Abstracting Abstraction in Search II: Complexity Analysis

Christer Bäckström and **Peter Jonsson**
Department of Computer Science, Linköping University
SE-581 83 Linköping, Sweden
christer.backstrom@liu.se   peter.jonsson@liu.se

Abstract

Modelling abstraction as a function from the original state space to an abstract state space is a common approach in combinatorial search. Sometimes this is too restricted, though, and we have previously proposed a framework using a more flexible concept of transformations between labelled graphs. We also proposed a number of properties to describe and classify such transformations. This framework enabled the modelling of a number of different abstraction methods in a way that facilitated comparative analyses. It is of particular interest that these properties can be used to capture the concept of refinement without backtracking between levels; how to do this has been an open question for at least twenty years. In this paper, we continue our previous research by analysing the complexity of testing the various transformation properties for both explicit and implicit graph representations.

1 Introduction

Abstraction in combinatorial search is often modelled as a function f from the vertices of one graph to the vertices of another graph, the latter graph being the abstraction of the first. It is also common that the function is a homomorphism. This approach is very natural and has proven useful in many cases, cf. Holte et al. (1996), Helmert, Haslum, and Hoffmann (2007) and Zilles and Holte (2010). It is not always sufficient, however; one such case is abstraction in planning. Motivated by the desire to model and compare various abstraction methods for planning, we have earlier (Bäckström and Jonsson 2012) defined a more flexible abstraction framework based on *transformations* between labelled graphs. A transformation is a pair $\langle f, R \rangle$, where f is a *transformation function* and R is a *label relation*. The function f maps vertices in the original graph to sets of vertices in the abstract graph and the relation R specifies how labels in the two graphs are related, which implicitly specifies how subsets of arcs in the graphs are related. While the transformation concept can capture many abstraction methods, it is a more general concept which is not specifically tailored to abstraction. In order to classify, analyse and compare transformations we also introduced a number of transformation properties.

In our earlier publication (Bäckström and Jonsson 2012) we have demonstrated the usefulness of this framework in various ways. We have shown that a certain combination of our transformation properties exactly captures the **DPP** concept by Zilles and Holte (2010). A related concept is that of path/plan refinement without backtracking to the abstract level. Partial solutions to capturing this concept have been presented in the literature, for instance, the ordered monotonicity criterion by Knoblock, Tenenberg, and Yang (1991), the downward refinement property (**DRP**) by Bacchus and Yang (1994), and the simulation-based approach by Bundy et al. (1996). However, how to define general conditions that capture this concept has remained an open question in the literature until our previous paper. There we proved that different combinations of our transformation properties can be used to capture this concept, and we can even distinguish several different degrees of the concept. Furthermore, we reformulated five different abstraction methods in planning as transformations. This enabled us to analyse and compare the methods in new ways by determining their inherent transformation properties.

The reader is strongly recommended to also read our previous paper (Bäckström and Jonsson 2012) in connection with this one, since that paper focused on presenting the framework with motivating examples on how to use it, as well as proving various theorems about it. This paper must be viewed as a continuation of that work, focusing only on the computational complexity of deciding which properties a transformation has; certain aspects of the framework might seem unmotivated if reading this paper alone. Sections 2 to 5 presents the framework used in our previous paper, but omits all proofs and some of the discussions. All major definitions and results about the framework itself remain. The new results are the complexity analyses in Sections 6 to 8. The paper ends with a discussion of these results in Section 9.

2 STGs and STG Transformations

We first introduce our framework for studying abstractions. Although the definitions may appear somewhat complex and difficult to understand at first sight, there is a reason: we want to *prove* results, not merely devote ourselves to discussions. We begin by defining some general notation and concepts, then we introduce state transition graphs and our transformation concept.

If X is a set, then $|X|$ denotes the cardinality of X. A *partition* of a set X is a set P of non-empty subsets of X such that (1) $\cup_{p \in P} p = X$ and (2) for all $p, q \in P$, if $p \neq q$, then $p \cap q = \varnothing$. Let $f : X \to Y$ be a function, then $Rng(f) = \{f(x) \mid x \in X\}$ is the *range* of f. Often, a function f will be from X to 2^Y (for some sets X and Y). In this case, $Rng(f) \subseteq 2^Y$, that is, the value of f is a subset of Y, not an element in Y. For such functions we also define $f(Z) = \cup_{x \in Z} f(x)$ for all $Z \subseteq X$.

Definition 1. A *state transition graph (STG)* over a set L of labels is a tuple $\mathbb{G} = \langle S, E \rangle$ where S is a set of vertices and $E \subseteq S \times S \times L$ is a set of labelled arcs. The set of labels in $\mathbb{G}$ is implicitly defined as $L(\mathbb{G}) = L(E) = \{\ell \mid \langle s, t, \ell \rangle \in E\}$. A sequence $s_0, \ell_1, s_1, \ell_2, \ldots, \ell_k, s_k$ of states in S and labels in $L(E)$ is a *(state) path* in $\mathbb{G}$ if either (1) $k = 0$ or (2) $\langle s_{i-1}, s_i, \ell_i \rangle \in E$ for $1 \leq i \leq k$.

The set S is called a *state space* and its members *states*. More than one arc in the same direction between two states is allowed, as long as the arcs have different labels. The intention of the labels is to provide a means to identify a subset of arcs by assigning a particular label to these arcs. This is useful, for instance, in planning where a single action may induce many arcs in an STG. If all arcs have the same label, which is allowed, then the STG collapses to an ordinary directed graph. Arcs may be written as $\langle s, t \rangle$ and paths as $s_0, \ldots, s_k$ if the exact labels are not relevant.

Definition 2. Let $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ and $\mathbb{G}_2 = \langle S_2, E_2 \rangle$ be two STGs. A total function $f : S_1 \to 2^{S_2}$ is a *transformation function* from $\mathbb{G}_1$ to $\mathbb{G}_2$ if $Rng(f)$ is a partition of S_2. A *label relation* from $\mathbb{G}_1$ to $\mathbb{G}_2$ is a binary relation $R \subseteq L(\mathbb{G}_1) \times L(\mathbb{G}_2)$. An *(STG) transformation* from $\mathbb{G}_1$ to $\mathbb{G}_2$ is a pair $\tau = \langle f, R \rangle$ where f is a transformation function from $\mathbb{G}_1$ to $\mathbb{G}_2$ and R is a label relation from $\mathbb{G}_1$ to $\mathbb{G}_2$.

The transformation function f specifies how the transformation maps states from one STG to the other while the label relation R provides additional information about how sets of arcs are related between the two STGs. Note that f is formally a function from S_1 to 2^{S_2}, that is, it has a subset of S_2 as value. We use a function rather than a relation since it makes the theory clearer and simpler and is more in line with previous work in the area.

Example 3. Consider two STGs: $\mathbb{G}_1 : 00 \xrightarrow{a} 01 \xrightarrow{b} 10 \xrightarrow{a} 11$ and $\mathbb{G}_2 : 0 \xrightarrow{c} 1$. Also define $f_1 : \mathbb{G}_1 \to \mathbb{G}_2$ such that $f_1(xy) = \{x\}$. We see immediately that f_1 is a transformation function from $\mathbb{G}_1$ to $\mathbb{G}_2$. Define $f_2 : \mathbb{G}_1 \to \mathbb{G}_2$ such that $f_2(xy) = \{x, y\}$; this function is not a transformation function since $f_2(00) = \{0\}$ and $f_2(01) = \{0, 1\}$ which implies that $Rng(f_2)$ does not partition $\mathbb{G}_2$. Finally, the function $f_3(xy) = \{2x + y, 7 - 2x - y\}$ is a transformation function from $\mathbb{G}_1$ to $\mathbb{G}_3 = \langle \{0, \ldots, 7\}, \{\langle x, y, d \rangle \mid x \neq y\} \rangle$ since $Rng(f_3)$ partitions $\{0, \ldots, 7\}$ into $\{\{0, 7\}, \{1, 6\}, \{2, 5\}, \{3, 4\}\}$. The functions f_1 and f_3 are illustrated in Figure 1.

A high degree of symmetry is inherent in our transformation concept. It is, in fact, only a conceptual choice to say that one STG is the transformation from another and not the other way around. This symmetry simplifies our exposition
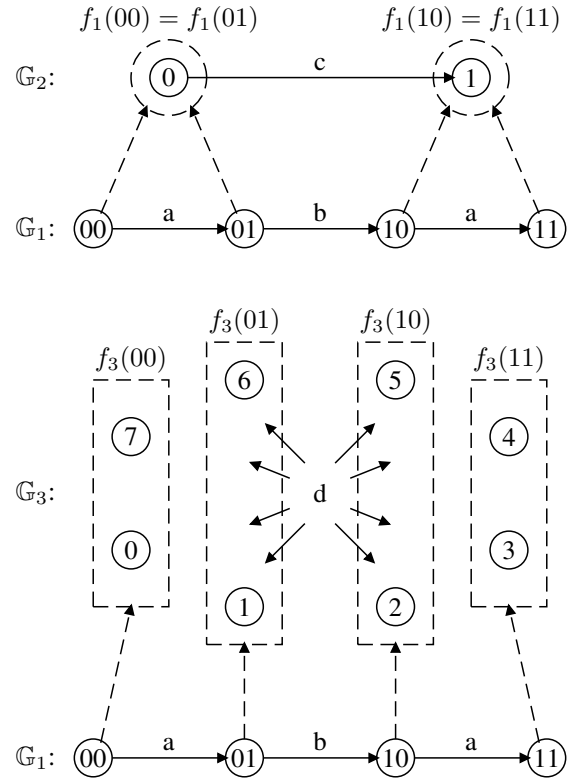

Figure 1: The functions f_1 and f_3 in Example 3.

considerably, but it does not automatically carry over also to all types of properties of transformations; some care must be exercised in such cases, as we will see examples of later.

Definition 4. Let $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ and $\mathbb{G}_2 = \langle S_2, E_2 \rangle$ be two STGs, let f be a transformation function from $\mathbb{G}_1$ to $\mathbb{G}_2$ and let R be a label relation from $\mathbb{G}_1$ to $\mathbb{G}_2$. Then, the *reverse transformation function* $\overline{f} : S_2 \to 2^{S_1}$ is defined as $\overline{f}(t) = \{s \in S_1 \mid t \in f(s)\}$ and the *reverse label relation* $\overline{R} \subseteq L(\mathbb{G}_2) \times L(\mathbb{G}_1)$ is defined as $\overline{R}(\ell_2, \ell_1)$ iff $R(\ell_1, \ell_2)$.

Consider the functions f_1 and f_3 from Example 3 once again. We see that $\overline{f_1}(0) = \{00, 01\}$ and $\overline{f_1}(1) = \{10, 11\}$, while $\overline{f_3}(0) = \overline{f_3}(7) = \{00\}$, $\overline{f_3}(1) = \overline{f_3}(6) = \{01\}$, $\overline{f_3}(2) = \overline{f_3}(5) = \{10\}$ and $\overline{f_3}(3) = \overline{f_3}(4) = \{11\}$.

Lemma 5. *Let f be a transformation function from an STG $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ to an STG $\mathbb{G}_2 = \langle S_2, E_2 \rangle$. Then:*

1) for all $s_1 \in S_1$, $s_2 \in S_2$, $s_1 \in \overline{f}(s_2)$ iff $s_2 \in f(s_1)$.

2) $\overline{f}$ is a transformation function from $\mathbb{G}_2$ to $\mathbb{G}_1$.

3) If $\langle f, R \rangle$ is a transformation from $\mathbb{G}_1$ to $\mathbb{G}_2$, then $\langle \overline{f}, \overline{R} \rangle$ is a transformation from $\mathbb{G}_2$ to $\mathbb{G}_1$.

3 Path Refinement

If we want to find a path in an STG by abstraction, then we must transform this STG into an abstract STG and find a path in the latter. We must then somehow refine this abstract path into a path in the original STG. Preferably, we want to do this without backtracking to the abstract level. We define

three different kinds of path refinements that achieve this, with varying degrees of practical usefulness.

Definition 6. Let $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ and $\mathbb{G}_2 = \langle S_2, E_2 \rangle$ be two STGs and let f be a transformation function from $\mathbb{G}_1$ to $\mathbb{G}_2$. Let $\sigma = t_0, t_1, \ldots, t_k$ be an arbitrary path in $\mathbb{G}_2$. Then:

1) σ is *trivially downward state refinable* if there are two states $s_0 \in \overline{f}(t_0)$ and $s_\ell \in \overline{f}(t_k)$ s.t. there is a path in $\mathbb{G}_1$ from s_0 to s_ℓ.

2) σ is *weakly downward state refinable* if there is a sequence $s_0, s_1, \ldots, s_k$ of states in S_1 such that $s_i \in \overline{f}(t_i)$ for all i s.t. $0 \leq i \leq k$ and there is a path from s_{i-1} to s_i in $\mathbb{G}_1$ for all i $(1 \leq i \leq k)$.

3) σ is *strongly downward state refinable* if for every i s.t. $1 \leq i \leq k$, there is a path from s_{i-1} to s_i in $\mathbb{G}_1$ for all $s_{i-1} \in \overline{f}(t_{i-1})$ and all $s_i \in \overline{f}(t_i)$.

Trivial path refinement only requires that if there is a path between two states in the abstract graph, then there is a path between two corresponding states in the original graph. The two paths need not have any other connection at all. The other two refinements tie the two paths to each other in such a way that the states along the abstract path are useful for finding the ground path. Earlier attempts to capture refinement without backtracking to higher levels appear in the literature, eg. Knoblock, Tenenberg, and Yang (1991) and Bacchus and Yang (1994), but these are more limited and inherently tied to specific abstraction methods.

4 Properties of Transformations

As a tool to analyse and classify transformations in a general way, we define the following properties.

Definition 7. Let $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ and $\mathbb{G}_2 = \langle S_2, E_2 \rangle$ be two STGs and let $\tau = \langle f, R \rangle$ be a transformation from $\mathbb{G}_1$ to $\mathbb{G}_2$. Then τ can have the following *method properties*:

$\mathbf{M}_\uparrow$: $|f(s)| = 1$ for all $s \in S_1$.

$\mathbf{M}_\downarrow$: $|\overline{f}(s)| = 1$ for all $s \in S_2$.

$\mathbf{R}_\uparrow$: If $\langle s_1, t_1, \ell_1 \rangle \in E_1$, then there is some $\langle s_2, t_2, \ell_2 \rangle \in E_2$ such that $R(\ell_1, \ell_2)$.

$\mathbf{R}_\downarrow$: If $\langle s_2, t_2, \ell_2 \rangle \in E_2$, then there is some $\langle s_1, t_1, \ell_1 \rangle \in E_1$ such that $R(\ell_1, \ell_2)$.

$\mathbf{C}_\uparrow$: If $R(\ell_1, \ell_2)$ and $\langle s_1, t_1, \ell_1 \rangle \in E_1$, then there is some $\langle s_2, t_2, \ell_2 \rangle \in E_2$ such that $s_2 \in f(s_1)$ and $t_2 \in f(t_1)$.

$\mathbf{C}_\downarrow$: If $R(\ell_1, \ell_2)$ and $\langle s_2, t_2, \ell_2 \rangle \in E_2$, then there is some $\langle s_1, t_1, \ell_1 \rangle \in E_1$ such that $s_1 \in \overline{f}(s_2)$ and $t_1 \in \overline{f}(t_2)$.

Properties $\mathbf{M}_\uparrow$/$\mathbf{M}_\downarrow$ (*upwards/downwards many-one*) depend only on f and may thus hold also for f itself. The intention of $\mathbf{M}_\uparrow$ is to say that f maps every state in $\mathbb{G}_1$ to a single state in $\mathbb{G}_2$. We often write $f(s) = t$ instead of $t \in f(s)$ when f is $\mathbf{M}_\uparrow$ and analogously for $\overline{f}$. Properties $\mathbf{R}_\uparrow$/$\mathbf{R}_\downarrow$ (*upwards/downwards related*) depend only on R and may thus hold also for R itself. The intention behind $\mathbf{R}_\uparrow$ is that if there is a non-empty set of arcs in $\mathbb{G}_1$ with a specific label, then there is at least one arc in $\mathbb{G}_2$ that is explicitly specified via R to correspond to this arc set. Properties $\mathbf{C}_\uparrow$/$\mathbf{C}_\downarrow$ (*upwards/downwards coupled*) describe the connection between f and R. The intention behind $\mathbf{C}_\uparrow$ is to provide a way

to tie up f and R to each other and require that arcs that are related via R must go between states that are related via f. We use a double-headed arrow when a condition holds both upward and downward. For instance, $\mathbf{C}_\updownarrow$ (*up-down coupled*) means that both $\mathbf{C}_\uparrow$ and $\mathbf{C}_\downarrow$ hold. These classifications retain the symmetric nature of transformations. For instance, $\langle f, R \rangle$ is a $\mathbf{C}_\downarrow$ transformation from $\mathbb{G}_1$ to $\mathbb{G}_2$ if and only if $\langle \overline{f}, \overline{R} \rangle$ is an $\mathbf{C}_\uparrow$ transformation from $\mathbb{G}_2$ to $\mathbb{G}_1$.

Example 8. We reconsider Example 3. The function $f_1 : \mathbb{G}_1 \to \mathbb{G}_2$ is $\mathbf{M}_\uparrow$ but is not $\mathbf{M}_\downarrow$ while function $f_3 : \mathbb{G}_1 \to \mathbb{G}_3$ is $\mathbf{M}_\downarrow$ but not $\mathbf{M}_\uparrow$. Define $R = \{a, b\} \times \{c\}$ and note that the transformation $\langle f_1, R \rangle : \mathbb{G}_1 \to \mathbb{G}_2$ has both property $\mathbf{R}_\uparrow$ and $\mathbf{R}_\downarrow$. Furthermore, $\langle f_1, R \rangle$ is neither $\mathbf{C}_\downarrow$ nor $\mathbf{C}_\uparrow$. One may also note that if $R' = \{a, b\} \times \{d\}$ then $\langle f_3, R' \rangle$ is $\mathbf{C}_\uparrow$ but not $\mathbf{C}_\downarrow$

As a further example, the homomorphism-based abstraction concept used by Zilles and Holte (2010) and others, is captured by the class of $\mathbf{M}_\uparrow \mathbf{R}_\updownarrow \mathbf{C}_\downarrow$ transformation functions.

In addition to these properties we also define a number of properties that describe various ways in which paths in one graph can be refined into paths in the other graph. We first need to define a concept of reachability, though.

Definition 9. Let $\mathbb{G} = \langle S, E \rangle$ be an STG. Then for all $s \in S$, the set $\mathcal{R}(s)$ of *reachable states* from s is defined as $\mathcal{R}(s) = \{t \in S \mid$ there is a path from s to t in $\mathbb{G}\}$. We extend this s.t. for all $T \subseteq S$, $\mathcal{R}(T) = \cup_{s \in T} \mathcal{R}(s)$.

When we consider two STGs $\mathbb{G}_1$ and $\mathbb{G}_2$ simultaneously we write $\mathcal{R}_1(\cdot)$ and $\mathcal{R}_2(\cdot)$ to clarify which graph the reachability function refers to. The refinement properties can now be defined as follows.

Definition 10. Let $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ and $\mathbb{G}_2 = \langle S_1, E_1 \rangle$ be two STGs and let f be a transformation function from $\mathbb{G}_1$ to $\mathbb{G}_2$. Then f can have the following *instance properties*:[1].

$\mathbf{P_{k\downarrow}}$: For every path $t_0, \ldots, t_k$ in S_2, there are $s_0, \ldots, s_k \in S_1$ s.t. $s_i \in \overline{f}(t_i)$ for all i $(0 \leq i \leq k)$ and $s_i \in \mathcal{R}_1(s_{i-1})$ for all i $(1 \leq i \leq k)$.

$\mathbf{P_{k\uparrow}}$: For every path $s_0, \ldots, s_k$ in S_1, there are $t_0, \ldots, t_k \in S_2$ s.t. $t_i \in f(s_i)$ for all i $(0 \leq i \leq k)$ and $t_i \in \mathcal{R}_2(t_{i-1})$ for all i $(1 \leq i \leq k)$.

$\mathbf{P_{T\downarrow}}$: $\mathbf{P_{1\downarrow}}$ holds.

$\mathbf{P_{T\uparrow}}$: $\mathbf{P_{1\uparrow}}$ holds.

$\mathbf{P_{W\downarrow}}$: $\mathbf{P_{k\downarrow}}$ holds for all $k > 0$.

$\mathbf{P_{W\uparrow}}$: $\mathbf{P_{k\uparrow}}$ holds for all $k > 0$.

$\mathbf{P_\downarrow}$: If $t \in \mathcal{R}_2(f(s))$, then $\overline{f}(t) \cap \mathcal{R}_1(s) \neq \varnothing$.

$\mathbf{P_\uparrow}$: If $t \in \mathcal{R}_1(s)$, then $f(t) \cap \mathcal{R}_2(f(s)) \neq \varnothing$.

$\mathbf{P_{S\downarrow}}$: If $t \in \mathcal{R}_2(f(s))$, then $\overline{f}(t) \subseteq \mathcal{R}_1(s)$.

$\mathbf{P_{S\uparrow}}$: If $t \in \mathcal{R}_1(s)$, then $f(t) \subseteq \mathcal{R}_2(f(s))$.

We briefly explain the downward properties and note that the upward properties can be understood in an analogous, but not fully symmetric, way. Consider a path $t_0, t_1, \ldots, t_k$ in the abstract graph, for some $k > 0$. If property $\mathbf{P_{k\downarrow}}$ holds, then there are states $s_0, s_1, \ldots, s_k$ in the original

---

[1] The earlier KR paper contains a typo in the definitions of $\mathbf{P_{k\downarrow}}$ and $\mathbf{P_{k\uparrow}}$, which is corrected here.

graph such that there is a path from s_0 to s_k passing through all of $s_1, \ldots, s_{k-1}$ in order. Consider the example in Figure 2. This transformation function f satisfies $\mathbf{P_{1\downarrow}}$ since both single-arc paths t_0, t_1 and t_1, t_2 in $\mathbb{G}_2$ have corresponding paths in $\mathbb{G}_1$. However, f does not satisfy $\mathbf{P_{2\downarrow}}$ since the path t_0, t_1, t_2 does not have a corresponding path in $\mathbb{G}_1$; we can go from $\overline{f}(t_0)$ to $\overline{f}(t_1)$ and from $\overline{f}(t_1)$ to $\overline{f}(t_2)$ but we cannot go all the way from $\overline{f}(t_0)$ to $\overline{f}(t_2)$.
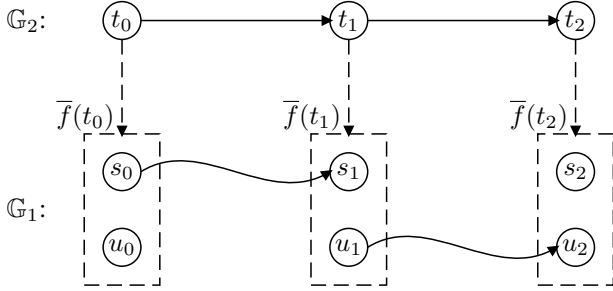

Figure 2: A transformation that is $\mathbf{P_{1\downarrow}}$ but not $\mathbf{P_{2\downarrow}}$.

The following relationships hold:

$$\mathbf{P_{S\downarrow}} \Rightarrow \mathbf{P_{\downarrow}} \Rightarrow \mathbf{P_{W\downarrow}} \Rightarrow \mathbf{P_{T\downarrow}}$$
$$\mathbf{P_{T\downarrow}} \not\Rightarrow \mathbf{P_{W\downarrow}} \not\Rightarrow \mathbf{P_{\downarrow}} \not\Rightarrow \mathbf{P_{S\downarrow}}$$

Property $\mathbf{P_{\updownarrow}}$ is a generalisation to our transformation functions of the **DPP** property (Zilles and Holte 2010), and it coincides with **DPP** if the functions are restricted in the same way as theirs. Some of the other $\mathbf{P}$ properties can be used to capture different degrees of path refinement as follows.

Theorem 11. *Let $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ and $\mathbb{G}_2 = \langle S_1, E_1 \rangle$ be two STGs and let $\tau = \langle f, R \rangle$ be a transformation from $\mathbb{G}_1$ to $\mathbb{G}_2$. Then every path in $\mathbb{G}_2$ is*

1) trivially downward state refinable iff τ is $\mathbf{P_{T\downarrow}}$,
2) weakly downward state refinable iff τ is $\mathbf{P_{W\downarrow}}$ and
3) strongly downward state refinable iff τ is $\mathbf{P_{S\downarrow}}$.

Zilles and Holte (2010) proved that testing if **DPP** holds is in **P** for explicit graphs and **PSPACE**-complete for implicit graphs. We will later generalise these results to hold also for testing $\mathbf{P_{\updownarrow}}$.

5 MSTRIPS and Implicit STGs

To be able to specify STGs implicitly with succinct representations we will introduce the MSTRIPS concept, which is a straightforward generalisation of STRIPS to multi-valued state variables. Other such formalisms are possible and the choice does not matter much for the forthcoming results. MSTRIPS uses state spaces that are induced by variables. A state is then defined as a vector of values for these variables. We will, however, do this a bit differently and use a state concept based on sets of variable-value pairs. While this make the basic definitions slightly more complicated, it will simplify the forthcoming definitions and proofs.

Definition 12. A *variable set* V is a set of objects called *variables*. A *domain function* D for V is a function that maps every variable $v \in V$ to a corresponding *domain* D_v

of values. An *atom* over V and D is a pair $\langle v, x \rangle$ (usually written as $(v = x)$) such that $v \in V$ and $x \in D_v$. A *state* is a set of atoms and $V \cdot D = \cup_{v \in V}(\{v\} \times D_v)$ denotes the set of all possible atoms over V and D. A state $s \subseteq V \cdot D$ is
  1) *consistent* if each $v \in V$ occurs at most once in s,
  2) *total* if each $v \in V$ occurs exactly once in s.
The filter functions $\mathcal{T}$ and $\mathcal{C}$ are defined for all $S \subseteq V \cdot D$ as:
  1) $\mathcal{C}(S) = \{s \subseteq S \mid s \text{ is consistent }\}$.
  2) $\mathcal{T}(S) = \{s \subseteq S \mid s \text{ is total }\}$.
Let $s, t \in \mathcal{C}(V \cdot D)$, $U \subseteq V$ and $v \in V$. Then $V(s) = \{v \mid (v = x) \in s\}$, $s[U] = s \cap (U \cdot D)$, $s[v] = s[\{v\}]$ and $s \bowtie t = s[V - V(t)] \cup t$.

Note that $\mathcal{T}(V \cdot D)$ is the set of all total states over V and D and $\mathcal{C}(V \cdot D)$ is the set of all consistent states. Unless otherwise specified, states will be assumed total and we will usually write state rather than total state.

We can define MSTRIPS as follows.

Definition 13. An MSTRIPS *frame* is a tuple $\boldsymbol{f} = \langle V, D, A \rangle$ where V is a variable set, D is a domain function for V and A is a set of *actions*. Each action $a \in A$ has a *precondition* $\mathrm{pre}(a) \in \mathcal{C}(V \cdot D)$ and a *postcondition* $\mathrm{post}(a) \in \mathcal{C}(V \cdot D)$. The STG $\mathbb{G}(\boldsymbol{f}) = \langle S, E \rangle$ for $\boldsymbol{f}$ is defined s.t. 1) $S = \mathcal{T}(V \cdot D)$ and 2) $E = \{\langle s, t, a \rangle \mid a \in A, \mathrm{pre}(a) \subseteq s \text{ and } t = s \bowtie \mathrm{post}(a)\}$. A sequence $\omega = a_1, \ldots, a_k$ of actions in A is a *plan* from a state $s \in S$ to a state $t \in S$ if there is a path $s_0, a_1, s_1, a_2 \ldots, a_k, s_k$ in $\mathbb{G}(\boldsymbol{f})$ s.t. $s_0 = s$ and $s_k = t$.

An MSTRIPS STG is an STG specified as an MSTRIPS frame, with the actions used as labels.

6 Computational Problems

We define the following decision problems.

Transformation Function
INSTANCE: Two state spaces S_1 and S_2 and a function $f : S_1 \to 2^{S_2}$.
QUESTION: Is f a transformation function from S_1 to S_2?

X-Test
INSTANCE: Two STGs $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ and $\mathbb{G}_2 = \langle S_2, E_2 \rangle$ and a transformation $\tau = \langle f, R \rangle$ from S_1 to S_2.
QUESTION: Does τ satisfy property X?

For properties that have up and down variants, we will usually omit the arrow when the complexity is the same for both directions. For instance, **M**-Test refers collectively to $\mathbf{M_{\uparrow}}$-Test, $\mathbf{M_{\downarrow}}$-Test and $\mathbf{M_{\updownarrow}}$-Test.

The STGs may be given either explictly or implicitly as MSTRIPS frames. The function f and the relation R are assumed polynomial-time computable, and for Transformation Function we assume that f returns some special marker symbol for undefined input values (which is possible since f is polynomial-time computable). Note that in the case of explicit STGs we have that if f is polynomial time then also $\overline{f}$ is polynomial-time computable. However, this implication does not necessarily hold for implicit STGs.

Lemma 14. *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two arbitrary STGs and let f be an arbitrary polynomial-time computable transformation function from $\mathbb{G}_1$ to $\mathbb{G}_2$. Then:*

1) $\overline{f}$ *is polynomial-time computable if* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *are explicit.*

2) Deciding if $\overline{f}(s) \neq \varnothing$ *for arbitrary* $s \in S_2$ *is **NP**-complete in the general case if* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *are implicit.*

Proof. 1) Straightforward since we can enumerate the states in both STGs in polynomial time.

2) *Membership:* Guess a state $t \in S_1$ and check (in polynomial time) whether $s \in f(t)$, or not.
Hardness: Reduction from Satisfiability. Let φ be an arbitrary formula over the variables $\mathbf{x} = x_1; \ldots; x_n$. Let $V_1 = \{x_1, \ldots, x_n\}$ and $V_2 = \{y\}$ where $D_v = \{0, 1\}$ for all $v \in V_1 \cup V_2$. Let S_1 and S_2 be the corresponding state spaces. Define the transformation function f from $\mathbb{G}_1$ to $\mathbb{G}_2$ s.t. $f(\mathbf{x}) = \{1\}$ iff $\varphi(\mathbf{x})$ is true. Then, $\overline{f}(1)$ is the set of models for φ, so φ is satisfiable if and only if $\varphi \neq \varnothing$. $\square$

It should be noted that the size of $\overline{f}(s)$ is not even polynomially bounded in the general case.

Table 1 summarizes the complexity results that will be proven in the following two sections.

Problem	Explicit STGs	Implicit STGs
Transf. Func.	in **P**	$\mathbf{\Pi}_2^p$-complete
M-Test	in **P**	co**NP**-complete
R-Test	in **P**	in **P**
$\mathbf{C}_\uparrow$-Test	in **P**	co**NP**-complete
$\mathbf{C}_\downarrow$-Test	in **P**	$\mathbf{\Pi}_2^p$-complete
$\mathbf{P}_k$-Test	in **P**	**PSPACE**-complete
$\mathbf{P_W}$-Test	co**NP**-complete	**PSPACE**-complete
P-Test, $\mathbf{P_S}$-Test	in **P**	**PSPACE**-complete

Table 1: Complexity results.

7 Explicit STGs

In this section we analyse the decision problems from the previous section for explicit STGs.

Theorem 15. *Transformation Function, M-Test, R-Test and C-Test are in **P** for explicit STGs.*

Proof sketch. Straightforward since we can enumerate the vertices and arcs in polynomial time. $\square$

Theorem 16. $\mathbf{P}_k$*-Test is in **P** for all* $k > 0$ *and explicit STGs.*

Proof sketch. We prove the $\mathbf{P}_{k\downarrow}$ case. Let $t_0, \ldots, t_k$ be a path in $\mathbb{G}_2$. Obviously $|\overline{f}(t_i)| \leq |S_1|$ for each i s.t. $0 \leq i \leq k$. Hence, there are at most $|S_1|^k$ combinations of states $s_0, \ldots, s_k$ s.t. $s_i \in \overline{f}(t_i)$ to check for a path from s_0 to s_k via $s_1, \ldots, s_{k-1}$. Furthermore, there at most $|S_2|^k$ paths of length k in $\mathbb{G}_2$ (since we can ignore the labels). It follows that we can check all these in polynomial time.

The $\mathbf{P}_{k\uparrow}$ case is analogous. $\square$

Theorem 17. *P-Test and P_S-Test are in **P** for explicit STGs.*

Proof. Immediate from Definition 10 since $\overline{f}$, $\mathcal{R}_1$ and $\mathcal{R}_2$ can be computed in polynomial time for explicit STGs. $\square$

Theorem 18. $\mathbf{P_W}$*-Test is coNP-complete for explicit STGs.*

Proof. We prove the $\mathbf{P_{W\downarrow}}$ case.
Membership: The complementary problem of testing if $\mathbf{P_{W\downarrow}}$ does not hold can be solved as follows. Guess a path $t_0, \ldots, t_k$ of length k in $\mathbb{G}_2$, for some k s.t. $1 \leq k \leq |S_2|$. Define the sequence $T_0, \ldots, T_k$ of subsets of S_1 s.t.

1) $T_0 = \overline{f}(t_0)$ and
2) $T_i = \overline{f}(t_i) \cap \mathcal{R}_1(T_{i-1})$ for $0 < i \leq k$.

This sequence can be computed in polynomial time since we can compute $\overline{f}$ and $\mathcal{R}_1$ in polynomial time. Furthermore, there is a path $s_0, \ldots, s_k$ in $\mathbb{G}_1$ s.t. $s_i \in \overline{f}(t_i)$ for all i ($0 \leq i \leq k$) if and only if $T_k \neq \varnothing$. Hence, this problem is in **NP** so it follows that testing if $\mathbf{P_{W\downarrow}}$ holds is in co**NP**.

Hardness: Reduction from Unsatisfiability. Let $F = \{c_1, \ldots, c_m\}$ be a set of clauses where each clause $c_j \in F$ is a set of literals over the variables $x_1, \ldots, x_n$. Construct an STG $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ such that:

$$S_1 = S_1^0 \cup \ldots \cup S_1^{m+1}, \text{ where}$$
$$S_1^j = \{f_0^j, \ldots, f_{n+1}^j, t_0^j, \ldots, t_{n+1}^j\} \text{ for } 0 \leq j \leq m+1.$$
$$E_1 = E_1^0 \cup \ldots \cup E_1^{m+1}, \text{ where}$$
$$E_1^0 = \{\langle t_0^0, f_1^0 \rangle, \langle t_0^0, t_1^0 \rangle\} \cup$$
$$\{\langle f_i^0, f_{i+1}^0 \rangle, \langle f_i^0, t_{i+1}^0 \rangle, \langle t_i^0, f_{i+1}^0 \rangle, \langle t_i^0, t_{i+1}^0 \rangle \mid$$
$$1 \leq i < n\},$$
$$E_1^j = \{\langle t_0^j, f_1^j \rangle \mid \overline{x_1} \notin c_j\} \cup \{\langle t_0^j, t_1^j \rangle \mid x_1 \notin c_j\} \cup$$
$$\{\langle f_i^j, f_{i+1}^j \rangle, \langle t_i^j, f_{i+1}^j \rangle \mid 1 \leq i < n, \overline{x_{i+1}} \notin c_j\} \cup$$
$$\{\langle f_i^j, t_{i+1}^j \rangle, \langle t_i^j, t_{i+1}^j \rangle \mid 1 \leq i < n, x_{i+1} \notin c_j\} \cup$$
$$\{\langle f_n^j, t_{n+1}^j \rangle, \langle t_n^j, t_{n+1}^j \rangle\}, \text{ for } 1 \leq j \leq m, and$$
$$E_1^{m+1} = \{\langle f_n^{m+1}, t_{n+1}^{m+1} \rangle, \langle t_n^{m+1}, t_{n+1}^{m+1} \rangle\} \cup$$
$$\{\langle f_i^{m+1}, f_{i+1}^{m+1} \rangle, \langle f_i^{m+1}, t_{i+1}^{m+1} \rangle,$$
$$\langle t_i^{m+1}, f_{i+1}^{m+1} \rangle, \langle t_i^{m+1}, t_{i+1}^{m+1} \rangle \mid 1 \leq i < n\}.$$

Also define an STG $\mathbb{G}_2 = \langle S_2, E_2 \rangle$ such that:
$$S_2 = \{f_0, \ldots, f_{n+1}, t_0, \ldots, t_{n+1}\} \text{ and}$$
$$E_2 = \{\langle t_0, f_1 \rangle, \langle t_0, t_1 \rangle, \langle f_n, t_{n+1} \rangle, \langle t_n, t_{n+1} \rangle\} \cup$$
$$\{\langle f_i, f_{i+1} \rangle, \langle f_i, t_{i+1} \rangle, \langle t_i, f_{i+1} \rangle, \langle t_i, t_{i+1} \rangle \mid$$
$$1 \leq i < n\}.$$

For each j such that $0 \leq j \leq m+1$, define the subgraph $\mathbb{G}_1^j = \langle S_1^j, E_1^j \rangle$ of $\mathbb{G}_1$. These STGs are illustrated in Figure 3. Define the transformation function f from $\mathbb{G}_1$ to $\mathbb{G}_2$ such that

$$f(t_i^j) = \{t_i\} \text{ for } 0 \leq i \leq n+1 \text{ and } 0 \leq j \leq m+1$$
$$f(f_i^j) = \{f_i\} \text{ for } 1 \leq i \leq n \text{ and } 0 \leq j \leq m+1$$

We first note that every path in $\mathbb{G}_2$ from t_0 to t_{n+1} must pass either f_i or t_i, but not both, for every i such that $1 \leq i \leq n$. That is, there is exactly one path from t_0 to t_{n+1} for every possible truth assignment for $x_1, \ldots, x_n$. We also note that $\mathbb{G}_1^0$ and $\mathbb{G}_1^{m+1}$ are both isomorphic to $\mathbb{G}_2$ except that in $\mathbb{G}_1^0$ there are no arcs to t_{n+1}^0 and in $\mathbb{G}_1^{m+1}$ there are no arcs from t_0^{m+1}. Hence, any path in $\mathbb{G}_2$ of length at most n is always weakly refinable to a path in either $\mathbb{G}_1^0$ or $\mathbb{G}_1^{m+1}$. We further note that for arbitrary j such that $1 \leq j \leq m$, every
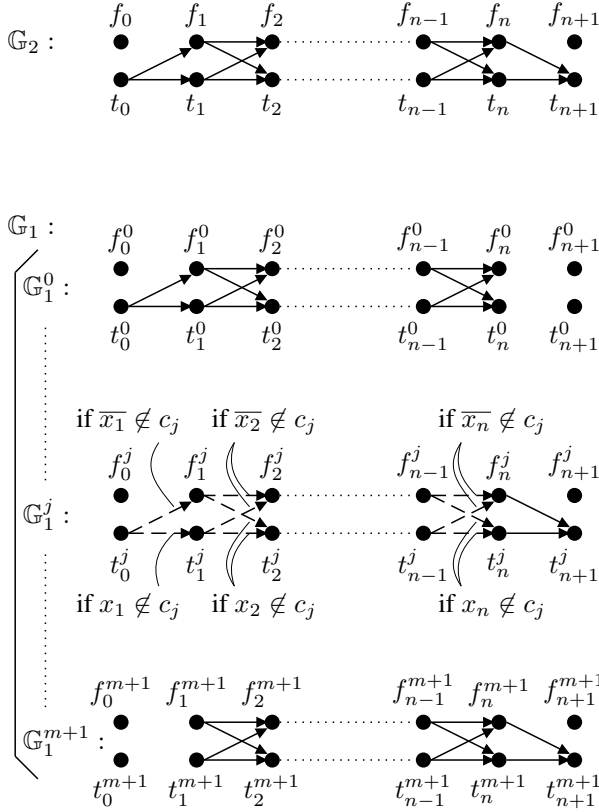
Figure 3: The STG $\mathbb{G}_2$ and the components of the STG $\mathbb{G}_1$ in the proof of Theorem 18 (only one generic component is shown for $\mathbb{G}_1^1 \ldots, \mathbb{G}_1^m$). Dashed arcs occur or not dependent on the literals of the corresponding clause.

path in $\mathbb{G}_1^j$ from t_0^j to t_{n+1}^j corresponds to a truth assignment for $x_1, \ldots, x_n$ that does not satisfy clause c_j. Consider an arbitrary path σ_2 of length $n + 1$ in $\mathbb{G}_2$, which must be from t_0 to t_{n+1}. Assume σ_2 is refinable to a path σ_1 in $\mathbb{G}_1$. Since neither $\mathbb{G}_1^0$ nor $\mathbb{G}_1^{m+1}$ has any path of length $n + 1$, there must be some j such that $1 \leq j \leq m$ and σ_1 is a path in $\mathbb{G}_1^j$, i.e. clause c_j is not satisfied by the assignment. Instead assume that σ_2 is not refinable to any path in $\mathbb{G}_1$. Then all clauses $c_1, \ldots, c_m$ must be satisfied by the assignment.

That is, every path of length at most n in $\mathbb{G}_2$ is always weakly refinable to a path in $\mathbb{G}_1$ and every path of length $n + 1$ in $\mathbb{G}_2$ is weakly refinable to a path in $\mathbb{G}_1$ if and only if F is not satisfiable. Since $\mathbb{G}_2$ has no paths longer than $n + 1$ it follows that f is $\mathbf{P_{W\downarrow}}$ if and only F is not satisfiable. The theorem follows since Unsatisfiability is coNP-hard. $\square$

8 Complexity for Implicit STGs

We now turn to implicit STGs represented as MSTRIPS frames. The symbol ";" denotes sequence concatenation.

Theorem 19. *Transformation Function is $\mathbf{\Pi}_2^p$-complete for* MSTRIPS *STGs.*

Proof. Hardness: Reduction from $\forall\exists$-SAT. Let $\psi = \forall\mathbf{x}\exists\mathbf{y}.\varphi(\mathbf{x}, \mathbf{y})$ be a $\forall\exists$-SAT formula s.t. $\mathbf{x} =$

$x_1, \ldots, x_m$ and $\mathbf{y} = y_1, \ldots, y_n$. Define $V_1 = \{x_1, \ldots, x_m, y_1, \ldots, y_n, z\}$ and $V_2 = \{x_1, \ldots, x_m, z\}$, where $D_v = \{0, 1\}$ for all $v \in V_1 \cup V_2$. Let $S_1 = V_1 \cdot D_1$ and $S_2 = V_2 \cdot D_2$ be the corresponding state spaces. Define a function $f : S_1 \to 2^{S_2}$ as follows:

$f(\mathbf{x}; \mathbf{y}; 0) = \{\mathbf{x}; 1\}$ if $\varphi(\mathbf{x}, \mathbf{y})$ is true.
$f(\mathbf{x}; \mathbf{y}; 0) = \{\mathbf{x}; 0\}$ if $\varphi(\mathbf{x}, \mathbf{y})$ is false.
$f(\mathbf{x}; \mathbf{y}; 1) = \{\mathbf{x}; 0\}$ always.

Note that testing whether φ is true for an assignment is polynomial time. Also note that $|f(\mathbf{x}; \mathbf{y}; z)| = 1$ for all $\mathbf{x}$, $\mathbf{y}$ and z, so f is total and maps every state in S_1 to exactly one state in S_2. Hence, $Rng(f)$ is a partition of S_2 if and only if $f(S_1) = S_2$. For all $\mathbf{x}$ there are always $\mathbf{y}$ and z such that $f(\mathbf{x}; \mathbf{y}; z) = \{\mathbf{x}; 0\}$, but there are $\mathbf{y}$ and z such that $f(\mathbf{x}; \mathbf{y}; z) = \{\mathbf{x}; 1\}$ if and only if there is some $\mathbf{y}$ such that $\varphi(\mathbf{x}, \mathbf{y})$ is true. That is, $f(S_1) = S_2$ if and only if ψ is satisfiable. Since f is a transformation function from S_1 to S_2 if and only if $Rng(f)$ is a partition of S_2, it follows that f is a transformation function if and only if ψ is satisfiable. We conclude that Transformation Function is $\mathbf{\Pi}_2^p$-hard.

Membership: We must verify that f is total and that $Rng(f)$ is a partition of S_2. We can verify that f is not total by guessing a state $s \in S_1$ and verifying that $f(s)$ is undefined. This is in NP since f is polynomial time, so verifying that f is total is in coNP. Verifying that $f(s) \neq \varnothing$ for all $s \in S_1$ is in coNP by an analogous argument. To verify that $f(s)$ and $f(t)$ are disjoint if $f(s) \neq f(t)$, consider the complementary problem of verifying that they are not disjoint in this case. This can be done by guessing two states s and t and verifying that $f(s) \neq f(t)$ but $f(s) \cap f(t) \neq \varnothing$. Hence, this problem is in NP, so the original problem is in coNP. Finally, we need to prove that $f(S_1) = S_2$, i.e. that

$$\forall t \in S_2 \exists s \in S_1 . t \in f(s).$$

Consider the complementary problem, that is, to verify that

$$\exists t \in S_2 \neg \exists s \in S_1 . t \in f(s).$$

We can guess t and use an oracle for coNP to verify there is no s s.t. $t \in f(s)$. Hence, this problem is in $\mathbf{NP}^{\mathbf{coNP}} = \mathbf{NP}^{\mathbf{NP}}$ so the original problem is in $\mathbf{coNP}^{\mathbf{NP}} = \mathbf{\Pi}_2^p$. $\square$

Theorem 20. *M-Test is coNP-complete for* MSTRIPS *STGs.*

Proof. Membership: We first prove the $\mathbf{M}_\uparrow$ case. Consider the complementary problem of deciding if f does not satisfy $\mathbf{M}_\uparrow$. This can be solved by guessing a state $s \in S_1$ and then verify that $|f(s)| > 1$. Hence, this problem is in NP so testing if τ is $\mathbf{M}_\uparrow$ is in coNP.

Since $\overline{f}$ is not guaranteed to be polynomial-time computable we must handle the $\mathbf{M}_\downarrow$ case differently. We first note that for arbitrary state $s \in S_2$, $|\overline{f}(s)| > 1$ iff there are two states $t, t' \in S_1$ s.t. $t \neq t'$ and $s \in f(t) \cap f(t')$. Consider the complementary problem of deciding if τ does not satisfy $\mathbf{M}_\downarrow$. We can check this by guessing two states $t, t' \in S_1$ and then verify that $t \neq t'$ and $f(t) \cap f(t') \neq \varnothing$. Hence, this problem is in NP so testing if τ is $\mathbf{M}_\downarrow$ is in coNP.

Hardness: We first prove the $\mathbf{M}_\uparrow$ case by reduction from Unsatisfiability. Let $\varphi(\mathbf{x})$ be a formula over $\mathbf{x} =$

$x_1, \ldots, x_m$. Define $V_1 = V_2 = \{x_1, \ldots, x_m, z\}$, where $D_v = \{0,1\}$ for all $v \in V_1$. Let $S_1 = V_1 \cdot D_1 = S_2$ be the corresponding state spaces. Define $f : S_1 \to 2^{S_2}$ s.t.

$f(\mathbf{x}; 0) = \{\mathbf{x}; 0\}$ if $\varphi(\mathbf{x})$ is false.
$f(\mathbf{x}; 1) = \{\mathbf{x}; 1\}$ if $\varphi(\mathbf{x})$ is false.
$f(\mathbf{x}; 0) = f(\mathbf{x}; 1) = \{\mathbf{x}; 0, \mathbf{x}; 1\}$ if $\varphi(\mathbf{x})$ is true.

We note that f is a transformation function from $\mathbb{G}_1$ to $\mathbb{G}_2$. Furthermore, $f(\mathbf{x}; 0) \neq f(\mathbf{x}; 1)$ if $\varphi(\mathbf{x})$ is false but $f(\mathbf{x}; 0) = f(\mathbf{x}; 1)$ if $\varphi(\mathbf{x})$ is true. That is, $|f(\mathbf{x}; z)| = 1$ for all $\mathbf{x}$ and z if and only if $\varphi(\mathbf{x})$ is not satisfiable. It follows that $\mathbf{M}_\uparrow$-Test is coNP-hard.

For the $\mathbf{M}_\downarrow$ case, we first note that every element in $f(S_1)$ is either of the form $\{s\}$ or $\{s, t\}$, where $s, t \in S_1$. Hence, we have $\overline{f}(\{s\}) = f(s)$ and $\overline{f}(\{s, t\}) = f(s) = f(t)$, i.e. $\overline{f}$ is polynomial-time computable. It then follows from the symmetry between $\mathbf{M}_\uparrow$ and $\mathbf{M}_\downarrow$ that testing $\mathbf{M}_\downarrow$ is coNP-hard since we can swap $\mathbb{G}_1$ with $\mathbb{G}_2$ and f with $\overline{f}$. □

Theorem 21. *R-Test is in **P** for* MSTRIPS *STGs.*

Proof. We prove the $\mathbf{R}_\uparrow$ case and $\mathbf{R}_\downarrow$ is analogous. For every $a_1 \in A_1$, both $\mathrm{pre}(a_1)$ and $\mathrm{post}(a_1)$ are consistent by definition so a_1 defines at least one arc in $\mathbb{G}_1$. The analogous holds for A_2. To check if $\mathbf{R}_\uparrow$ holds it is thus sufficient to check that for every $a_1 \in A_1$ there is some $a_2 \in A_2$ s.t. $R(a_1, a_2)$. This is obviously polynomial time. □

Theorem 22. *$C_\uparrow$-Test is coNP-complete for* MSTRIPS *STGs.*

Proof. Membership: Consider the complementary problem of deciding if τ is not $\mathbf{C}_\uparrow$. Guess an $a_1 \in A_1$, an $a_2 \in A_2$ and two $s, t \in S_1$ s.t. $\langle s, t, a_1 \rangle \in E_1$. Then we check that there are no $s' \in f(s)$ and $t' \in f(t)$ s.t. $\langle s', t', a_2 \rangle \in E_2$. Since f is polynomial-time computable, there can be at most a polynomial number of such pairs. Hence, this problem is in NP so deciding if τ is $\mathbf{C}_\uparrow$ is in coNP.

Hardness: Reduction from Unsatisfiability. Let $\varphi(\mathbf{x})$ be a formula over $\mathbf{x} = x_1, \ldots, x_m$. Define $V_1 = V_2 = \{x_1, \ldots, x_m, y, z\}$, where $D_v = \{0,1\}$. Let S_1 and S_2 be the corresponding state spaces. Define the action sets $A_1 = A_2 = \{a\}$ where $\mathrm{pre}(a) = \{(y = 0), (z = 0)\}$ and $\mathrm{post}(a) = \{(z = 1)\}$. Consider the MSTRIPS frames $\boldsymbol{f}_1 = \boldsymbol{f}_2 = \langle V_1, D_1, A_1 \rangle$ and their corresponding STGs $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ and $\mathbb{G}_2 = \langle S_2, E_2 \rangle$. Obviously, both E_1 and E_2 have an arc from $\mathbf{x}; 0; 0$ to $\mathbf{x}; 0; 1$ for every $\mathbf{x}$ and no other arcs. Define $f : S_1 \to 2^{S_2}$ s.t.

$f(\mathbf{x}; y; z) = \{\mathbf{x}; y; z\}$ if $\varphi(\mathbf{x})$ is false.
$f(\mathbf{x}; 0; z) = \{\mathbf{x}; 1; z\}$ if $\varphi(\mathbf{x})$ is true.
$f(\mathbf{x}; 1; z) = \{\mathbf{x}; 0; z\}$ if $\varphi(\mathbf{x})$ is true.

We note that f is a transformation function from $\mathbb{G}_1$ to $\mathbb{G}_2$. Furthermore, f map $\mathbf{x}; 0; 0$ and $\mathbf{x}; 0; 1$ to the sets of themselves if and only if $\varphi(\mathbf{x})$ is false. Also define the relation $R = \{\langle a, a \rangle\}$ and the transformation $\tau = \langle f, R \rangle$. We must prove that φ is unsatisfiable if and only if τ is $\mathbf{C}_\uparrow$.

if: Suppose $\mathbf{C}_\uparrow$ holds. Let $\langle s, t, a \rangle$ be an arbitrary arc in E_1. Then $s = \mathbf{x}; 0; 0$ and $t = \mathbf{x}; 0; 1$ for some $\mathbf{x}$. Since $R(a, a)$ holds there must be an arc $\langle f(s), f(t), a \rangle$ in E_2. This is only possible if $f(s) = s$ and $f(t) = t$. Since $\langle s, t, a \rangle$ was chosen arbitrarily, we have $f(\mathbf{x}; 0; 0) = \{\mathbf{x}; 0; 0\}$ and

$f(\mathbf{x}; 0; 1) = \{\mathbf{x}; 0; 1\}$ for all $\mathbf{x}$. It thus follows from the definition of f that there is no $\mathbf{x}$ s.t. $\varphi(\mathbf{x})$ is true.

only if: Suppose φ is unsatisfiable. Then $f(\mathbf{x}; 0; 0) = \{\mathbf{x}; 0; 0\}$ and $f(\mathbf{x}; 0; 1) = \{\mathbf{x}; 0; 1\}$ for all $\mathbf{x}$. Hence, $\mathbf{C}_\uparrow$ holds. □

Theorem 23. *$C_\downarrow$-Test is Π_2^p-complete for* MSTRIPS *STGs.*

Proof. Membership: Consider the complementary problem of deciding if τ does not satisfy $\mathbf{C}_\downarrow$, that is, checking that

$$\exists s, t, \ell, \ell' (\langle s, t, \ell \rangle \in E_2 \wedge R(\ell', \ell) \wedge \neg \varphi)$$

holds, where φ is

$$\exists s', t' (\langle s', t', \ell' \rangle \in E_1 \wedge s' \in \overline{f}(s) \wedge t' \in \overline{f}(t)).$$

Consider the problem of deciding if φ holds. This problem is in NP since φ is equivalent to

$$\exists s', t' (\langle s', t', \ell' \rangle \in E_1 \wedge s \in f(s') \wedge t \in f(t')).$$

Hence, the complementary problem C of deciding if $\neg\varphi$ holds is in coNP. We can thus check that $\mathbf{C}_\downarrow$ does not hold by guessing an arc $\langle s, t, \ell \rangle \in E_2$ and a label ℓ' and then use an oracle for C to verify that there is no corresponding arc in E_1. It follows that the problem is in $\mathbf{NP}^{\mathbf{coNP}} = \mathbf{NP}^{\mathbf{NP}}$. That is, checking if $\mathbf{C}_\downarrow$ holds is in $\mathbf{coNP}^{\mathbf{NP}} = \Pi_2^p$.

Hardness: Reduction from $\forall\exists$-SAT. Let $\psi = \forall\mathbf{x}\exists\mathbf{y}.\varphi(\mathbf{x}, \mathbf{y})$ be a $\forall\exists$-SAT formula s.t. $\mathbf{x} = x_1, \ldots, x_m$ and $\mathbf{y} = y_1, \ldots, y_n$. Define $V_2 = \{x_1, \ldots, x_m, z, u\}$ and $V_1 = V_2 \cup \{y_1, \ldots, y_n\}$, where $D_v = \{0,1\}$ for all $v \in V_1$. Let $S_1 = V_1 \cdot D_1$ and $S_2 = V_2 \cdot D_2$ be the corresponding induced state spaces. Define the action sets $A_1 = A_2 = \{a\}$ where $\mathrm{pre}(a) = \{(z = 0), (u = 0)\}$ and $\mathrm{post}(a) = \{(u = 1)\}$. Consider the MSTRIPS frames $\boldsymbol{f}_1 = \langle V_1, D, A_1 \rangle$ and $\boldsymbol{f}_2 = \langle V_2, D, A_2 \rangle$ with their corresponding STGs $\mathbb{G}_1 = \langle S_1, E_1 \rangle$ and $\mathbb{G}_2 = \langle S_2, E_2 \rangle$. Define $f : S_1 \to 2^{S_2}$ as follows:

$f(\mathbf{x}; \mathbf{y}; z; u) = \{\mathbf{x}; z; u\}$ if $\varphi(\mathbf{x}, \mathbf{y})$ is true.
$f(\mathbf{x}; \mathbf{y}; 0; u) = \{\mathbf{x}; 1; u\}$ if $\varphi(\mathbf{x}, \mathbf{y})$ is false.
$f(\mathbf{x}; \mathbf{y}; 1; u) = \{\mathbf{x}; 0; u\}$ if $\varphi(\mathbf{x}, \mathbf{y})$ is false.

We note that f is a transformation function from $\mathbb{G}_1$ to $\mathbb{G}_2$. Furthermore, f map $\mathbf{x}; 0; 0$ and $\mathbf{x}; 0; 1$ to the sets of themselves if and only if $\varphi(\mathbf{x})$ is false. Also define the relation $R = \{\langle a, a \rangle\}$ and the transformation $\tau = \langle f, R \rangle$. We must prove that ψ is true if and only if τ is $\mathbf{C}_\downarrow$.

if: Suppose $\mathbf{C}_\downarrow$ holds. Let $\langle s, t, a \rangle$ be an arbitrary arc in E_2. Then $s = \mathbf{x}; 0; 0$ and $t = \mathbf{x}; 0; 1$ for some $\mathbf{x}$. Since $R(a, a)$ holds and $\mathbf{C}_\downarrow$ holds there must be an arc $\langle s', t', a \rangle \in E_1$ s.t. $s' \in \overline{f}(s)$ and $t' \in \overline{f}(t)$. By definition of f this is only possible if $\{s\} = f(s')$ and $\{t\} = f(t')$. Since $\langle s, t, a \rangle$ was chosen arbitrarily, we have $f(\mathbf{x}; \mathbf{y}; 0; 0) = \{\mathbf{x}; 0; 0\}$ and $f(\mathbf{x}; \mathbf{y}; 0; 1) = \{\mathbf{x}; 0; 1\}$ for at least one $\mathbf{y}$ for every $\mathbf{x}$. It follows from the definition of f that ψ must be true.

only if: Suppose ψ is true. Then for every $\mathbf{x}$ there is at least one $\mathbf{y}$ s.t. $f(\mathbf{x}; \mathbf{y}; 0; 0) = \{\mathbf{x}; 0; 0\}$ and $f(\mathbf{x}; \mathbf{y}; 0; 1) = \{\mathbf{x}; 0; 1\}$, i.e. $\mathbf{x}; \mathbf{y}; 0; 0 \in \overline{f}(\mathbf{x}; 0; 0)$ and $\mathbf{x}; \mathbf{y}; 0; 1 \in \overline{f}(\mathbf{x}; 0; 1)$. It follows that $\mathbf{C}_\downarrow$ is satisfied. □

Theorem 24. *P_T-Test, P_k-Test, P_W-Test, P-Test and P_S-Test are* PSPACE-*complete for* MSTRIPS *STGs, even if f is the identity function.*

Proof sketch. Hardness: We first prove the $\mathbf{P_{T\downarrow}}$ case, i.e. the case $\mathbf{P_{1\downarrow}}$. Let $f = \langle V, D, A\rangle$ be an MSTRIPS frame and let $\mathbb{G}_1 = \langle S_1, E_1\rangle$ be the STG for f. Let s and t be arbitrary states in S_1. Construct the STG $\mathbb{G}_2 = \langle S_2, E_2\rangle$ s.t. $S_2 = S_1$ and $E_2 = \{\langle s, t\rangle\}$. Let f be the identity function. There is only one path in $\mathbb{G}_2$, the path from s to t consisting of one arc. Checking if there is a corresponding path from s to t in $\mathbb{G}_1$ requires deciding if there is a plan from s to t in $\mathbb{G}_1$, which is **PSPACE**-complete (Bylander 1994, Theorem 3.1). Hence, it is **PSPACE**-hard to check if $\mathbf{P_{T\downarrow}}$ holds.

The same construction can be used to prove that all the other properties are also **PSPACE**-hard.

Membership: We prove only the $\mathbf{P_{k\downarrow}}$ case since the others are similar. First consider the complementary problem of verifying that $\mathbf{P_{k\downarrow}}$ does not hold. The algorithm NotRefinable in Figure 4 solves this problem: it accepts if and only if there is at least one path of length k or shorter in $\mathbb{G}_2$ that is not refinable to a path in $\mathbb{G}_1$. Note that if it rejects in line 4 or 6, then there is not even a path of length k, and all paths are refinable. The algorithm runs in nondeterministic polynomial space, so the problem is in **NPSPACE**. However, testing if $\mathbf{P_{k\downarrow}}$ holds is then also in **NPSPACE** since **NPSPACE** is closed under complement. The theorem follows since **NPSPACE=PSPACE**. $\square$

```
1    function NotRefinable(k,𝔾₁,𝔾₂)
2       guess s, t ∈ S₂, s′ ∈ f̄(s) and t′ ∈ f̄(t)
3       while k > 0 do
4          if s = t then reject
5          guess u ∈ S₂ and u′ ∈ f̄(u)
6          if there is no ⟨s, u⟩ ∈ E₂ then reject
7          if there is no path from s′ to u′ in 𝔾₁ then accept
8             s := u, s′ := u′, k := k − 1
9       reject
```

Figure 4: Algorithm for disproving $\mathbf{P_{k\downarrow}}$.

Note that the theorem implies that all properties are **PSPACE**-hard even if we know that $\mathbf{M_\downarrow}$ holds.

9 Discussion

We only discuss the complexity aspects of the framework here, and refer to Bäckström and Jonsson (2012) for discussion of the framework in general.

Although all **P** properties are **PSPACE**-hard for implicit graphs, it is sometimes possible to get around this. For instance, we know (Bäckström and Jonsson 2012, Theorem 27) that if $\mathbf{M_\uparrow R_\uparrow C_\uparrow}$ holds then $\mathbf{P_{S\uparrow}}$ must hold too. Hence, a co**NP**-complete test is sufficient to guarantee that $\mathbf{P_{S\uparrow}}$ holds in this case.

The difference in complexity for testing $\mathbf{C_\uparrow}$ and $\mathbf{C_\downarrow}$ in the case of implicit graphs may, perhaps, seem counterintuitive given the symmetry of these properties. The reason for this is that we only assume f to be polynomial-time computable, leaving the complexity of $\overline{f}$ unspecified. As soon as we know that also $\overline{f}$ is polynomial, then both directions are co**NP**-complete to test.

It is interesting that $\mathbf{P_W}$ is harder than the other **P** properties for explicit graphs. This warrants the question whether it is harder also for implicit graphs, but that this is not visible due to the coarseness of the class **PSPACE**. However, in the case of explicit graphs, testing $\mathbf{P_W}$ immediately becomes polynomial if we know that **M** holds in the same direction.

Theorem 25. $\mathbf{P_{W\uparrow}}$-*Test* ($\mathbf{P_{W\downarrow}}$-*Test*) *is in* $\mathbf{P}$ *for explicit STGs if* τ *is* $\mathbf{M_\uparrow}$ ($\mathbf{M_\downarrow}$).

Recall that f is a homomorphism if $\langle s, t\rangle \in E_1$ implies that $\langle f(s), f(t)\rangle \in E_2$.

Proof sketch. We prove that $\mathbf{P_{W\uparrow}}$ holds if and only if f is a homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2^+ = \langle S_2, E_2^+\rangle$, where E_2^+ is the transitive closure of E_2. (The $\mathbf{P_{W\downarrow}}$ case is analogous since $\overline{f}$ must then be a homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1^+$.)

if: Suppose f is a homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2^+$. Let $\sigma = s_0, \ldots, s_k$ be an arbitrary path in $\mathbb{G}_1$. Then $f(s_0), \ldots, f(s_k)$ is a path in $\mathbb{G}_2^+$, so there must be a path in $\mathbb{G}_2$ from $f(s_0)$ to $f(s_k)$ passing through $f(s_1), \ldots, f(s_{k-1})$. It follows that $\mathbf{P_{W\uparrow}}$ holds since σ was chosen arbitrarily.

only if: Suppose f is not a homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2^+$. Then there is some $\langle s, t\rangle \in E_1$ s.t. $\langle f(s), f(t)\rangle \notin E_2^+$. Hence, there is no path from $f(s)$ to $f(t)$ in $\mathbb{G}_2$ so no path in $\mathbb{G}_1$ containing the arc $\langle s, t\rangle$ is refinable. It follows that $\mathbf{P_{W\uparrow}}$ does not hold.

The theorem follows since it is a polynomial-time problem to check if f (or $\overline{f}$) is a homomorphism. $\square$

References

Bacchus, F., and Yang, Q. 1994. Downward refinement and the efficiency of hierarchical problem solving. *Artif. Intell.* 71(1):43–100.

Bäckström, C., and Jonsson, P. 2012. Abstracting abstraction in search with applications to planning. In *Proc. 13th Int'l Conf. Principles Knowledge Repr. and Reasoning, (KR-2012), Rome, Italy.*

Bundy, A.; Giunchiglia, F.; Sebastiani, R.; and Walsh, T. 1996. Calculating criticalities. *Artif. Intell.* 88(1-2):39–67.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artif. Intell.* 69(1-2):165–204.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *Proc. 17th Int'l Conf. Automated Planning and Scheduling, (ICAPS-2007), Providence, Rhode Island, USA*, 176–183.

Holte, R. C.; Mkadmi, T.; Zimmer, R. M.; and MacDonald, A. J. 1996. Speeding up problem solving by abstraction: A graph oriented approach. *Artif. Intell.* 85(1-2):321–361.

Knoblock, C. A.; Tenenberg, J. D.; and Yang, Q. 1991. Characterizing abstraction hierarchies for planning. In *Proc. 9th Nat'l Conf. Artif. Intell. (AAAI'91), Anaheim, CA, USA*, 692–697.

Zilles, S., and Holte, R. C. 2010. The computational complexity of avoiding spurious states in state space abstraction. *Artif. Intell.* 174(14):1072–1092.