, or appear in A^n, or not appear in the sequence as the durative action Da' ends (in the plan of which $\mathbf{A}$ is a subsequence) after the end of Da. In the first and the third case we have, respectively, that $\{a^{inv}, a'^{end}\}$ or $\{a^{end}, a'^{inv}\}$ is non-interfering. In the second case, $\{a^{end}, a'^{end}\}$, is either mutex or non executable. Note that this second case can only happen when Da and Da' are not equivalent (they must differ in at least a fragment by the standing assumption made on simple plans). This shows that also (Aiii) doe not hold. Therefore $\mathcal{GA}^{\mathrm{st}}(\gamma) = \{a^{\mathrm{st}}\}$. This proves (i) in Definition 52.

Suppose now that that (ii) in Definition 52 does not hold true for $\mathbf{A}$. Let $j > 1$ be the first index for which (ii) is violated, namely for which we can find $Da' \in \mathcal{GA}^d(\gamma)$ such that $a'^{\mathrm{st}} \in A^j$ while $A^{j'} \cap \mathcal{GA}^{\mathrm{end}}(\gamma) = \emptyset$ for every $0 < j' < j$. The pair $\{a^{\mathrm{inv}}, a'^{\mathrm{st}}\}$ is non-interfering by Definition 41 of an admissible sequence. Moreover, arguing as in previous point above regarding the timing of the end fragments of Da and Da', we conclude that either one of the pairs $\{a^{1inv}, a^{2end}\}$, $\{a^{1end}, a^{2inv}\}$ must be non-interfering or the pair $\{a^{1end}, a^{2end}\}$ must be non-interfering and executable. Since $(\{a^{\mathrm{st}}\}, A^2, \dots, A^{j-1}, \{a'^{\mathrm{st}}\})$ is γ-reachable, it follows from assumption B in Definition 56 that there must exist $0 < j' < j$ such that $A^{j'} \not\subseteq \mathcal{GA}(\gamma)_{\mathrm{irr}}$. Let j' be the first index for which this happens. Since, by construction, $A^{j'} \cap (\mathcal{GA}^{\mathrm{st}}(\gamma) \cup \mathcal{GA}^{\mathrm{end}}(\gamma)) = \emptyset$, there must exist $b \in A^{j'} \cap \in \mathcal{GA}(\gamma)_{\mathrm{rel}}$. Clearly, $\{a^{\mathrm{inv}}, b\}$ is non interfering and since $(\{a^{\mathrm{st}}\}, A^2, \dots, A^{j'-1}, \{b\})$ is γ-reachable and by construction $A^h \subseteq \mathcal{GA}(\gamma)_{\mathrm{irr}}$ for every $h = 2, \dots, j' - 1$, property C in Definition 56 is violated. Therefore this proves (ii) in Definition 52.

Suppose now that $A^j \cap (\mathcal{GA}^{\mathrm{st}}(\gamma) \cup \mathcal{GA}^{\mathrm{end}}(\gamma)) = \emptyset$ for every $j = 2, \dots, n-1$. If (iii) in Definition 52 does not hold true for $\mathbf{A}$, consider $j > 1$ to be the first index for which (iii) is violated, namely A^j is not γ-irrelevant, and let $b \in A^j$ be any action which is not γ-irrelevant. Clearly, $b \in\in \mathcal{GA}(\gamma)_{\mathrm{rel}}$ and, arguing like above, we deduce that Da and b again violate property C. Proof is thus complete. □

Based on the previous results, we conclude with a simple sufficient condition for the invariance, which is very useful in analysing concrete cases.

Corollary 58. *Consider a template $\mathcal{T}$ and suppose that, for every instance γ,*

- *every $Da \in \mathcal{GA}^d(wk, \gamma)$ is such that Da_* is weakly γ-safe of type (a);*

- *every $a \in \mathcal{GA} \setminus (\mathcal{GA}^{st}(wk, \gamma) \cup \mathcal{GA}^{end}(wk, \gamma))$ is either γ-irrelevant or γ-balanced.*

Then, $\mathcal{T}$ is invariant.

Proof. It is clear that condition (i) and (ii) of Theorem 53 are satisfied. In order to check that $\mathcal{GA}^d$ is relevant non intertwining, we show that the properties (i) and (ii) of Proposition 57 are satisfied. Fix any instance γ. Note that the assumption that Da_* is weakly γ-safe of type (a) implies, in particular, that $a^{*\mathrm{st}}$ and a^{st} are irrelevant and thus strongly γ-safe. Therefore property (i) is satisfied. We now show that, choosing $\mathcal{GA}^d(\gamma) = \mathcal{GA}^d(\mathrm{wk}, \gamma)$, properties A, B, and C of the pairwise relevant non overlapping property are satisfied

Consider $Da^1, Da^2 \in \mathcal{GA}^d(\mathrm{wk}, \gamma)$. It follows from the fact that Da^1_* and Da^2_* are both weakly γ-safe of type (a) (see Remark 47) that

$$Pre^+_{a^{i\,st}_\gamma} = \{q^i\} \subseteq Eff^-_{a^{i\,st}_\gamma},\ i = 1, 2 \tag{16}$$

If $a^{1\,\mathrm{st}}$ and $a^{2\,\mathrm{st}}$ are non-interfering, it follows that $q^1 \neq q^2$ and, in this case, $\{a^{1\,\mathrm{st}}, a^{2\,\mathrm{st}}\}$ is γ-unreachable. This proves property A. In order to prove B, we go back to (16) and we consider two possible cases. If $q^1 = q^2$ we have that $q^1 \in \Gamma^-_{a^{1\,\mathrm{st}}} \cap Pre^+_{a^{2\,\mathrm{st}}}$ and, since $q^1 \in \gamma(\mathcal{T})$, for sure $q^1 \notin Eff^+_{a''}$ for any a'' that is γ-irrelevant. This implies that condition (ii) of Definition 54 is satisfied. If instead $q^1 \neq q^2$, we have that the condition (iii) of Definition 54 is instead satisfied. In any case this says that the pair $(a^{1\,\mathrm{st}}, a^{2\,\mathrm{st}})$ is $\mathcal{GA}(wk, \gamma)_{\mathrm{irr}}$–unreachable. This proves B. Finally C can be proven exactly like B. □

Our results have a broader application than the case in Corollary 58 as shown in the following example.

Example 13. *Consider a domain with two durative actions Da^1 and Da^2 (shown in Figure 10) and a template $\mathcal{T}$ with just an instance γ such that $\gamma(\mathcal{T}) = \{q, q'\}$. Da^1 is strongly γ-safe, while Da^2 is weakly γ-safe of type (a). Note that Corollary 58 cannot be applied since a^{1end} is γ-bounded. Indeed, the template is not invariant: Figure 10 shows a triple intertwinement of two copies of Da^1 and Da^2 that leads to a sequence that is not individually γ-safe. This sequence can certainly be thought as the happening sequence of an executable simple plan. Consequently, $\mathcal{T}$ is not invariant.*

If we modify the durative action Da^1 adding an overall condition $Pre_{a^{1inv}} = \{q\}$, the intertwining proposed would no longer be an admissible sequence. Even if Corollary 58 can still not be applied, we now show that, with this modification, the set $\mathcal{GA}^d = \{Da^1, Da^2\}$ is pairwise relevant non overlapping. To show this, we fix $\mathcal{GA}^d(\gamma) = \{Da^1, Da^2\}$ in Definition 56. Conditions A and B must be checked for any ordered pair of durative actions. In our case, there are four possibilities: two identical copies of Da^1 or of Da^2, the pair Da^1, Da^2 or the pair Da^2, Da^1. Note that if we pick two copies of Da^1, conditions (Aiii) and (Biii) hold true because of the overall condition. If we pick two copies of Da^2, (Ai) holds true. Moreover, considering that $\mathcal{GA}(\gamma)_{irr} = \emptyset$, it follows that the pair (a^{2st}, a^{2st}) is $\mathcal{GA}(\gamma)_{irr}$-unreachable (condition (ii) in Definition 54 is satisfied). Therefore two copies of Da^2 satisfy condition (Bii). Finally, if we consider the pair Da^1, Da^2 or Da^2, Da^1, we have that $Pre^+_{a^{1st}, a^{2st}} = \{q, q'\}$ so that $\{a^{1st}, a^{2st}\}$ is γ-unreachable and thus (Aii) is satisfied. Using again condition (ii) in Definition 54 we can check that both pairs (a^{1st}, a^{2st}) and (a^{2st}, a^{1st}) are $\mathcal{GA}(\gamma)_{irr}$-unreachable, so that (Bii) hold true. Finally, note that condition C is empty in our case since $\mathcal{GA}(\gamma)_{rel} = \emptyset$. Thanks to Proposition 57 and Theorem 53, we conclude that $\mathcal{T}$ is invariant.

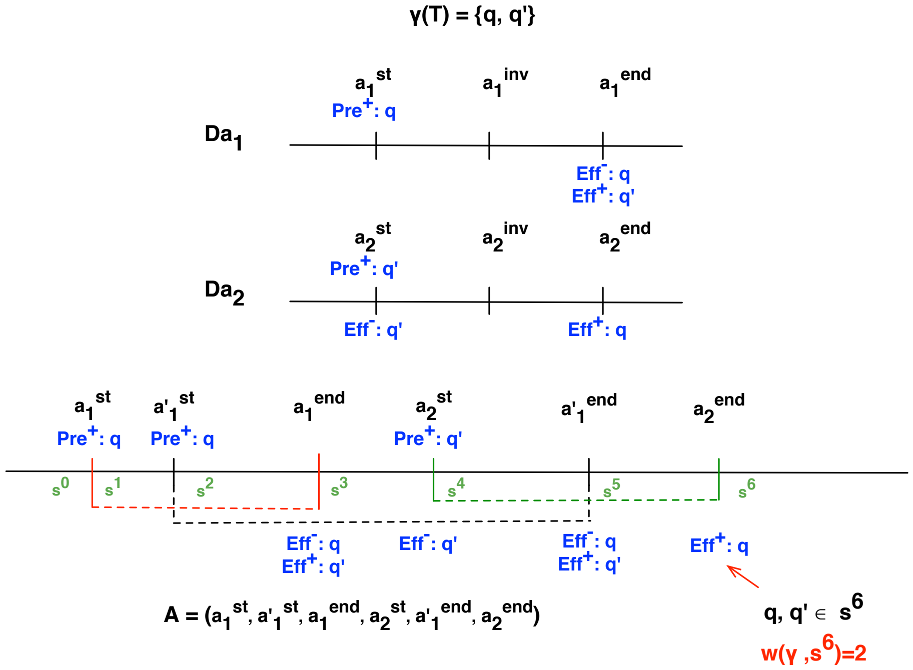


Figure 10: A triple intertwinement of two copies of the action schemas Da^1 and Da^2 leads to a sequence that is not individually γ-safe. See Example 13.

7. Safety of Action Schemas for a Template

In Section 6, we present results that guarantee the invariance of a template, Theorems 51 and 53 and Corollary 58. To be applied, we need to check that all instantaneous and durative actions satisfy a safety condition as well as that other extra properties, which prevent potentially dangerous simultaneous happenings or intertwinements among actions, hold true. Since we aim to find invariants off-line quickly and efficiently, our algorithm does not work at the level of actions. Instead, it reasons at the *lifted* level and uses the structure of the action schemas, i.e. their conditions and effects, to decide whether the ground instantiations of these schemas are safe or not. Our main goal in this section is to obtain lifted versions of Theorems 51 and 53 and Corollary 58.

In this and in the following sections, we make an important assumption on the groundings of the action schemas. Precisely, for any action schema α, instantaneous or durative, we always assume that the grounding functions we consider $gr : V_\alpha \to O$ are injective. This restriction plays a crucial role in the formulations of our results and, on the other hand, does not essentially entail any loss of generality, if we allow a suitable modification of the domain's action schemas. First, one can observe that, for many domains, injectivity follows automatically from type restrictions and explicit constraints (i.e. given two variables x_1 and x_2 of an action α, the preconditions of α contain the requirement $x_1 \neq x_2$). If there are variables that can be bound to the same value, we can proceed as follows. Given a subset of variables $W \subseteq V$, we define the action schema α^W by substituting any variable $w \in W$ with an amalgamated symbol $[W]$ in all the formulas of α. For example, if we have an action schema $\mathtt{move}(\mathtt{rbt}, \mathtt{loc}_1, \mathtt{loc}_2)$, we create an additional schema `moveSame(rbt, loc)`. The set of free variables of the new action schema α^W is thus $V_{\alpha^W} = (V_\alpha \setminus W) \cup \{[W]\}$. Note how an injective grounding for α^W corresponds to a grounding gr of α such that $gr(w_1) = gr(w_2)$ for every $w_1, w_2 \in W$ while injectivity is otherwise maintained, namely, $gr(v_1) \neq gr(v_2)$ if $\{v_1, v_2\} \nsubseteq W$. Adding all action schemas α^W, as W varies among all possible subsets of V_α, is equivalent to consider all possible grounding functions for the original action schema α. This procedure potentially leads to an exponential increase (in the cardinality $|V_\alpha|$) of the number of action schemas. However, in practical applications, the possibility of non-injectivity is typically limited to few variables, and so the procedure leads to a controlled growth of the number of schemas and, consequently, of the computation complexity.

Coherently with assumption (2) concerning singleton or sets of actions, we assume that any action schema satisfies the condition:

$$Pre_\alpha^+ \cap Pre_\alpha^- = \emptyset \tag{17}$$

Note that conditions (17) together with the injectivity constraint for groundings automatically imply that any grounded action $a = gr(\alpha)$ satisfies the condition (2).

In general, we call *liftable* a property P of ground actions if, given an action schema α, if one instantiation $a^* = gr^*(\alpha)$ satisfies P, then all instantiations $a = gr(\alpha)$ satisfy P. In this case, we say that the action schema α satisfies property P.

The results presented in this and the next sections achieve two main goals. On the one hand, they show that the properties of safety introduced for instantaneous and durative actions in Sections 4 and 5 are liftable as well the non intertwining properties, even if in a weaker sense, behind the formulation of Theorems 51 and 53. On the other hand, they will give efficient characterisations of such properties at the lifted level, which we use in our algorithmic implementation (see Section 9).

In the remaining part of this section, we analyse action schemas and their ground instantiations. We show that strong safety is liftable and work out a complete characterisation of this property at the lifted level. Next section is devoted to lifting properties for durative actions.

7.1. Structure and properties of action schemas

We start with the following definition that introduces the key concept of matching. It couples an action schema to a template and allows us to understand if, in the ground world, a ground formula appearing in an action schema is or is not in $\gamma(\mathcal{T})$.

Definition 59 (Matching)**.** *Consider a template $\mathcal{T} = (C, \mathcal{F}_C)$ and an action schema $\alpha \in \mathcal{A}$. A formula l that appears in α is said to* match *$\mathcal{T}$ via the template's component $c = \langle r/k, p\rangle \in C$ if:*

(i) $\mathrm{Rel}[l] = \langle r/k\rangle$*; and*

(ii) if l is universally quantified, $\mathrm{VarQ}[l] = \{p\}$.

*Given two formulas l and l' in α, we say that they are $\mathcal{T}$-*coupled *(and we write $l \sim_{\mathcal{T}} l'$) if:*

(a) l and l' individually match $\mathcal{T}$ via the components c and c'; and

(b) if $(c, i) \sim_{\mathcal{F}_C} (c', j)$, $Var[l, i] = Var[l', j]$.

We now fix a template $\mathcal{T}$ and an action schema α and study the properties of the relation $\sim_{\mathcal{T}}$ on the literals of α that match $\mathcal{T}$, introduced above. First, we have the following simple result.

Proposition 60. *For a template $\mathcal{T} = (C, \mathcal{F}_C)$ and an action schema α, $\sim_{\mathcal{T}}$ is an equivalence relation.*

Proof. Reflexivity and symmetry are obvious from the definition. Regarding transitivity, assume that l^1, l^2, l^3 are three formulas in α matching $\mathcal{T}$ through the components c^1, c^2, c^3 respectively, such that $l^1 \sim_{\mathcal{T}} l^2$ and $l^2 \sim_{\mathcal{T}} l^3$. Suppose that $(c^1, i) \in F_{c^1}$ and $(c^3, j) \in F_{c^3}$ are such that $(c^1, i) \sim_{\mathcal{F}_C} (c^3, j)$. The fact that $\mathcal{F}_C$ is an admissible partition implies that there exists $(c^2, h) \in F_{c^2}$ such that $(c^1, i) \sim_{\mathcal{F}_C} (c^2, h) \sim_{\mathcal{F}_C} (c^3, j)$. The assumptions $l^1 \sim_{\mathcal{T}} l^2$ and $l^2 \sim_{\mathcal{T}} l^3$ yield $Var[l^1, i] = Var[l^2, h] = Var[l^3, j]$. This shows that $l^1 \sim_{\mathcal{T}} l^3$. □

Definition 61 ($\mathcal{T}$-Class)**.** *For a template $\mathcal{T} = (C, \mathcal{F}_C)$ and an action schema α, an equivalence class of literals with respect to $\sim_{\mathcal{T}}$ is called a $\mathcal{T}$-*class.

We now derive a more concrete description of the concept of matching.

Remark 62. *Suppose that l is a formula in the action schema α that matches the template $\mathcal{T}$ via the component $c = \langle r/k, p\rangle$. The possible structure of l is shown below:*

$$\begin{array}{ll} p = k, & l = r(v_0, \ldots, v_{k-1}),\ \forall i\ v_i \in V_\alpha \\ p < k, & l = r(v_0, \ldots, v_{k-1}),\ \forall i\ v_i \in V_\alpha \\ p < k, & l = (\forall v_p :\ r(v_0, \ldots, v_{k-1})),\ \forall i \neq p\ v_i \in V_\alpha \end{array} \tag{18}$$

Suppose now that l^1 and l^2 are two formulas in the action schema α that match the template $\mathcal{T}$ via $c^1 = \langle r^1/k^1, p^1\rangle$ and $c^2 = \langle r^2/k^2, p^2\rangle$, respectively. We can represent, for $i = 1, 2$, $l^i = r^i(v^i_0, \ldots v^i_{k^i-1})$ (or $l^i = (\forall v_{p^i} :\ r^i(v^i_0, \ldots v^i_{k^i-1}))$ if $p^i < k^i$). The $\mathcal{T}$ coupling condition $l^1 \sim_{\mathcal{T}} l^2$ is equivalent to require that for any pair of fixed variables $(c^1, j) \in F_{c^1}$ and $(c^2, h) \in F_{c^2}$, it holds

$$(c^1, j) \sim_{\mathcal{F}_C} (c^2, h) \ \Rightarrow\ v^1_j = v^2_h \tag{19}$$

We now consider a grounding function gr for α and an instance γ for $\mathcal{T}$. If l is a formula in α that matches $\mathcal{T}$ via the component $c = \langle r/k, p\rangle$, depending on the structure of l as illustrated in (18), we have that

$$\begin{array}{ll} l = r(v_0, \ldots, v_{k-1}), & gr(l) = \{r(x_0, \ldots, x_{k-1}) \,|\, x_i = gr(v_i)\, \forall i\} \\ l = (\forall v_p :\ r(v_0, \ldots, v_{k-1})), & gr(l) = \{r(x_0, \ldots, x_{k-1}) \,|\, x_p \in \mathcal{O},\ x_i = gr(v_i)\, \forall i \neq p\} \end{array} \tag{20}$$

Considering the definition of $\gamma(\mathcal{T})$ in (3) and the fact that the quantified case above can only appear when $p < k$, it follows that $gr(l)$ is either a subset of $\gamma(\mathcal{T})$ or it must have empty intersection with $\gamma(\mathcal{T})$. This motivates the following definition:

Definition 63 (Coherence)**.** *gr and γ are coherent over l if $gr(l) \subseteq \gamma(\mathcal{T})$.*

Referring to the representation (20), coherence can be equivalently expressed as

$$gr(v_j) = \gamma(c, j),\ \forall j \neq p \tag{21}$$

The following result is immediate from the conditions (21):

Proposition 64. *Consider a template $\mathcal{T} = (C, \mathcal{F}_C)$ and an action schema α. Let l be a formula in the action schema α that matches $\mathcal{T}$. Then, for every grounding function gr of α, it is possible to find an instance γ of $\mathcal{T}$ such that gr and γ are coherent over l and viceversa.*

The following result explains how coherence interacts with the equivalence relation $\sim_{\mathcal{T}}$.

Lemma 65. *Consider a template $\mathcal{T} = (C, \mathcal{F}_C)$ with an instance γ and an action schema α with a grounding gr. Assume that gr and γ are coherent over a formula l^1 in α and let l^2 be another formula in α that matches $\mathcal{T}$. Then, gr and γ are coherent over l^2 if and only if $l^2 \sim_{\mathcal{T}} l^1$.*

Proof. We assume that for $i = 1, 2$, l^i matches $\mathcal{T}$ through components $c^i = \langle r^i/k^i, p^i \rangle$. We now represent each l^i as in Remark 62:

$$l^i = r^i(v^i_0, \dots v^i_{k^i-1}) \text{ or } l^i = (\forall v_{p^i} : \; r^i(v^i_0, \dots v^i_{k^i-1}))$$

The fact that gr and γ are coherent over l^1 yields

$$gr(v^1_j) = \gamma(c^1, j), \; \forall j \neq p^1 \tag{22}$$

Suppose now that $l_2 \sim_{\mathcal{T}} l_1$ and consider any fixed variable (c^2, h) of c^2 for some $h \in \{0, \dots, k^2 - 1\} \setminus \{p^2\}$. From the fact that $\mathcal{F}_C$ is an admissible partition, it follows that we can find a fixed variable (c^1, j) of c^1 such that $(c^1, j) \sim_{\mathcal{F}_C} (c^2, h)$. The $\mathcal{T}$ coupling condition (19) implies that $v^1_j = v^2_h$. This yields, using (22) and the fact that γ is an instance

$$gr(v^2_h) = gr(v^1_j) = \gamma(c^1, j) = \gamma(c^2, h)$$

We have thus proven that

$$gr(v^2_h) = \gamma(c^2, h), \; \forall h \neq p^2 \tag{23}$$

On the other hand, assume now that gr and γ are coherent over l^2 and pick any fixed variables (c^1, j) and (c^2, h) of c^1 and c^2 respectively such that $(c^1, j) \sim_{\mathcal{F}_C} (c^2, h)$. Using the definition of an instance and equations (22) and (23), we obtain that $gr(v^2_h) = gr(v^1_j)$. The standing assumption that gr is injective yields $v^1_j = v^2_h$. This shows that the $\mathcal{T}$ coupling condition (19) expressed in Remark 62 holds true. Therefore, $l_2 \sim_{\mathcal{T}} l_1$. □

The following result immediately follows from the definition of coherence and Lemma 65.

Proposition 66. *Suppose that M is a subset of formulas appearing in α. Then, $gr(M) \cap \gamma(\mathcal{T}) = gr(M \cap L)$ where L is the $\mathcal{T}$-class of formulas of α on which gr and γ are coherent.*

Proposition 66 has an important practical consequence. Once gr and γ have been fixed, only the part of α made of formulas in the class L where gr and γ are coherent affect the part of state dynamics concerning the set $\gamma(\mathcal{T})$. Precisely, if $a = gr(\alpha)$, it follows from the definition of a_γ (see Remark 15) that:

$$Pre^{\pm}_{a_\gamma} = gr(Pre^{\pm}_{\alpha} \cap L), \quad Eff^{\pm}_{a_\gamma} = gr(Eff^{\pm}_{\alpha} \cap L)$$

Considering that, by Proposition 16, a is strongly γ-safe if and only if a_γ is also strongly γ-safe, the property of strong safety of an action schema α does not depend on the formulas in α that do not match $\mathcal{T}$. Hence, in principle, such a property should be analysed by studying the restrictions of α to the different $\mathcal{T}$-classes L of matching formulas. This intuition leads to the following definition.

Definition 67 (Pure Action Schemas)**.** *Considering a template $\mathcal{T}$, an action schema α and a $\mathcal{T}$-class L of formulas in α, we define α_L to be the action schema where we only consider formulas belonging to L. More precisely, α_L is the action schema such that*

$$Pre^{\pm}_{\alpha_L} = Pre^{\pm}_{\alpha} \cap L, \quad Eff^{\pm}_{\alpha_L} = Eff^{\pm}_{\alpha} \cap L$$

We call α_L a pure action schema*.*

Example 14 (*Floortile* domain)**.** *Consider the template $\mathcal{T}_{ft}$ in Example 2 and the action schema $\alpha =$*`paintUp`st*: $Pre^{+}_{\alpha} = \{$`robotAt(r, x)`, `clear(y)`$\}$, $Eff^{-}_{\alpha} = \{$`clear(y)`$\}$.*

Note that both formulas `robotAt(r, x)` and `clear(y)` in α match $\mathcal{T}_{ft}$ and form two different $\mathcal{T}$-classes because they do not satisfy condition (ii) in Definition 59: $L_1 = \{$`robotAt(r, x)`$\}$ and $L_2 = \{$`clear(y)`$\}$.

Consider the instance γ_1 that associates `tile1` to each fixed variable in the components of $\mathcal{T}_{ft}$ and grounding function $gr(\mathtt{r})=$`rbt1`, $gr(\mathtt{x}) =$ `tile1` and $gr(\mathtt{y}) =$ `tile2`. In this case, gr and γ_1 are coherent on the $\mathcal{T}$-class L_1.

We have two pure action schemas corresponding to α: α_{L_1} and α_{L_2}. α_{L_1} has the following specification: $Pre^{+}_{\alpha_{L_1}} = \{\mathtt{robotAt(r, x)}\}$ and α_{L_2}: $Pre^{+}_{\alpha_{L_2}} = \{\mathtt{clear(y)}\}, Eff^{-}_{\alpha_{L_2}} = \{\mathtt{clear(y)}\}$.

7.2. Pure Action Schema Classification

We now carry on a detailed analysis of pure action schemas, showing in particular how the check for strong safety for an action $a = gr(\alpha)$ can be efficiently performed at the lifted level working with the different pure action schemas α_L.

We fix an action schema α and a $\mathcal{T}$-class L of its formulas. First, we introduce a concept of weight at the level of formulas in L that allows us to distinguish between simple and universally quantified formulas. Precisely, given $l \in L$, we put $w_l = 1$ if l is simple, while $w_l = \omega$ if l is universally quantified and where $\omega = |O|$. For a subset $A \subseteq L$, we define $w(A) = \sum_{l \in A} w_l$. Note that $w(\cdot)$ simply coincides with the notion of cardinality in case all formulas in L are simple. If we consider a grounding function gr for α, then for every subset $A \subseteq L$, it holds:

$$|gr(A)| = w(A) \tag{24}$$

Similarly, if c is a component of $\mathcal{T}$, we define w_c equal to 1 or to ω if c, respectively, does not have or does have a counted variable.

We need a last concept:

Definition 68 (Coverage)**.** *Take a component $c \in \mathcal{T}$. We let L_c to be the subset of formulas in L that match $\mathcal{T}$ through the component c. A subset of formulas $M \subseteq L$ is said to* cover *the component c, if $w(M \cap L_c) = w_c$. M is said to* cover *$\mathcal{T}$, if M covers every component $c \in \mathcal{T}$.*

Remark 69. *If we consider a component $c \in \mathcal{T}$, we have that all ground atoms generated by c are in $gr(M)$ if and only if M covers c. In particular, $\gamma(\mathcal{T}) = gr(M)$ if and only if M covers $\mathcal{T}$.*

We now give a classification of the pure action schemas α_L, formally analogous to the one introduced for action sets in Definitions 17 and 19: we simply replace preconditions and effects of a_γ with those of α_L and the concept of cardinality with that of weight.

Definition 70 (Classification of Pure Action Schemas)**.** *The pure action schema α_L is:*

- unreachable *for $\mathcal{T}$ if $w(Pre^{+}_{\alpha_L}) \geq 2$;*

- heavy *for $\mathcal{T}$ if $w(Pre^+_{\alpha_L}) \leq 1$ and $w(Eff^+_{\alpha_L}) \geq 2$;*

- irrelevant *for $\mathcal{T}$ if $w(Pre^+_{\alpha_L}) \leq 1$ and $w(Eff^+_{\alpha_L}) = 0$;*

- relevant *for $\mathcal{T}$ if $w(Pre^+_{\alpha_L}) \leq 1$ and $w(Eff^+_{\alpha_L}) = 1$.*

Definition 71 (Classification of Relevant Action Schemas)**.** *The pure relevant action schema α_L is* weighty *when it has a single relevant precondition: $w(Pre^+_{\alpha_L}) = 1$. A is* weightless *if $w(Pre^+_{\alpha_L}) = 0$.*

A weighty action schema α_L is either:

- balanced *for $\mathcal{T}$ if $Pre^+_{\alpha_L} \subseteq Eff^+_{\alpha_L} \cup Eff^-_{\alpha_L}$;*

- unbalanced *for $\mathcal{T}$ if $Pre^+_{\alpha_L} \cap (Eff^+_{\alpha_L} \cup Eff^-_{\alpha_L}) = \emptyset$;*

A weightless action schema α_L is either:

- bounded *for $\mathcal{T}$ if L covers $\mathcal{T}$;*

- unbounded *for $\mathcal{T}$ if L does not cover $\mathcal{T}$.*

The following result clarifies the relation with the corresponding grounded actions.

Proposition 72. *Consider an action schema α, a $\mathcal{T}$-class L of its formulas, a grounding function gr and an instance γ coherent over L. Put $a = gr(\alpha)$. Then, α_L satisfies any of the properties expressed in Definitions 70 and 71 if and only if a satisfies the corresponding γ-property as defined in Definitions 17 and 19.*

Proof. Immediate consequence of the fact that $a_\gamma = gr(\alpha_L)$, of Equation (24), and of Remark 69. □

We are now ready to give the following final result concerning strong safety of general action schemas. It shows how strong safety can be seen as a property of an action schema and can be studied by analysing its pure parts.

Corollary 73. *Strong safety is a liftable property. Moreover, an action schema α is strongly safe if and only if, for every $\mathcal{T}$-class of formulas L of α, α_L is unreachable, irrelevant, balanced or bounded.*

Proof. Suppose that $a = gr(\alpha)$ for some gr and let γ be an instance. Then, $a_\gamma = gr(\alpha_L)$ where L is the $\mathcal{T}$-class on which gr and γ are coherent. Result is now a straightforward consequence of Proposition 72 and Corollary 21. □

Example 15 (*Floortile* domain)**.** *Consider the template $\mathcal{T}_{ft}$ and the action schema $\alpha = \texttt{paintUp}^{st}$ in Example 14. The two pure action schemas α_{L_1} and α_{L_2} are both irrelevant and hence strongly safe. Hence, α is strongly safe.*

Now consider the action schema $\alpha' = \texttt{paintUp}^{end}$ with specification: $Eff^+_{\alpha'} = \{\texttt{painted(y, c)}\}$. This is a pure action schema. It is unbounded and thus not strongly safe.

An immediate consequence of Corollary 23 is:

Corollary 74. *Given a template $\mathcal{T}$, $\mathcal{T}$ is invariant if for each $\alpha \in \mathcal{A}$, α is strongly safe.*

8. Durative action schemas

Our goal now is to work out proper lifted versions of the properties of durative actions presented in Section 5, in particular those involved in the statement of our main results, Theorems 51 and 53. Some of these properties concern just one durative action (e.g. safety), while others involve more actions (e.g. non-interfering, irrelevant-unreachable). We start analysing the first type of properties, presenting, in particular, an explicit characterisation of safety for durative actions at the lifted level.

We always use the following notation. Take a durative action schema $D\alpha = (\alpha^{\mathrm{st}}, \alpha^{\mathrm{inv}}, \alpha^{\mathrm{end}})$ and a grounding function gr for $D\alpha$. We put $Da = gr(D\alpha)$, where $Da = (a^{\mathrm{st}}, a^{\mathrm{inv}}, a^{\mathrm{end}})$ with $a^{\mathrm{st}} = gr(\alpha^{\mathrm{st}})$, $a^{\mathrm{inv}} = gr(\alpha^{\mathrm{inv}})$, and $a^{\mathrm{end}} = gr(\alpha^{\mathrm{end}})$.

Our first goal is to lift the assumptions (8) and (9) on durative actions. First define, for a generic action schema α, the subsets of *postconditions*:

$$\Gamma^+_\alpha = (Pre^+_\alpha \setminus Eff^-_\alpha) \cup Eff^+_\alpha, \quad \Gamma^-_\alpha = (Pre^-_\alpha \setminus Eff^+_\alpha) \cup Eff^-_\alpha$$

We will make the standing assumption that every durative action schema $D\alpha = (\alpha^{\mathrm{st}}, \alpha^{\mathrm{inv}}, \alpha^{\mathrm{end}})$ satisfies the relations

$$\Gamma^+_{\alpha^{\mathrm{st}}} \cap Pre^-_{\alpha^{\mathrm{inv}}} = \emptyset, \quad \Gamma^-_{\alpha^{\mathrm{st}}} \cap Pre^+_{\alpha^{\mathrm{inv}}} = \emptyset \tag{25}$$

$$Pre^+_{\alpha^{\mathrm{end}}} \cap Pre^-_{\alpha^{\mathrm{inv}}} = \emptyset, \quad Pre^+_{\alpha^{\mathrm{end}}} \cap Pre^-_{\alpha^{+\mathrm{inv}}} = \emptyset \tag{26}$$

Since, we recall, grounding functions are supposed to be injective, (25) and (26) are equivalent to requiring that any grounding function $Da = gr(D\alpha)$ satisfies conditions (8) and (9).

Also, we define the auxiliary durative action schema $D\alpha_* = (\alpha^{\mathrm{st}}_*, \alpha^{\mathrm{end}}_*)$ where α^{st}_* and α^{end}_* are the action schema such that:

$$\begin{array}{ll} Eff^\pm_{\alpha^{\mathrm{st}}_*} = Eff^\pm_{\alpha^{\mathrm{st}}}, & Pre^\pm_{\alpha^{\mathrm{st}}_*} = Pre^\pm_{\alpha^{\mathrm{st}}} \cup (Pre^\pm_{\alpha^{\mathrm{inv}}} \setminus Eff^\pm_{\alpha^{\mathrm{st}}}) \\ Eff^\pm_{\alpha^{\mathrm{end}}_*} = Eff^\pm_{\alpha^{\mathrm{end}}}, & Pre^\pm_{\alpha^{\mathrm{end}}_*} = Pre^\pm_{\alpha^{\mathrm{end}}} \cup Pre^\pm_{\alpha^{\mathrm{inv}}} \end{array}$$

$Da_* = gr(D\alpha_*)$ is the corresponding auxiliary action already defined in Section 5.2.

8.1. Safety of durative action schemas

We now fix a template $\mathcal{T}$ and start to analyse safety. We consider a durative action schema $D\alpha$, its auxiliary action schema $D\alpha_*$ and its groundings $Da = gr(D\alpha)$ and $Da_* = gr(D\alpha_*)$. Strong safety for durative actions reduces to strong safety of its components and it is thus a liftable property. In consequence, we can talk about the strong safety of $D\alpha$ or $D\alpha_*$: this is equivalent to the strong safety of all its groundings, $Da = gr(D\alpha)$ or, respectively, $Da_* = gr(D\alpha_*)$. Checking such property at the lifted level can be done by applying Corollary 73 to the start and end fragments.

We now want to characterise simple safety of the auxiliary durative action $Da_* = gr(D\alpha_*)$ at the lifted level. First we consider executability.

Proposition 75. *Executability of auxiliary durative actions is a lifted property. Precisely, $D\alpha_*$ is executable if and only if*

$$\Gamma^+_{\alpha^{st}_*} \cap Pre^-_{\alpha^{end}_*} = \emptyset = \Gamma^-_{\alpha^{st}_*} \cap Pre^+_{\alpha^{end}_*} \tag{27}$$

Proof. Immediate consequence of Proposition 30. □

Assume now $D\alpha_*$ to be executable. Fix a grounding gr and put $Da_* = gr(D\alpha_*)$. Consider an instance γ and let L be the the $\mathcal{T}$-class of formulas in $D\alpha$ on which gr and γ are coherent. Put $D\alpha_L = (\alpha_L^{\mathrm{st}}, \alpha_L^{\mathrm{inv}}, \alpha_L^{\mathrm{end}})$ and $D\alpha_{*L} = (\alpha_{*L}^{\mathrm{st}}, \alpha_{*L}^{\mathrm{end}})$. Note that $Da_{*\gamma} = gr(D\alpha_{*L})$. Therefore, since simple γ-safety of Da_* only depends on $Da_{*\gamma}$ (since executability has already been assumed), we expect that such property can be formulated in terms of the pure auxiliary durative action schema $D\alpha_{*L}$. To this aim, we now give, for such durative action schemas, the same classification introduced for durative actions in Definition 48. First, we need a further concept:

Definition 76 (Reachable action schemas)**.** *$D\alpha_{*L}$ is said to be* reachable *if it is executable and*

$$w(Pre^+_{\alpha^{st}_{*L}} \cup (Pre^+_{\alpha^{end}_{*L}} \setminus Eff^+_{\alpha^{st}_{*L}})) \leq 1$$

Proposition 77. *If gr and γ are coherent over L and $Da_* = gr(D\alpha_*)$, we have that $Da_{*\gamma}$ is γ-reachable if and only if $D\alpha_{*L}$ is reachable.*

Proof. Immediate consequence of Propositions 31 and 66 and of equation (24). □

Definition 78 (Safe durative action schemas)**.** *When $D\alpha_{*L}$ is such that*

(i) *$D\alpha_{*L}$ is reachable;*

(ii) *α^{st}_{*L} is strongly safe;*

(iii) *α^{end}_{*L} is unbounded;*

(iv) *$D\alpha_{*L}$ satisfies any of the conditions below:*

  (a) *α^{st}_{*L} irrelevant, $w(Pre^+_{\alpha^{st}_{*L}}) = 1$, $Pre^+_{\alpha^{st}_{*L}} \subseteq Eff^-_{\alpha^{st}_{*L}}$;*

  (b) *α^{st}_{*L} irrelevant, $w(Pre^+_{\alpha^{st}_{*L}}) = 1$, $Pre^+_{\alpha^{st}_{*L}} \nsubseteq Eff^-_{\alpha^{st}_{*L}}$, $Pre^+_{\alpha^{st}_{*L}} \subseteq Eff_{\alpha^{end}_L}$;*

  (c) *α^{st}_{*L} irrelevant, $w(Pre^+_{\alpha^{st}_{*L}}) = 0$, $Pre^-_{\alpha^{st}_{*L}} \cup Eff^-_{\alpha^{st}_{*L}} \cup Eff_{\alpha^{end}_L}$ covers $\mathcal{T}$;*

  (d) *α^{st}_{*L} relevant, $Eff^+_{\alpha^{st}_L} \subseteq Eff_{\alpha^{end}_L}$.*

we say that $D\alpha_{*L}$ is weakly safe of type (x) *where $x \in \{a, b, c, d\}$.*

Corollary 79. *Safety for durative auxiliary actions is a liftable property. $Da_* = gr(D\alpha_*)$ is safe if and only if:*

- *$D\alpha_*$ is executable;*
- *For every $\mathcal{T}$-class L of formulas in $D\alpha$, one of the following conditions hold:*
  - *$D\alpha_{*L}$ is strongly safe;*
  - *α^{st}_{*L} is strongly safe and $D\alpha_{*L}$ is unreachable;*
  - *$D\alpha_{*L}$ is weakly safe of type (x) where $x \in \{a, b, c, d\}$.*

Proof. Immediate consequence of previous definitions and Proposition 46. □

α	$\texttt{paintUp}^{\text{st}}$	$\texttt{paintUp}^{\text{inv}}$	$\texttt{paintUp}^{\text{end}}$
Pre^+_α	$\{\texttt{robotAt(r, x)}$ $\texttt{clear(y)}\}$	$\{\texttt{robot} - \texttt{has(r, c)}$ $\texttt{up(y, x)}\}$	$\emptyset$
Eff^+_α	$\emptyset$	$\emptyset$	$\{\texttt{painted(y, c)}\}$
Eff^-_α	$\{\texttt{clear(y)}\}$	$\emptyset$	$\emptyset$

Table 4: Durative action schema `paintUp` (abbreviated specification). See Example 16.

Example 16 (*Floortile* domain)**.** *Consider our usual template:*

$$\mathcal{T}_{ft} = (\{\langle \texttt{robotAt}/2, 0\rangle, \langle \texttt{painted}/2, 1\rangle, \langle \texttt{clear}/1, 1\rangle\}$$

and the action schema:

$$D\alpha = \texttt{paintUp} : (\texttt{paintUp}^{st}, \texttt{paintUp}^{inv}, \texttt{paintUp}^{end})$$

where the single instantaneous action schemas have the specifications as in Table 4.

In this action schema, we have three formulas that match $\mathcal{T}_{ft}$: `robotAt(r, x)`, `clear(y)` and `painted(y, c)`. They form two $\mathcal{T}$-classes: $L_1 = \{$`robotAt(r, x)`$\}$ and $L_2 = \{$`clear(y)`, `painted(y, c)`$\}$. Note that in this case $\texttt{paintUp}^{st}_{L_i}$ is equal to $\texttt{paintUp}^{st}_{*L_i}$ for $i = 1, 2$ and the same holds for $\texttt{paintUp}^{end}_{L_i}$.

The pure action schemas $\texttt{paintUp}^{st}_{L_1}$, $\texttt{paintUp}^{st}_{L_2}$ and $\texttt{paintUp}^{end}_{L_1}$ are strongly safe because they are irrelevant. The pure schema $\texttt{paintUp}^{end}_{L_2}$ is unbounded.

The pure durative action schema $\texttt{paintUp}_{L_1}$ is strongly safe because $\texttt{paintUp}^{st}_{L_1}$ and $\texttt{paintUp}^{end}_{L_1}$ are strongly safe since they are irrelevant.

The pure schema $\texttt{paintUp}_{L_2}$ is weakly safe of type (a) since:

- $\texttt{paintUp}_{L_2}$ *is reachable because* $\texttt{paintUp}^{st}_{L_2}$ *is reachable and* $\texttt{paintUp}^{end}_{L_2}$ *does not contain preconditions;*
- $\texttt{paintUp}^{st}_{L_2}$ *is strongly safe since it is irrelevant;*
- $\texttt{paintUp}^{end}_{L_2}$ *is unbounded;*
- $w(Pre^+_{\texttt{paintUp}^{st}_{L_2}}) = 1$ *because the preconditions at start consist in* `clear(y)`*;*
- $Pre^+_{\texttt{paintUp}^{st}_{L_2}} \subseteq Eff^-_{\texttt{paintUp}^{st}_{L_2}}$ *because the delete effects at start also contain* `clear(y)`*.*

8.2. Lifting properties of multiple actions

In this section, we study how properties that involve more than one action (e.g. mutex) can be lifted. This requires working simultaneously with different groundings and, for this reason, additional concepts are needed.

Consider two action schemas α^1 and α^2 (instantaneous or durative) with set of the variables V_{α^1} and V_{α^2}, respectively. Whenever we consider two groundings gr^1 and gr^2 for α^1 and α^2, respectively, the pairwise properties of the two actions $a^i = gr^i(\alpha^i)$ (e.g. properties regarding the sequence (a^1, a^2) or the set $\{a^1, a^2\}$) are non liftable, as in general they may depend on the specific groundings chosen. A key aspect is the possible presence, in the two action schemas, of pairs of variables $v^i \in V^{if}$ such that $gr^1(v^1) = gr^2(v^2)$: this may cause the same ground atom to appear in the two actions a^1 and a^2, which in principle can affect the validity of certain properties, such as non-interference. To cope with this complexity at the lifted level, we introduce a concept of *reduced union* of the two sets V_{α^1} and V_{α^2} to be used as a common set of variables for the two schemas.

We define a *matching* between α^1 and α^2 as any subset $\mathcal{M} \subseteq V_{\alpha^1} \times V_{\alpha^2}$ such that:

- If (v^1, v^2), $(w^1, v^2) \in \mathcal{M}$, then $v^1 = w^1$;
- If (v^1, v^2), $(v^1, w^2) \in \mathcal{M}$, then $v^2 = w^2$.

We now define the set $V_{\alpha^1} \sqcup_{\mathcal{M}} V_{\alpha^2}$ obtained by $V_{\alpha^1} \cup V_{\alpha^2}$ by reducing each pair of variables $v^1 \in V_{\alpha^1}$ and $v^2 \in V_{\alpha^2}$ such that $(v^1, v^2) \in \mathcal{M}$ to a new variable, denoted as v^1v^2. Note that in the case when $\mathcal{M} = \emptyset$, no reduction takes place and $V_{\alpha^1} \sqcup_{\emptyset} V_{\alpha^2} = V_{\alpha^1} \cup V_{\alpha^2}$.

For a matching $\mathcal{M}$, we have natural maps $\pi^i_{\mathcal{M}} : V^{if} \to V_{\alpha^1} \sqcup_{\mathcal{M}} V_{\alpha^2}$ associating to each variable v^i, v^i itself or the new reduced variable v^iv^j in case $(v^i, v^j) \in \mathcal{M}$. The two schemas α^1 and α^2 can thus be rewritten in this new alphabet by formally substituting each variable $v^i \in V^{if}$ in their formulas with $\pi^i_{\mathcal{M}}(v^i)$. If l^i is a formula of α^i, we denote by $\pi^i_{\mathcal{M}}(l^i)$ the formula obtained with this substitution. Similarly, if A^i is a set of formulas of α^i, we put $\pi^i_{\mathcal{M}}(A^i) = \{\pi^i_{\mathcal{M}}(l^i) \,|\, l^i \in A^i\}$.

On the formulas of the two schemas, expressed in the common variable set $V_{\alpha^1} \sqcup_{\mathcal{M}} V_{\alpha^2}$, we can jointly apply set theoretic operators. If l^i is a formula of α^i and A^i is a set of formulas of α^i, for $i = 1, 2$, we will use the notation $l^1 =_{\mathcal{M}} l^2$ for $\pi^1_{\mathcal{M}}(l^1) = \pi^2_{\mathcal{M}}(l^2)$ and $l^1 \in_{\mathcal{M}} A^2$ for $\pi^1_{\mathcal{M}}(l^1) \in \pi^2_{\mathcal{M}}(A^2)$. Similarly, we put $A^1 *_{\mathcal{M}} A^2 = \pi^1_{\mathcal{M}}(A^1) * \pi^2_{\mathcal{M}}(A^2)$ where $* \in \{\cup, \cap, \setminus\}$.

We now investigate the relation between matchings and specific groundings of the two schemas.

Definition 80 (Coherent grounding functions)**.** *Consider two action schemas α^1 and α^2 and a matching $\mathcal{M}$ between them. Two grounding functions gr^1 and gr^2 for α^1 and α^2, respectively, are said to be* $\mathcal{M}$-adapted *if given $v^i \in V^{if}$ for $i = 1, 2$, it holds $gr^1(v^1) = gr^2(v^2)$ if and only if $(v^1, v^2) \in \mathcal{M}$.*

Remark 81. *Note that, for two groundings gr^1 and gr^2, if we consider $\mathcal{M} = \{(v^1, v^2) \,|\, gr^1(v^1) = gr^2(v^2)\}$ we have that $\mathcal{M}$ is a matching (recall that maps gr^i are injective) and gr^1 and gr^2 are $\mathcal{M}$-adapted.*

Coherent groundings can be factorised through the reduced set $V_{\alpha^1} \sqcup_{\mathcal{M}} V_{\alpha^2}$:

Proposition 82. *Consider two action schemas α^1 and α^2, a matching $\mathcal{M}$ between them, and grounding functions gr^i for α^i, $i = 1, 2$. The following conditions are equivalent:*

(i) gr^1 and gr^2 are $\mathcal{M}$-adapted;

(ii) there exists an injective function $gr : V_{\alpha^1} \sqcup_{\mathcal{M}} V_{\alpha^2} \to O$ such that, $gr^i = gr \circ \pi^i_{\mathcal{M}}$ for $i = 1, 2$.

Suppose that gr^1 and gr^2 are two $\mathcal{M}$-adapted groundings of α^1 and α^2 and let gr be the function as in (ii) of the previous proposition. If A^i is a set of formulas of α^i, for $i = 1, 2$, for any set theoretic operation $* \in \{\cup, \cap, \setminus\}$ it holds that:

$$gr^1(A^1) * gr^2(A^2) = gr(\pi^1_{\mathcal{M}}(A^1)) * gr(\pi^2_{\mathcal{M}}(A^2)) = gr(A^1 *_{\mathcal{M}} A^2) \tag{28}$$

This follows from Proposition 82 and the fact that gr is injective. An iterative use of (28) shows that any set theoretic expression on the two grounded actions $gr^i(\alpha^i)$ is in bijection (through gr)

with a corresponding expression on the two action schemas α^i expressed in the common reduced set $V_{\alpha^1} \sqcup_{\mathcal{M}} V_{\alpha^2}$. As a consequence, any property of actions (with the standing assumption of $\mathcal{M}$-adapted groundings) that can be expressed by set theoretic operations on their formulas can be reformulated by rewriting these formulas in the new alphabet $V_{\alpha^1} \sqcup_{\mathcal{M}} V_{\alpha^2}$. This is the key observation in order to lift properties of pairs of actions. To be more concrete, we consider the example of non-interfering actions, which will be needed in what follows.

Definition 83 (Mutex simple action schemas)**.** *We say that two action schemas α^1 and α^2 are $\mathcal{M}$* non-interfering *if for $i \neq j$*

$$Eff^+(\alpha^i) \cap_{\mathcal{M}} Eff^-(\alpha^j) = \emptyset$$

$$Pre(\alpha^i) \cap_{\mathcal{M}} Eff(\alpha^j) = \emptyset$$

If α^1 and α^2 are not $\mathcal{M}$ non-interfering, they are called $\mathcal{M}$-mutex.

Definition 84 (Executable action schemas)**.** *We say that a set of two $\mathcal{M}$* non-interfering *action schemas $\{\alpha^1, \alpha^2\}$ is $\mathcal{M}$*-executable *if for $i \neq j$*

$$Pre^+(\alpha^i) \cap_{\mathcal{M}} Pre^-(\alpha^j) = \emptyset\,.$$

Proposition 85. *Consider two action schemas α^1 and α^2, a matching $\mathcal{M}$ between them and grounding functions gr^1 and gr^2 for, respectively, α^1 and α^2, that are $\mathcal{M}$-adapted. Put $a^i = gr^i(\alpha^i)$ for $i = 1, 2$. Then,*

(i) α^1 and α^2 are $\mathcal{M}$-mutex if and only if a^1 and a^2 are mutex;

(ii) $\{\alpha^1, \alpha^2\}$ is $\mathcal{M}$-executable if and only if $\{a^1, a^2\}$ is executable.

Proof. Immediate consequence of Definitions 88 and 84 and of equation (28). □

Remark 86. *Note that certain properties that depend on the matching $\mathcal{M}$ have a monotonic behaviour, i.e. if they are true for a matching $\mathcal{M}$, they remain true for a larger matching $\mathcal{M}' \supseteq \mathcal{M}$. This is the case, for instance, of properties that can be expressed in terms of identities between formulas of type $l^1 =_{\mathcal{M}} l^2$, such as the $\mathcal{M}$-mutex property.*

To cope with properties related to a template and its instantiations, it is useful to introduce a family of matchings induced by the presence of formulas in the two schemas matching in a template. Precisely, consider now a template $\mathcal{T} = (C, \mathcal{F}_C)$ and two action schemas α^1 and α^2. Consider $\mathcal{T}$-classes L^i of formulas of α^i for $i = 1, 2$. There is a natural way to associate a matching to L^1 and L^2 as follows. Pick formulas $l^i \in L^i$ for $i = 1, 2$ and consider components $c^i \in C$ such that l^i matches $\mathcal{T}$ through c^i. Put:

$$\mathcal{M}_{L^1,L^2} = \{(Var[l^1, j], Var[l^2, h]) \,|\, (c^1, j) \sim_{\mathcal{F}_C} (c^2, h)\} \tag{29}$$

It immediately follows from the definition of $\mathcal{T}$-coupled pairs of formulas (Definition 59) that $\mathcal{M}_{L^1,L^2}$ does not depend on the particular formulas l^i chosen, but only on the $\mathcal{T}$-classes L^i.

Essentially, in $\mathcal{M}_{L^1,L^2}$, we are rewriting variables in the formulas of L^1 and L^2 that correspond to $\mathcal{F}_C$-equivalent variables in the template $\mathcal{T}$. The next proposition shows the role played by such a matching.

Proposition 87. *Consider two groundings gr^1 and gr^2 for α^1 and α^2, respectively, which are $\mathcal{M}$-adapted. Then the following facts hold:*

(i) for an instance γ for $\mathcal{T}$, if L^i are the $\mathcal{T}$-classes of formulas of α^i on which gr^i and γ are coherent. Then, $\mathcal{M}_{L^1,L^2} \subseteq \mathcal{M}$;

(ii) for $\mathcal{T}$-classes of formulas L^i of α^i, if $\mathcal{M}_{L^1,L^2} \subseteq \mathcal{M}$, there exists just one instance γ of $\mathcal{T}$ such that gr^i and γ are coherent on L^i.

Proof. Fix, for $i = 1, 2$, $l^i \in L^i$. Assume that l^i matches $\mathcal{T}$ through components $c^i = \langle r^i/k^i, p^i \rangle$ and represent l^i as in Remark 62:

$$l^i = r^i(v^i_0, \dots v^i_{k^i-1}) \text{ or } l^i = (\forall v_{p^i} : \ r^i(v^i_0, \dots v^i_{k^i-1}))$$

(i): Let $j \neq p^1$ and $h \neq p^2$ be such that $(c^1, j) \sim_{\mathcal{F}_C} (c^2, h)$. The fact that gr^i and γ are coherent yields:

$$gr^1(v^1_j) = \gamma(c^1, j) = \gamma(c^2, h) = gr^2(v^2_h)$$

This implies, by Definition 80, that $(v^1_j, v^2_h) \in \mathcal{M}$. By definition (29), we thus have $\mathcal{M}_{L^1,L^2} \subseteq \mathcal{M}$.

(ii): Choose γ in such a way that gr^1 and γ are coherent: $\gamma(c^1, j) = gr^1(v^1_j)$ for every $j \neq p^1$. Fix now $h \neq p^2$ and choose $j \neq p^1$ such that $(c^1, j) \sim_{\mathcal{F}_C} (c^2, h)$. Then, since $\mathcal{M}_{L^1,L^2} \subseteq \mathcal{M}$,

$$\gamma(c^2, h) = \gamma(c^1, j) = gr^1(v^1_j) = gr^2(v^2_h)$$

This implies that gr^2 too is coherent with γ. □

We are now ready to lift the properties used in Section 6. We start with unreachability for fragments of durative action schemas.

Definition 88 (Unreachable durative action schemas)**.** *Take two durative action schemas $D\alpha^1$, $D\alpha^2$ and the corresponding $\mathcal{T}$-classes of formulas L^1 and L^2.*

1. *We say that $(\{\alpha^{1inv}, \alpha^{2inv}\}, \{\alpha^{1end}, \alpha^{2end}\})$ is (L^1, L^2)-*unreachable *if at least one of the following conditions is satisfied:*

   (i) $Pre^+_{\alpha^{1inv}} \cap_{\mathcal{M}_{L^1,L^2}} Pre^-_{\alpha^{2end}} \neq \emptyset$;

   (ii) $Pre^-_{\alpha^{1inv}} \cap_{\mathcal{M}_{L^1,L^2}} Pre^+_{\alpha^{2end}} \neq \emptyset$;

   (iii) $w(Pre^+_{\alpha^{1end}_{*L^1}} \cup_{\mathcal{M}} Pre^+_{\alpha^{2end}_{*L^2}}) \geq 2$ for every matching $\mathcal{M} \supseteq \mathcal{M}_{L^1,L^2}$.

2. *We say that $(\{\alpha^{1st}, \alpha^{2st}\}, \{\alpha^{1inv}, \alpha^{2inv}\})$ is (L^1, L^2)-*unreachable *if at least one of the following conditions is satisfied:*

   (i) $\Gamma^+_{\alpha^{1st}} \cap_{\mathcal{M}_{L^1,L^2}} Pre^-_{\alpha^{2inv}} \neq \emptyset$;

   (ii) $\Gamma^-_{\alpha^{1st}} \cap_{\mathcal{M}_{L^1,L^2}} Pre^+_{\alpha^{2inv}} \neq \emptyset$;

   (iii) $w(Pre^+_{\alpha^{1st}_{*L^1}} \cup_{\mathcal{M}} Pre^+_{\alpha^{2st}_{*L^2}}) \geq 2$ for every matching $\mathcal{M} \supseteq \mathcal{M}_{L^1,L^2}$.

Note that the check of property (iii) of the above definition in principle involves all possible matchings containing $\mathcal{M}_{L^1,L^2}$. In Section 9.2, we propose an efficient check of this condition that exhibits a computational complexity of polynomial order in the number of variables and formulas of the domain.

Proposition 89. *Suppose that $D\alpha^1$, $D\alpha^2$ are two durative action schemas and gr^1, gr^2 two corresponding grounding functions. Put $Da^i = gr(D\alpha^i)$ and consider an instance γ. Let L^i be the $\mathcal{T}$-class of formulas of $D\alpha^i$ on which gr^i and γ are coherent.*

- *If $(\{\alpha^{1inv}, \alpha^{2inv}\}, \{\alpha^{1end}, \alpha^{2end}\})$ is (L^1, L^2)-unreachable, then $(\{a^{1st}, a^{2st}\}, \{a^{1inv}, a^{2inv}\})$ is γ-unreachable.*
- *If $(\{\alpha^{1st}, \alpha^{2st}\}, \{\alpha^{1inv}, \alpha^{2inv}\})$ is (L^1, L^2)-unreachable, then $(\{a^{1st}, a^{2st}\}, \{a^{1inv}, a^{2inv}\})$ is γ -unreachable.*

Proof. We only prove the first point, the second being analogous. Let $\mathcal{M}$ be the matching such that gr^1 and gr^2 are $\mathcal{M}$-adapted. It follows from Proposition 87 that $\mathcal{M} \supseteq \mathcal{M}_{L^1,L^2}$. Note that the conditions (i), (ii) expressed in Definition 88, if true for $\mathcal{M}_{L^1,L^2}$, are also true for such matching $\mathcal{M}$, because of Remark 86. Consequently we know that at least one of the conditions (i), (ii), (iii) expressed in Definition 88 holds true for such $\mathcal{M}$. It then follows from (28) that at least one of the following conditions hold

(ib) $Pre^+_{a^{1\mathrm{inv}}} \cap Pre^-_{a^{2\mathrm{end}}} \neq \emptyset$;

(iib) $Pre^-_{a^{1\mathrm{inv}}} \cap Pre^+_{a^{2\mathrm{end}}} \neq \emptyset$;

(iiib) $|Pre^+_{a^{1\mathrm{inv}}_\gamma} \cup Pre^+_{a^{1\mathrm{end}}_\gamma} \cup Pre^+_{a^{2\mathrm{inv}}_\gamma} \cup Pre^+_{a^{2\mathrm{end}}_\gamma}| \geq 2.$

By virtue of Propositions 30 and 31 this implies that $(\{a^{1\mathrm{inv}}, a^{2\mathrm{inv}}\}, \{a^{1\mathrm{end}}, a^{2\mathrm{end}}\})$ is γ-unreachable. □

We now give the lifted version of relevant right isolated.

Definition 90 (Relevant right isolated schemas)**.** *For a template $\mathcal{T}$, the set of durative action schemas $\mathcal{A}^d$ is said to be* relevant right isolated *if, for every $D\alpha^1, D\alpha^2 \in \mathcal{A}^d$, corresponding $\mathcal{T}$-classes L^1, L^2 of formulas of each of them such that $D\alpha^i_{L^i}$ are both not strongly safe, one of the following conditions is satisfied (we use the notation $\mathcal{M} = \mathcal{M}_{L^1,L^2}$):*

(i) $|Eff^+_{\alpha^{1end}_{L^1}} \cup_{\mathcal{M}} Eff^+_{\alpha^{2end}_{L^2}}| \leq 1$*;*

(ii) at least one of the two pairs $\{\alpha^{1end}, \alpha^{2end}\}$, $\{\alpha^{1inv}, \alpha^{2inv}\}$ is either $\mathcal{M}$-mutex or non $\mathcal{M}$-executable;

(iii) $(\{\alpha^{1inv}, \alpha^{2inv}\}, \{\alpha^{1end}, \alpha^{2end}\})$ is (L^1, L^2)-unreachable.

Proposition 91. *For a template $\mathcal{T}$, suppose that the set of durative action schemas $\mathcal{A}^d$ is relevant right isolated. Then $\mathcal{G}\mathcal{A}^d$ is also relevant right isolated.*

Proof. Fix any instance γ and consider $Da^1, Da^2 \in \mathcal{G}\mathcal{A}^d(\mathrm{wk}, \gamma)$. Let $D\alpha^i$ and gr^i, for $i = 1, 2$, durative schemas and groundings such that $Da^i = gr^i(D\alpha^i)$. Let L^i be the $\mathcal{T}$-class of formulas of each schema $D\alpha^i$ such that gr^i and γ are coherent over L^i for $i = 1, 2$. Therefore, $D\alpha^1$ and $D\alpha^2$ must satisfy one of the conditions (i) to (iii) in the Definition 90. Let $\mathcal{M}$ be the matching respect to which gr^1 and gr^2 are adapted (in the sense of Remark 81). We know from Proposition 87 that $\mathcal{M} \supseteq \mathcal{M}_{L^1,L2}$. Note now that if condition (i) holds true, it also holds true for such larger $\mathcal{M}$ (Remark 86) and this yields condition (i) of Definition 50. Similarly, condition (ii) yields the same condition with this new $\mathcal{M}$ (Remark 86) from which condition (ii) in Definition 90 follows using Proposition 85. Finally, if condition (iii) holds true, then condition (iii) in Definition 90 follows by using Proposition 89. Therefore, by Definition 90 we have that the two durative action schemas $D\alpha^1$ and $D\alpha^2$ must satisfy one of the conditions (i) to (iii) in the definition. From the fact that gr^1 and gr^2 are $\mathcal{M}$-adapted, it follows that conditions (i) of Definition 90 yields condition (i) of Definition 50. Condition (ii) and (iii) in Definition 50 finally follow conditions (ii) and (iii) in Definition 90 using Propositions 85 and 89. □

We are now ready to propose the lifted version of our first invariant result Theorem 51 .

Corollary 92. *Consider a template $\mathcal{T}$ and suppose that the set of instantaneous action schemas $\mathcal{A}^i$ and that of durative action schemas $\mathcal{A}^d$ satisfy the following properties:*

(i) every $\alpha \in \mathcal{A}^i$ is strongly safe;

(ii) for every $D\alpha \in \mathcal{A}^d$ and every $\mathcal{T}$-class L such that $D\alpha_L$ is not strongly safe, $D\alpha_{*L}$ is reachable and strongly safe;

(iii) $\mathcal{A}^d$ is relevant right isolated.

Then, $\mathcal{T}$ is invariant.

In order to lift the remaining results on the invariance of a template, a key point is to lift the fundamental Definition 56 of pairwise relevant non overlapping. To this aim it will be convenient to introduce some compact notation concerning sets of action schemas and relative classes. We define

$$\begin{array}{rcl} \mathcal{A}^dC(\mathcal{T}) & = & \{(D\alpha, L) \,|\, D\alpha \in \mathcal{A}^d,\ L\ \ \mathcal{T}\text{-class of } D\alpha\} \\ \mathcal{A}^dC(wk, \mathcal{T}) & = & \{(D\alpha, L) \in \mathcal{A}^dC(\mathcal{T}),\ D\alpha_L \text{ weakly safe}\} \\ \mathcal{A}C(\mathcal{T}) & = & \{(\alpha, L) \,|\, \alpha \in \mathcal{A},\ L\ \ \mathcal{T}\text{-class of } \alpha\} \end{array}$$

(α, L) and $(D\alpha, L)$ are called, respectively, schema-class and durative schema-class pairs.

We now propose a lifted version of the property of unreachability expressed in Definition 54.

Definition 93 (MC-unreachable schemas)**.** *Consider a template $\mathcal{T}$ and a subset $MC \subseteq \mathcal{A}C(\mathcal{T})$ such that for every $(\alpha, L) \in MC$, α_L is irrelevant. Consider a pair of action schemas $\alpha^1, \alpha^2 \in \mathcal{A}$ and relative classes L^1 and L^2, respectively. (α^1, α^2) is* $(MC; L^1, L^2)$-unreachable *if any of the following conditions is satisfied:*

(i) there exist $l^1 \in \Gamma^+_{\alpha^1}$, $l^2 \in Pre^-_{\alpha^2}$ with $l^1 =_{\mathcal{M}_{L^1,L^2}} l^2$ such that, for every schema-class $(\alpha, L) \in MC$ and for every matching $\mathcal{M}$ between α^1 and α for which $\mathcal{M} \supseteq \mathcal{M}_{L^1,L}$, we have that $l^1 \notin_{\mathcal{M}} Eff^-_{\alpha}$;

(ii) there exist $l^1 \in \Gamma^-_{\alpha^1}$, $l^2 \in Pre^+_{\alpha^2}$ with $l^1 =_{\mathcal{M}_{L^1,L^2}} l^2$ such that, for every schema-class $(\alpha, L) \in MC$ and for every matching $\mathcal{M}$ between α^1 and α for which $\mathcal{M} \supseteq \mathcal{M}_{L^1,L}$, we have that $l^1 \notin_{\mathcal{M}} Eff^+_{\alpha}$;

(iii) $w(Pre^+_{\alpha^1_{L^1}} \cup_{\mathcal{M}} (Pre^+_{\alpha^2_{L^2}} \setminus_{\mathcal{M}} Eff^+_{\alpha^1_{L^1}})) \geq 2$ for every matching $\mathcal{M} \supseteq \mathcal{M}_{L^1,L^2}$.

Note that the check of all these properties in principle involve all possible matchings containing $\mathcal{M}_{L^1,L^2}$. In Section 9.2, we propose an efficient check of this condition that exhibits a computational complexity of polynomial order in the number of variables and formulas of the domain.

The following result shows how the notion of unreachability in Definition 93 is the lifted version of the one expressed in Definition 54.

Proposition 94. *Consider a template $\mathcal{T}$ and a subset $MC \subseteq \mathcal{A}C(\mathcal{T})$ such that for every $(\alpha, L) \in MC$, α_L is irrelevant. Consider a pair of action schemas $\alpha^1, \alpha^2 \in \mathcal{A}$ and relative groundings gr^1 and gr^2. Put $a^i = gr(\alpha^i)$ and consider an instance γ. Let L^i be the $\mathcal{T}$-class of formulas of α^i on which gr^i and γ are coherent. If (α^1, α^2) is $(MC; L^1, L^2)$-unreachable, then (a^1, a^2) is M-unreachable where M is the set of actions a so obtained: for every $(\alpha, L) \in MC$ and gr grounding of α such that gr and γ are coherent over L, we put $a = gr(\alpha)$.*

Proof. Let $\mathcal{M}$ be the matching respect to which the two groundings gr^1 and gr^2 are $\mathcal{M}$-adapted. By Proposition 87, we have that $\mathcal{M} \supseteq \mathcal{M}_{L^1,L^2}$. Suppose (i) holds and put $q = gr^1(l^1) = gr^2(l^2) \in \Gamma_a^{1+} \cap Pre_a^{2-}$. Consider now any action $a \in M$ and let α be an action schema such that $a = gr(\alpha)$ for some grounding gr. Let L be the $\mathcal{T}$-class of formulas of α on which gr and γ are coherent. By construction, it follows that $(\alpha, L) \in MC$. Consider now the matching $\tilde{\mathcal{M}}$ between α^1 and α respect to which gr^1 and gr are $\tilde{\mathcal{M}}$-adapted. We have that $\tilde{\mathcal{M}} \supseteq \mathcal{M}_{L^1,L}$. Then, by (i) we have that $l^1 \notin_{\tilde{\mathcal{M}}} Eff_\alpha^+$ that implies that $q \notin Eff_a^-$. This shows that condition (i) of Definition 54 is satisfied. Similarly, one can prove that condition (ii) of Definition 93 yields condition (ii) of Definition 54. Finally the fact that (iii) of Definition 93 yields condition (iii) of Definition 54 follows from a repeated application of relation (28). □

We are now ready to lift Definition 56. Given a set of durative actions and classes $\mathcal{A}^d C(\mathcal{T})^* \subseteq \mathcal{A}^d C(\mathcal{T})$, we put

$$\begin{aligned}
\mathcal{A}^{\mathrm{st}} C(\mathcal{T})^* &= \{(\alpha^{\mathrm{st}}, L) \in \mathcal{A}C(\mathcal{T}) \,|\, (D\alpha, L) \in \mathcal{A}^d C(\mathcal{T})^*\},\\
\mathcal{A}^{\mathrm{end}} C(\mathcal{T})^* &= \{(\alpha^{\mathrm{end}}, L) \in \mathcal{A}C(\mathcal{T}) \,|\, (D\alpha, L) \in \mathcal{A}^d C(\mathcal{T})^*\}\\
\mathcal{A}C(\mathcal{T})^*_{irr} &= \{(\alpha, L) \in \mathcal{A}C(\mathcal{T}) \setminus (\mathcal{A}^{\mathrm{st}} C(\mathcal{T})^* \cup \mathcal{A}^{\mathrm{end}} C(\mathcal{T})^*) \,|\, \alpha_L \text{ irrelevant}\},\\
\mathcal{A}C(\mathcal{T})^*_{rel} &= \{(\alpha, L) \in \mathcal{A}C(\mathcal{T}) \setminus (\mathcal{A}^{\mathrm{st}} C(\mathcal{T})^* \cup \mathcal{A}^{\mathrm{end}} C(\mathcal{T})^*) \,|\, \alpha_L \text{ relevant}\}
\end{aligned}$$

Definition 95 (Pairwise relevant non-overlapping action schemas)**.** *For a template $\mathcal{T}$, the set of durative action schemas $\mathcal{A}^d$ is said to be* pairwise relevant non-overlapping *if we can find a set of durative schema-class pairs $\mathcal{A}^d C(\mathcal{T})^* \subseteq \mathcal{A}^d C(\mathcal{T})$ with $\mathcal{A}^d C(\mathcal{T})^* \supseteq \mathcal{A}^d C(wk, \mathcal{T})$ such that the following properties are satisfied:*

A. *for every $(D\alpha^1, L^1), (D\alpha^2, L^2) \in \mathcal{A}^d C(\mathcal{T})^*$, denoted $\mathcal{M} = \mathcal{M}_{L^1,L^2}$, one of the following conditions holds true:*

   (Ai) at least one of the two pairs $\{\alpha^{1st}, \alpha^{2st}\}$, $\{\alpha^{1inv}, \alpha^{2inv}\}$ is either $\mathcal{M}$-mutex or non $\mathcal{M}$-executable;

   (Aii) $(\{\alpha^{1st}, \alpha^{2st}\}, \{\alpha^{1inv}, \alpha^{2inv}\})$ is (L^1, L^2)-unreachable.

   (Aiii) the pairs $\{\alpha^{1inv}, \alpha^{2end}\}$, $\{\alpha^{1end}, \alpha^{2inv}\}$ are $\mathcal{M}$-mutex and, if $(D\alpha^1, L^1) \neq (D\alpha^2, L^2)$, the pair $\{\alpha^{1end}, \alpha^{2end}\}$ is either $\mathcal{M}$-mutex or non $\mathcal{M}$-executable.

B. *for every $(D\alpha^1, L^1), (D\alpha^2, L^2) \in \mathcal{A}^d C(\mathcal{T})^*$, denoted $\mathcal{M} = \mathcal{M}_{L^1,L^2}$, one of the following conditions holds true:*

   (Bi) $\{\alpha^{1inv}, \alpha^{2st}\}$ is $\mathcal{M}$-mutex;

   (Bii) $\{\alpha^{1st}, \alpha^{2st}\}$ is $(\mathcal{A}C(\mathcal{T})^*_{irr}, L^1, L^2)$-unreachable;

   (Biii) the pairs $\{\alpha^{1inv}, \alpha^{2end}\}$, $\{\alpha^{1end}, \alpha^{2inv}\}$ are $\mathcal{M}$-mutex and the pair $\{\alpha^{1end}, \alpha^{2end}\}$ is either $\mathcal{M}$-mutex or non $\mathcal{M}$-executable.

C. *for every* $(D\alpha^1, L^1) \in \mathcal{A}^d C(\mathcal{T})^*$, $(\alpha^2, L^2) \in \mathcal{A}C(\mathcal{T})^*_{rel}$, *denoted* $\mathcal{M} = \mathcal{M}_{L^1,L^2}$, *one of the following conditions is satisfied:*

   (Ci) $\{\alpha^{1inv}, \alpha^2\}$ *is* $\mathcal{M}$*-mutex;*

   (Cii) $\{\alpha^{1st}, \alpha^2\}$ *is* $(\mathcal{A}C(\mathcal{T})^*_{irr}, L^1, L^2)$*-unreachable;*

Proposition 96. *Consider a template* $\mathcal{T}$*. If the set of durative action schemas* $\mathcal{A}^d$ *is* pairwise relevant non-overlapping*, then the corresponding set of durative actions* $\mathcal{GA}^d$ *is pairwise relevant non-overlapping.*

Proof. Consider the set of of durative action schemas and classes $\mathcal{A}^d C(\mathcal{T})^*$ in Definition 95. Fix any instance γ and define $\mathcal{GA}^d(\gamma)$ as the set of durative actions Da obtained as follows: for every $(D\alpha, L) \in \mathcal{A}^d C(\mathcal{T})^*$ consider the grounding gr of $D\alpha$ such that gr and γ are coherent over L and put $Da = gr(D\alpha)$. We now show that properties A, B, and C of Definition 56 hold with respect to this choice of $\mathcal{GA}^d(\gamma)$.

To prove A and B we fix $Da^1, Da^2 \in \mathcal{GA}^d(\gamma)$. Let $D\alpha^i$ and gr^i, for $i = 1, 2$, durative schemas and groundings such that $Da^i = gr(D\alpha^i)$. Let L^i be the $\mathcal{T}$-class of formulas of each schema $D\alpha^i$ such that gr^i and γ are coherent over L^i for $i = 1, 2$. By the way $\mathcal{GA}^d(\gamma)$ has been defined above, we have that $(D\alpha^i, L^i) \in \mathcal{A}^d C(\mathcal{T})^*$. Let $\mathcal{M}$ be the matching respect to which gr^1 and gr^2 are adapted (in the sense of Remark 81). We know from Proposition 87 that $\mathcal{M} \supseteq \mathcal{M}_{L^1,L2}$.

We know that the two pairs $(D\alpha^i, L^i)$ satisfy one of the conditions (Ai) to (Aiii). Notice now that if (Ai) or (Aiii) is satisfied, it is also satisfied with respect to the larger matching $\mathcal{M}$ (Remark 86). Using Proposition 85 we then conclude that Da^1, Da^2 satisfy the corresponding condition (Ai) or (Aiii) in Definition 56. Finally, If instead $(D\alpha^i, L^i)$ satisfy (Aii), then condition (Aii) in Definition 56 follows for Da^1, Da^2 by using Proposition 89. Condition A for the pair Da^1, Da^2 is thus satisfied.

We now come to condition B. We know that the pair $(D\alpha^i, L^i)$ satisfy one of the conditions (Bi) to (Biii). Arguing like in previous point, we get that if they satisfy either (Bi) or (Biii), then Da^1, Da^2 satisfy the corresponding (Bi) or (Biii) in Definition 56. Instead (Bii) for $(D\alpha^i, L^i)$ yield (Bii) for Da^1, Da^2 as consequence of Proposition 94.

We finally consider condition C. To this aim, we fix $Da^1 \in \mathcal{GA}^d(\gamma)$ and $a^2 \in \mathcal{GA}(\gamma)_{rel}$. Consider action schema $D\alpha^1 \in \mathcal{A}^d$ and $\alpha^2 \in \mathcal{A}$ and corresponding groundings gr^i, for $i = 1, 2$, such that $Da^1 = gr^1(D\alpha^1)$ and $a^2 = gr^2(\alpha^2)$. Let L^i be the $\mathcal{T}$-class of formulas of each schema such that gr^i and γ are coherent over L^i for $i = 1, 2$. By the way $\mathcal{GA}^d(\gamma)$ has been defined above, we have that $(D\alpha^1, L^1) \in \mathcal{A}^d C(\mathcal{T})^*$ while $(\alpha^2, L^2) \in \mathcal{A}C(\mathcal{T})^*_{rel}$. We thus have that $D\alpha^1$ and α^2 must satisfy one of the two properties (Ci) or (Cii). Arguing as in previous points we obtain that these two properties imply the corresponding one (Ci) or (Cii) for Da^1 and a^2.

Proof is thus complete. □

We are now ready to give the lifted versions of our main resultsTheorem 53 and Corollary 58. Proofs are straightforward consequences of our previous definitions and results.

Corollary 97. *Consider a template* $\mathcal{T}$ *and suppose that the set of instantaneous action schemas* $\mathcal{A}^i$ *and that of durative action schemas* $\mathcal{A}^d$ *satisfy the following properties:*

(i) every $\alpha \in \mathcal{A}^i$ *is strongly safe;*

(ii) for every $D\alpha \in \mathcal{A}^d$, $D\alpha^{st}$ *is strongly safe and* $D\alpha_*$ *is safe;*

(iii) $\mathcal{A}^d$ *is pairwise relevant non-overlapping.*

Then, $\mathcal{T}$ is invariant.

Corollary 98. *Consider a template $\mathcal{T}$ and suppose that the set of instantaneous action schemas $\mathcal{A}^i$ and that of durative action schemas $\mathcal{A}^d$ satisfy the following properties:*

(i) for every $(D\alpha, L) \in \mathcal{A}^d C(wk, \mathcal{T})$, then $D\alpha_{*L}$ is weakly safe of type (a);

(ii) for every $(\alpha, L) \in \mathcal{A}C(\mathcal{T}) \setminus (\mathcal{A}^{st}C(wk, \mathcal{T}) \cup \mathcal{A}^{end}C(wk, \mathcal{T}))$, then, α_L is either irrelevant or balanced.

Then, $\mathcal{T}$ is invariant.

We end this section by presenting two examples from the IPCs in which we apply Corollaries 98 and 92 to demonstrate the invariance of the templates under consideration. Corollary 97 is the most general one and can be used in more complex cases.

Example 17 (*Floortile* domain)**.** *Consider our usual template:*

$$\mathcal{T}_{ft} = (\{\langle \texttt{robotAt}/2, 0\rangle, \langle \texttt{painted}/2, 1\rangle, \langle \texttt{clear}/1, 1\rangle\}$$

The actions schemas in the domains are: $\mathcal{A}^d$ = {`changeColor,paintUp,paintDown, up,down,right,left` }*. The schemas* `paintUp` *and* `paintDown` *are symmetrical and differ only on formulas not matching $\mathcal{T}_{ft}$. They have the same $\mathcal{T}$-classes L_1 =* {`robotAt(r, x)`} *and L_2 =* {`clear(y), painted(y, c)`}*. As seen in Example 16, the pure schemas* $\texttt{paintUp}^{st}_{*L_1}$ *and* $\texttt{paintUp}^{end}_{*L_1}$ *are irrelevant and* $\texttt{paintUp}_{*L_2}$ *is weakly safe of type (a). The same holds for* $\texttt{paint-down}_{*L_1}$ *and* $\texttt{paint-down}_{*L_2}$.

The schemas `up, down, right, left` *are also symmetrical and differ only on formulas not in the components of $\mathcal{T}_{ft}$. They have the same $\mathcal{T}$-classes L_3 =* {`robotAt(r, x), clear(x)`} *and L_4 =* {`robotAt(r, y), clear(y)`}*. The schemas* $\texttt{up}_{*L_i}$, $\texttt{down}_{*L_i}$, $\texttt{right}_{*L_i}$ *and* $\texttt{left}_{*L_i}$*, with $i = 3, 4$, are all weakly safe of type (a).*

The schema `changeColor` *has no formula matching the template, hence its start and end fragments are both irrelevant.*

By Corollary 98, the template $\mathcal{T}_{ft}$ is invariant.

Example 18 (*Depot* domain)**.** *Consider the domain* Depot *(see Appendix B) and the template:*

$$\mathcal{T}_{dp} = (\{\langle \texttt{lifting}/2, 1\rangle, \langle \texttt{available}/1, 1\rangle\}$$

Invariants of this template mean that a hoist can be in two possible states: lifting a crate or available. The action schemas in the domains are all durative:

$$\mathcal{A}^d = \{\texttt{drive}, \texttt{lift}, \texttt{drop}, \texttt{load}, \texttt{unload}\}$$

We indicate them as $D\alpha^1, \ldots, D\alpha^5$ respectively and, given $D\alpha^i$, its variables as $x_i, y_i, \ldots$.

To demonstrate that $\mathcal{T}_{dp}$ is invariant, we want to apply Corollary 92. We start with condition (ii) since $\mathcal{A}^i$ is empty.

The action $D\alpha^1$ =`drive` *has no formulas that match the template so it is strongly safe. The other schemas have respectively $\mathcal{T}$-classes L_i = {$\texttt{lifting}(x_i, y_i)$, $\texttt{available}(x_i)$}. There are only two fragments of the durative actions that are not strongly safe as they are unbounded: $\alpha^{3end}_{L_3}$ and $\alpha^{4end}_{L_4}$. However, their auxiliary versions $\alpha^{3end}_{*L_3}$ and $\alpha^{4end}_{*L_4}$ are strongly safe since they are balanced (when the over all condition $\texttt{lifting}(x_3, y_3)$ is added to the end effects, it matches the* *delete effect* $\texttt{lifting}(x_3, y_3)$ *and balances the add effect* $\texttt{available}(x_3)$*; similar considerations hold for* $D\alpha^4$*). Reachability for* $\alpha^{3end}_{*L_3}$ *and* $\alpha^{4end}_{*L_4}$ *is a straightforward check. In consequence, condition (ii) holds.*

We now need to verify condition (iii) of Corollary 92, i.e. $\mathcal{A}^d$ *is relevant right isolated. Under the re-writing* $\mathcal{M}_{L_3L_4}$*, we have that* $x_3 = x_4$ *and* $y_3 = y_4$ *and therefore* $Eff^+_{\alpha^{3end}_{L_3}} \cup_{\mathcal{M}_{L_3L_4}} Eff^+_{\alpha^{4end}_{L_4}} = \{\texttt{available}(x_3) = \texttt{available}(x_4)\}$*. Hence condition (i) of Definition 90 is satisfied.*

We can conclude that $\mathcal{T}_{dp}$ *is an invariant template.*

Example 19 (*Rovers* domain)**.** *Consider the domain Rovers (see Appendix C) and the template:*

$$\mathcal{T}_{rv} = (\{\langle \texttt{full}/1, 0\rangle, \langle \texttt{empty}/1, 0\rangle\}$$

Invariants of this template would mean that the store of a rover can be in two possible states: empty or full. Since rovers' stores can only contain one object, this seems a promising template.

To verify that it is invariant, we now analyse the safety of each durative schemas in the domain. The schemas in the domains are all durative:

$$\mathcal{A}^d = \{\texttt{navigate}, \texttt{sample\_soil}, \texttt{sample\_rock}, \texttt{drop}, \texttt{calibrate}, \texttt{take\_image},$$

$$\texttt{communicate\_soil\_data}, \texttt{communicate\_rock\_data}, \texttt{communicate\_image\_data}\}$$

We indicate them as $D\alpha^1, \ldots, D\alpha^9$ *respectively and, given* $D\alpha^i$*, its variables as* $x_i, y_i, \ldots$.

The actions $D\alpha^1$ *and* $D\alpha^5 - D\alpha^9$ *have no formulas that match the template so they are irrelevant and, consequently, strongly safe. We are left with:* $D\alpha^2 = \texttt{sample\_soil}$, $D\alpha^3 = \texttt{sample\_rock}$ *and* $D\alpha^4 = \texttt{drop}$*. Each of them has just one* $\mathcal{T}$*-class* $L_i = \{\texttt{empty}(x_i),\ \texttt{full}(x_i)\}$ *for* $i = 2, 3, 4$*. Note that* $D\alpha^2_{L_2}$ *and* $D\alpha^3_{L_3}$ *are both weakly safe of type (a). Instead,* $D\alpha^4_{L_4}$ *is strongly safe:* $\alpha^{4st}_{L_4}$ *is irrelevant and* $\alpha^{4end}_{L_4}$ *is relevant bounded.*

Note that we cannot conclude invariance by using either Corollary 92 (as not all schemas are strongly safe) or Corollary 98 (because of $\alpha^{4end}_{L_4}$ *that is relevant bounded). We now directly show that, surprisingly,* $\mathcal{T}_{rv}$ *is not invariant. If* γ *is any instance of* $\mathcal{T}$*, we consider groundings* gr^i *for* $i = 2, 4$ *such that* gr^i *and* γ *are coherent over* L_i *and we put* $Da^i = gr^i(D\alpha^i)$*. Now we are essentially in the case analysed in Example 13: intertwining two copies of* Da^4 *and one of* Da^2 *leads to an admissible not individually* γ*-safe sequence.*

If we modify the durative action schema $D\alpha^4$ *adding an overall condition* $Pre\alpha^{4inv} = \{\texttt{full}(x_4)\}$*, by arguing similarly to Example 13, we can prove that* $\mathcal{A}^d$ *is pairwise relevant non overlapping and thus conclude, using Corollary 97, that* $\mathcal{T}$ *is now invariant.*

This example shows how our invariant synthesis can be used as a debugging tool and as a method to improve the modelling of planning domains. In this case, the addition of an overall condition is sufficient to prevent erroneous physical phenomena as a store being full and empty at the same time.

9. Guess, Check and Repair Algorithm

As with related techniques (Gerevini and Schubert, 1998, 2000; Rintanen, 2000; Helmert, 2009), our algorithm for finding invariants implements a *guess, check and repair* approach. It starts by generating a set of initial simple templates. For each template $\mathcal{T}$, it then applies the results stated in the previous sections to check its invariance. If $\mathcal{T}$ is invariant, the algorithm outputs it. On the other hand, if the algorithm does not manage to prove the invariance of $\mathcal{T}$, it discards it. Before rejection, however, the algorithm tries to fix the template by generating a set of new templates that are guaranteed not to fail for the same reasons as $\mathcal{T}$. In turn, these new templates need to be checked against the invariance conditions as they might fail due to other reasons.

9.1. Guessing initial templates

When we create the set of initial templates, we ignore constant relations, i.e. relations whose ground atoms have the same truth value in all the states (for example, type predicates). In fact, they are trivially invariants and so are typically not interesting.

For each modifiable relation r with arity k, we generate $k + 1$ initial templates. They all have one component and zero or one counted variable (which can be in any position from 0 to $k - 1$): $\langle r/k, k\rangle$ (no counted variable) and $\langle r/k, p\rangle$ with $p \in \{0, \ldots, k - 1\}$. Since the templates have one component, there is only one possible admissible partition $\mathcal{F}_C$, with $C = \{c\}$. Hence, we construct the template $\mathcal{T} = (C, \mathcal{F}_C)$.

Example 20 (*Floortile* domain)**.** *Consider the components* $c_1 = \langle \texttt{robotAt}, 2, 1\rangle$*. Put* $\mathcal{F}_C = \{F_1\}$ *where* $F_1 = \{(c_1, 0)\}$*. An initial template is* $\mathcal{T}_1 = (\{c_1\}, \{F_1\})$*. Intuitively, invariants of* $\mathcal{T}_1$ *mean that a robot can occupy only one position at any time and our algorithm validates it as an invariant. Another initial template is built by considering the component* $c_2 = \langle \texttt{robotAt}, 2, 0\rangle$ *and the partition* $\mathcal{F}_C = \{F_2\}$ *where* $F_2 = \{(c_2, 1)\}$*. We have another initial template:* $\mathcal{T}_2 = (\{c_2\}, \{F_2\})$*. Invariants of this template mean that a tile cannot be occupied by more than one robot, which is not true in general, and our algorithm correctly discards it. Finally, consider the component* $c_3 = \langle \texttt{robotAt}, 2, 2\rangle$ *and the partition* $\mathcal{F}_C = \{F_3\}$ *where* $F_3 = \{(c_3, 0), (c_3, 1)\}$*. Another initial template is* $\mathcal{T}_3 = (\{c_3\}, \{F_3\})$*. This is also not an invariant and is rejected.*

If we repeat this process with every modifiable relation r in the Floortile *domain, we obtain the full set of initial templates.*

9.2. Checking conditions for invariance

We apply the results stated in the previous sections to check the invariance of a template. In particular, we apply our most operative results: Corollaries 74 , 92 , 97, 98. Remarkably, all these results work at the level of action schemas, not ground actions.

We first need to verify if all the instantaneous action schemas $\mathcal{A}$ in the domain, both the native ones and those obtained from the fragmentation of durative actions, respect the strong safety conditions. We then check safety conditions that only involve durative action schemas that are not strongly safe. Finally, we validate additional conditions that avoid the intertwinement of potentially dangerous durative actions. Given the different computational complexity of our results (see considerations below), our algorithm checks the applicability of them in the following order: first, Corollary 74, which involves only conditions for instantaneous schemas, then Corollary 98, which considers safety conditions for individual action schemas, and finally Corollaries 92 and 97, which need to verify conditions involving pairs of durative action schemas. To implement this procedure, we apply the decision tree shown in Figure 11 to the set of action schemas $\mathcal{A}$. The leaves labelled as *Possibly Not Invariant* arise when our sufficient results do not apply. In this case, we cannot assert anything about the invariance of the template.

Our checks involve the analysis of all $\mathcal{T}$-classes in each action schema α in the domain. Since the $\mathcal{T}$-classes form a partition of the set of formulas in the schema that match the template, the maximum number of $\mathcal{T}$-classes is equal to the number of such formulas. We can estimate this term with the product $\omega \cdot |C|$ where ω is the maximum number of formulas in any schema that

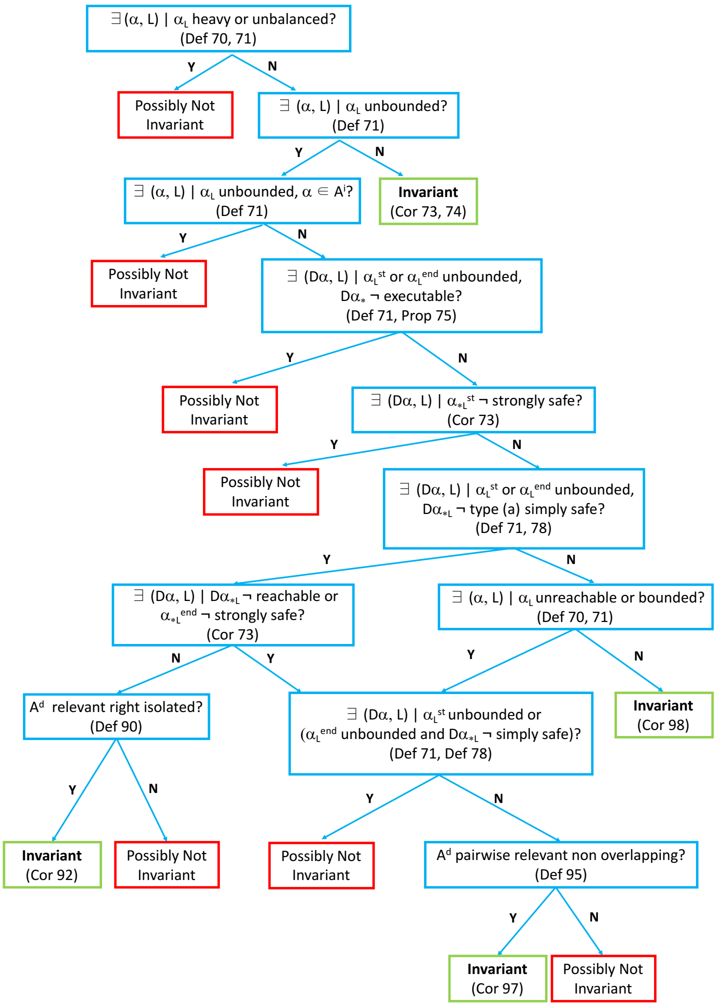

$\exists(\alpha, L) \mid \alpha_L$ heavy or unbalanced? (Def 70, 71)
Y
N
Possibly Not Invariant
$\exists$ $(\alpha, L) \mid \alpha_L$ unbounded? (Def 71)
Y
N
$\exists$ $(\alpha, L) \mid \alpha_L$ unbounded, $\alpha \in A^i$? (Def 71)
Invariant (Cor 73, 74)
Y
N
Possibly Not Invariant
$\exists$ $(D\alpha, L) \mid \alpha_L^{st}$ or α_L^{end} unbounded, $D\alpha_* \neg$ executable? (Def 71, Prop 75)
Y
N
Possibly Not Invariant
$\exists$ $(D\alpha, L) \mid \alpha_{*L}^{st} \neg$ strongly safe? (Cor 73)
Y
N
Possibly Not Invariant
$\exists$ $(D\alpha, L) \mid \alpha_L^{st}$ or α_L^{end} unbounded, $D\alpha_{*L} \neg$ type (a) simply safe? (Def 71, 78)
Y
N
$\exists$ $(D\alpha, L) \mid D\alpha_{*L} \neg$ reachable or $\alpha_{*L}^{end} \neg$ strongly safe? (Cor 73)
$\exists$ $(\alpha, L) \mid \alpha_L$ unreachable or bounded? (Def 70, 71)
N
Y
Y
N
A^d relevant right isolated? (Def 90)
$\exists$ $(D\alpha, L) \mid \alpha_L^{st}$ unbounded or $(\alpha_L^{end}$ unbounded and $D\alpha_{*L} \neg$ simply safe)? (Def 71, Def 78)
Invariant (Cor 98)
Y
N
Y
N
Invariant (Cor 92)
Possibly Not Invariant
Possibly Not Invariant
A^d pairwise relevant non overlapping? (Def 95)
Y
N
Invariant (Cor 97)
Possibly Not Invariant

Figure 11: Decision Tree for deciding the invariance of a template $\mathcal{T}$.

share the same relation and $|C|$ is the cardinality of the template's component set C. We deduce that all safety checks for individual schemas (both the instantaneous and the durative ones) have a computational complexity of the order of $M \cdot |\mathcal{A}| \cdot \omega \cdot |C|$, where M is the maximum number of the formulas appearing in any schema and $|\mathcal{A}|$ is the total number of schemas. Consequently, this is the computational complexity of Corollaries 74 and 98 that only involve safety checks.

The check of the right relevant isolated property of Definition 90, which is needed in Corollary 92, and the check of the pairwise relevant non overlapping property of Definition 95, which is used in Corollary 97, involve instead schema-class pairs and possibly families of matchings. When the condition to be checked only involves a fixed matching, its computational complexity is of the order of $M^2 \cdot |\mathcal{A}|^2 \cdot \omega^2 \cdot |C|^2$. This leaves out all the checks of the unreachability conditions for pairs of schemas, namely, (iii) of Definition 90 and (Aii), (Bii), (Cii) of Definition 95 that need to verify the conditions expressed in Definitions 88 and 93. Below, we make some detailed considerations on how these conditions can also be efficiently checked at the algorithmic level.

- **Check of Definition 88:** The two instances of conditions (i) and (ii) involve a fixed matching. The two instances of condition (iii) instead are quantified over the all possible matchings containing $\mathcal{M}_{L^1,L^2}$. Consider the first case (the second one being analogous). We first check if $w(Pre^+_{\alpha^{i\mathrm{end}}_{*L^i}}) \geq 2$ for $i = 1$ or for $i = 2$: if this is the case, then condition (iii) is verified. If $Pre^+_{\alpha^{i\mathrm{end}}_{*L^i}} = \{l^i\}$ and the formulas l^i are such that $\mathrm{Rel}[l^1] \neq \mathrm{Rel}[l^2]$ then, again, condition (iii) is verified. If, instead, $\mathrm{Rel}[l^1] = \mathrm{Rel}[l^2]$, we show that condition (iii) is never satisfied. Indeed, in this case, the two formulas match the template through the same component $c = \langle r/k, p \rangle$. We write $l^i = r(v^i_0, \ldots, v^i_{k-1})$ for $i = 1, 2$ and we note that $(v^1_j, v^2_j) \in \mathcal{M}_{L^1,L^2}$ for every $j \neq p$. Therefore, if $p = k$ (all variables are fixed in the component), necessarily, $l^1 =_{\mathcal{M}_{L^1,L^2}} l^2$. If instead $p < k$, then $l^1 =_{\mathcal{M}} l^2$ when we consider $\mathcal{M} = \mathcal{M}_{L^1,L^2} \cup \{(v^1_p, v^2_p)\}$. Finally, condition (iii) is certainly not verified if $Pre^+_{\alpha^{i\mathrm{end}}_{*L^i}} = \emptyset$ for $i = 1$ or $i = 2$. For two given schema-class pairs, this check has thus complexity M^2.

- **Check of Definition 93:** All three properties in the above definition, in principle, involve checking a condition over all possible matchings containing $\mathcal{M}_{L^1,L^2}$. We first consider (i). Suppose that we find l^1 and l^2 that satisfy $l^1 =_{\mathcal{M}_{L^1,L^2}} l^2$. Now fix $(\alpha, L) \in MC$. If for every $l \in Eff^-_\alpha$ we have that $\mathrm{Rel}[l] \neq \mathrm{Rel}[l^1]$, then (i) holds. If instead there are $l \in Eff^-_\alpha$ such that $\mathrm{Rel}[l] = \mathrm{Rel}[l^1] = \langle r/k \rangle$, for each of them we proceed as follows. We write $l^1 = r(v^1_0, \ldots, v^1_{k-1})$ and $l = r(v_0, \ldots, v_{k-1})$ and for each $j = 0, \ldots, k-1$ we consider the two variables v^1_j and v_j. If there exists j such that either $(v^1_j, w) \in \mathcal{M}_{L^1,L}$ for some $w \in V_\alpha \setminus \{v_j\}$, or $(w^1, v_j) \in \mathcal{M}_{L^1,L}$ for some $w^1 \in V_{\alpha^1} \setminus \{v^1_j\}$, then for sure $(v^1_j, v_j) \notin \mathcal{M}$ for any matching $\mathcal{M} \supseteq \mathcal{M}_{L^1,L}$ and (i) is verified. Now consider the case in which (i) is not verified. Now $\mathcal{M} = \mathcal{M}_{L^1,L} \cup \{(v^1_j, v_j) \mid j = 1, \ldots, k\}$ is a matching such that $l^1 =_{\mathcal{M}} l$.

  Checking condition (ii) is analogous to (i). For what (iii) is concerned, note that when we need to check this property, the schemas involved will be reachable and not heavy. For property (iii) to hold, we must have $Pre^+_{\alpha^i_{L^i}} = \{l^i\}$ for $i = 1, 2$. Following the same argument as in the previous item, if $\mathrm{Rel}[l^1] = \mathrm{Rel}[l^2]$, then (iii) is not verified. If instead $\mathrm{Rel}[l^1] \neq \mathrm{Rel}[l^2]$, we consider the set $Eff^+_{\alpha^1_{L^1}}$. If it is empty or if $Eff^+_{\alpha^1_{L^1}} = \{l\}$ and $\mathrm{Rel}[l] \neq \mathrm{Rel}[l^2]$, then (iii) is verified. Otherwise, it is not. We conclude that for two given schema-class pairs, this check has complexity $M^3 \cdot |\mathcal{A}| \cdot \omega \cdot |C| \cdot N$ where N is the maximum number of variables in any formula in the domain.

The above considerations allow to conclude that the check of the right relevant isolated property has complexity $M^2 \cdot |\mathcal{A}|^2 \cdot \omega^2 \cdot |C|^2$, while the check of the pairwise relevant non overlapping property has complexity $M^3 \cdot |\mathcal{A}|^3 \cdot \omega^3 \cdot |C|^3 \cdot N$. These are also the complexity needed to check the properties of, respectively, Corollaries 92 and 97.

Example 21 (*Floortile* domain)**.** *The variables for the computational complexity analysis are as follows:*

$$|\mathcal{A}| = 14,\ M = 4,\ \omega = 1, |C| = 3, N = 2$$

9.3. Repairing templates

When, in analysing an action schema α, we reach a failure node in our decision tree, we discard the template $\mathcal{T}$ under consideration since we cannot prove its invariance. This might be because of two reasons: either $\mathcal{T}$ is not an invariant or our sufficient conditions are not powerful enough to capture it. Before discarding the template, however, we try to fix it in such a way to obtain new templates for which it might be possible to prove invariance under our conditions. In particular, based on the schema α, we enlarge the set of components of the template by adding suitable formulas that appear in the preconditions and negative effects of α since they can be useful to prove that α is weakly or strongly safe.

More precisely, if the algorithm rejects $\mathcal{T}$ because it finds an instantaneous schema that is heavy or unbalanced (first step in the decision tree), no fixes are possible for $\mathcal{T}$. Since α leads to a weight greater or equal to two for at least an instance of $\mathcal{T}$, enlarging the set of components of $\mathcal{T}$ cannot help in repairing the template. Similarly, if there are durative schemas that are non executable or unreachable, no fixes are possible since these properties cannot be changed by adding components. However, when a failure node is reached in the presence of unbounded schemas, enlarging the set of components might prove useful in making them weakly or strongly safe schemas. We operate as follows: for each unbounded schema α, we try to turn it into a balanced action schema and, when α is the end fragment of a durative action $D\alpha$, we alternatively attempt to make $D\alpha$ a weakly safe schema, as defined in Definition 78.

Take a template $\mathcal{T} = (C, \mathcal{F}_C)$ that has been rejected by the algorithm, put k the number of fixed variables for $\mathcal{T}$ and m the number of its components. Consider an unbounded schema α with relevant formula l. We look for another formula l' in α with the following characteristics:

(i) $\mathrm{Rel}[l'] = \langle r'/a' \rangle$, where $a' = k$ or $a' = (k+1)$;

(ii) There exists a bijection β from the set of the variables of l to the set of the variables of l' such that $Arg[i, l] = Arg[\beta(i), l']$ for every $i \in I$;

(iii) $l' \in Pre^+_\alpha \cap Eff^-_\alpha$.

If α is the end fragment of a durative action $D\alpha$, then condition (iii) can be substituted with one of the alternative following conditions:

(iv) $l' \in Pre^+_{\alpha^{\mathrm{st}}_*} \cap Eff^-_{\alpha^{\mathrm{st}}_*}$

(iiv) $l' \in Pre^+_{\alpha^{\mathrm{st}}_*} \cap Eff^-_{\alpha^{\mathrm{end}}_*}$

For each formula l' that satisfies conditions (i), (ii) and one of conditions (iii), (iv) and (v), we create a new component $c' = \langle r'/k', p' \rangle$, where $p' \in \{0, \ldots, k'\}$, and one new template $\mathcal{T}' = (C', \mathcal{F}'_C)$, where $C' = C \cup \{c'\}$ and $\mathcal{F}'_C$ is an admissible partition of $F_{C'}$ such that for each $c_1, c_2 \in C$, we have that $(c_1, i) \sim_{F_{C'}} (c_2, j)$ if and only if $(c_1, i) \sim_{F_C} (c_2, j)$ and $(c, i) \sim_{F_{C'}} (c', j)$ if and only if $Arg[i, l] = Arg[j, l']$ (or, equivalently, $j = \beta(i)$).

If we find a formula l' that satisfies condition (iii), the schema α is guaranteed to be balanced for $\mathcal{T}'$; if the formula l' satisfies condition (iv), α is guaranteed to be weakly safe of type (a) for $\mathcal{T}'$; finally, if the formula l' satisfies condition (v), α is guaranteed to be weakly safe of type (b) for $\mathcal{T}'$.

Example 22 (*Floortile* domain)**.** *Consider the template $\mathcal{T}_2 = (\{c_2\}, \{F_2\})$ as indicated in Example 21 and the action schema α =*`up`^{end}*: $Pre_\alpha = \emptyset$, $Eff_\alpha^+ = \{$*`robotAt(r, y), clear(x)`*$\}$. The formula* `robotAt(r, y)` *matches $\mathcal{T}_2$ and forms a $\mathcal{T}$-class $L_1 = \{$*`robotAt(r, y)`*$\}$. The pure action schema α_{L_1} is unbounded as well as the end parts of the other schemas that indicate movements. If we apply our decision tree to $\mathcal{T}_2$ and the set of actions $\mathcal{A}$, we cannot prove that $\mathcal{T}_2$ is an invariant since the unbounded schemas are not weakly safe. Before discarding $\mathcal{T}_2$, we try to fix it. In particular, the formula* `clear(y)` *satisfies conditions (i), (ii) and (iv) above. If we add it to $\mathcal{T}_2$, we obtain a new template $\mathcal{T}_2' = (\{c_2, c_2'\}, \{F_2'\})$ where $c_2' = \langle$*`clear`*$, 1, 1\rangle$ with $F_{c_2'} = \{(c_2', 0)\}$ and $F_2' = \{(c_2, 1), (c_2', 0)\}$. If we apply our decision tree to this new template, we can prove that $\mathcal{T}_2'$ is an invariant since Corollary 98 can be successfully applied (all schemas are either strongly safe or weakly safe of type (a)). Intuitively, invariants of this template mean that a tile is either clear or occupied by a robot.*

10. Experimental Results

To evaluate the performance of our Temporal Invariant Synthesis, referred as TIS in what follows, we have performed a number of experiments on the IPC benchmarks. We implemented the TIS algorithm, reported in Section 9, in the Python language and conducted the experiments by using a Quad Core 2.6 GHz Intel i5 processor with a memory of 4 GB.

In Section 10.1, we focus on the number and quality of the invariants found by the TIS. In Section 10.2, we present a comparison between our TIS and the invariant synthesis that is used within the planner TFD (Eyerich et al., 2009), both in terms of the invariants found and of the state variables that can be synthesised based on such invariants. Finally, in Section 10.3, we present experiments that show the impact of using the TIS on the performance of two state-of-the-art planners that use a variable/value representation.

10.1. Quality of the representation

In this section, we discuss the number and quality of the invariants found by the TIS and the efficiency of our algorithm.

Figure 12 shows the invariants that our technique finds for all the IPC temporal domains (from IPC'02 to IPC'14). Each set in Figure 12 corresponds to a set C of components, which are separated by a comma and indicated with the relation name (arity is omitted here for brevity), the positions of the fixed variables (not enclosed in square brackets) and the position of the counted variable (enclosed in square brackets). For example, {at 0 [1], in 0 [1]} indicates the invariant with the components $c_1 = \langle$`at`$/2, 1\rangle$ and $c_2 = \langle$`in`$/2, 1\rangle$. The admissible partitions are not shown for brevity, however in most of the cases the only admissible partition is the trivial one.

For comparison purposes only, we have devised a variant of our invariant synthesis technique, which refer to as Simple Invariant Synthesis (SIS). The SIS is the simplest possible extension of Helmert's original technique to temporal domains. It fragments each durative schema in three instantaneous schemas and then applies Helmert's original technique, i.e. it judges safe only

Depots (IPC'02):
{clear [0]}
{lifting 0 [1], available 0}
{in 0 [1], at 0 [1], lifting 1 [0]}
{in 0 [1], on 0 [1], lifting 1 [0]}
{clear 0, in 0 [1], on 1 [0], lifting 1 [0]}

DriverLog (IPC'02):
{empty 0, driving 1 [0]}
{driving 0 [1], in 0 [1], at 0 [1]}

ZenoTravel (IPC'02):
{in 0 [1], at 0 [1]}

Rover (IPC'02):
{at 0 [1]}
{channel_free [0]}
{at_rock_sample [0]}
{at_soil_sample [0]}
{available [0]}
{at_soil_sample 0, have_soil_analysis 1 [0]}
{at_rock_sample 0, have_rock_analysis 1 [0]}
{at_rock_sample [0], at_soil_sample [0], full [0]}

Satellite (IPC'02):
{power_avail [0], power_on [0]}
{pointing 0 [1]}

Airport (IPC'04):
{facing 0 [1]}
{is-pushing [0]}
{at-segment 0 [1]}
{is-moving 0, is-pushing 0}
{is-moving [0], is-pushing [0]}
{airborne 0 [1], at-segment 0 [1]}
{at-segment 1 [0], not_occupied 0}
{airborne 0 [1], is-moving 0, is-pushing 0}
{is-parked 0 [1], is-moving 0, is-pushing 0}
{not_occupied [0], is-moving [0], is-pushing [0]}

Pipesworld - no tankage (IPC'04):
{normal 0, push-updating 0, pop-updating 0}
{normal [0], push-updating [0], pop-updating [0]}

Pipesworld - tankage (IPC'04):
{first 1 [0]}
{occupied 0, not-occupied 0}
{normal [0], not-occupied [0]}
{normal 0, push-updating 0, pop-updating 0}
{normal [0], push-updating [0], pop-updating [0]}
{push-updating [0], occupied [0], pop-updating [0]}

UMTS (IPC'04):
{initiated 0 [1]}
{initiated 1 [0]}

Openstack (IPC'06/'08):
{waiting [0]}
{stacks-avail [0]}
{started 0, waiting 0}
{started [0], waiting [0]}
{waiting 0, started 0, shipped 0}
{waiting [0], started [0], shipped [0]}

Storage (IPC'06):
{at 0 [1]}
{on 0 [1], lifting 1 [0]}
{lifting 0 [1], available 0}

TPP (IPC'06):
{at 0 [1]}

Trucks (IPC'06):
{free 1 0, in 1 2 [0]}
{in 2 1 [0], free 0 1}

Crewplanning (IPC'08):
{unused [0]}
{currentday 0 [1]}

Elevators-strips (IPC'08):
{lift-at 0 [1]}
{passengers 0 [1]}
{passenger-at 0 [1], boarded 0 [1]}

Elevators-numeric (IPC'08):
{lift-at 0 [1]}
{passenger-at 0 [1], boarded 0 [1]}

Modeltrain (IPC'08):
{idle [0]}
{switch-exit 0 [1]}
{tail-segment 0 [1]}
{head-segment 0 [1]}
{switch-entrance 0 [1]}
{next-train 0 [1], last-train-in-tail-segment 0}
{next-train 1 [0], first-train-in-head-segment 0}

Openstacks-strips (IPC'08):
{waiting [0]}
{not-made [0]}
{stacks-avail [0]}
{started 0, waiting 0}
{made 0, not-made 0}
{started [0], waiting [0]}
{made [0], not-made [0]}
{waiting 0, started 0, shipped 0}
{waiting [0], started [0], shipped [0]}

Parcprinter (IPC'08/'11):
{notprintedwith 0 1 [2]}
{notprintedwith 0 2 [1]}
{notprintedwith 1 2 [0]}
{timepoint 0 [1], location 0 [1]}
{hasimage 0 1 [2], notprintedwith 0 1 [2]}

Pegsol (IPC'08/'11):
{occupied [0]}
{free 0, occupied 0}

Sokoban (IPC'08/'11):
{at 0 [1]}
{clear [0]}
{at 1 [0], clear 0}

Transport (IPC'08):
{in 0 [1], at 0 [1]}
{ready-loading [0]}

Woodworking (IPC'08):
{idle [0]}
{unused [0]}
{unused 0, wood 0 [1]}
{unused 0, treatment 0 [1]}
{unused 0, surface-condition 0 [1]}

Floortile (IPC'11/'14):
{clear [0]}
{robot-at 0 [1]}
{robot-has 0 [1]}
{clear 0, robot-at 1 [0]}
{painted 0 [1], clear 0, robot-at 1 [0]}

Parking (IPC'11/'14):
{car-clear 0, behind-car 1 [0]}
{curb-clear 0, at-curb-num 1 [0]}
{behind-car 0 [1], at-curb-num 0 [1]}

TurnAndOpen (IPC'11/'14):
{closed [0]}
{at-robby 0 [1]}
{closed 0, open 0}
{closed [0], open [0]}
{carry 0 2 [1], free 0 1}

Matchcellar (IPC'11/'14):
{unused [0]}
{unused 0, light 0}
{unused [0], light [0]}

MapAnalyser (IPC'14):
{starting 1 [0]}
{starting 0 [1]}
{available 0, in_place 0}
{at_jun 0 [1], starting 0 [1]}
{available [0], in_place [0]}

RTAM (IPC'14):
{on_fire [0]}
{trapped [0]}
{uncertified [0]}
{off_fire 0, on_fire 0}
{loaded 0 [1], at 0 [1]}
{off_fire [0], on_fire [0]}
{untrapped 0, trapped 0}
{certified 0, uncertified 0}
{available 0, loaded 1 [0]}
{untrapped [0], trapped [0]}
{certified [0], uncertified [0]}
{loaded 0 [1], waiting 0, uncertified 0}
{certified 0, delivered 0, uncertified 0}
{certified [0], delivered [0], uncertified [0]}
{delivered 0, waiting 0, uncertified 0, loaded 0 [1]}

Figure 12: Invariants found for the temporal domains of all the IPCs. Each invariant is enclosed in braces where the relation names indicate the components of the invariant, the numbers not enclosed in square brackets indicate the positions of the fixed variables in the list of variables of the corresponding relation and numbers enclosed in square brackets indicate the position of the counted variables.

instantaneous schemas that in our classification are irrelevant or balanced. We use the SIS technique to explore the impact on runtime of the sophisticated checks that we need to perform to make sure that the durative actions that are not balanced are indeed safe.

The second and the third columns of Table 5 compare the number of invariants found by the TIS to those found by SIS for the temporal domains of the last three competitions: IPC'08, IPC'11, and IPC'14. The fourth column reports the TIS runtime (RT) for generating the invariants. The numbers tell us that the TIS computational time to calculate invariants is negligible and that there is no significant delay associated with the checks required by our algorithm, in particular the complex ones involving pairs of schemas. While these checks do not impact the computational time, they allow us to find a more comprehensive set of invariants than a simpler technique such as the SIS.

The last column in Table 5 (# Fix) reports the number of invariants obtained by repairing failed templates in our TIS algorithm and gives us an indication of the importance of the repairing step in our algorithm.

Domains	# Inv SIS	# Inv TIS	TIS RT	# Fix TIS
IPC'08				
CrewPlanning	0	**2**	0.29	0
Elevators-Num	0	**2**	0.02	1
Elevators-Str	0	**3**	0.02	1
Modeltrain	3	**7**	0.23	2
Openstacks-Adl	2	**7**	0.01	4
Openstacks-Num	4	**8**	0.06	6
Openstacks-Num-Adl	2	**5**	0.01	4
Openstacks-Str	4	**9**	0.09	6
Parcprinter	5	5	0.59	2
Pegsol	0	**2**	0.002	1
Sokoban	0	**3**	0.01	1
Transport	0	**2**	0.01	1
Woodworking	2	**5**	0.20	3
IPC'11				
Floortile	0	**5**	0.05	2
Matchcellar	3	3	0.003	2
Parking	0	**3**	0.03	3
Storage	0	**3**	0.05	2
TMS	0	0	0.02	0
TurnAndOpen	2	**5**	0.03	2
IPC'14				
Driverlog	0	**2**	0.03	2
Mapanalyser	3	**5**	0.04	4
RTAM	6	**15**	0.20	8
Satellite	0	**2**	0.01	1

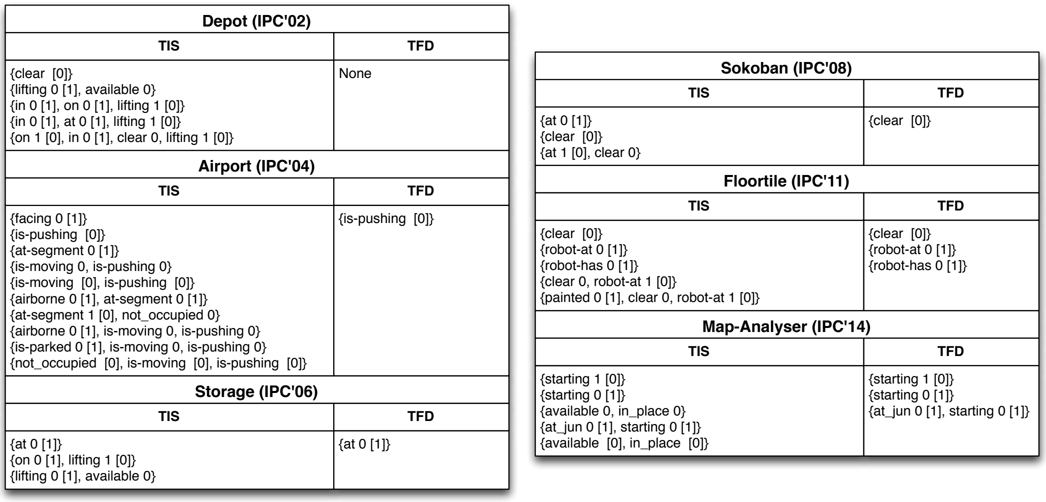
Table 5: Number of invariants (# Inv) obtained by using the Simple Invariant Synthesis (SIS) and the Temporal Invariant Synthesis (TIS) against the temporal domains of the IPC'08, IPC'11 and IPC'14, TIS run time (RT) for generating invariants and number of invariants obtained by the TIS via repairing failed templates (# Fix).

10.2. Comparison with the Temporal Fast Downward Invariant Synthesis

Currently, it is difficult to compare our technique for generating temporal invariants from lifted domains to related techniques since they either handle non temporal domains only (Fox and Long, 1998; Gerevini and Schubert, 2000; Rintanen, 2000, 2008; Helmert, 2009) or work at the ground level of the representation (Rintanen, 2014, 2017). The approach that appears most similar to ours is the invariant synthesis implemented within the Temporal Fast Downward (TFD) planner (Eyerich et al., 2009), which we refer to as TFD-IS (simply TFD in the tables and figures). This technique also works on lifted domains, but it refines the results obtained at this level by using information on reachable ground atoms. In this section, we present a comparison between our TIS and the TFD-IS with respect to: (i) the invariants found by the two approaches; and (ii) the state variables that can be synthesised based on such invariants. In comparing invariants, we consider the temporal domains of all IPCs, while in comparing state variables we focus on those domains in which the TFD-IS and the TIS output different invariants. We start with a brief description of the TFD-IS.

Our knowledge of the TFD-IS is based entirely on a manual inspection of the code[6] and an analysis of the results that the code produce as there is no formal account of such a technique. The TFD-IS is an extension of Helmert's original synthesis (described in Section 11.2) devised to deal with temporal and numeric domains. As the original technique, TFD-IS employs a guess, check and repair approach to find invariants. The algorithm analyses the temporal schemas directly, without splitting them into their start, overall and end fragments. Only two types of relevant durative action schemas are evaluated as safe: those that in our classification are balanced at start and irrelevant at end and those that are weakly safe of type (a). In all the other cases, the action schemas are labelled as unsafe and the candidate invariant is dismissed.

Figure 13 shows a few examples of the different sets of invariants found by the TFD-IS and our TIS, taking one temporal domain from each IPC competitions.



Depot (IPC'02)	
TIS	TFD

{clear [0]}
{lifting 0 [1], available 0}
{in 0 [1], on 0 [1], lifting 1 [0]}
{in 0 [1], at 0 [1], lifting 1 [0]}
{on 1 [0], in 0 [1], clear 0, lifting 1 [0]}
	None

Airport (IPC'04)	
TIS	TFD

{facing 0 [1]}
{is-pushing [0]}
{at-segment 0 [1]}
{is-moving 0, is-pushing 0}
{is-moving [0], is-pushing [0]}
{airborne 0 [1], at-segment 0 [1]}
{at-segment 1 [0], not_occupied 0}
{airborne 0 [1], is-moving 0, is-pushing 0}
{is-parked 0 [1], is-moving 0, is-pushing 0}
{not_occupied [0], is-moving [0], is-pushing [0]}
	{is-pushing [0]}

Storage (IPC'06)	
TIS	TFD

{at 0 [1]}
{on 0 [1], lifting 1 [0]}
{lifting 0 [1], available 0}
	{at 0 [1]}

Sokoban (IPC'08)	
TIS	TFD

{at 0 [1]}
{clear [0]}
{at 1 [0], clear 0}
	{clear [0]}

Floortile (IPC'11)	
TIS	TFD

{clear [0]}
{robot-at 0 [1]}
{robot-has 0 [1]}
{clear 0, robot-at 1 [0]}
{painted 0 [1], clear 0, robot-at 1 [0]}
	{clear [0]}
	{robot-at 0 [1]}
	{robot-has 0 [1]}

Map-Analyser (IPC'14)	
TIS	TFD

{starting 1 [0]}
{starting 0 [1]}
{available 0, in_place 0}
{at_jun 0 [1], starting 0 [1]}
{available [0], in_place [0]}
	{starting 1 [0]}
	{starting 0 [1]}
	{at_jun 0 [1], starting 0 [1]}

Figure 13: Examples of the different invariants produced by the TIS and the TFD-IS taking one domain from each IPC competition.

Table 6 shows the number of invariants found by the two techniques in all the domains of the IPC competitions. In 12 cases out of 33 cases the TIS outperforms the TFD synthesis, and in all

[6] TFD-0.4 code is available at http://gki.informatik.uni-freiburg.de/tools/tfd/index.html

the other cases the two synthesis output the same invariants. In several cases, the difference in the number of invariants is significant (e.g. for *Depots*, the TIS finds five invariants against zero of the TFD-IS; for *Airport*, the TIS finds ten invariants against one of the TFD-IS).

Domains	# Inv TFD	# Inv TIS
IPC'02		
Depots	0	**5**
DriverLog	2	2
ZenoTravel	1	1
Rover	5	**8**
Satellite	2	2
IPC'04		
Airport	1	**10**
Pipesworld - no tankage	0	**2**
Pipesworld - tankage	1	**6**
UMTS	2	2
IPC'06		
Openstack	6	6
Pathways	0	0
Storage	1	**3**
TPP	1	1
Trucks	2	2
IPC'08		
CrewPlanning	2	2
Elevators-Num	2	2
Elevators-Str	3	3
Modeltrain	6	**7**
Openstacks-Num	8	8
Openstacks-Num-Adl	5	5
Openstacks-Str	9	9
Parcprinter	5	5
Pegsol	1	**2**
Sokoban	1	**3**
Transport	2	2
Woodworking	5	5
IPC'11		
Floortile	3	**5**
Matchcellar	3	3
Parking	3	3
TMS	0	0
TurnAndOpen	5	5
IPC'14		
Mapanalyser	3	**5**
RTAM	11	**15**

Table 6: Number of invariants (# Inv) found by using the TIS and the TFD-IS against all the IPC temporal domains.

Let us analyse now how the different number of invariants reflects on the number of state variables that are generated based on them. In this context, together with the TIS and the TFD-IS, we also consider the case in which no invariants are used to synthesise state variables, referred as NIS (No Invariant Synthesis). In this case, a state variable with two truth values (true and false) is produced for each atom in the domain. We use the NIS as a baseline for our experiments. Table 7 reports the comparison for those IPC temporal domains in which the TIS and the TFD-IS give different outputs and, for brevity, shows three problems for each domain. The Table shows that, by increasing the number of invariants found, the TIS gives rise to a more compact representation than the other two techniques. In all the domains the TIS produces a significant reduction in the number of state variables in comparison with them. In *Sokoban*, for example, the reduction is greater than one order of magnitude.

10.3. Performance in Temporal Planners

We have performed a number of additional experiments in order to evaluate the impact of using the state variables generated by the TIS on the performance of those planners that use a variable/value representation. In particular, we focus here on the performance of two planners: Temporal Fast Downward (TFD) (Eyerich et al., 2009) and POPF-SV, a version of POPF (Coles et al., 2010) that makes use of multi-valued state variables[7].

[7] This version of POPF has been made available to us by their authors Andrew Coles and Amanda Coles.

Domains	# SV		
	NIS	TFD	TIS
Depots - p1	46	46	14
Depots - p10	198	198	32
Depots - p20	758	758	67
Rover - p1	35	32	25
Rover - p10	125	105	77
Rover- p20	480	289	204
Airport - p10	218	218	172
Airport - p30	7068	7068	3828
Airport - p50	18071	18071	9145
Pipes - NoTank - p10	100	100	98
Pipes - NoTank - p30	527	527	522
Pipes - NoTank - p50	1225	1225	1216
Pipes - Tank - p10	148	122	96
Pipes - Tank - p30	647	590	525
Pipes - Tank - p50	1385	1240	1151
Storage - p10	98	86	38
Storage - p20	546	456	136
Storage - p30	1930	1630	350
Pegsol - p10	67	67	34
Pegsol - p20	67	67	34
Pegsol - p30	67	67	34
Modeltrain - p10	589	383	191
Modeltrain - p20	588	380	188
Modeltrain - p30	1270	750	390
Sokoban - p10	490	490	72
Sokoban - p20	127	127	37
Sokoban - p30	1131	1131	75
Floortile - p1	64	40	16
Floortile - p10	126	67	26
Floortile - p19	186	97	36
MapAnalyser - p1	215	179	174
MapAnalyser - p10	752	677	670
MapAnalyser - p20	854	729	722
RTAM - p1	1279	341	311
RTAM - p10	1498	407	374
RTAM - p20	3114	677	614

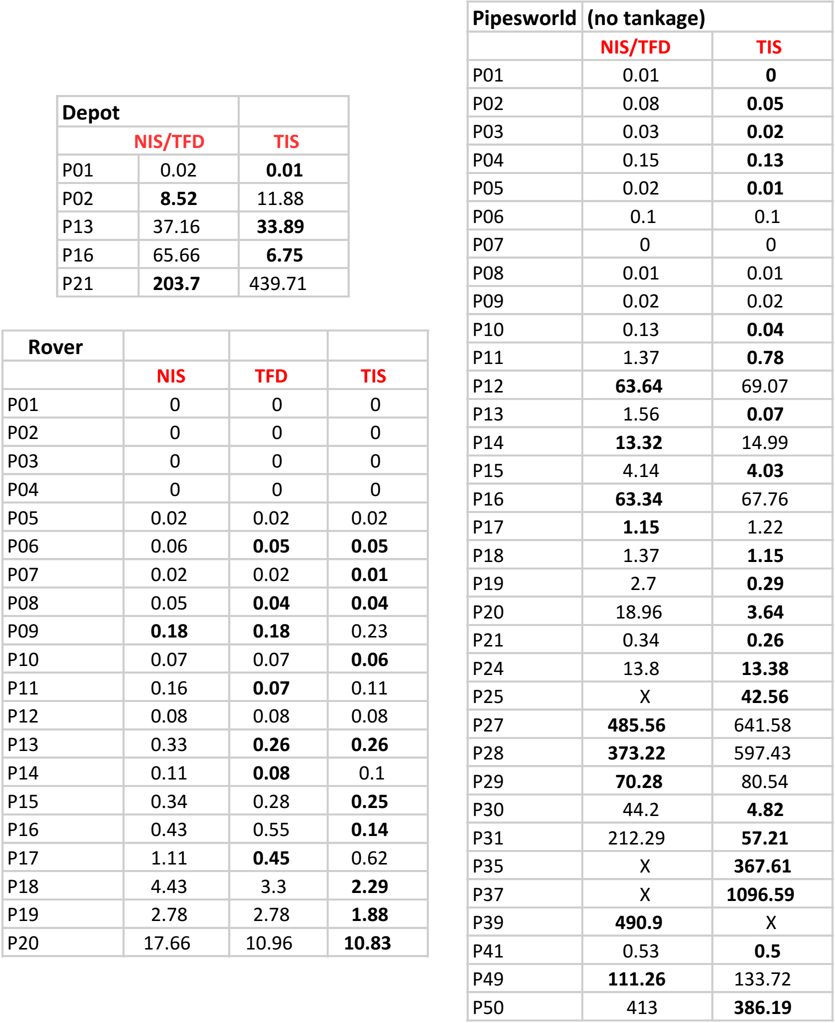
Table 7: Number of state variables (# SV) that are obtained by instantiating invariants coming from : (1) No Invariant Synthesis (NIS); (2) TFD Invariant Synthesis (TDF); and (3) Temporal Invariant Synthesis (TIS). We focus on the temporal IPC domains in which the TIS and the TDF find different invariants. We consider three problems for each domain (first, intermediate and last instance of the benchmark).

10.3.1. TFD: Temporal Fast Downward

TFD is a planning system for temporal and numeric problems based on Fast Downward (FD) (Helmert, 2006), which is limited to non temporal and non numeric domains. TFD uses a multi-valued variable representation called "Temporal Numeric SAS^{+}" (TN-SAS^{+}), which is a direct extension of the "Finite Domain Representation" (FDR) used within FD to handle tasks with time and numeric fluents. TN-SAS^{+} captures all the features of PDDL - Level 3 and represents planning tasks by using: i) a set of state variables, which are divided into logical and numeric state variables; ii) a set of axioms, which are used to represent logical dependencies and arithmetic sub-terms; and iii) a set of durative actions, which comprise: a) a duration variable; b) start, persistent and end conditions; and c) start and end effects.

TFD translates PDDL2.1 tasks into TN-SAS^{+} tasks first and then performs a heuristic search in the space of time-stamped states by using a context-enhanced additive heuristic (Helmert and Geffner, 2008) extended to handle time and numeric fluents. The translation from PDDL2.1 to TN-SAS^{+} works in four steps. First, the PDDL instance is normalised, i.e. types are removed and conditions and effects are simplified. Then, an instance where all the formulas are ground is produced through a grounding step and the invariant synthesis (which we have previously indicated as TFD-IS) is applied to generate invariants (the grounding and the invariant synthesis can be performed in parallel). Starting from the invariants provided by the invariant synthesis and the ground domain, a set of multi-valued state variables is generated. Finally, a set of actions is obtained starting from PDDL actions, which describe how the state variables change over time.

In Figures 14 - 16, we compare TFD with two alternative versions, one in which we substitute the original TFD-IS with our technique, the TIS, and one in which we substitute the TFD-IS with the NIS (No Invariant Synthesis), which we use as a baseline for our experiments. We focus on the IPC domains in which the TFD-IS and the TIS give different outputs (see Table 6). The search time (ST) is in seconds and, following the conventions of the IPCs, the timeout has been set to 1800 seconds. Problems for which a plan could not be found within the timeout by all three techniques do not appear in the table.



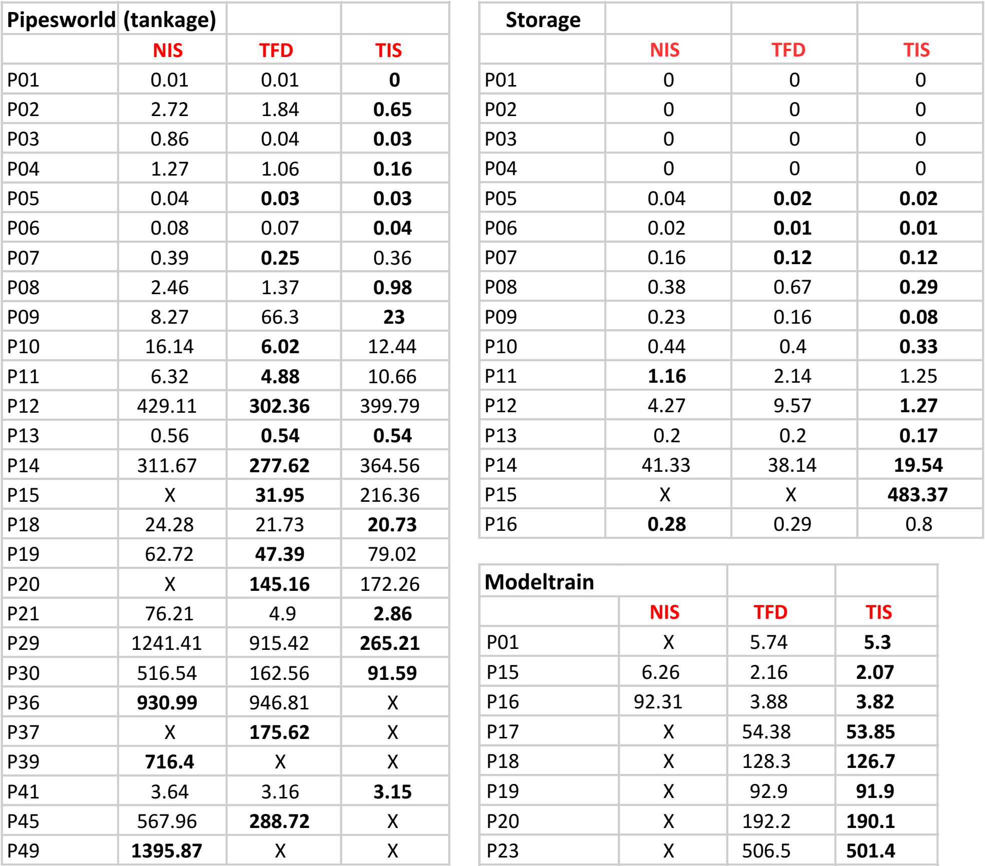
Depot	
	NIS/TFD	TIS
P01	0.02	**0.01**
P02	**8.52**	11.88
P13	37.16	**33.89**
P16	65.66	**6.75**
P21	**203.7**	439.71

Rover			
	NIS	TFD	TIS
P01	0	0	0
P02	0	0	0
P03	0	0	0
P04	0	0	0
P05	0.02	0.02	0.02
P06	0.06	**0.05**	**0.05**
P07	0.02	0.02	**0.01**
P08	0.05	**0.04**	**0.04**
P09	**0.18**	**0.18**	0.23
P10	0.07	0.07	**0.06**
P11	0.16	**0.07**	0.11
P12	0.08	0.08	0.08
P13	0.33	**0.26**	**0.26**
P14	0.11	**0.08**	0.1
P15	0.34	0.28	**0.25**
P16	0.43	0.55	**0.14**
P17	1.11	**0.45**	0.62
P18	4.43	3.3	**2.29**
P19	2.78	2.78	**1.88**
P20	17.66	10.96	**10.83**

Pipesworld	(no tankage)	
	NIS/TFD	TIS
P01	0.01	**0**
P02	0.08	**0.05**
P03	0.03	**0.02**
P04	0.15	**0.13**
P05	0.02	**0.01**
P06	0.1	0.1
P07	0	0
P08	0.01	0.01
P09	0.02	0.02
P10	0.13	**0.04**
P11	1.37	**0.78**
P12	**63.64**	69.07
P13	1.56	**0.07**
P14	**13.32**	14.99
P15	4.14	**4.03**
P16	**63.34**	67.76
P17	**1.15**	1.22
P18	1.37	**1.15**
P19	2.7	**0.29**
P20	18.96	**3.64**
P21	0.34	**0.26**
P24	13.8	**13.38**
P25	X	**42.56**
P27	**485.56**	641.58
P28	**373.22**	597.43
P29	**70.28**	80.54
P30	44.2	**4.82**
P31	212.29	**57.21**
P35	X	**367.61**
P37	X	**1096.59**
P39	**490.9**	X
P41	0.53	**0.5**
P49	**111.26**	133.72
P50	413	**386.19**

Figure 14: Search time (ST) in seconds for the planner TFD against IPC temporal domains. Three versions of the invariant synthesis are used: (1) No Invariant Synthesis (NIS); (2) the original TFD synthesis; and (3) our Temporal Invariant Synthesis (TIS). We focus on the domains in which the TIS and the TFD synthesis give different outputs. The timeout used is 1800 seconds. Problems for which all the techniques do not find a plan within the timeout do not appear in the tables.

In several cases, the lower number of state variables guaranteed by the TIS speeds up the TFD planner, for example in the domains *Modeltrain*, *Sokoban* and *Storage*, and, for some problems, the gain is very high, for example *Storage - p12, p14, p15* and *Sokoban - p03, p05, p07, p16*. In domains such as *Rover*, *Mapanalyser* and *Floortile*, when TFD manages to solve the problems, it is so fast that the impact of the number of state variables is negligible. There are also a few cases that present a mixed flavour, see for example *Depots* as well as both versions of *Pipesworld*

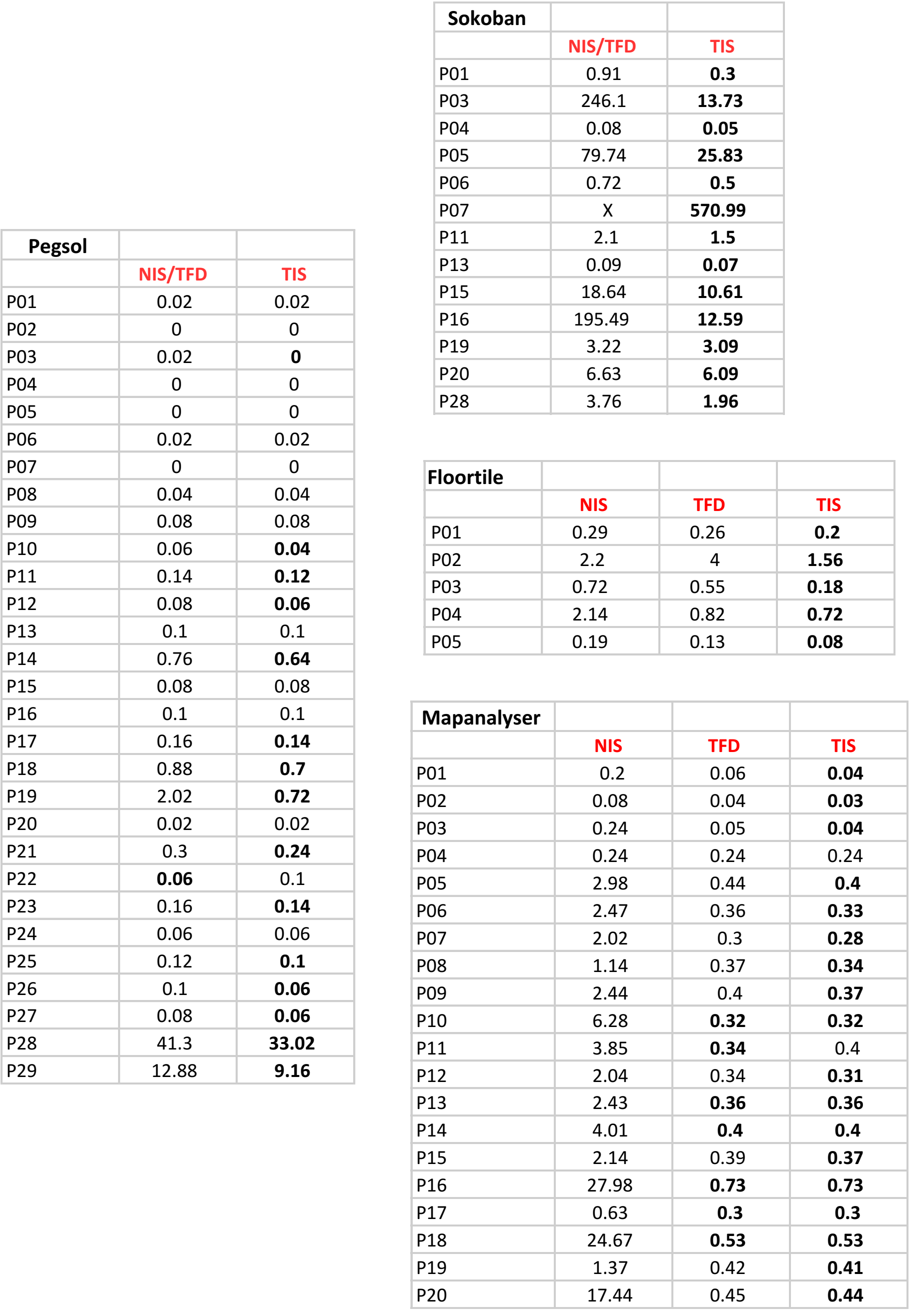


Pipesworld (tankage)			
	NIS	TFD	TIS
P01	0.01	0.01	**0**
P02	2.72	1.84	**0.65**
P03	0.86	0.04	**0.03**
P04	1.27	1.06	**0.16**
P05	0.04	**0.03**	**0.03**
P06	0.08	0.07	**0.04**
P07	0.39	**0.25**	0.36
P08	2.46	1.37	**0.98**
P09	8.27	66.3	**23**
P10	16.14	**6.02**	12.44
P11	6.32	**4.88**	10.66
P12	429.11	**302.36**	399.79
P13	0.56	**0.54**	**0.54**
P14	311.67	**277.62**	364.56
P15	X	**31.95**	216.36
P18	24.28	21.73	**20.73**
P19	62.72	**47.39**	79.02
P20	X	**145.16**	172.26
P21	76.21	4.9	**2.86**
P29	1241.41	915.42	**265.21**
P30	516.54	162.56	**91.59**
P36	**930.99**	946.81	X
P37	X	**175.62**	X
P39	**716.4**	X	X
P41	3.64	3.16	**3.15**
P45	567.96	**288.72**	X
P49	**1395.87**	X	X

Storage			
	NIS	TFD	TIS
P01	0	0	0
P02	0	0	0
P03	0	0	0
P04	0	0	0
P05	0.04	**0.02**	**0.02**
P06	0.02	**0.01**	**0.01**
P07	0.16	**0.12**	**0.12**
P08	0.38	0.67	**0.29**
P09	0.23	0.16	**0.08**
P10	0.44	0.4	**0.33**
P11	**1.16**	2.14	1.25
P12	4.27	9.57	**1.27**
P13	0.2	0.2	**0.17**
P14	41.33	38.14	**19.54**
P15	X	X	**483.37**
P16	**0.28**	0.29	0.8

Modeltrain			
	NIS	TFD	TIS
P01	X	5.74	**5.3**
P15	6.26	2.16	**2.07**
P16	92.31	3.88	**3.82**
P17	X	54.38	**53.85**
P18	X	128.3	**126.7**
P19	X	92.9	**91.9**
P20	X	192.2	**190.1**
P23	X	506.5	**501.4**

Figure 15: Search time (ST) in seconds for the planner TFD against IPC temporal domains. Three versions of the invariant synthesis are used: (1) No Invariant Synthesis (NIS); (2) the original TFD synthesis; and (3) our Temporal Invariant Synthesis (TIS). We focus on the domains in which the TIS and the TFD synthesis give different outputs. The timeout used is 1800 seconds. Problems for which all the techniques do not find a plan within the timeout do not appear in the tables.



Pegsol		
	NIS/TFD	TIS
P01	0.02	0.02
P02	0	0
P03	0.02	**0**
P04	0	0
P05	0	0
P06	0.02	0.02
P07	0	0
P08	0.04	0.04
P09	0.08	0.08
P10	0.06	**0.04**
P11	0.14	**0.12**
P12	0.08	**0.06**
P13	0.1	0.1
P14	0.76	**0.64**
P15	0.08	0.08
P16	0.1	0.1
P17	0.16	**0.14**
P18	0.88	**0.7**
P19	2.02	**0.72**
P20	0.02	0.02
P21	0.3	**0.24**
P22	**0.06**	0.1
P23	0.16	**0.14**
P24	0.06	0.06
P25	0.12	**0.1**
P26	0.1	**0.06**
P27	0.08	**0.06**
P28	41.3	**33.02**
P29	12.88	**9.16**

Sokoban		
	NIS/TFD	TIS
P01	0.91	**0.3**
P03	246.1	**13.73**
P04	0.08	**0.05**
P05	79.74	**25.83**
P06	0.72	**0.5**
P07	X	**570.99**
P11	2.1	**1.5**
P13	0.09	**0.07**
P15	18.64	**10.61**
P16	195.49	**12.59**
P19	3.22	**3.09**
P20	6.63	**6.09**
P28	3.76	**1.96**

Floortile			
	NIS	TFD	TIS
P01	0.29	0.26	**0.2**
P02	2.2	4	**1.56**
P03	0.72	0.55	**0.18**
P04	2.14	0.82	**0.72**
P05	0.19	0.13	**0.08**

Mapanalyser			
	NIS	TFD	TIS
P01	0.2	0.06	**0.04**
P02	0.08	0.04	**0.03**
P03	0.24	0.05	**0.04**
P04	0.24	0.24	0.24
P05	2.98	0.44	**0.4**
P06	2.47	0.36	**0.33**
P07	2.02	0.3	**0.28**
P08	1.14	0.37	**0.34**
P09	2.44	0.4	**0.37**
P10	6.28	**0.32**	**0.32**
P11	3.85	**0.34**	0.4
P12	2.04	0.34	**0.31**
P13	2.43	**0.36**	**0.36**
P14	4.01	**0.4**	**0.4**
P15	2.14	0.39	**0.37**
P16	27.98	**0.73**	**0.73**
P17	0.63	**0.3**	**0.3**
P18	24.67	**0.53**	**0.53**
P19	1.37	0.42	**0.41**
P20	17.44	0.45	**0.44**

Figure 16: Search time (ST) in seconds for the planner TFD against IPC temporal domains. Three versions of the invariant synthesis are used: (1) No Invariant Synthesis (NIS); (2) the original TFD synthesis; and (3) our Temporal Invariant Synthesis (TIS). We focus on the domains in which the TIS and the TFD synthesis give different outputs. The timeout used is 1800 seconds. Problems for which all the techniques do not find a plan within the timeout do not appear in the tables.

(tankage and no tankage). The reduction in the number of the variables sometimes constitutes an advantage for the planner (see for example *Depots - p16*, *PipesworldNoTankage - p13, p20, p25, p30, p31,p35, p37*, *PipesworldTankage - p09, p29, p30*), while other times it seems to be detrimental for the search (see *Depots - p21*, *PipesworldNoTankage - p27, p28, p39*, *PipesworldTank age - p36, p37, p39, p45, p49*). It should be noted that in *Depots* and *PipesworldNoTankage*, the TFD-IS does not find any invariants so we are effectively comparing the binary encoding based on NIS against the multi-value state variables encoding based on our TIS. It is interesting to see that in *PipesworldTankage* the are several instances (for example, p36, p39, p49) in which the binary encoding seems to have the best impact on performances.

In terms of problem coverage, the two techniques perform similarly. In some cases, finding more invariants help in solving difficult problems, see for example *PipesworldNoTankage - p25, p35, p37*, *Storage - p15*, and *Sokoban - p07*. However, there are also cases in which the opposite is true, for example in *PipesworldNoTankage - p39*, *PipesworldTankage - p36, p37, p45, p49*. Note that there are also problems in which the binary representation works best: *PipesworldTank age - p39, p49*.

The intuition of the influence of the number of state variables on planning performance comes from the observation that TFD uses graph structures for the computation of the heuristic estimates whose complexity is strongly influenced by this number. Our results indicate that, while this intuition is probably correct, there are other details in the heuristic computation that need to be considered and the connection between the number of state variables and the graphs' structure must be analysed more in depth. The experiments show that the TIS has the potential to speed up search and to improve the coverage of planners, but more research is needed to understand how a more compact representation can be exploited at its full potential across all the domains and problems. For example, it would be interesting to analyse the performance of TFD when different subsets of the invariants are used to generate state variables. We did not perform such experiments as this analysis is beyond the scope of this work.

10.3.2. POPF-SV: Forward-Chaining Partial-Order Planner with State Variables

POPF-SV is a version of the forward-chaining temporal planner POPF (Coles et al., 2010) that reads a variable/value representation of the domain and uses it for performing an inference step in a pre-processing phase and for reducing the size of the states during search. In particular, POPF-SV reads a standard PDDL task along with its corresponding TN-SAS$^+$ translation. This translation is the same as in TFD and has been described in the previous section. This implies that the invariant synthesis used in POPF-SV is the TFD-IS. Differently from TFD, POPF-SV reasons with both the original PDDL domain and the TN-SAS$^+$ version of the domain. The multi-valued state variable representation of the task is not used in the heuristic computation, but it is employed for two different purposes. An inference step based on the state variables is performed to support temporal preferences. This step extracts rules that are then used during search (for example, is it possible to have action a within 10 time units of action b). The second use of the invariant analysis aims to make the state representation more efficient. Only one value for each multi-valued state variable needs to be stored within a state since if one is true then the others must necessarily be false. This property results in massive savings in memory. This is particularly beneficial for POPF as memory is generally what causes the planner to fail (rather than time).

In Figures 17 - 19, we compare POPF-SV with two alternative versions, one in which we substitute the original TFD-IS used within POPF-SV with the our technique, the TIS, and one in which we substitute the TFD-IS with the NIS, which we use as a baseline for our experiments.

We focus on the IPC domains in which the TFD-IS and the TIS give different outputs (see Table 6). The search time (ST) is in seconds and, following the conventions of the IPCs, the timeout has been set to 1800 seconds. Problems for which a plan could not be found within the timeout by all three techniques do not appear in the table.

The tables clearly show that for POPF-SV dealing with fewer state variables is beneficial to the algorithm across all the domains. In the domains that are more challenging for the planner, such as *Depots*, *PipesworldTankage*, *PipesworldNoTankage Sokoban*, and *Floortile*, the gain is significant. These results are in line with our intuition that a larger set of invariants help to obtain more compact representations, which in turn have a positive impact on planners' performance. The three versions work similarly in terms of coverage.

10.4. Beyond the IPC

The generality of our approach to synthesise invariants from lifted temporal domains cannot be fully appreciated by considering IPC domains only. This is because the durative actions of such domains present a rather uniform structure: most of the actions are either balanced or weakly safe of type (a). However, in domains that describe practical applications, other types of actions are used. For example, in data processing, web services composition, production and software domains (Barnes et al., 2013; Ghosh and Ghosh, 2010; Liu et al., 2007; Golden, 2003), the creation of new objects from an empty set is often required. Typically, the actions that create the new objects are bounded or weakly safe of type (c). The TFD-IS fail to identify invariants in such situations as they regard such actions as unsafe.

As an example, let us consider the following simple case.

Example 23 (*DriverLog* domain)**.** *Consider the domain DriverLog (see Appendix D) and the template:*

$$\mathcal{T}_{dl} = (\{\langle \texttt{empty}/1, 0\rangle, \langle \texttt{driving}/2, 1\rangle\})$$

Invariants of this template mean that a truck can be in two possible states: empty or being driven by a driver. The action schemas in the domains are all durative:

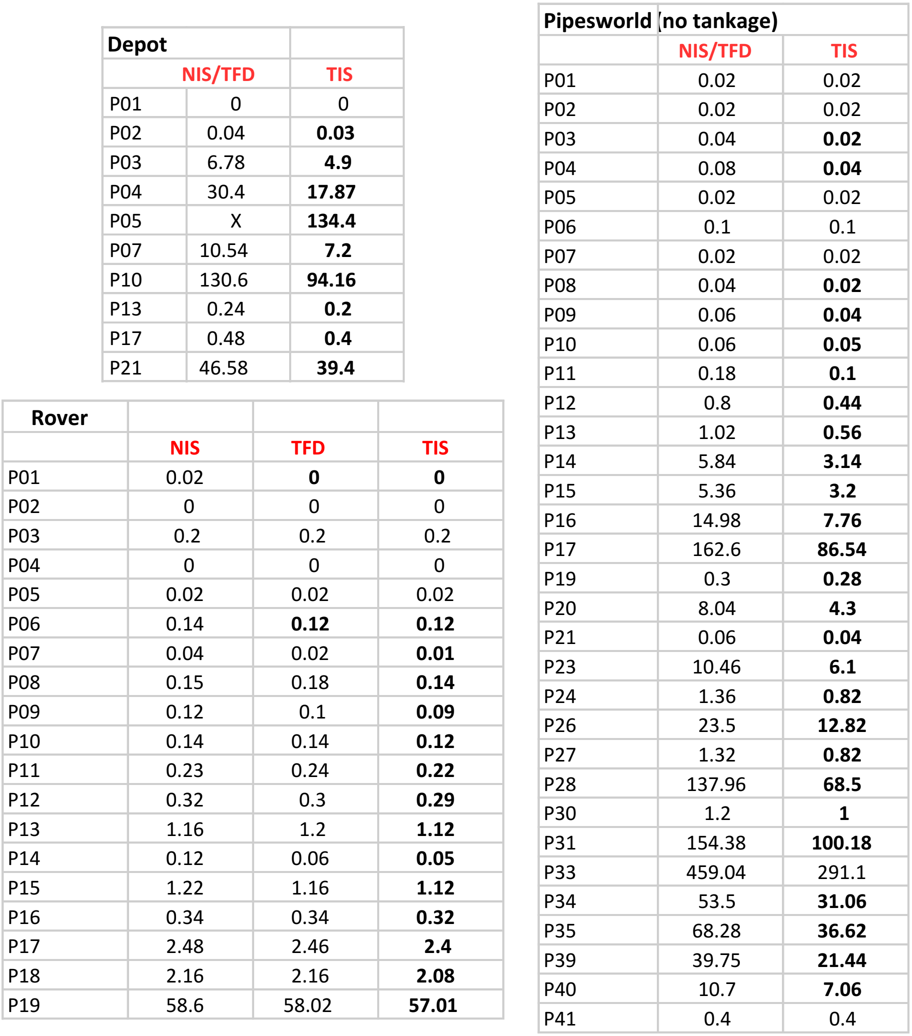
$$\mathcal{A}^d = \{\texttt{load}, \texttt{unload}, \texttt{board}, \texttt{disembark}, \texttt{drive}, \texttt{walk}\}$$

The schemas `load`, `unload`, `drive`, `walk` *have no formulas that match the template so they are strongly safe. The other two schemas, which we indicate as $D\alpha^1, D\alpha^2$ with variables as $x_i, y_i, \ldots$, have respectively $\mathcal{T}$-classes $L_i = \{\texttt{empty}(x_i), \texttt{driving}(y_i, x_i)\}$ and they are weakly safe of type (a). By Corollary 98, $\mathcal{T}_{dl}$ is invariant.*

Let us now introduce a new action in the domain, `buyTruck`*, which represents the action of buying a new truck, with the following specification:*

```
(:durative-action buyTruck
  :parameters (?new_trk - truck ?dpt - depot)
  :duration (= ?duration 5)
  :condition (and (at start (forall (?d - driver) (not (driving ?d ?new_trk))))
                  (at start (forall (?l - location) (not (at ?new_trk ?l))))
                  (at start (not (empty ?new_trk))))
  :effect (and (at start (empty ?new_trk)) (at start (at ?new_trk ?dpt))))
```

This new action, $D\alpha^3$, has $\mathcal{T}$-class $L_3 = \{\texttt{empty}(x_3), \texttt{driving}(y_3, x_3)\}$. Note that $D\alpha^3_{L_3}$ is weakly safe of type (c). Therefore, we can no longer apply Corollary 98 to infer that, with the addition of this new schema, the template $\mathcal{T}_{dl}$ remains invariant. We have to use instead the

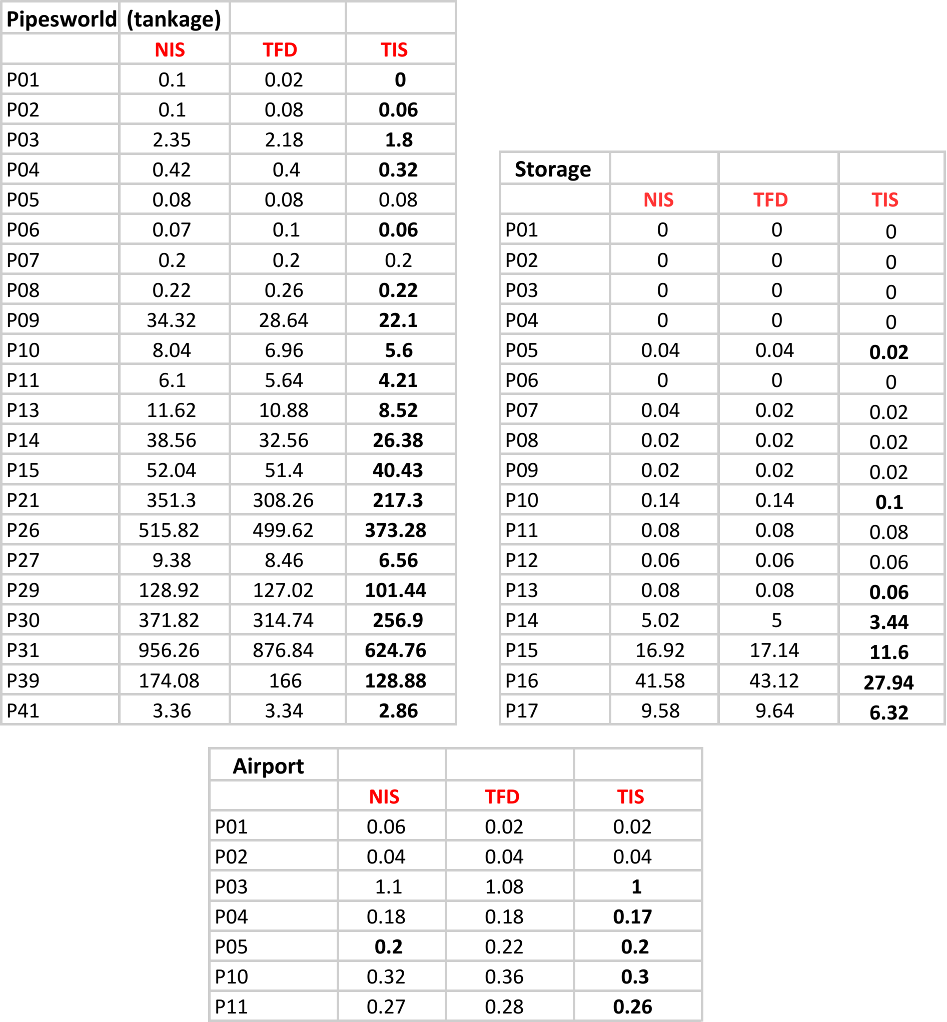


Depot		
	NIS/TFD	TIS
P01	0	0
P02	0.04	**0.03**
P03	6.78	**4.9**
P04	30.4	**17.87**
P05	X	**134.4**
P07	10.54	**7.2**
P10	130.6	**94.16**
P13	0.24	**0.2**
P17	0.48	**0.4**
P21	46.58	**39.4**

Rover			
	NIS	TFD	TIS
P01	0.02	**0**	**0**
P02	0	0	0
P03	0.2	0.2	0.2
P04	0	0	0
P05	0.02	0.02	0.02
P06	0.14	**0.12**	**0.12**
P07	0.04	0.02	**0.01**
P08	0.15	0.18	**0.14**
P09	0.12	0.1	**0.09**
P10	0.14	0.14	**0.12**
P11	0.23	0.24	**0.22**
P12	0.32	0.3	**0.29**
P13	1.16	1.2	**1.12**
P14	0.12	0.06	**0.05**
P15	1.22	1.16	**1.12**
P16	0.34	0.34	**0.32**
P17	2.48	2.46	**2.4**
P18	2.16	2.16	**2.08**
P19	58.6	58.02	**57.01**

Pipesworld (no tankage)		
	NIS/TFD	TIS
P01	0.02	0.02
P02	0.02	0.02
P03	0.04	**0.02**
P04	0.08	**0.04**
P05	0.02	0.02
P06	0.1	0.1
P07	0.02	0.02
P08	0.04	**0.02**
P09	0.06	**0.04**
P10	0.06	**0.05**
P11	0.18	**0.1**
P12	0.8	**0.44**
P13	1.02	**0.56**
P14	5.84	**3.14**
P15	5.36	**3.2**
P16	14.98	**7.76**
P17	162.6	**86.54**
P19	0.3	**0.28**
P20	8.04	**4.3**
P21	0.06	**0.04**
P23	10.46	**6.1**
P24	1.36	**0.82**
P26	23.5	**12.82**
P27	1.32	**0.82**
P28	137.96	**68.5**
P30	1.2	**1**
P31	154.38	**100.18**
P33	459.04	291.1
P34	53.5	**31.06**
P35	68.28	**36.62**
P39	39.75	**21.44**
P40	10.7	**7.06**
P41	0.4	0.4

Figure 17: Search time (ST) in seconds for the planner POPF-SV against IPC temporal domains. Three versions of the invariant synthesis are used: (1) No Invariant Synthesis (NIS); (2) the original POPF-SV synthesis (i.e. TFD-IS); and (3) our Temporal Invariant Synthesis (TIS). We focus on the domains in which the TIS and the TFD-IS give different outputs. The timeout used is 1800 seconds. Problems for which all the techniques do not find a plan within the timeout do not appear in the tables.

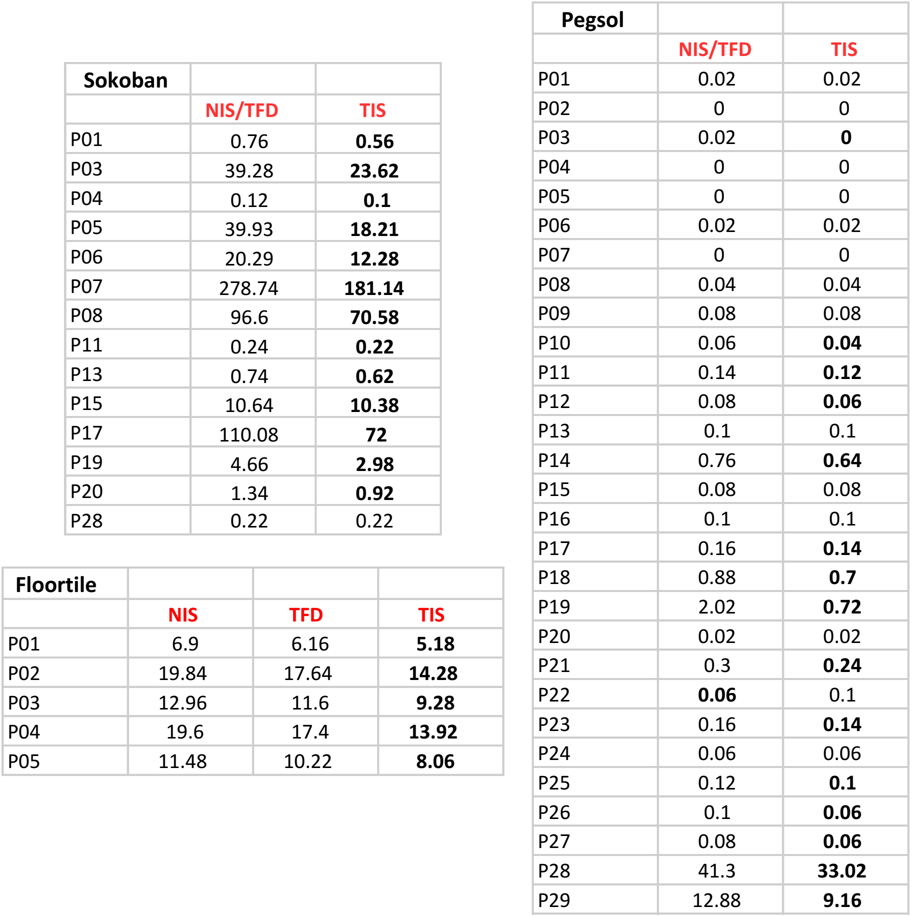


Pipesworld (tankage)			
	NIS	TFD	TIS
P01	0.1	0.02	**0**
P02	0.1	0.08	**0.06**
P03	2.35	2.18	**1.8**
P04	0.42	0.4	**0.32**
P05	0.08	0.08	0.08
P06	0.07	0.1	**0.06**
P07	0.2	0.2	0.2
P08	0.22	0.26	**0.22**
P09	34.32	28.64	**22.1**
P10	8.04	6.96	**5.6**
P11	6.1	5.64	**4.21**
P13	11.62	10.88	**8.52**
P14	38.56	32.56	**26.38**
P15	52.04	51.4	**40.43**
P21	351.3	308.26	**217.3**
P26	515.82	499.62	**373.28**
P27	9.38	8.46	**6.56**
P29	128.92	127.02	**101.44**
P30	371.82	314.74	**256.9**
P31	956.26	876.84	**624.76**
P39	174.08	166	**128.88**
P41	3.36	3.34	**2.86**

Storage			
	NIS	TFD	TIS
P01	0	0	0
P02	0	0	0
P03	0	0	0
P04	0	0	0
P05	0.04	0.04	**0.02**
P06	0	0	0
P07	0.04	0.02	0.02
P08	0.02	0.02	0.02
P09	0.02	0.02	0.02
P10	0.14	0.14	**0.1**
P11	0.08	0.08	0.08
P12	0.06	0.06	0.06
P13	0.08	0.08	**0.06**
P14	5.02	5	**3.44**
P15	16.92	17.14	**11.6**
P16	41.58	43.12	**27.94**
P17	9.58	9.64	**6.32**

Airport			
	NIS	TFD	TIS
P01	0.06	0.02	0.02
P02	0.04	0.04	0.04
P03	1.1	1.08	**1**
P04	0.18	0.18	**0.17**
P05	**0.2**	0.22	**0.2**
P10	0.32	0.36	**0.3**
P11	0.27	0.28	**0.26**

Figure 18: Search time (ST) in seconds for the planner POPF-SV against IPC temporal domains. Three versions of the invariant synthesis are used: (1) No Invariant Synthesis (NIS); (2) the original POPF-SV synthesis (i.e. TFD-IS); and (3) our Temporal Invariant Synthesis (TIS). We focus on the domains in which the TIS and the TFD-IS give different outputs. The timeout used is 1800 seconds. Problems for which all the techniques do not find a plan within the timeout do not appear in the tables.



Sokoban		
	NIS/TFD	TIS
P01	0.76	**0.56**
P03	39.28	**23.62**
P04	0.12	**0.1**
P05	39.93	**18.21**
P06	20.29	**12.28**
P07	278.74	**181.14**
P08	96.6	**70.58**
P11	0.24	**0.22**
P13	0.74	**0.62**
P15	10.64	**10.38**
P17	110.08	**72**
P19	4.66	**2.98**
P20	1.34	**0.92**
P28	0.22	0.22

Floortile			
	NIS	TFD	TIS
P01	6.9	6.16	**5.18**
P02	19.84	17.64	**14.28**
P03	12.96	11.6	**9.28**
P04	19.6	17.4	**13.92**
P05	11.48	10.22	**8.06**

Pegsol		
	NIS/TFD	TIS
P01	0.02	0.02
P02	0	0
P03	0.02	**0**
P04	0	0
P05	0	0
P06	0.02	0.02
P07	0	0
P08	0.04	0.04
P09	0.08	0.08
P10	0.06	**0.04**
P11	0.14	**0.12**
P12	0.08	**0.06**
P13	0.1	0.1
P14	0.76	**0.64**
P15	0.08	0.08
P16	0.1	0.1
P17	0.16	**0.14**
P18	0.88	**0.7**
P19	2.02	**0.72**
P20	0.02	0.02
P21	0.3	**0.24**
P22	**0.06**	0.1
P23	0.16	**0.14**
P24	0.06	0.06
P25	0.12	**0.1**
P26	0.1	**0.06**
P27	0.08	**0.06**
P28	41.3	**33.02**
P29	12.88	**9.16**

Figure 19: Search time (ST) in seconds for the planner POPF-SV against IPC temporal domains. Three versions of the invariant synthesis are used: (1) No Invariant Synthesis (NIS); (2) the original POPF-SV synthesis (i.e. TFD-IS); and (3) our Temporal Invariant Synthesis (TIS). We focus on the domains in which the TIS and the TFD-IS give different outputs. The timeout used is 1800 seconds. Problems for which all the techniques do not find a plan within the timeout do not appear in the tables.

more general Corollary 97. Assumptions (i) and (ii) are evident. We need to show property (iii), that $\mathcal{A}^d$ is pairwise relevant non-overlapping (see Definition 95). To this aim, we fix $\mathcal{A}^d C(\mathcal{T})^* = \mathcal{A}^d C(wk, \mathcal{T}) = \{(D\alpha^1, L_1), (D\alpha^2, L_2), (D\alpha^3, L_3)\}$ and we check that properties A, B, and C hold true.

The verification of properties A and B for the pair $(D\alpha^1, L_1)$ and $(D\alpha^2, L_2)$ can be carried on using the same argument at the lifted level previously used in the proof of Corollary 58 that works for generic weakly safe actions of type (a). We now consider the pairs $(D\alpha^i, L_i)$ and $(D\alpha^3, L_3)$ for $i = 1, 2$. Note that since $Pre^+_{\alpha_i^{st}} \cap_{\mathcal{M}_{L_i,L_3}} Pre^-_{\alpha_3^{st}} \neq \emptyset$ for $i = 1, 2$, it follows that property (Ai) is in both cases satisfied. Moreover, since $\{\alpha^{i\,st}, \alpha_3^{st}\}$ is $(\mathcal{A}C(\mathcal{T})^*_{irr}, L_i, L_3)$-unreachable (property (i) in Definition 93 is satisfied), it follows that property (Bii) is satisfied. Finally, since $\mathcal{A}C(\mathcal{T})^*_{rel} = \emptyset$, property C is vacuously verified. Therefore all assumptions of Corollary 97 are satisfied and, consequently, $\mathcal{T}_{dl}$ is invariant.

The TFD-IS*, however, does not account for the safety of bounded actions and so classifies the new action as unsafe. In consequence, it does not produce the template $\mathcal{T}_{dl}$ when the new action is introduced. The FD invariant synthesis will have a similar behaviour.*

11. Related Work

Several approaches to invariant synthesis are available in the literature. In what follows, we present these approaches in depth by highlighting differences and similarities with our technique.

11.1. Previous work on synthesising temporal invariants

The invariant synthesis presented in Bernardini and Smith (2011a) represents a preliminary version of the technique described in this paper. Bernardini and Smith (2011a) lacks a rigorous presentation of the theoretical framework behind the synthesis of invariants. In addition, the algorithm does not capture all the cases in which unsafe intertwinements between durative actions can happen, which results in the generation of unsafe invariants that could be violated under some circumstances. For example, for the domain *ZenoTravel* (IPC'02), the algorithm classifies $\mathcal{T}_{zt} = (\{\langle \texttt{fuel-level}/2, 1\rangle\})$ as an invariant. However, it can be shown that $\mathcal{T}_{zt}$ is not an invariant because the schema `refuel` is weakly safe of type (b) and can give rise to dangerous intertwinements that can invalidate the invariant conditions.

11.2. Fast Downward and Temporal Fast Downward

Helmert (2009) presents a translation from a subset of PDDL2.2 into FDR (Finite Domain Representation), a multi-valued planning task formalism used within the planner Fast Downward (Helmert, 2006). The translation only handles non-temporal and non-numeric PDDL2.2 domains, the so-called "PDDL Level 1" (equivalent to STRIPS (Fikes and Nilsson, 1971) with the extensions known as ADL (Pednault, 1986). One of the steps of this translation is the identification of mutual exclusion invariants and it is an extension of the technique presented in Edelkamp and Helmert (1999) developed for STRIPS.

When considering sequential domains, the invariant synthesis presented in this paper works similarly to Helmert's one. In particular, both work at the lifted level, while all the other related techniques discussed below work at the ground level. Both techniques start from simple invariant candidates and check them against conditions that ensure invariance by analysing the structure of the action schemas in the domain. When a candidate is rejected, they both try to refine it to create a new stronger candidate, which is then checked from scratch.

However, in contrast with our technique, Helmert's method considers a schema safe only when the weight transitions from one, to zero and back to one. Potentially safe transitions from zero to one are ignored. This simplified analysis results in the identification of a smaller set of invariants with respect to our technique. For example, Helmert's invariant synthesis labels as unsafe all the action schemas that add a relevant formula without deleting that formula or another relevant one, even when the preconditions impose that the weight is zero when the action schema is applied. In this way, Helmert's invariant synthesis misses invariants that our technique is able to find (see Example 23).

Chen et al. (2009) builds on Helmert's invariant synthesis and his multi-valued domain formulation to synthesise *long-distance* mutual exclusions (londex), which capture constraints over actions and facts not only at the same time step but also across multiple steps. The londex has been successfully used in SAT-based planners to improve their performance. In future work, we will explore how the concept of londex can be extended to temporal domains.

Within the context of Temporal Fast Downward (TFD) (Eyerich et al., 2009), a simple extension of Helmert's invariant synthesis is used to deal with temporal and numeric domains of the ICPs. See Section 10.2 for a description of such a technique.

11.3. Rintanen's Invariant Synthesis

An algorithm for inferring invariants in propositional STRIPS domains is proposed by Rintanen (2000, 2008). It synthesises not only mutual-exclusion invariants, but also other types of invariants. The algorithm works on a ground representation of the domain and, starting from an inductive definition of invariants as formulae that are true in the initial state and are preserved by the application of every action, the algorithm is based on an iterative computation of a fix-point, which is useful for reasoning about all the invariants of a domain at the same time rather than inferring some invariants first and then use them for inferring others.

Rintanen's algorithm uses a guess, check and repair approach but, unlike our technique, it starts from stronger invariant candidates and then progressively weakens them if they are not preserved by the actions. Thus, the *repair* phase consists in considering a less general invariant instead of a more general one. For example, let us consider the schema $\sigma = x \neq y \rightarrow P(x, y) \vee Q(y, z)$ as a potential invariant (all the invariants considered have this implicative form). One of the weakening operation consists of identifying two variables. In this case, if z is set equal to x, the weaker candidate $\sigma = x \neq y \rightarrow P(x, y) \vee Q(y, x)$ is obtained and checked.

This technique has been successfully used within both Graphplan based planners (Blum and Furst, 1997), where it helps to identify unreachable subgoals, and SAT-based planners (Kautz and Selman, 1999), where it can be useful to reduce the amount of search needed. However, although its implementation is limited to invariants involving two formulas at the most, it incurs a high performance penalty on large instances.

In Rintanen (2014), Rintanen extends the original algorithm presented in Rintanen (2000, 2008) in order to handle temporal domains. As the original algorithm, the temporal one works on ground domains, not using a lifted representation at any stage. The format of the invariants found is $l_1 V(r)\ l_2$, where l_1 and l_2 are positive or negative ground facts, r is a floating point number, and the formula says that either l_1 is true or l_2 is true over the interval [0..r] relative to the current time point. If $r = inf$, the formula means that if l_1 is false, then l_2 will remain true forever. Since Rintanen's invariant synthesis exploits the initial conditions and the ground representation of the domain, it usually finds a broader range of invariants than our technique. However, this makes the invariant synthesis suffer high computational cost. Reachability analysis on a ground representation of the planning instances is computationally very expensive and, while our algorithm takes a few seconds to run, Rintanen's synthesis employs tens of minutes to find invariants in several domains (see Table 1 in Rintanen (2014)). In recent work (Rintanen, 2017), Rintanen has proposed a hybrid algorithm that performs the basic invariance tests with a ground method, but ground the actions and formulas only with respect to a smaller number of objects in order to contain complexity.

We do not directly compare our technique against Rintanen's algorithm in Section 10 because the two techniques aim to find different types of invariants (our focuses on mutual exclusion invariants, while Rintanen's tackles a broad range of invariant types) and they work on different representations of the problem (lifted versus ground). However, in what follows, we give examples of the output of Rintanen's technique for completeness.

Let us consider the *Crewplanning* domain (IPC6 and IPC7). For each crew member c_i, Rintanen's algorithm finds ground invariants of the type:

$$not\ current\_day - c_i - d_j\ V(inf)\ not\ current\_day - c_i - d_k$$

which means that if it is day d_j for the crew member c_i, it cannot be day d_k at the same time. All these invariants correspond to the lifted invariant *current_day* 0 [1] that is found by our invariant synthesis. For the same domain, however, Rintanen's algorithm finds additional invariants that express temporal relations between atoms. Our technique does not aim to find this type of invariants. For example, Rintanen's method finds temporal invariants of the form:

$$done\_sleep - c_i - d_k\ V(255)\ not\ done\_mealc_i - d_{(k+2)}$$

which means that, for the crew member c_i, the atom *done_meal* in day $k + 1$ becomes true 255 time units after the atom $done_sleep$ was true in day k. In fact, in day k, *done_sleep* is made true by the end effects of the action *sleep*. From this time point, in order to make *done_meal* true the day after $k + 1$, two actions need to be executed: *post_sleep*, with duration 195, and *have_meal* with duration 60, for a total time separation of 255 time units. For the *Crewplanning* domain, the run time of our algorithm is 0.29 seconds, while Rintanen's one has a runtime of 1 minute and 23.24 seconds for hard instances. This is actually one of the best run times, since for problems such as *Parcprinter*, *Elevators*, *Sokoban*, *Transport-numeric* and other the algorithm has a run time of more than 4 hours. Given these run times, it does not seem plausible to use Rintanen's algorithm as a pre-processing steps to improve search in planning, which is one of the most important cases of use of invariant synthesis algorithms.

11.4. DISCOPLAN

DISCOPLAN (*DIScovering State COnstraints for PLANning*) (Gerevini and Schubert, 1998) is a technique for generating invariants from PDDL Level 1 tasks. DISCOPLAN supports conditional effects without compiling them. DISCOPLAN discovers not only mutual exclusion invariants, but also other types of invariants: static predicates, simple implicative, (strict) single valuedness and n-valuedness, anti-simmetry, OR and XOR invariants.

Considering mutual exclusion invariants, DISCOPLAN uses a guess, check and repair approach similar to our approach: an hypothetical invariant is generated by analysing simultaneously the preconditions and the effects of each action to see whether an instantiation of a literal is deleted whenever another instantiation of the same literal is added. Then, this candidate is checked against all the other actions and the initial conditions. If the hypothetical invariant is not found to be valid, then all the unsafe actions are collected together and a set of possible refinements are generated. However, whereas our technique tries to refine a candidate as soon as an unsafe action is found, DISCOPLAN tries to address all the causes of unsafety at the same time while generating refinements. This approach leads to more informed choices on how to refine hypothetical invariants and can result in the identification of more invariants. However, it is more expensive from a computational point of view, which is why DISCOPLAN is often inefficient on big instances.

DISCOPLAN can be used not only for finding invariants, but also for inferring action-variable domains. An action-variable domain is a set including all the objects that can be used to instantiate the variables of an action. Such sets of possible tuples of variables are found by forward propagation of ground atoms from the initial state. This technique is related to the reachability analysis performed by Graphplan (Blum and Furst, 1997), but does not implement mutual exclusion calculation.

DISCOPLAN is usually used in combination with SAT encodings of planning problems. In particular, a pre-processing step is performed over the domain under consideration in order to find invariants and variable domains, then the domain as well as the invariants and the variable domains are translated into SAT. Finally, a SAT-based planner is used to solve the resulting translated domain. SAT-based planners (Kautz and Selman, 1999; Huang et al., 2010) show significant speed-up when invariants and action-variable domains are used.

11.5. Type Inference Module

TIM (*Type Inference Module*) (Fox and Long, 1998) uses a different approach for finding invariants in PDDL Level 1 domains. More precisely, TIM is a pre-preprocessing technique for inferring object types on the basis of the actions and the initial state. Data obtained from this computation are then used for inferring invariants. TIM recognises four kinds of invariants (invariants of types 1 and 2 correspond to mutual exclusion invariants):

1. *Identity invariants* (for example, considering the domain *Blockworld*, two objects cannot be at the same place at the same time);
2. *Unique state invariants* (for example, every object must be in at most one place at any time point);
3. *State membership invariants* (for example, every object must be in at least one place at any time point); and
4. *Resource invariants* (for example, in a 3-blocks world, there are 4 surfaces).

The invariants found by TIM have been exploited for improving performance within the planner STAN (Fox and Long, 2011).

11.6. Knowledge representation and engineering

In addition to works that address the creation of invariants directly, there are works in the literature that highlight the importance of multi-valued state variables for debugging domain descriptions and for assisting the domain designer in building correctly encoded domains (Fox and Long, 1998; Bernardini and Smith, 2011b; Cushing et al., 2007). In particular, Cushing et al. (2007) analyse well-studied IPC temporal and numeric domains and reveal several modelling errors that affect such domains. This analysis lead the authors to suggest better ways of describing temporal domains. They identify as a central feature to do so the direct specification of multi-valued state variables and show how this can help domain experts to write correct models.

Other works in the literature use the creation of invariants and state variables as an intermediate step in the translation from PDDL to other languages. In particular, Huang et al. (2010) introduce SASE, a novel SAT encoding scheme based on the SAS+ formalism Helmert (2009). The state variables (extracted from invariants) used by SASE play a key role in achieving its efficiency. Since our technique generates a broader set of invariants than related techniques, it gives rise to SAS+ tasks with smaller sets of state variables. We speculate that this would in turn reverberate positively on SAT-based planners that use a SASE encoding. Testing of this hypothesis is part of our future work.

12. Conclusions and Future Work

In this paper, we present a technique for automatically finding mutual exclusion invariants in lifted temporal planning domains expressed in PDDL2.1. Our technique builds on Helmert's invariant synthesis (Helmert, 2009), but generalised it and extends it to temporal domains. Synthesising invariants for temporal tasks is much more complex than for tasks with instantaneous actions because actions can occur simultaneously or concurrently and interfere with each other. For this reason, a simple generalisation of Helmert's approach does not work in temporal settings. In extending the theory to capture the temporal case, we have had to formulate invariance conditions that take into account the full temporal structure of the actions as well as the possible interactions between them. As a result, we have constructed a technique that is significantly more comprehensive than the related ones. Our technique is presented here formally and proofs are offered that support its soundness.

Since our technique, differently from the majority of related approaches, works at the lifted level of the representation, it is very efficient. The experimental results show that its run time is negligible, while it allows us to find a wider set of invariants, which in turn results in synthesising a smaller number of state variables to represent a domain. The experiments also indicate that the temporal planners that use state variables to represent the world may benefit from dealing with a smaller number of state variables.

Our approach to finding invariants can be incorporated in any translation from PDDL2.1 to a language based on multi-valued state variables. For example, we have used (a simplified version of) the temporal invariant synthesis described in this paper in our translator from PDDL2.1 to NDDL, which is the domain specification language of the planner EUROPA2 (Bernardini and Smith, 2008b). EUROPA2 has been the core planning technology for several NASA space mission operations. It uses a language based on multi-valued state variables that departs from PDDL2.1 in several ways. The use of our translator from PDDL2.1 to NDDL has facilitated the testing of EUROPA2 against domains of the IPCs originally expressed in PDDL2.1 (Bernardini and Smith, 2007, 2008a). This has originally motivated our work on temporal invariant synthesis.

In future work, we plan to extend our experimental evaluation by incorporating our invariant synthesis in other planners that use a multi-valued variable representation and that are not currently publicly available. This will allow us to assess more exhaustively the impact that handling fewer state variables has on the performance of temporal planners. The experimental results shown in this paper provide evidence that more research is needed in this direction.

In addition, we plan to exploit the metric information encoded in planning domains to find a broader range of invariants. Invariants for domains with metric fluents are interesting and challenging. We envisage that there are two kinds of situations to be considered: those in which it can be shown that a linear combination of fluents is invariant (relevant to domains with linear effects on variables) and those in which metric fluents interact with propositional fluents in a more complex structure. For example, one might think of a domain encoding the act of juggling in which the number of balls in the air plus the number in the hands is a constant, but the balls in the hand might be encoded propositionally (for example, by a literal `holding_left` and so on), while those in the air as a count. Finding the invariant in this case is a challenging problem since it crosses the propositional and metric fluent spaces.

Finally, as shown in Example 19, our technique can be a valuable basis for debugging temporal planning domains. We intend to work in this direction by incorporating our technique in validation tools such as VAL (Howey et al., 2004).

Appendix A: PDDL2.1 Specification of the *Floortile* Domain

```
(define (domain floor-tile)
 (:requirements :typing :durative-actions)
 (:types robot tile color - object)

(:predicates
  (robotAt ?r - robot ?x - tile)
  (up ?x - tile ?y - tile)
  (down ?x - tile ?y - tile)
  (right ?x - tile ?y - tile)
  (left ?x - tile ?y - tile)
  (clear ?x - tile)
  (painted ?x - tile ?c - color)
  (robot-has ?r - robot ?c - color)
  (availableColor ?c - color)
  (free-color ?r - robot))

(:durative-action change-color
  :parameters (?r - robot ?c - color ?c2 - color)
  :duration (= ?duration 5)
  :condition (and (at start (robot-has ?r ?c))
                  (over all (availableColor ?c2)))
  :effect (and (at start (not (robot-has ?r ?c)))
               (at end (robot-has ?r ?c2))))

(:durative-action paintUp
  :parameters (?r - robot ?y - tile ?x - tile ?c - color)
  :duration (= ?duration 2)
  :condition (and (over all (robot-has ?r ?c))
                  (at start (robotAt ?r ?x))
                  (over all (up ?y ?x))
                  (at start (clear ?y)))
  :effect (and (at start (not (clear ?y)))
               (at end (painted ?y ?c))))

(:durative-action paint-down
  :parameters (?r - robot ?y - tile ?x - tile ?c - color)
  :duration (= ?duration 2)
  :condition (and (over all (robot-has ?r ?c))
                  (at start (robotAt ?r ?x))
                  (over all (down ?y ?x))
                  (at start (clear ?y)))
  :effect (and (at start (not (clear ?y)))
               (at end (painted ?y ?c))))

(:durative-action up
  :parameters (?r - robot ?x - tile ?y - tile)
  :duration (= ?duration 3)
  :condition (and (at start (robotAt ?r ?x))
                  (over all (up ?y ?x))
                  (at start (clear ?y)))
  :effect (and
               (at start (not (robotAt ?r ?x)))
               (at end (robotAt ?r ?y))
               (at start (not (clear ?y)))
               (at end (clear ?x))))
(:durative-action down
  :parameters (?r - robot ?x - tile ?y - tile)
  :duration (= ?duration 1)
  :condition (and (at start (robotAt ?r ?x))
                  (over all (down ?y ?x))
                  (at start (clear ?y)))
  :effect (and (at start (not (robotAt ?r ?x)))
               (at end (robotAt ?r ?y))
               (at start (not (clear ?y)))
               (at end (clear ?x))))

(:durative-action right
  :parameters (?r - robot ?x - tile ?y - tile)
  :duration (= ?duration 1)
  :condition (and (at start (robotAt ?r ?x))
                  (over all (right ?y ?x))
                  (at start (clear ?y)))
  :effect (and (at start (not (robotAt ?r ?x)))
               (at end (robotAt ?r ?y))
               (at start (not (clear ?y)))
               (at end (clear ?x))))

(:durative-action left
  :parameters (?r - robot ?x - tile ?y - tile)
  :duration (= ?duration 1)
  :condition (and (at start (robotAt ?r ?x))
                  (over all (left ?y ?x))
                  (at start (clear ?y)))
  :effect (and (at start (not (robotAt ?r ?x)))
               (at end (robotAt ?r ?y))
               (at start (not (clear ?y)))
               (at end (clear ?x))))
)
```

Appendix B: PDDL2.1 Specification of the *Depot* Domain

```
(define (domain Depot)
(:requirements :typing :durative-actions)
(:types place locatable - object
        depot distributor - place
        truck hoist surface - locatable
        pallet crate - surface)

(:predicates (at ?x - locatable ?y - place)
             (on ?x - crate ?y - surface)
             (in ?x - crate ?y - truck)
             (lifting ?x - hoist ?y - crate)
             (available ?x - hoist)
             (clear ?x - surface))

(:durative-action Drive
:parameters (?x - truck ?y - place ?z - place)
:duration (= ?duration 10)
:condition (and (at start (at ?x ?y)))
:effect (and (at start (not (at ?x ?y))) (at end (at ?x ?z))))
(:durative-action Lift
:parameters (?x - hoist ?y - crate ?z - surface ?p - place)
:duration (= ?duration 1)
:condition (and (over all (at ?x ?p)) (at start (available ?x))
                (at start (at ?y ?p)) (at start (on ?y ?z))
                (at start (clear ?y)))
:effect (and (at start (not (at ?y ?p))) (at start (lifting ?x ?y))
             (at start (not (clear ?y)))(at start (not (available ?x)))
             (at start (clear ?z)) (at start (not (on ?y ?z)))))

(:durative-action Drop
:parameters (?x - hoist ?y - crate ?z - surface ?p - place)
:duration (= ?duration 1)
:condition (and (over all (at ?x ?p)) (over all (at ?z ?p))
                (over all (clear ?z)) (over all (lifting ?x ?y)))
:effect (and (at end (available ?x)) (at end (not (lifting ?x ?y)))
             (at end (at ?y ?p)) (at end (not (clear ?z)))
             (at end (clear ?y))(at end (on ?y ?z))))

(:durative-action Load
:parameters (?x - hoist ?y - crate ?z - truck ?p - place)
:duration (= ?duration 3)
:condition (and (over all (at ?x ?p)) (over all (at ?z ?p))
                (over all (lifting ?x ?y)))
:effect (and (at end (not (lifting ?x ?y))) (at end (in ?y ?z))
             (at end (available ?x))))

(:durative-action Unload
:parameters (?x - hoist ?y - crate ?z - truck ?p - place)
:duration (= ?duration 4)
:condition (and (over all (at ?x ?p)) (over all (at ?z ?p))
                (at start (available ?x)) (at start (in ?y ?z)))
:effect (and (at start (not (in ?y ?z))) (at start (not (available ?x)))
             (at start (lifting ?x ?y))))

)
```

Appendix C: PDDL2.1 Specification of the *Rovers* Domain

```
(define (domain Rover)
(:requirements :typing :durative-actions)
(:types rover waypoint store camera mode lander objective)

(:predicates (at ?x - rover ?y - waypoint)
             (at_lander ?x - lander ?y - waypoint)
             (can_traverse ?r - rover ?x - waypoint ?y - waypoint)
             (equipped_for_soil_analysis ?r - rover)
             (equipped_for_rock_analysis ?r - rover)
             (equipped_for_imaging ?r - rover)
             (empty ?s - store)
             (have_rock_analysis ?r - rover ?w - waypoint)
             (have_soil_analysis ?r - rover ?w - waypoint)
             (full ?s - store)
             (calibrated ?c - camera ?r - rover)
             (supports ?c - camera ?m - mode)
             (available ?r - rover)
             (visible ?w - waypoint ?p - waypoint)
             (have_image ?r - rover ?o - objective ?m - mode)
             (communicated_soil_data ?w - waypoint)
             (communicated_rock_data ?w - waypoint)
             (communicated_image_data ?o - objective ?m - mode)
             (at_soil_sample ?w - waypoint)
             (at_rock_sample ?w - waypoint)
             (visible_from ?o - objective ?w - waypoint)
             (store_of ?s - store ?r - rover)
             (calibration_target ?i - camera ?o - objective)
             (on_board ?i - camera ?r - rover)
             (channel_free ?l - lander))

(:durative-action navigate
:parameters (?x - rover ?y - waypoint ?z - waypoint)
:duration (= ?duration 5)
:condition (and (over all (can_traverse ?x ?y ?z)) (at start (available ?x))
              (at start (at ?x ?y)) (over all (visible ?y ?z)))
:effect (and (at start (not (at ?x ?y))) (at end (at ?x ?z))))

(:durative-action sample_soil
:parameters (?x - rover ?s - store ?p - waypoint)
:duration (= ?duration 10)
:condition (and (over all (at ?x ?p)) (at start (at ?x ?p))
                (at start (at_soil_sample ?p))
                (at start (equipped_for_soil_analysis ?x))
                (at start (store_of ?s ?x)) (at start (empty ?s)))
:effect (and (at start (not (empty ?s))) (at end (full ?s))
             (at end (have_soil_analysis ?x ?p))
             (at end (not (at_soil_sample ?p)))))

(:durative-action sample_rock
:parameters (?x - rover ?s - store ?p - waypoint)
:duration (= ?duration 8)
:condition (and (over all (at ?x ?p)) (at start (at ?x ?p))
                (at start (at_rock_sample ?p))
                (at start (equipped_for_rock_analysis ?x))
                (at start (store_of ?s ?x))
                (at start (empty ?s)))
:effect (and (at start (not (empty ?s))) (at end (full ?s))
             (at end (have_rock_analysis ?x ?p))
             (at end (not (at_rock_sample ?p)))))

(:durative-action drop
:parameters (?x - rover ?s - store)
:duration (= ?duration 1)
:condition (and (at start (store_of ?s ?x)) (at start (full ?s)))
:effect (and (at end (not (full ?s))) (at end (empty ?s))))

(:durative-action calibrate
 :parameters (?r - rover ?i - camera ?t - objective ?w - waypoint)
 :duration (= ?duration 5)
 :condition (and (at start (equipped_for_imaging ?r))
                 (at start (calibration_target ?i ?t)) (over all (at ?r ?w))
                 (at start (visible_from ?t ?w)) (at start (on_board ?i ?r)))
 :effect (at end (calibrated ?i ?r)) )

(:durative-action take_image
 :parameters (?r - rover ?p - waypoint ?o - objective ?i - camera ?m - mode)
 :duration (= ?duration 7)
 :condition (and (over all (calibrated ?i ?r))
                 (at start (on_board ?i ?r))
                 (over all (equipped_for_imaging ?r))
                 (over all (supports ?i ?m) )
                 (over all (visible_from ?o ?p))
                 (over all (at ?r ?p)))
 :effect (and (at end (have_image ?r ?o ?m)) (at end (not (calibrated ?i ?r)))))

(:durative-action communicate_soil_data
 :parameters (?r - rover ?l - lander ?p - waypoint ?x - waypoint ?y - waypoint)
 :duration (= ?duration 10)
 :condition (and (over all (at ?r ?x)) (over all (at_lander ?l ?y))
                 (at start (have_soil_analysis ?r ?p))
                 (at start (visible ?x ?y)) (at start (available ?r))
                 (at start (channel_free ?l)))
 :effect (and (at start (not (available ?r))) (at start (not (channel_free ?l)))
              (at end (channel_free ?l))
              (at end (communicated_soil_data ?p))(at end (available ?r))))

(:durative-action communicate_rock_data
 :parameters (?r - rover ?l - lander ?p - waypoint ?x - waypoint ?y - waypoint)
 :duration (= ?duration 10)
 :condition (and (over all (at ?r ?x)) (over all (at_lander ?l ?y))
                 (at start (have_rock_analysis ?r ?p))
                 (at start (visible ?x ?y)) (at start (available ?r))
                 (at start (channel_free ?l)))
 :effect (and (at start (not (available ?r))) (at start (not (channel_free ?l)))
              (at end (channel_free ?l))
              (at end (communicated_rock_data ?p))(at end (available ?r))))

(:durative-action communicate_image_data
 :parameters (?r - rover ?l - lander ?o - objective ?m - mode ?x - waypoint
              ?y - waypoint)
 :duration (= ?duration 15)
 :condition (and (over all (at ?r ?x)) (over all (at_lander ?l ?y))
                 (at start (have_image ?r ?o ?m))
                 (at start (visible ?x ?y)) (at start (available ?r))
                 (at start (channel_free ?l)))
 :effect (and (at start (not (available ?r))) (at start (not (channel_free ?l)))
              (at end (channel_free ?l))
              (at end (communicated_image_data ?o ?m)) (at end (available ?r)))))
```

Appendix D: PDDL2.1 Specification of the *DriverLog* Domain

```
(define (domain driverlog)
  (:requirements :typing :durative-actions)
  (:types location locatable - object
                      driver truck obj - locatable)
  (:predicates
                (at ?obj - locatable ?loc - location)
                (in ?obj1 - obj ?obj - truck)
                (driving ?d - driver ?v - truck)
                (link ?x ?y - location) (path ?x ?y - location)
                (empty ?v - truck))
(:durative-action LOAD-TRUCK
  :parameters
   (?obj - obj ?truck - truck ?loc - location)
  :duration (= ?duration 2)
  :condition (and (over all (at ?truck ?loc)) (at start (at ?obj ?loc)))
  :effect (and (at start (not (at ?obj ?loc))) (at end (in ?obj ?truck))))

(:durative-action UNLOAD-TRUCK
  :parameters
   (?obj - obj ?truck - truck ?loc - location)
  :duration (= ?duration 2)
  :condition (and (over all (at ?truck ?loc)) (at start (in ?obj ?truck)))
  :effect (and (at start (not (in ?obj ?truck))) (at end (at ?obj ?loc))))

(:durative-action BOARD-TRUCK
  :parameters
   (?driver - driver ?truck - truck ?loc - location)
  :duration (= ?duration 1)
  :condition (and (over all (at ?truck ?loc)) (at start (at ?driver ?loc))
             (at start (empty ?truck)))
  :effect (and (at start (not (at ?driver ?loc)))
          (at end (driving ?driver ?truck)) (at start (not (empty ?truck)))))

(:durative-action DISEMBARK-TRUCK
  :parameters
   (?driver - driver ?truck - truck ?loc - location)
  :duration (= ?duration 1)
  :condition (and (over all (at ?truck ?loc))
                  (at start (driving ?driver ?truck)))
  :effect (and (at start (not (driving ?driver ?truck)))
               (at end (at ?driver ?loc)) (at end (empty ?truck))))

(:durative-action DRIVE-TRUCK
  :parameters
   (?truck - truck ?loc-from - location ?loc-to - location ?driver - driver)
  :duration (= ?duration 10)
  :condition (and (at start (at ?truck ?loc-from))
                  (over all (driving ?driver ?truck))
                  (at start (link ?loc-from ?loc-to)))
  :effect (and (at start (not (at ?truck ?loc-from)))
               (at end (at ?truck ?loc-to))))

(:durative-action WALK
  :parameters
   (?driver - driver ?loc-from - location ?loc-to - location)
  :duration (= ?duration 20)
  :condition (and (at start (at ?driver ?loc-from))
                  (at start (path ?loc-from ?loc-to)))
  :effect (and (at start (not (at ?driver ?loc-from)))
               (at end (at ?driver ?loc-to)))))
```

Acknowledgements

We thank Malte Helmert, Gabriele Roger and Jussi Rintanen for making their code for synthesising invariants available, William Cushing for helpful discussions about the configurations of temporal actions and Maria Fox and Derek Long for insightful discussions on the semantics of PDDL2.1. This work has been supported by Royal Holloway University of London, Politecnico di Torino and the NASA Exploration Systems Program.

References

Barnes, J. M., Pandey, A., Garlan, D., 2013. Automated planning for software architecture evolution. In: 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 213–223.

Bernardini, S., Smith, D. E., 2007. Developing domain-independent search control for EUROPA2. In: Proc. of the Workshop on Heuristics for Domain-independent Planning: Progress, Ideas, Limitations, Challanges, 17th International Conference on Automated Planning and Scheduling (ICAPS'07).

Bernardini, S., Smith, D. E., 2008a. Automatically generated heuristic guidance for EUROPA2. In: Proc. of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (iSAIRAS'08).

Bernardini, S., Smith, D. E., 2008b. Translating pddl2.2. into a constraint-based variable/value language. In: Proc. of the Workshop on Knowledge Engineering for Planning and Scheduling, 18th International Conference on Automated Planning and Scheduling (ICAPS'08).

Bernardini, S., Smith, D. E., 2011a. Automatic synthesis of temporal invariants. In: Proc. of the Ninth Symposium on Abstraction, Reformulation and Approximation (SARA-11). Parador de Cardona, Spain.

Bernardini, S., Smith, D. E., 2011b. Finding mutual exclusion invariants in temporal planning domains. In: Proc. of the Seventh International Workshop on Planning and Scheduling for Space (IWPSS-11). Darmstadt, Germany.

Blum, A., Furst, M., 1997. Fast planning through planning graph analysis. Artificial Intelligence 90, 281–300.

Bonet, B., Geffner, H., 2001. Planning as Heuristic Search. Artificial Intelligence 129 (1-2), special issue on Heuristic Search.

Chen, Y., Huang, R., Xing, Z., Zhang, W., 2009. Long-distance mutual exclusion for planning. Artificial Intelligence 173 (2), 365 – 391.

Chien, S., Rabideau, G., Knight, R., Sherwood, R., Engelhardt, B., Mutz, D., Estlin, T., B.Smith, Fisher, F., Barret, T., Stebbins, G., Tran, D., 2000. ASPEN - Automated Planning and Scheduling for Space Missions Operations. In: 6th International Conference on Space Operations.

Coles, A. J., Coles, A. I., Fox, M., Long, D., 2010. Forward-Chaining Partial-Order Planning. In: Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10).

Cushing, W., Weld, D., Kambhampati, S., Mausam, Talamadupula, K., 2007. Evaluating Temporal Planning Domains. In: Proc. of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS-07). pp. 105–112.

Do, M. B., Kambhampati, S., 2001. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. Journal of Artificial Intelligence Research 132, 151–182.

Edelkamp, S., Helmert, M., 1999. Exhibiting Knowledge in Planning Problems to Minimize State Encoding Length. In: Proc. of the Fifth European Conference on Planning (ECP'99). pp. 135–147.

Edelkamp, S., Helmert, M., 2001. The model checking integrated planning system (MIPS). AI Magazine 22 (3), 67–71.

Edelkamp, S., Hoffmann, J., 2004. PDDL2.2: The language for the classical part of the 4th International Planning Competition. Tech. Rep. 195, Albert-Ludwigs-Universität Freiburg.

Eyerich, P., Mattmller, R., Rger, G., 2009. Using the Context-enhanced Additive Heuristic for Temporal and Numeric Planning. In: Proc. of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS-09). pp. 49–64.

Fikes, R., Nilsson, N., 1971. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2(3-4), 189–208.

Fox, M., Long, D., 1998. The automatic inference of state invariants in TIM. Journal of Artificial Intelligence Research 9, 367421.

Fox, M., Long, D., 2003. PDDL 2.1: An extension to PDDL for expressing temporal planning domains. Journal of Artificial Intelligence Research 20, 61–124.

Fox, M., Long, D., 2011. Efficient implementation of the plan graph in STAN. jair 10, 87115.

Frank, J., Jónsson, A., 2003. Constraint Based Attribute and Interval Planning. Journal of Constraints 8 (4), 339–364, special Issue on Planning.

Fratini, S., Pecora, F., Cesta, A., 2008. Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. Archives of Control Sciences 18 (2), 5–45.

Gerevini, A., Saetti, A., I., S., 2006. An approach to temporal planning and scheduling in domains with predictable exogenous events. Journal of Artificial Intelligence Research 25, 187–231.

Gerevini, A., Schubert, L., 1998. Inferring state constraints for domain-independent planning. In: Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI-98). pp. 905–912.

Gerevini, A., Schubert, L., 2000. Discovering state constraints in discoplan: Some new results. In: In Proc. of the 17th National Conference on Artificial Intelligence (AAAI-2000). pp. 761–767.

Ghallab, M., Laruelle, H., 1994. Representation and Control in IxTeT, a Temporal Planner. In: Proc. of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94). AAAI Press, pp. 61–67.

Ghosh, N., Ghosh, S. K., 2010. An Intelligent Approach for Security Management of an Enterprise Network Using Planner. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 187–214.

Golden, K., 2003. A Domain Description Language for Data Processing . In: Proceedings of the ICAPS'03 Workshop on PDDL.

Haslum, P., Botea, A., Helmert, M., Bonet, B., Koenig, S., 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In: Proc. of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07).

Helmert, M., 2006. The Fast Downward planning system. Journal of Artificial Intelligence Research 26, 191–246.

Helmert, M., 2009. Concise finite-domain representations for PDDL planning tasks. Artificial Intelligence 3 (17), 503–535.

Helmert, M., Geffner, H., 2008. Unifying the Causal Graph and Additive Heuristics. In: Proc. of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS-08). pp. 140–147.

Hoffmann, J., Nebel, B., 2001. The FF planning system: fast plan generation through heuristic search. Journal of Artificial Intelligence Research 14, 253–302.

Howey, R., Long, D., Fox, M., 2004. Val: automatic plan validation, continuous effects and mixed initiative planning using pddl. In: 16th IEEE International Conference on Tools with Artificial Intelligence. pp. 294–301.

Huang, R., Chen, Y., Zhang, W., 2010. A novel transition based encoding scheme for planning as satisfiability. In: Proc. of the Twenty-Forth National Conference on Artificial Intelligence (AAAI-10). Vol. 2. AAAI Press, pp. 89–94.

Kautz, H., Selman, B., 1999. Unifying SAT-based and Graph-based Planning. In: Proc. of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99). Morgan Kaufmann Publishers Inc., pp. 318–325.

Liu, Z., Ranganathan, A., Riabov, A., 2007. A planning approach for message-oriented semantic web service composition. In: Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2. AAAI'07. AAAI Press, pp. 1389–1394.

McDermott, D., 2000. The 1998 AI Planning Systems Competition. AI Magazine 21 (2), 35–55.

Muscettola, N., 1994. HSTS: Integrating Planning and Scheduling. In: Zweben, M., Fox, M. (Eds.), Intelligent Scheduling. Morgan Kauffmann, pp. 451–469.

Pednault, E., 1986. Toward a mathematical theory of plan synthesis. Ph.D. thesis, Stanford University, Department of Electrical Engineering.

Richter, S., Westphal, M., Sep. 2010. The LAMA Planner: Guiding Cost-based Anytime Planning with Landmarks. Journal of Artificial Intelligence Research 39 (1), 127–177.

Rintanen, J., 2000. An Iterative Algorithm for Synthesizing Invariants. In: Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI-00). pp. 806–811.

Rintanen, J., 2008. Regression for classical and nondeterministic planning. In: Proc. of the 18th European Conference on Artificial Intelligence (ECAI-08).

Rintanen, J., 2014. Constraint-Based Algorithm for Computing Temporal Invariants. In: Proc. of the European Conference on Logic in Artificial Intelligence,.

Rintanen, J., 2017. Schematic invariants by reduction to ground invariants. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17). AAAI Press, pp. 3644–3650.

Vidal, V., Jun. 2004. The YAHSP Planning System: Forward Heuristic Search with Lookahead Plans Analysis. In: Proceedings of the 4th International Planning Competition (IPC-2004). Whistler, BC, Canada, pp. 59–60.