Endomorphisms of Classical Planning Tasks

Rostislav Horčík[1], **Daniel Fišer**[1,2]
[1] Czech Technical University in Prague, Faculty of Electrical Engineering, Prague, Czech Republic
[2] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
xhorcik@fel.cvut.cz, danfis@danfis.cz

Abstract

Detection of redundant operators that can be safely removed from the planning task is an essential technique allowing to greatly improve performance of planners. In this paper, we employ structure-preserving maps on labeled transition systems (LTSs), namely endomorphisms that are well known from model theory, in order to detect redundancy. Computing endomorphisms of an LTS induced by a planning task is typically infeasible, so we show how to compute some of them on concise representations of planning tasks such as finite domain representations and factored LTSs. We formulate the computation of endomorphisms as a constraint satisfaction problem (CSP) that can be solved by an off-the-shelf CSP solver. Finally, we experimentally verify that the proposed method can find a sizable number of redundant operators on the standard benchmark set.

Introduction

One way to improve performance of classical planners (regardless of the employed planning technique) is by reducing the size of the input planning task by removing operators and facts so that at least one (optimal) plan is preserved. The most commonly used method for the reduction of planning tasks is a reachability analysis in forward and backward direction using h^m heuristics (Haslum and Geffner 2000; Alcázar and Torralba 2015). This method prunes operators that are either unreachable or that lead to a dead-end state. Nevertheless, even if we successfully remove all unreachable and dead-end operators, the remaining planning task may still be unnecessarily large because of many alternative plans it may contain.

Recently, Fišer, Torralba, and Shleyfman (2019) proposed a technique combining so-called operator mutexes (op-mutexes) and symmetries in such a way that allows to identify operators that are redundant in a sense that removing them still preserves at least one (optimal) plan. The main idea behind this technique is based on the observation that if two operators cannot occur simultaneously in an optimal plan and there is a symmetry mapping one operator to another, then one of the operators can be safely removed. This method works only on planning tasks exhibiting non-trivial symmetries.

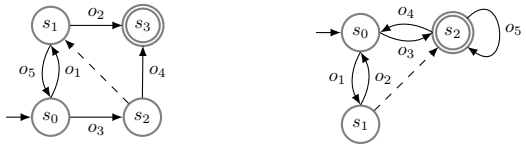

Figure 1: Examples of transition systems with redundant states. The dashed arrows show where to map redundant states.

In this paper, we develop a new pruning method allowing to detect redundant parts of the state space which cannot be detected by a reachability analysis. Although it can, in some cases, detect the same redundant operators as the method based on op-mutexes and symmetries, our method does not depend on the existence of symmetries.

Consider a simple transition system depicted in Fig. 1 (left) where the cost of o_1 and o_2 is less than the cost of o_3 and o_4, respectively, and o_5 has a unit cost. It is obvious that the plan going through the state s_2 is sub-optimal and we can safely omit the state s_2 and operators o_3 and o_4. However, the transition system has no symmetries due to different costs of actions and the operator o_5. Moreover, all states are reachable and there are no dead-ends. Our solution to detect such a redundancy is to look for a structure-preserving self-map on the transition system which maps plans to (possibly cheaper) plans. In the situation described above, such mapping could map the state s_2 to s_1, o_3 to o_1, o_4 to o_2, and fix the remaining states and operators. More precisely, since transition systems can be understood as first-order structures, we just employ standard structure-preserving maps from model theory, namely homomorphisms (see, e.g., Hodges 1997). In particular, we are interested in endomorphisms which are homomorphisms having the same domain and co-domain. It is well known that homomorphisms preserve existential-positive formulas (e.g. Hodges 1997, Section 2.4). As the existence of a plan of a certain length can be expressed by such a formula, it follows that an endomorphism maps plans to plans. Consequently, if we find an endomorphism whose image is strictly smaller than the original transition system, then the states and operators which are not in its image are necessarily redundant.

Another example depicted in Fig. 1 (right) shows a transition system having no non-trivial symmetries due to the operator o_5 and the goal state. Moreover, all states are reachable and there are no dead-ends. Still, the state s_1 and operators o_1, o_2 are redundant. However, there is an endomorphism mapping s_1 to s_2 and operators o_1, o_2 to o_3, o_4, respectively, provided o_3 (resp. o_4) is not more expensive than o_1 (resp. o_2).

The endomorphisms are closely related to symmetries of transition systems. Unlike the symmetries, the endomorphisms need not be bijective. Note that none of the endomorphisms shown in Fig. 1 is bijective.

Once we know we look for endomorphisms, the necessary second step is to describe algorithms computing them. Since classical planning tasks are described in concise representations and it is infeasible to construct their whole state space, we cannot hope for computing all existing endomorphisms. However, we can find at least some of them on concise representations. We propose two methods. The first one uses a set of abstract transition systems (e.g., projections) whose synchronized product is the whole state space. The second method uses the finite domain representation. In both methods, the actual endomorphisms are expressed as solutions to a constraint satisfaction problem (CSP) and solved by a standard CSP solver. The computation itself can be seen as constraint optimization because some of the solutions are better than others.

Background

An **FDR planning task** Π is specified by a tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$. $\mathcal{V}$ is a finite set of **variables**, each variable $V \in \mathcal{V}$ has a finite **domain** $\mathrm{dom}(V)$. A **fact** $\langle V, v \rangle$ is a pair of a variable $V \in \mathcal{V}$ and one of its values $v \in \mathrm{dom}(V)$. The set of all facts is denoted by $\mathcal{F} = \{\langle V, v \rangle \mid V \in \mathcal{V}, v \in \mathrm{dom}(V)\}$, and the set of facts of variable V is denoted by $\mathcal{F}_V = \{\langle V, v \rangle \mid v \in \mathrm{dom}(V)\}$. A **partial state** p is a variable assignment over some variables $\mathrm{vars}(p) \subseteq \mathcal{V}$. We write $p[V]$ for the value assigned to the variable $V \in \mathrm{vars}(p)$ in the partial state p. We also identify p with the set of facts contained in p, i.e., $p = \{\langle V, p[V] \rangle \mid V \in \mathrm{vars}(p)\}$. A partial state s is a **state** if $\mathrm{vars}(s) = \mathcal{V}$. I is an **initial state**. G is a partial state called **goal**, and a state s is a **goal state** iff $G \subseteq s$. The set of all states is denoted by $\mathcal{S}_\Pi$.

$\mathcal{O}$ is a finite set of **operators**, each operator $o \in \mathcal{O}$ has a precondition $\mathrm{pre}(o)$ and effect $\mathrm{eff}(o)$, which are partial states over $\mathcal{V}$, and a cost $\mathrm{c}(o) \in \mathbb{R}_0^+$. An operator o is **applicable** in a state s iff $\mathrm{pre}(o) \subseteq s$. The **resulting state** of applying an applicable operator o in a state s is another state $o[\![s]\!]$ such that $o[\![s]\!][V] = \mathrm{eff}(o)[V]$ for every $V \in \mathrm{vars}(\mathrm{eff}(o))$, and $o[\![s]\!][V] = s[V]$ for every $V \in \mathcal{V} \setminus \mathrm{vars}(\mathrm{eff}(o))$.

A sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ is applicable in a state s_0 if there are states $s_1, \ldots, s_n$ such that o_i is applicable in s_{i-1} and $s_i = o_i[\![s_{i-1}]\!]$ for $i \in \{1, \ldots, n\}$. The resulting state of this application is $\pi[\![s_0]\!] = s_n$ and $\mathrm{c}(\pi) = \sum_{i=1}^{n} \mathrm{c}(o_i)$ denotes the cost of this sequence of operators. A sequence of operators π is called a **plan** iff π is applicable in I and $\pi[\![s]\!]$ is a goal state, π is called an **optimal plan** if its cost is minimal among all plans.

A **labeled transition system** (LTS) is a tuple $\Theta = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star \rangle$, where $\mathcal{S}$ is a finite set of **states**, $\mathcal{L}$ is a finite set of **labels** with associated cost $\mathrm{c}(l) \in \mathbb{R}_0^+$ to each label $l \in \mathcal{L}$, $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$ is a set of **transitions**, $s_I \in \mathcal{S}$ is the **initial state**, and $S_\star \subseteq \mathcal{S}$ is a set of **goal states**. We write $s \xrightarrow{l} s'$ to refer to a transition from s to s' with the label l. A sequence of labels $\pi = \langle l_1, \ldots, l_n \rangle$ is called a **plan** in Θ if there exist $s_{i-1} \xrightarrow{l_i} s_i \in \mathcal{T}$ for every $i \in \{1, \ldots, n\}$ such that $s_0 = s_I$ and $s_n \in S_\star$. The cost of the plan π is $\mathrm{c}(\pi) = \sum_{i=1}^{n} \mathrm{c}(l_i)$. A plan π is called **optimal plan** if the cost of π is minimal among all plans.

The **state space** of a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ is the LTS $\Theta_\Pi = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star \rangle$, where $\mathcal{S} := \mathcal{S}_\Pi$, $s_I := I$, $s \in S_\star$ iff $G \subseteq s$, the labels $\mathcal{L}$ are the operators $\mathcal{O}$ with the given costs, and $s \xrightarrow{o} s'$ is a transition in $\mathcal{T}$ if o is applicable in s and $o[\![s]\!] = s'$.

Given two transition systems $\Theta = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star \rangle$ and $\Theta' = \langle \mathcal{S}', \mathcal{L}, \mathcal{T}', s'_I, S'_\star \rangle$ with a common set of labels $\mathcal{L}$, a **synchronized product** of Θ and Θ' is another transition system $\Theta \otimes \Theta' = \langle \mathcal{S}'', \mathcal{L}, \mathcal{T}'', s''_I, S''_\star \rangle$, where $\mathcal{S}'' = \mathcal{S} \times \mathcal{S}'$, $s''_I = \langle s_I, s'_I \rangle$, $S''_\star = S_\star \times S'_\star$, and $\mathcal{T}'' = \{\langle \langle s, s' \rangle, l, \langle t, t' \rangle \rangle \mid \langle s, l, t \rangle \in \mathcal{T}, \langle s', l, t' \rangle \in \mathcal{T}'\}$.

A **factored LTS** of a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ is a tuple $\langle \Theta_1, \ldots, \Theta_n \rangle$ of LTSs (also called factors) with a common set of labels such that $\Theta_1 \otimes \cdots \otimes \Theta_n = \Theta_\Pi$. Let $\mathcal{V} = \{V_1, \ldots, V_n\}$ denote the set of variables of Π. An **atomic factored LTS** $\langle \Theta_{V_1}, \ldots, \Theta_{V_n} \rangle$ is a factored LTS where Θ_{V_i}, for every $i \in \{1, \ldots, n\}$, is an atomic projection of Π to the variable V_i, i.e., $\Theta_{V_i} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star \rangle$, where $\mathcal{S} = \mathrm{dom}(V_i)$; $\mathcal{L} = \mathcal{O}$; $s_I = I[V_i]$; $S_\star = \{G[V_i]\}$ if $V_i \in \mathrm{vars}(G)$ or $S_\star = \mathcal{S}$ otherwise; and the set of transitions $\mathcal{T}$ consists of transitions $s \xrightarrow{o} t$ for the state $s = \mathrm{pre}(o)[V_i]$ if $V_i \in \mathrm{vars}(\mathrm{pre}(o))$ or $s \in \mathrm{dom}(V_i)$ otherwise, and for the state $t = \mathrm{eff}(o)[V_i]$ if $V_i \in \mathrm{vars}(\mathrm{eff}(o))$ or $t = s$ otherwise.

We have to also recall the **Constraint Satisfaction Problem** (CSP) (for details see, e.g., Russell and Norvig 2010, Chapter 6). Suppose we are given a set X of variables, a finite domain D where the variables can take values, and a set of constraints $\mathcal{C}$. The problem is whether there exists an assignment $h \colon X \to D$ of values D to variables X that satisfies all the constraints. Each constraint $C \in \mathcal{C}$ is specified as a pair $C = \langle \vec{x}, R \rangle$ where $\vec{x}$ is a tuple of variables from X of length n, and R is an n-ary relation on D. The assignment h satisfies the constraint $C = \langle \vec{x}, R \rangle$ if $h(\vec{x}) \in R$. CSPs are often formulated in such a way that each variable $x \in X$ has its domain of allowed values $D_x \subseteq D$. Note that each such restriction can be equivalently expressed as a unary constraint $\langle \langle x \rangle, D_x \rangle$.

For notational convenience, we will sometimes abuse the notation for sequences by referring to a sequence as a set.

Endomorphisms of Transition Systems

Labeled transition systems can be viewed as first-order relational structures used in logic and model theory. More precisely, given an LTS $\Theta = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star \rangle$, one can simply understand Θ as a first-order structure defined on $\mathcal{S} \cup \mathcal{L}$

endowed with unary predicates defining states and labels, a ternary relation $\mathcal{T}$, a constant s_I, and unary predicate $S_\star$. The following definition is nothing else than the usual definition of homomorphism used in model theory (see, e.g., Hodges 1997; Libkin 2004).[1]

Definition 1. Let $\Theta = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star \rangle$ and $\Theta' = \langle \mathcal{S}', \mathcal{L}', \mathcal{T}', s'_I, S'_\star \rangle$ denote two LTSs. A mapping $\alpha \colon \mathcal{S} \cup \mathcal{L} \to \mathcal{S}' \cup \mathcal{L}'$ is an **LTS homomorphism** from Θ to Θ' if the following conditions are satisfied:

(L1) $\alpha(s) \in \mathcal{S}'$ for every $s \in \mathcal{S}$, and

(L2) $\alpha(l) \in \mathcal{L}'$ for every $l \in \mathcal{L}$, and

(L3) $\alpha(s) \xrightarrow{\alpha(l)} \alpha(s') \in \mathcal{T}'$ for every $s \xrightarrow{l} s' \in \mathcal{T}$, and

(L4) $\mathrm{c}(\alpha(l)) \leq \mathrm{c}(l)$ for every $l \in \mathcal{L}$, and

(L5) $\alpha(s_I) = s'_I$, and

(L6) $\alpha(s) \in S'_\star$ for every $s \in S_\star$.

An LTS homomorphism from Θ to Θ is called an **LTS endomorphism** of Θ.

For notational convenience, we extend the mapping α to sequences $\alpha(\langle x_1, \ldots, x_n \rangle) = \langle \alpha(x_1), \ldots, \alpha(x_n) \rangle$ and sets $\alpha(\{x_1, \ldots, x_n\}) = \{\alpha(x_1), \ldots, \alpha(x_n)\}$.

A bijective LTS homomorphism α whose inverse is an LTS homomorphism as well is called an **LTS isomorphism**. If, in addition, α is an LTS endomorphism then α is called an **LTS automorphism**. Note that LTS automorphisms are structural symmetries stabilized for the initial state and goals as described by Shleyfman et al. (2015).

Homomorphisms are important because they preserve existential-positive formulas. In fact, every formula preserved by homomorphisms on all finite structures is equivalent to an existential-positive formula (see Rossman 2008). Consequently, they have to preserve plans. Thus the following theorem follows immediately but we provide an elementary proof for the sake of the reader.

Theorem 2. *Let α denote an LTS endomorphism of Θ. If π is a plan in Θ, then $\alpha(\pi)$ is a plan in Θ.*

Proof. Let $\Theta = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star \rangle$ and let $\pi = s_0 \xrightarrow{l_1} s_1 \xrightarrow{l_2} \cdots \xrightarrow{l_{n-1}} s_{n-1} \xrightarrow{l_n} s_n$, where $s_0 = s_I$ and $s_n \in S_\star$. It follows directly from Definition 1 that (L5) $s_0 = s_I = \alpha(s_I) = \alpha(s_0)$, and (L6) $s_n \in S_\star$ implies $\alpha(s_n) \in S_\star$, and (L3) $s_{i-1} \xrightarrow{l_i} s_i \in \mathcal{T}$ implies $\alpha(s_{i-1}) \xrightarrow{\alpha(l_i)} \alpha(s_i) \in \mathcal{T}$ for every $i \in \{1, \ldots, n\}$. Therefore, $\alpha(\pi)$ is a plan in Θ. □

Corollary 3. *Let α denote an LTS endomorphism of Θ. If π is an optimal plan in Θ, then $\alpha(\pi)$ is an optimal plan in Θ.*

Proof. It follows from (L4) that $\mathrm{c}(\pi) \geq \mathrm{c}(\alpha(\pi))$. Since $\mathrm{c}(\pi)$ is minimal among all plans, it follows that $\mathrm{c}(\pi) = \mathrm{c}(\alpha(\pi))$. The rest follows from Theorem 2. □

[1] The only subtlety is hidden in the cost function. One can regard it as a sort of "fuzzy" predicate which has to be preserved by the homomorphism like other "crisp" predicates. Alternatively, it is possible to encode the condition (L4) as preservation of unary predicates $C_l = \{l' \in \mathcal{L} \mid \mathrm{c}(l') \leq \mathrm{c}(l)\}$ for each label l.

Note that the condition (L4) is required only for the preservation of optimal plans. So, it is possible to weaken the definition of LTS endomorphisms by removing this condition, which could be useful for satisficing planning where we are interested in any plans and not necessarily in the optimal ones. However, we leave this for future work.

The fact that LTS endomorphisms preserve (optimal) plans means that once we have an endomorphism α of an LTS, we can prune the planning task by keeping only its image $\{\alpha(x) \mid x \in \mathcal{S} \cup \mathcal{L}\}$ and throwing away everything else. Formally, we define redundant labels as labels that can be removed as long as there is at least one optimal plan left.

Definition 4. Let Θ denote an LTS with labels $\mathcal{L}$, and let $X \subseteq \mathcal{L}$ denote a set of labels. We say that X is **redundant** in Θ if there exists an optimal plan π in Θ such that $\pi \cap X = \emptyset$.

Theorem 5. *Let α denote an LTS endomorphism of Θ with labels $\mathcal{L}$. Then $X = \{l \in \mathcal{L} \mid \forall l' \in \mathcal{L} : \alpha(l') \neq l\}$ is redundant in Θ.*

Proof. It follows from Corollary 3 that for every optimal plan π such that $\pi \cap X \neq \emptyset$ it holds that $\alpha(\pi)$ is also an optimal plan. And from the definition of X it follows that $\alpha(\pi) \cap X = \emptyset$. □

In practice, we would like to find the largest possible set of redundant labels, which means that we would like to find an LTS endomorphism with the smallest image. Interestingly, the smallest image is unique (up to an isomorphism) and it is called a **core** of the structure. This notion originally comes from graph theory (Hell and Nešetřil 1992), but it turned out that it works also for finite relational structures (e.g., Rossman 2008). The only endomorphisms of the core are automorphisms. Therefore, in theory, there is a unique optimal endomorphism. Unfortunately, it is an NP-complete problem to test whether a finite structure is a core or not. Nevertheless, any non-trivial endomorphism (i.e., endomorphism whose image is smaller than its preimage) allows to prune the LTS.

Now we have come to the question of how to actually compute endomorphisms of an LTS. It is well known that homomorphisms between finite first-order structures can be viewed as solutions to a CSP (e.g., Libkin 2004, Section 14.3). For the case of LTSs the CSP formulation has two kinds of variables: a variable X_s for every state $s \in \mathcal{S}$, and a variable X_l for every label $l \in \mathcal{L}$. The constraints directly corresponds to the conditions (L1)–(L6) from Definition 1. The domain of each X_s is the set of states (L1), and the domain of X_l is the set of all labels whose cost is less than or equal to $\mathrm{c}(l)$ (L2, L4). There is a constraint $\langle \langle X_{s_I} \rangle, \{s_I\} \rangle$ for the initial state s_I (L5), and $\langle \langle X_{s_G} \rangle, S_\star \rangle$ for every goal state $s_G \in S_\star$ (L6). Finally, there is a constraint $\langle \langle X_s, X_l, X_t \rangle, \mathcal{T} \rangle$ for each transition $s \xrightarrow{l} t$ (L3). A solution to such CSP with the minimum number of values assigned to the label variables X_l allows to prune the largest possible amount of labels.

Of course, the above formulation is not practical because we would need to construct the whole state space (LTS) in order to construct the CSP formulation. However, we show

in the next section that we can use a similar approach on factored transition systems.

Endomorphisms of Factored LTSs

We start by showing that endomorphisms of multiple LTSs naturally extend over their synchronized product as long as the mapping of labels is the same for all LTSs.

Definition 6. Given two sets of states $\mathcal{S}$ and $\mathcal{S}'$, a set of labels $\mathcal{L}$, and two mappings $\alpha\colon \mathcal{S}\cup\mathcal{L} \to \mathcal{S}\cup\mathcal{L}$ and $\alpha'\colon \mathcal{S}'\cup\mathcal{L} \to \mathcal{S}'\cup\mathcal{L}$ such that $\alpha(l) = \alpha'(l)$ for every $l \in \mathcal{L}$, $\beta = \alpha\otimes\alpha'$ denotes a new mapping $\beta\colon (\mathcal{S}\times\mathcal{S}')\cup\mathcal{L} \to (\mathcal{S}\times\mathcal{S}')\cup\mathcal{L}$ such that $\beta(l) = \alpha(l) = \alpha'(l)$ for every $l \in \mathcal{L}$ and $\beta(s, s') = \langle\alpha(s), \alpha'(s')\rangle$ for every $s \in \mathcal{S}$ and every $s' \in \mathcal{S}'$.

Theorem 7. *Let $\Theta = \langle\mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star\rangle$, and $\Theta' = \langle\mathcal{S}', \mathcal{L}, \mathcal{T}', s'_I, S'_\star\rangle$ denote two LTSs with a common set of labels, and let α and α' denote LTS endomorphisms of Θ and Θ', respectively. If $\alpha(l) = \alpha'(l)$ for every $l \in \mathcal{L}$, then $\beta = \alpha \otimes \alpha'$ is an LTS endomorphism of $\Theta \otimes \Theta'$.*

Proof. (L1), (L2) and (L4–6) follow trivially from the construction of β and $\Theta \otimes \Theta'$. For (L3), let $\langle s, s'\rangle \xrightarrow{l} \langle t, t'\rangle$ be a transition in $\Theta \otimes \Theta'$. By the definition of transitions in $\Theta \otimes \Theta'$, we have $s \xrightarrow{l} t$ in Θ and $s' \xrightarrow{l} t'$ in Θ'. Thus $\alpha(s) \xrightarrow{\alpha(l)} \alpha(t) \in \mathcal{T}$ and similarly $\alpha'(s') \xrightarrow{\alpha(l)} \alpha'(t') \in \mathcal{T}'$ as $\alpha(l) = \alpha'(l)$. By Definition 6 we have $\beta(s, s') = \langle\alpha(s), \alpha'(s')\rangle$, $\beta(t, t') = \langle\alpha(t), \alpha'(t')\rangle$, and $\beta(l) = \alpha(l)$. Hence $\beta(s, s') \xrightarrow{\beta(l)} \beta(t, t')$ is a transition in $\Theta \otimes \Theta'$. □

Let $\langle\Theta_1, \ldots, \Theta_n\rangle$ denote a factored LTS of a planning task Π. It follows from Theorem 7 that if we find an LTS endomorphism α_i for each individual factor Θ_i and all these endomorphisms agree on the common set of labels, then $\alpha_1 \otimes \cdots \otimes \alpha_n$ is an LTS endomorphism of the whole state space Θ_Π (i.e., $\Theta_1 \otimes \cdots \otimes \Theta_n$). Therefore, we can use this mapping for the inference of redundant labels (operators) according to Theorem 5.

The construction of CSP for the factored LTS follows the same construction laid out in the previous section: Each factor Θ_i is encoded individually, but all encodings for all factors share the same set of variables for labels. This way, every solution to such CSP will produce the same mapping over the set of labels in all factors.

The size of the resulting CSP is polynomial in the size of the factored LTS. Assume we have F many factors each having at most M states, i.e., for an atomic factored LTS $F = |\mathcal{V}|$, and M is the size of the largest domain. Since LTSs corresponding to planning problems are deterministic, each label l can induce at most M many transitions on each factor. So there are at most $|\mathcal{L}| \cdot M \cdot F$ transitions to be encoded into constraints. Moreover, there are at most $|\mathcal{L}| \cdot M$ admissible triples of values for each such constraint. Thus the size of constraints encoding the condition (L3) is bounded by $3 \cdot |\mathcal{L}|^2 \cdot M^2 \cdot F$. The sizes of all other constraints are negligible in comparison to (L3).

Example 8. Fig. 2 shows an example of a planning task. It is easy to see that the factor Θ_{V_1} has an endomorphism α such that $\alpha(v_1) = \alpha(v_2) = \alpha(v_3) = v_3$, $\alpha(o_1) = \alpha(o_2) = \alpha(o_3) = o_3$, $\alpha(p_1) = \alpha(p_2) = \alpha(p_3) = p_3$, and $\alpha(v_0) = v_0$ and $\alpha(v_4) = v_4$. The factor Θ_{V_2} has an endomorphism α' such that $\alpha'(o) = \alpha(o)$ for all operators $o \in \{o_1, o_2, o_3, p_1, p_2, p_3\}$ and $\alpha'(v_b) = \alpha'(v_c) = \alpha'(v_a) = v_a$. By Theorem 7, $\alpha \otimes \alpha'$ is an endomorphism on the synchronized product of both factors. Therefore, the labels (operators) o_1, o_2, p_1, p_2 are redundant by Theorem 5.

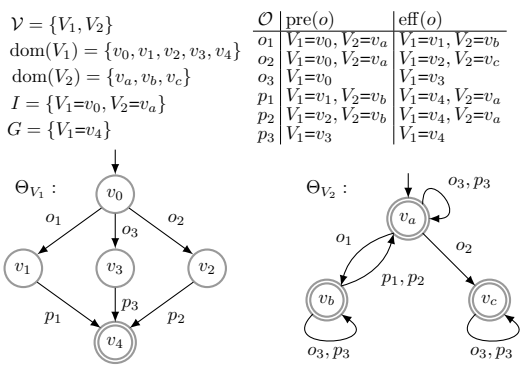

Figure 2: Example planning task with atomic factored LTS $\langle\Theta_{V_1}, \Theta_{V_2}\rangle$. All operators have a unit cost.

Endomorphisms of Finite Domain Representations

The size of the CSP formulation for the factored LTSs can be still computationally prohibitive in practice, because the number of transitions even on relatively small factors may be too large. In this section, we define endomorphisms directly on FDR and show how to formulate a CSP for finding such endomorphisms that is smaller than the CSP for the endomorphisms of the corresponding atomic factored LTS. The price for the smaller CSP is that there can be solutions to the CSP for the atomic factored LTS that are not FDR endomorphisms.

Definition 9. Let $\Pi = \langle\mathcal{V}, \mathcal{O}, I, G\rangle$ be an FDR planning task with facts $\mathcal{F}$. A mapping $\phi\colon \mathcal{F} \cup \mathcal{O} \to \mathcal{F} \cup \mathcal{O}$ is an **FDR endomorphism** of Π if the following conditions are satisfied[2]:

(F1) $\phi(\mathcal{F}_V) \subseteq \mathcal{F}_V$ for every $V \in \mathcal{V}$, and
(F2) $\phi(\mathcal{O}) \subseteq \mathcal{O}$, and
(F3) $\phi(\mathrm{pre}(o)) \supseteq \mathrm{pre}(\phi(o))$ for every $o \in \mathcal{O}$, and
(F4) $\phi(\mathrm{eff}(o)) = \mathrm{eff}(\phi(o))$ for every $o \in \mathcal{O}$, and
(F5) $\mathrm{c}(\phi(o)) \leq \mathrm{c}(o)$ for every $o \in \mathcal{O}$, and
(F6) $\phi(I) = I$, and
(F7) $\phi(G) = G$.

Theorem 10. *Every FDR endomorphism of Π induces an LTS endomorphism of Θ_Π.*

Proof. Let ϕ denote an FDR endomorphism of Π. Note that each fact can be mapped only to facts from the same variable (F1). Therefore, $\mathrm{vars}(p) = \mathrm{vars}(\phi(p))$ holds for every

[2] As in the case of the LTS endomorphism, we extend ϕ to sequences and sets. In particular, we can apply ϕ to partial states.

(partial) state p. So, it is easy to see that $\phi(s) \in \mathcal{S}_\Pi$ for every state s (so (L1) holds); and the initial state is preserved (L5); and the goal states are preserved (L6). It is also easy to see that (L2) and (L4) hold because ϕ maps operators to operators (F2) with possibly smaller costs (F5). What remains to show is that the conditions (F3) and (F4) preserve transitions within Θ_Π (L3).

Let $s \xrightarrow{o} o[\![s]\!]$ denote a transition in Θ_Π. Now, we need to show that $\phi(s) \xrightarrow{\phi(o)} \phi(o[\![s]\!])$ is also a transition in Θ_Π. We already showed that $\phi(s), \phi(o[\![s]\!]) \in \mathcal{S}_\Pi$ and $\phi(o) \in \mathcal{O}$, so what remains to show is that (i) $\phi(o)$ is applicable in $\phi(s)$ and (ii) $\phi(o)[\![\phi(s)]\!] = \phi(o[\![s]\!])$.

By the definition of transitions in Θ_Π we have $\mathrm{pre}(o) \subseteq s$, and from (F3) we have $\mathrm{pre}(\phi(o)) \subseteq \phi(\mathrm{pre}(o)) \subseteq \phi(s)$. Therefore, $\phi(o)$ is applicable in $\phi(s)$.

For (ii), note that $\mathrm{vars}(\mathrm{eff}(o)) = \mathrm{vars}(\phi(\mathrm{eff}(o))) = \mathrm{vars}(\mathrm{eff}(\phi(o)))$. Further observe that $p[V] = p'[V]$ implies $\phi(p)[V] = \phi(p')[V]$ for any partial states p, p' and any variable $V \in \mathcal{V}$. Therefore, for every $V \in \mathrm{vars}(\mathrm{eff}(o))$ it holds that $\phi(o[\![s]\!])[V] = \phi(\mathrm{eff}(o))[V] = \mathrm{eff}(\phi(o))[V] = \phi(o)[\![\phi(s)]\!][V]$, and for every $V' \in \mathcal{V} \setminus \mathrm{vars}(\mathrm{eff}(o))$ it holds that $\phi(o[\![s]\!])[V'] = \phi(s)[V'] = \phi(o)[\![\phi(s)]\!][V']$. □

Each FDR endomorphism induces endomorphisms on its respective variable projections and all these endomorphisms agree on labels. So, every FDR endomorphism is an endomorphism of the atomic factored LTS of the same FDR planning task. On the other hand, there are endomorphisms of the atomic factored LTS that are not FDR endomorphisms.

Example 11. In the example planning task in Fig. 2, there is an FDR endomorphism ϕ such that $\phi(o_2) = o_1, \phi(p_2) = p_1$, $\phi(v_2) = v_1$, and $\phi(v_c) = v_b$ (the rest of the mapping is identity, i.e., $\phi(x) = x$ for $x \notin \{o_2, p_2, v_2, v_c\}$). Note that in the definition of ϕ we tacitly identified facts with their values, e.g. $\langle V_1, v_2 \rangle$ was identified with v_2.

Therefore, the operators (labels) o_2, p_2 are redundant. However, the LTS endomorphism $\alpha \otimes \alpha'$ from Example 8 is not an FDR endomorphism. For instance, we cannot map o_1 to o_3 because $\mathrm{vars}(\mathrm{eff}(o_1)) \neq \mathrm{vars}(\mathrm{eff}(o_3))$ which is required for the condition (F4) to hold.

To formulate an FDR endomorphism as a CSP we strengthen the condition (F3) as $\phi(\mathrm{pre}(o)) = \mathrm{pre}(\phi(o))$ for all operators $o \in \mathcal{O}$ to simplify the CSP formulation, and we directly encode the conditions (F1)–(F7) as constraints. For every variable $V \in \mathcal{V}$ and every value $v \in \mathrm{dom}(V)$ we define a CSP variable X_v with the domain $\mathrm{dom}(V)$ (F1) (we assume w.l.o.g. that variable domains are pairwise disjoint), and for every operator $o \in \mathcal{O}$ we define a CSP variable X_o with the domain $D_o = \{o' \in \mathcal{O} \mid \mathrm{c}(o') \leq \mathrm{c}(o)\}$ (F2, F5). The initial state $I = \{\langle V_1, v_1 \rangle, \ldots, \langle V_n, v_n \rangle\}$ is encoded as the constraint $\langle \langle X_{v_1}, \ldots, X_{v_n} \rangle, \{\langle v_1, \ldots, v_n \rangle\} \rangle$ (F6), and the goal $G = \{\langle V_{i_1}, v_{i_1} \rangle, \ldots, \langle V_{i_k}, v_{i_k} \rangle\}$ as the the constraint $\langle \langle X_{v_{i_1}}, \ldots, X_{v_{i_k}} \rangle, \{\langle v_{i_1}, \ldots, v_{i_k} \rangle\} \rangle$ (F7).

Finally, every operator o is encoded with two constraints $C_{\mathrm{pre}(o)}$ and $C_{\mathrm{eff}(o)}$ as follows. Let $o \in \mathcal{O}$ denote an operator with $\mathrm{pre}(o) = \{\langle V_{i_1}, v_{i_1} \rangle, \ldots, \langle V_{i_k}, v_{i_k} \rangle\}$. The strengthened condition (F3) is encoded with $C_{\mathrm{pre}(o)} = \langle \langle X_o, X_{v_{i_1}}, \ldots, X_{v_{i_k}} \rangle, \mathcal{U}_o \rangle$ where $\mathcal{U}_o = \{\langle o', \mathrm{pre}(o')[V_{i_1}], \ldots, \mathrm{pre}(o')[V_{i_k}] \rangle \mid o' \in D_o, \mathrm{vars}(\mathrm{pre}(o')) = \mathrm{vars}(\mathrm{pre}(o))\}$. The condition (F4) is encoded with $C_{\mathrm{eff}(o)}$ constructed analogously to $C_{\mathrm{pre}(o)}$ but from $\mathrm{eff}(o)$.

The size of the CSP formulation is polynomial in the size of FDR and, again, the size of operator constraints $C_{\mathrm{pre}(o)}$ and $C_{\mathrm{eff}(o)}$ is the most critical. For each operator $o \in \mathcal{O}$ there are $2 \cdot |\mathcal{O}|$ constraints $C_{\mathrm{pre}(o)}$ and $C_{\mathrm{eff}(o)}$ altogether. The arities of these constraints are bounded by the number of variables $\mathcal{V}$. For each of them, there are at most $|\mathcal{O}|$ many $|\mathcal{V}|$-tuples of allowed values. Thus the size of these constraints is bounded by $2 \cdot |\mathcal{O}|^2 \cdot |\mathcal{V}|$. Nevertheless, this bound is often too pessimistic because the domains of preconditions and effects are often much smaller than $|\mathcal{V}|$ and operators are rarely allowed to be mapped to all operators.

If we rewrite the bound for factored LTS $3 \cdot |\mathcal{L}|^2 \cdot M^2 \cdot F$ for the case of an atomic factored LTS of the FDR planning task, we get $3 \cdot |\mathcal{O}|^2 \cdot d_{\max}^2 \cdot |\mathcal{V}|$, where $d_{\max}$ is the size of the largest domain. So, it is easy to see that for a given FDR planning tasks, the FDR endomorphism has a smaller CSP formulation then the corresponding LTS endomorphism (even if we consider the too pessimistic bound for the FDR endomorphism).

Related Work

The investigated endomorphisms are related to the already studied notion of structural symmetry (e.g., Pochter, Zohar, and Rosenschein 2011; Shleyfman et al. 2015), which is nothing else than an automorphism of the transition system in terms of model theory. Moreover, the homomorphisms are related to abstractions. In particular, abstractions form a certain subclass of homomorphisms because they always act as identity on the labels and are always surjective. However, abstractions were used mainly for computing heuristics and not for a detection of redundant operators.

The most commonly used method for removing redundant operators is reachability analysis using h^m heuristic, usually for $m = 2$, in forward and backward direction (Alcázar and Torralba 2015). This method can, however, find only the redundant operators that are either unreachable or lead to a dead-end state, i.e., they can never be part of any plan. Our proposed method via endomorphisms is therefore complementary—it can, in some cases, prune unreachable or dead-end states/operators, but it does not dominate h^m-based methods. Consider, for example, an LTS with an unreachable part which cannot be mapped into the reachable part by an endomorphism. Conversely, our method finds redundant operators in situations where h^m pruning fails. Consider for instance the examples in Fig. 1 as there are neither unreachable states nor dead-ends.

Another related method is the combination of operator mutexes (op-mutexes) and symmetries proposed by Fišer, Torralba, and Shleyfman (2019). The main idea behind this technique is based on the observation that if two operators cannot co-occur in the same optimal plan (i.e., they form an op-mutex) and there is a symmetry (stabilized for both the initial state and goals) mapping one operator to another, then one of the operators can be safely removed. So, as our

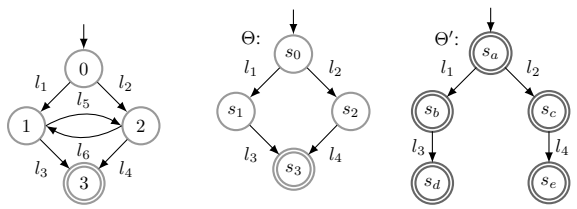

Figure 3: (a) an example of a symmetric LTS; (b) a factored LTS $\langle \Theta, \Theta' \rangle$.

method, this method can prune operators that can be part of some plan, but it requires an existence of a non-trivial symmetry, whereas our method does not need any symmetries. Again, consider the LTS depicted in Fig. 1 (right) having no non-trivial symmetry. Conversely, there are symmetric transition systems where the method employing op-mutexes finds redundant operators but our method fails. Consider the LTS depicted in Fig. 3 (a). All labels have a unit cost. The shown LTS is in fact a core. The operators l_1 and l_2 form an op-mutex and there is a symmetry mapping l_1 to l_2, which means that l_1 (or l_2) can be removed as redundant.

Our method is also tightly related to methods applying a dominance relation to prune states and operators during search. Torralba and Hoffmann (2015) used cost-simulations which are dominance relations respecting transitions. A binary relation $\preceq \subseteq \mathcal{S} \times \mathcal{S}$ is said to be **goal-respecting** if $s \preceq t$, $s \in S_\star$ implies $t \in S_\star$, and $\preceq$ is called a **cost-simulation** for Θ if, whenever $s \preceq t$, for every transition $s \xrightarrow{l} s'$ there is a transition $t \xrightarrow{l'} t'$ such that $s' \preceq t'$ and $\mathrm{c}(l') \leq \mathrm{c}(l)$.

Lemma 12. *An LTS endomorphism α (viewed as a relation) is a goal-respecting cost-simulation.*

Proof. An endomorphism α induces a binary relation $\preceq$ on states consisting of pairs of the form $\langle s, \alpha(s) \rangle$, i.e., we define $s \preceq t$ iff $t = \alpha(s)$. Since $\alpha(S_\star) \subseteq S_\star$, we have $\alpha(s) \in S_\star$ provided that $s \in S_\star$, i.e., $\preceq$ is goal-respecting.

Next, we have to prove that $\preceq$ is a cost-simulation. Assume $s \preceq t$ and $s \xrightarrow{l} s'$. We have to find t' and l' such that $t \xrightarrow{l'} t'$, $s' \preceq t'$, and $\mathrm{c}(l') \leq \mathrm{c}(l)$. By definition of $\preceq$ we have $t = \alpha(s)$. As α is an endomorphism, it follows that $t = \alpha(s) \xrightarrow{\alpha(l)} \alpha(s')$ and $\mathrm{c}(\alpha(l)) \leq \mathrm{c}(l)$. Since $s' \preceq \alpha(s')$, $\alpha(s')$ is the witnessing state t' and $\alpha(l)$ is the witnessing label l'. □

Therefore, LTS endomorphisms are just functional goal-respecting cost-simulations. Similarly, as it is infeasible to search for endomorphisms directly on the LTS induced by a planning task, cost-simulations have to be computed on the factored LTS as well. Thus Torralba and Hoffmann (2015) defined a label-dominance (LD) simulation. Given a factored LTS $\langle \Theta_1, \ldots, \Theta_n \rangle$, a tuple $\langle \preceq_1, \ldots, \preceq_n \rangle$ of relations, each $\preceq_i$ being a goal-respecting relation on Θ_i, is called **label-dominance simulation** if, whenever $s \preceq_i t$ in Θ_i, for every transition $s \xrightarrow{l} s'$ in Θ_i there is a transition $t \xrightarrow{l'} t'$ in Θ_i such that $s' \preceq_i t'$ and $\mathrm{c}(l') \leq \mathrm{c}(l)$ and for all $j \neq i$ the label l' dominates l in Θ_j given $\preceq_j$. We say that l' dominates l in Θ_j given $\preceq_j$ if for all $s \xrightarrow{l} t$ in Θ_j there exists $s \xrightarrow{l'} t'$ in Θ_j such that $t \preceq_j t'$.

Note that particular relations $\preceq_i$ in the LD simulation are goal-respecting cost-simulations. Thus it is reasonable to ask whether the endomorphism obtained by Theorem 7 forms an LD simulation. The answer is negative as shown in the following example. Thus we get a separation of endomorphisms from LD simulations. Consider the factored LTS consisting of two factors Θ and Θ' depicted in Fig. 3 (b). Assume that $\mathrm{c}(l_1) \leq \mathrm{c}(l_2)$ and $\mathrm{c}(l_3) \leq \mathrm{c}(l_4)$. There are clearly non-trivial endomorphisms on both LTSs such that $l_2 \mapsto l_1$, $l_4 \mapsto l_3$, $s_2 \mapsto s_1$, $s_c \mapsto s_b$, and $s_e \mapsto s_d$. Consequently, by Theorem 7 there is a non-trivial endomorphism on the synchronized product $\Theta \otimes \Theta'$ mapping $\langle s_2, s_c \rangle$ to $\langle s_1, s_b \rangle$. Nevertheless, there is no LD simulation which would allow to prune state $\langle s_2, s_c \rangle$. Indeed, suppose there are cost-simulations $\preceq$ and $\preceq'$ respectively on Θ and Θ' such that $s_2 \preceq s_1$ and $s_c \preceq' s_b$. For the transition $s_2 \xrightarrow{l_4} s_3$ there has to be a transition going from s_1. The only one is $s_1 \xrightarrow{l_3} s_3$. Consequently, l_3 has to dominate l_4 in Θ' by the definition of LD-simulation. However, this is not the case because l_3 is not applicable in the state s_c.

Experimental Evaluation

The inference of endomorphisms and pruning of redundant operators was implemented[3] in C and experimentally evaluated on a cluster of computing nodes with Intel Xeon Scalable Gold 6146 processors. We used all planning domains from the optimal tracks of International Planning Competitions (IPCs) from 1998 to 2018 excluding the ones containing conditional effects after translation (leaving 65 domains). We used fact-alternating mutex groups (Fišer and Komenda 2018; Fišer 2020) for the construction of FDR variables and for removing unreachable and dead-end facts and operators. For comparison, we used pruning using h^2 heuristic in forward and backward direction (Alcázar and Torralba 2015) which we refer to as h^2 pruning.

For solving CSPs, we used CP Optimizer from IBM ILOG CPLEX Optimization Studio v12.9. This solver contains an objective function that returns the number of different values assigned to a specified set of variables. We use the minimization of such objective function over operator (label) variables to obtain optimal solutions. Moreover, CP Optimizer is able to report all sub-optimal solutions found during the search for the optimal solution—we use this feature to obtain a sub-optimal solution for tasks where the search terminates prematurely because of a time or memory limit.

In the first set of experiments we aimed at (i) finding which domains contain non-trivial (i.e., non-identity) endomorphisms; (ii) how many operators can be identified as redundant by endomorphisms; (iii) how often can we find an optimal solution and how often the search for the op-

[3] https://gitlab.com/danfis/cpddl, branch aaai21-endomorphism

domain	fdr without h^2				ts without h^2				h^2	fdr with h^2					ts with h^2				
	red	fail	sym	%avg	red	fail	sym	%avg	%avg	red	fail	sym	%avg	$\%avg^2$	red	fail	sym	%avg	$\%avg^2$
airport04 (50)	0	29	0	0.00	16	29	0	2.00	73.41	0	29	0	73.41	0.00	0	29	0	73.41	0.00
blocks00 (35)	0	0	0	0.00	35	0	35	70.24	0.00	0	0	0	0.00	0.00	35	0	35	70.24	70.24
caldera18 (20)	0	0	0	0.00	(5) 8	12	0	28.44	55.35	0	0	0	55.35	0.00	(5) 11	9	0	74.53	34.09
cavediving14 (20)	8	0	5	13.66	3	13	1	6.24	0.54	8	0	5	14.20	13.72	3	13	1	6.79	6.29
data-network18 (20)	0	0	0	0.00	19	1	0	2.81	0.00	0	0	0	0.00	0.00	19	1	0	2.81	2.81
mystery98 (30)	4	3	8	1.10	8	16	10	6.28	46.39	3	12	2	46.79	0.87	1	21	1	46.64	0.42
organic-synthesis18 (20)	0	14	0	0.00	6	14	0	55.42	89.96	0	14	0	89.96	0.00	3	14	0	92.89	27.78
parcprinter08 (30)	26	0	8	5.23	28	0	17	20.38	44.87	0	0	0	44.87	0.00	20	0	16	57.46	22.35
parcprinter11 (20)	18	0	8	5.57	19	0	13	17.64	44.20	0	0	0	44.20	0.00	12	0	10	55.96	20.25
pathways06 (30)	0	0	0	0.00	5	10	0	0.05	3.94	0	0	0	3.94	0.00	5	10	0	3.98	0.05
pegsol08 (30)	2	0	0	0.30	3	0	0	0.38	19.14	0	0	0	19.14	0.00	0	0	0	19.14	0.00
pipesworld-notank04 (50)	0	0	0	0.00	12	18	4	1.51	6.12	0	0	0	6.12	0.00	0	14	0	6.12	0.00
psr-small04 (50)	0	0	0	0.00	38	0	0	2.49	23.51	0	0	0	23.51	0.00	11	0	0	23.75	0.25
rovers06 (40)	38	0	5	21.74	17	21	5	8.82	46.68	38	0	5	57.99	21.74	17	21	5	51.61	8.82
spider18 (20)	0	1	0	0.00	3	17	0	0.43	13.26	0	2	0	13.26	0.00	0	17	0	13.26	0.00
storage06 (30)	0	0	0	0.00	2	13	1	1.37	0.00	0	0	0	0.00	0.00	2	13	1	1.37	1.37
tidybot11 (20)	0	0	0	0.00	9	11	0	14.70	52.16	0	0	0	52.16	0.00	0	8	0	52.16	0.00
tpp06 (30)	10	2	0	7.69	(2) 6	14	0	2.53	39.74	10	2	0	43.50	7.76	4	12	0	41.60	3.32
transport08 (30)	13	0	12	2.69	1	21	1	0.29	0.00	13	0	12	2.69	2.69	1	21	1	0.29	0.29
transport11 (20)	11	0	11	5.10	1	16	1	0.43	0.00	11	0	11	5.10	5.10	1	16	1	0.43	0.43
transport14 (20)	15	0	14	6.44	1	17	1	0.99	0.00	15	0	14	6.44	6.44	1	17	1	0.99	0.99
trucks06 (30)	0	14	0	0.00	12	18	0	1.19	29.46	0	0	0	29.46	0.00	0	16	0	29.46	0.00
visitall11 (20)	6	0	6	5.19	6	0	6	5.19	0.00	6	0	6	5.19	5.19	6	0	6	5.19	5.19
visitall14 (20)	6	0	6	1.28	6	0	6	1.28	0.00	6	0	6	1.28	1.28	6	0	6	1.28	1.28
woodworking08 (30)	0	0	0	0.00	30	0	0	21.52	48.54	0	0	0	48.54	0.00	23	0	0	49.94	2.61
woodworking11 (20)	0	0	0	0.00	20	0	0	21.08	49.01	0	0	0	49.01	0.00	18	0	0	50.54	2.95
overall from above (735)	157	63	83	3.13	314	261	101	10.34	24.10	110	59	61	25.97	2.69	199	252	84	30.22	7.94

Table 1: *red*: number of tasks with at least one redundant operator (all non-trivial endomorphisms were optimal, except in caldera18 and tpp06, where the number of optimal solutions is given in parenthesis); *fail*: number of tasks where the inference failed due to a time or memory limit; *sym*: number of tasks in which symmetries appear after pruning; *%avg*: average percentage of removed operators per task within the whole domain; *%avg*^{2}: same as *%avg* but the basis is the number of operators after h^2 pruning, i.e., it measures the number of removed operators on top of those already removed by h^2; *overall* is a sum for *red*, *fail*, and *sym*, and it is the average percentage of removed operators over all tasks for *%avg* and *%avg*^{2}.

timal solution fails because it requires too many computational resources; and (iv) whether the theoretical dominance of the LTS endomorphism over the FDR endomorphism can be actually measured on the standard benchmark set. To that end, we compared the FDR endomorphism (denoted by `fdr`) with the LTS endomorphism of the factored LTS constructed from projections to the found mutex groups (denoted by `ts`).

Table 1 shows the results where the time limit for the inference of endomorphisms was set to 90 seconds and the memory limit was set to 16 GB. The table shows only the domains in which at least one variant found at least one redundant operator (or in other words, where at least one variant found a non-trivial endomorphism). The left side of the table shows results without the h^2 pruning and the right side show the behaviour of our pruning when used after the tasks were already pruned using h^2.

Even with h^2 pruning, we were able to find non-trivial FDR endomorphisms in 9 domains (out of 65) and in most of other domains, we were able to prove that there exist only identity FDR endomorphisms. Non-trivial LTS endomorphisms after h^2 pruning were found in 20 domains, but the inference failed on a memory limit much more frequently (the time limit was an issue in a very few tasks). For example, we were not able to find out whether there is an LTS endomorphism in domains such as agricola18, airport04, parking11/14, or petri-net-alignment18. The relative number of pruned operators varied greatly depending on the domain, which is an expected behaviour.

The comparison between the variants with and without h^2 pruning clearly demonstrate that, on one hand, endomorphisms can sometimes prune a subset of unreachable operators found by h^2 (e.g., in airport04, pegsol08, pipesworld-notankage04, spider18, tidybot11, and trucks06). And on the other hand, endomorphisms are able to prune operators that are reachable and thus undetectable by h^2, e.g., blocks00, transport08/11/14, and visitall08/11 are domains without unreachable or dead-end operators and endomorphisms were able to detect redundant operators there.

As we already described in the section on endomorphisms of transition systems, the only non-identity endomorphisms of a core are automorphisms, i.e., symmetries. So, we were also wondering whether removing redundant operators using endomorphisms can expose symmetries (on the so-called Problem Description Graph (Pochter, Zohar, and Rosenschein 2011)) that could not be found otherwise. As can be seen in Table 1, this behaviour is actually quite common, which is a promising result considering other planning methods depending on structural symmetries of planning tasks (e.g., Sievers et al. 2015; Shleyfman et al. 2015; Gnad et al. 2017; Fišer, Torralba, and Shleyfman 2019).

domain	lmc		pot		ms		symba		scrp	
	B	E	B	E	B	E	B	E	B	E
blocks00 (35)	28	**35**	29	**35**	21	**35**	31	**35**	28	**35**
cavediving14 (20)	7	7	7	7	7	7	8	**10**	7	7
hiking14 (20)	11	11	14	14	14	14	**16**	15	16	16
parcprinter08 (30)	23	**24**	28	28	25	**26**	22	**24**	30	30
parcprinter11 (20)	18	**19**	20	20	19	19	17	**18**	20	20
rovers06 (40)	9	9	8	8	8	8	14	14	10	**11**
tetris14 (17)	9	9	17	17	13	13	11	11	**14**	13
transport14 (20)	6	6	7	**8**	7	**8**	9	9	10	10
visitall14 (20)	6	6	13	13	13	13	6	**7**	13	13
woodworking08 (30)	24	**25**	19	19	17	17	28	28	30	30
woodworking11 (20)	16	**17**	14	14	12	12	20	20	20	20
overall (1697)	885	**896**	1 025	**1 032**	935	**951**	1 002	**1 011**	1 113	**1 120**

Table 2: Number of solved tasks per domain and over all 65 domains. *B*: base variant with h^2 and without endomorphisms, *E*: pruning with h^2 and endomorphisms.

In the second set of experiments, we applied the pruning using endomorphisms in various planners and counted the number of solved tasks under 30 minutes time limit and 16 GB memory limit. Since FDR endomorphisms are easier to infer and LTS endomorphisms can be found in more domains, we decided to combine both methods. We run the inference of FDR endomorphisms and after that the inference of LTS endomorphisms both with 90 seconds time limit. However, if the inference of FDR endomorphisms fails on a memory limit, we skip the inference of LTS endomorphisms, because the previous data showed that it is also very likely to fail. Moreover, we combined our proposed pruning with the h^2 pruning, because the previous set of experiment showed that the combination of the two is able to prune more operators.

We used the heuristic search planner Fast Downward (Helmert 2006) with A* and the LM-Cut (`lmc`) heuristic (Helmert and Domshlak 2009); the merge-and-shrink (`ms`) heuristic with SCC-DFP merge strategy and non-greedy bisimulation shrink strategy (Helmert et al. 2014; Sievers, Wehrle, and Helmert 2016); the potential (`pot`) heuristic enhanced with disambiguations and optimized for all syntactic states with added constraint for the initial state (Seipp, Pommerening, and Helmert 2015; Fišer, Horčík, and Komenda 2020); and the Scorpion planner (`scrp`) (Seipp 2018; Seipp and Helmert 2018) that performed very well in the last IPC 2018; and the symbolic planner SymbA* (`symba`) (Torralba et al. 2017) used as a baseline in IPC 2018.

Table 2 shows that the number of solved tasks is moderately increased due to the pruning, and the extra time spent in the inference of endomorphisms impacts the results negatively only in a very few cases. The table shows only the domains in which the pruning with endomorphisms resulted in a different number of solved tasks than in the case without endomorphisms for at least one of the planners. To demonstrate that the pruning has a positive effect even on tasks that were solved by both variants within the time limit, we show the number of expanded states (excluding the last f layer) as scatter plots for `lmc`, `pot`, and `ms` in Fig. 4. One can clearly see, that, on one hand, the number of expanded states is rarely higher with pruning (due to a differently constructed heuristic function). And on the other hand, pruning using endomorphisms can greatly reduce the number of expanded states.

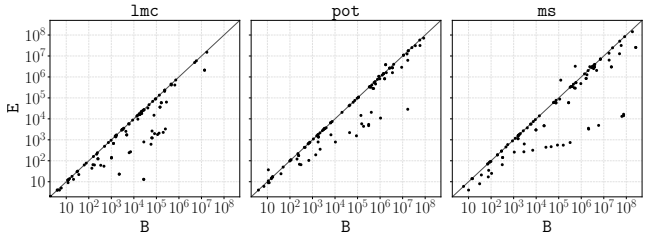

Figure 4: Number of expansions before the last f layer for tasks solved both with and without pruning using endomorphisms where at least one operator was pruned. *B* and *E* as in Table 2.

Conclusions

We introduced the notion of endomorphism into classical planning and we showed how to use it for detection of redundant operators. We proposed two methods for computing endomorphisms via a CSP solver. Experimental evaluation showed that a substantial number of IPC domains exhibit non-trivial endomorphisms. Nevertheless, it is necessary to further explore efficient methods for computing them, because, on one hand, our method for factored LTSs might produce too large CSP instances. And on the other hand, FDR endomorphisms are maybe too weak as only a few domains contain non-trivial FDR endomorphisms.

Acknowledgements

This work was funded by the Czech Science Foundation (grant no. 18-24965Y) and by DFG grant 389792660 as part of TRR 248 (see https://perspicuous-computing.science). The experimental evaluation was supported by the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

References

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Plan-

ning. In *Proc. ICAPS'15*, 2–6.

Fišer, D. 2020. Lifted Fact-Alternating Mutex Groups and Pruned Grounding of Classical Planning Problems. In *Proc. AAAI'20*, 9835–9842.

Fišer, D.; Horčík, R.; and Komenda, A. 2020. Strengthening Potential Heuristics with Mutexes and Disambiguations. In *Proc. ICAPS'20*, 124–133.

Fišer, D.; and Komenda, A. 2018. Fact-Alternating Mutex Groups for Classical Planning. *Journal of Artificial Intelligence Research* 61: 475–521.

Fišer, D.; Torralba, Á.; and Shleyfman, A. 2019. Operator Mutexes and Symmetries for Simplifying Planning Tasks. In *Proc. AAAI'19*, 7586–7593.

Gnad, D.; Torralba, Á.; Shleyfman, A.; and Hoffmann, J. 2017. Symmetry Breaking in Star-Topology Decoupled Search. In *Proc. ICAPS'17*, 125–134.

Haslum, P.; and Geffner, H. 2000. Admissible Heuristics for Optimal Planning. In *Proc. AIPS'00*, 140–149.

Hell, P.; and Nešetřil, J. 1992. The core of a graph. *Discrete Mathematics* 109(1-3): 117–126.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proc. ICAPS'09*, 162–169.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge & Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the Association for Computing Machinery* 61(3): 16.1–16.63.

Hodges, W. 1997. *A Shorter Model Theory*. Cambridge University Press. ISBN 978-0-521-58713-6.

Libkin, L. 2004. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer. ISBN 978-3-540-21202-7.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting Problem Symmetries in State-Based Planners. In *Proc. AAAI'11*.

Rossman, B. 2008. Homomorphism preservation theorems. *Journal of the Association for Computing Machinery* 55(3): 15:1–15:53.

Russell, S.; and Norvig, P. 2010. *Artificial Intelligence: A Modern Approach (Third Edition)*. Englewood Cliffs, NJ: Prentice-Hall. ISBN 0-13-103805-3.

Seipp, J. 2018. Fast Downward Scorpion. In *IPC 2018 planner abstracts*, 77–79.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *Journal of Artificial Intelligence Research* 62: 535–577.

Seipp, J.; Pommerening, F.; and Helmert, M. 2015. New Optimization Functions for Potential Heuristics. In *Proc. ICAPS'15*, 193–201.

Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; and Wehrle, M. 2015. Heuristics and Symmetries in Classical Planning. In *Proc. AAAI'15*, 3371–3377.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An Analysis of Merge Strategies for Merge-and-Shrink Heuristics. In *Proc. ICAPS'16*, 294–298.

Sievers, S.; Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015. Factored Symmetries for Merge-and-Shrink Abstractions. In *Proc. AAAI'15*, 3378–3385.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient symbolic search for cost-optimal planning. *Artificial Intelligence* 242: 52–79.

Torralba, Á.; and Hoffmann, J. 2015. Simulation-Based Admissible Dominance Pruning. In *Proc. IJCAI'15*, 1689–1695.