Generalized Label Reduction for Merge-and-Shrink Heuristics

Silvan Sievers and **Martin Wehrle** and **Malte Helmert**
Universität Basel
Basel, Switzerland
{silvan.sievers,martin.wehrle,malte.helmert}@unibas.ch

Abstract

Label reduction is a technique for simplifying families of labeled transition systems by dropping distinctions between certain transition labels. While label reduction is critical to the efficient computation of merge-and-shrink heuristics, current theory only permits reducing labels in a limited number of cases. We generalize this theory so that labels can be reduced in *every* intermediate abstraction of a merge-and-shrink tree. This is particularly important for efficiently computing merge-and-shrink abstractions based on *non-linear* merge strategies. As a case study, we implement a non-linear merge strategy based on the original work on merge-and-shrink heuristics in model checking by Dräger et al.

Introduction

State-space search is a fundamental problem in artificial intelligence. Many state spaces of interest, including those that arise in classical planning and in the verification of safety properties in model checking, can be compactly specified as a family of labeled transition systems (e. g., Helmert, Haslum, and Hoffmann 2008; Dräger, Finkbeiner, and Podelski 2009).

Label reduction identifies and eliminates semantically equivalent labels in such transition systems. It was originally introduced as an efficiency improvement for *merge-and-shrink abstractions* (Helmert, Haslum, and Hoffmann 2007). Later, Nissim, Hoffmann, and Helmert (2011a) showed that label reduction can (in some cases) exponentially reduce the representation size of abstractions based on *bisimulation*.

All implementations of merge-and-shrink abstractions described in the planning literature apply label reduction whenever possible: it has no negative impact on abstraction quality, is very fast to compute, and significantly reduces time and memory required to compute an abstraction. However, the current theory of merge-and-shrink abstractions only allows reducing labels in limited cases.

Broadly speaking, the merge-and-shrink approach consists in constructing a set of *atomic* transition systems, each corresponding to a single state variable of the problem, and then iteratively *merging* two transition systems into a larger one until only one transition system remains, which then induces a heuristic for an overall state-space search algorithm.

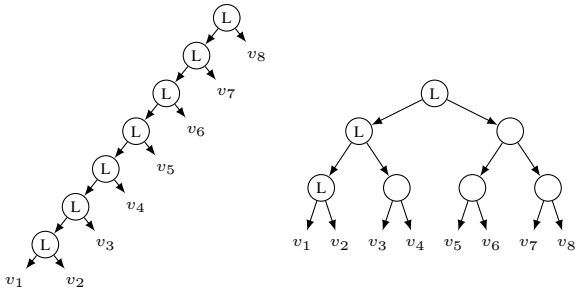

Figure 1: Two merge trees for a problem with 8 state variables. Previous theory allows reducing labels in the intermediate abstractions marked with an "L" when v_1 is the pivot.

Intermediate results can be *shrunk* to trade off computation effort against heuristic accuracy. A so-called *merge strategy* decides which transition systems to merge in each step of the algorithm. The merge strategy defines a binary tree over the atomic transition systems, the so-called *merge tree*.

Figure 1 shows two possible merge trees for a state space with 8 atomic transition systems, defined by state variables $v_1, \ldots, v_8$. The left part of the figure shows a merge tree which degenerates to a list; such merge trees correspond to so-called *linear merge strategies* (Helmert, Haslum, and Hoffmann 2007). The right part shows a complete merge tree, corresponding to a *non-linear merge strategy*. According to current theory (Nissim, Hoffmann, and Helmert 2011a), when defining a merge strategy, one must select a single leaf of the merge tree, called a *pivot*, and may only reduce labels after merge operations which correspond to ancestors of the pivot in the merge tree. In general, this means that with a complete tree over n atomic transition systems, only $O(\log n)$ of the merged transition systems can have their labels reduced.

We introduce a generalized concept of label reduction to overcome this limitation. The generalization is introduced in a declarative way, independently of the merge-and-shrink framework. It is conceptually much easier to understand than the previous theory, yet more powerful in the sense that it allows reducing to a smaller set of labels than previous techniques and in the sense that it can be applied safely to *every* intermediate abstraction of a merge tree.

Generalized label reduction is particularly beneficial for the efficient computation of merge-and-shrink abstractions with non-linear merge strategies. As a case study, we have implemented such a merge strategy, based on the original work on merge-and-shrink heuristics in model checking by Dräger, Finkbeiner, and Podelski (2006), which did not make use of label reduction. We show experimental results that highlight the usefulness of generalized label reduction in general and non-linear merge strategies in particular.

Planning Tasks

We present our techniques with the terminology of automated planning, but note that they are applicable to factored transition systems in general. We consider planning tasks in the SAS$^+$ formalism (Bäckström and Nebel 1995) augmented with action costs. A *planning task* is a 4-tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$, where $\mathcal{V}$ is a finite set of *state variables*, $\mathcal{O}$ is a finite set of *operators*, s_0 is the *initial state* and $s_\star$ is the *goal*.

Each variable $v \in \mathcal{V}$ has a finite domain $\mathcal{D}(v)$. A *partial state* s is a variable assignment on a subset of $\mathcal{V}$, denoted by $\mathit{vars}(s)$. We write $s[v]$ for the value assigned to $v \in \mathit{vars}(s)$, which must satisfy $s[v] \in \mathcal{D}(v)$. We say that s *complies* with partial state s' if $s[v] = s'[v]$ for all $v \in \mathit{vars}(s) \cap \mathit{vars}(s')$. A partial state s is a *state* if $\mathit{vars}(s) = \mathcal{V}$.

Each operator $o \in \mathcal{O}$ has a *precondition* $\mathit{pre}(o)$ and *effect* $\mathit{eff}(o)$, which are partial states, and a *cost* $c(o) \in \mathbb{R}_0^+$. An operator o is *applicable* in a state s if s complies with $\mathit{pre}(o)$, in which case o can be *applied*, resulting in the *successor state* s' that complies with $\mathit{eff}(o)$ and satisfies $s'[v] = s[v]$ for all $v \notin \mathit{vars}(\mathit{eff}(o))$.

The initial state s_0 is a state; the goal $s_\star$ is a partial state.

A *plan* is a sequence $o_1, \ldots, o_n \in \mathcal{O}$ of operators which are applicable, in order, to the initial state, resulting in a state that complies with the goal. Such a plan is *optimal* if $\sum_{i=1}^n c(o_i)$ is minimal among all plans. The objective of optimal planning is to find an optimal plan for a planning task or to prove that no plan exists.

Transition Systems and Merge-and-Shrink

We briefly recap the key ideas behind merge-and-shrink abstractions (e. g., Helmert, Haslum, and Hoffmann 2007). The central notion in this context is the explicit manipulation of *transition systems*. We define a transition system as a 4-tuple $\Theta = \langle S, L, T, S_\star \rangle$ where S is a finite set of *states*, L is a finite set of *labels*, $T \subseteq S \times L \times S$ is a set of (labeled) *transitions*, and $S_\star \subseteq S$ is the set of *goal states*. Each label $l \in L$ has a *cost* $c(l) \in \mathbb{R}_0^+$. Where it simplifies notation, we write $s \xrightarrow{l} s'$ to denote a transition $\langle s, l, s' \rangle$ from s to s' with label l, and we may write $s \xrightarrow{l} s' \in \Theta$ for $s \xrightarrow{l} s' \in T$.

A planning task naturally induces a transition system, which is usually too large to be represented explicitly. Instead, the merge-and-shrink approach works with a set X of smaller transition systems, which it iteratively transforms until only one transition system remains. This final transition system is then used to define a heuristic for solving the planning task.

The process starts by setting X to the set of *atomic* transition systems, which capture the behaviour of a single state variable. Then X is transformed by repeatedly applying one of the following two operations:

- *Merge:* Remove two transition systems $\Theta = \langle S, L, T, S_\star \rangle$ and $\Theta' = \langle S', L, T', S'_\star \rangle$ from X and replace them with their *synchronized product* $\Theta \otimes \Theta' = \langle S \times S', L, T^\otimes, S_\star \times S'_\star \rangle$, where a *synchronized transition* $\langle s, s' \rangle \xrightarrow{l} \langle t, t' \rangle \in T^\otimes$ exists iff $s \xrightarrow{l} t \in T$ and $s' \xrightarrow{l} t' \in T'$.
- *Shrink:* Remove a transition system $\Theta = \langle S, L, T, S_\star \rangle$ from X and replace it with the *abstract transition system* $\alpha(\Theta) := \langle \alpha(S), L, \{\langle \alpha(s), l, \alpha(t) \rangle \mid \langle s, l, t \rangle \in T\}, \alpha(S_\star) \rangle$, where α is an arbitrary function on S.

We remark that it is critical for merge operations (and hence for the correctness of the overall approach) that all transition systems work on a common set of labels. In the "basic" merge-and-shrink approach described in the paper by Helmert et al. (2007), this is always the set of operators of the underlying planning task. This changes when we make use of *label reduction*, described in the following section.

Before we move to label reduction, it is useful to introduce one more concept: the *global transition system* represented by X is the synchronized product (merge) of all elements in X, which we denote by $\bigotimes X$. (The product operator is associative and commutative modulo names of states, which we do not care about, so this is well-defined without having to specify an order on the individual merges.) At every stage of the merge-and-shrink algorithm, the current set X can be seen as a compact representation of $\bigotimes X$. In planning, initially $\bigotimes X$ equals the global transition system of the planning task (shown by Helmert et al., 2007). Merge steps do not change the represented global system, and shrink steps apply an abstraction to it.

Label Reduction: State of the Art

Label reduction adds a third class of transformations to the merge-and-shrink approach. It was first implemented, but not described, in the original application of merge-and-shrink abstractions to planning (Helmert, Haslum, and Hoffmann 2007). Nissim et al. (2011a) gave the first description; Helmert et al. (2014) discuss it more thoroughly. The key idea is to identify transition labels that can be combined into a single label without losing relevant information. Among other benefits, this can significantly reduce the representation size of the transition system because parallel transitions with different labels can collapse into a single transition.

The existing theory of label reduction is very complicated. We do not describe it in detail here: this would require much space, and a full description is not necessary for this paper. Details can be found in Section 5 of Nissim et al. (2011a) and Section 5 of Helmert et al. (2014). Here, it suffices to discuss three weaknesses of the current theory.

Firstly, the current theory largely attempts to define label reduction as a *local* concept considering individual transition systems: the central notion is that of a *label-reduced transition system*. This is fundamentally at odds with the purpose of labels in the merge-and-shrink framework to coordinate the joint behaviour of *all* transition systems in the

set X. If we change the labels in some, but not all transition systems in X, synchronization cannot work correctly.

The earlier papers address this difficulty by performing a kind of "just-in-time label reduction" that makes the labels of two transition systems correspond just before they are merged (which is the only point at which labels matter). This works, but the resulting theory is complex to understand and reason about, as different parts of the merge tree work with different labels. Consequently, current theory only permits reducing labels in certain cases, with other cases deemed to be unsafe and hence forbidden. Complications mainly arise in the case of *non-linear merge strategies*, and consequently, these were never correctly implemented.

Secondly, the current theory of label reduction is in a certain sense *syntax-based* while the rest of the merge-and-shrink framework is *semantic*. *Merge* operations and *shrink* operations are purely semantic: once a planning task (or other problem) is translated into atomic transition systems, the task description is not needed any more. Labels are opaque tokens that do not need to "stand for" anything. This greatly simplifies the theory of merge-and-shrink abstractions and makes them very flexible: they work for everything representable as transition systems.

Unfortunately, the current theory of label reduction needs to "look inside" the labels in order to decide which labels can be combined into one. For planning tasks, label reduction must treat labels as structured pairs of preconditions and effects, reintroducing and critically depending on the syntactic descriptions we would prefer not to have to reason about.

Thirdly, current theory cannot exploit label reductions that are enabled by shrinking. The decision how to reduce labels is completely independent of the shrink steps of the algorithm and hence needs to be correct for *all possible* shrink strategies. This severely limits simplification possibilities.

All these issues are addressed in the new theory of label reduction developed in the following section.

Label Reduction: New Theory

In this section, we introduce the new theory of label reduction and discuss its properties. Like the *merge* and *shrink* operations described earlier, we define label reduction as a transformation of the set X of transition systems:

- *Reduce labels:* Let τ be a *label mapping*, i. e., a function defined on the labels L of X, which satisfies $c(\tau(l)) \leq c(l)$ for all $l \in L$. Replace each transition system $\Theta = \langle S, L, T, S_\star \rangle \in X$ with the *label-reduced* system $\tau(\Theta) := \langle S, \tau(L), \{\langle s, \tau(l), t\rangle \mid \langle s, l, t\rangle \in T\}, S_\star\rangle$.

In words, label reduction means replacing all occurrences of each label l in all transition systems by the new label $\tau(l)$. (Of course, $\tau(l) = l$ is permitted.) When we choose to introduce a new label (i. e., $\tau(l) \notin L$), its cost can be set arbitrarily as long as it does not exceed $c(l)$. The operation is called label *reduction* because it is generally used to reduce the number of labels by choosing a non-injective function τ. (Using an injective function τ is possible, but pointless.)

It is worth emphasizing that, unlike previous definitions, label reduction always affects *all* transition systems simultaneously. As we will see in the following, this is sufficient to guarantee that label reduction is always "safe" to be applied. Unlike the previous theory, there is no need for pivot variables or to restrict label reduction to certain stages of the merge-and-shrink computation. Also, labels in the new theory always remain completely opaque objects (without associated "preconditions" and "effects").

However, there is a complication: the previous theory of label reduction reasoned about preconditions and effects to decide which labels can be combined to obtain *exact* label reductions, i. e., ones that do not introduce spurious transitions in $\bigotimes X$. With opaque labels, the question of exact label reduction must be addressed on the *semantic* level. Fortunately, we will see later that this is quite easy to do and more powerful than the previous syntax-based methods.

Properties of Label Reduction

To be able to use merge-and-shrink abstractions for admissible heuristics, we must guarantee that whenever a path from a given state s to some goal state exists in the actual problem, a corresponding path of at most the same cost exists in the final transition system computed.

Consider a transformation of a set of transition systems X with labels L into a new set X' with labels L' (e. g., merging, shrinking or reducing labels). We call such a transformation *transition-safe* if all transitions in $\bigotimes X$ have a corresponding transition in $\bigotimes X'$ (possibly with a different label) and goal states are preserved. Formally, the transformation is transition-safe if there exist functions α and τ mapping the states and labels of $\bigotimes X$ to the states and labels of $\bigotimes X'$ such that $\tau(L) = L'$, $s \xrightarrow{l} t \in \bigotimes X$ implies $\alpha(s) \xrightarrow{\tau(l)} \alpha(t) \in \bigotimes X'$ for all s, l, t, and $\alpha(s_\star)$ is a goal state of $\bigotimes X'$ for all goal states $s_\star$ of $\bigotimes X$.

We call a transformation *transition-exact* if additionally it does not give rise to any "new" transitions or goal states. Formally, the transformation is transition-exact if it is transition-safe, $s' \xrightarrow{l'} t' \in \bigotimes X'$ implies $s \xrightarrow{l} t \in \bigotimes X$ for *all* $s \in \alpha^{-1}(s')$ and $t \in \alpha^{-1}(t')$ and *some* $l \in \tau^{-1}(l')$, and for all goal states $s'_\star$ of $\bigotimes X'$ *all* states in the preimage $\alpha^{-1}(s'_\star)$ are goal states of $\bigotimes X$.

We call a transformation *cost-safe* if it cannot increase label costs and *cost-exact* if additionally it cannot decrease label costs. Formally, a transition-safe transformation must satisfy $c(\tau(l)) \leq c(l)$ for all $l \in L$, and a cost-exact one must satisfy $c(\tau(l)) = c(l)$ for all $l \in L$.

Finally, a transformation is *safe* if it is transition-safe and cost-safe and *exact* if it is transition-exact and cost-exact.

It is easy to verify that if each step in a sequence of transformations has one of these properties (e. g., is transition-safe), then the overall transformation also has it. (To prove this, compose the α and τ functions of each step.) Safe transformations give rise to admissible and consistent heuristics, and exact transformations give rise to perfect heuristics. Hence, it is important to verify that all transformations used in a merge-and-shrink heuristic computation are safe, and exact transformations are especially desirable.

Previous work on merge-and-shrink (e. g., Helmert et al. 2014) established that *merging* is always exact, *shrinking* is always safe, and shrinking based on *perfect bisimulation* is

exact. We now establish that in the new theory, label reduction is always safe.

Consider a label reduction with mapping τ that transforms $X = \{\Theta_1, \ldots, \Theta_n\}$ into $X' = \{\tau(\Theta_1), \ldots, \tau(\Theta_n)\}$. We first show that this label reduction is transition-safe. Here and in the following, we write states of $\bigotimes X$ and $\bigotimes X'$ as tuples $\langle s_1, \ldots, s_n \rangle$ where each s_i is a state of Θ_i. Consider some transition $\langle s_1, \ldots, s_n \rangle \xrightarrow{l} \langle t_1, \ldots, t_n \rangle \in \bigotimes X$. By the definition of products, we have $s_i \xrightarrow{l} t_i \in \Theta_i$ for all $1 \leq i \leq n$; by the definition of label reduction, we have $s_i \xrightarrow{\tau(l)} t_i \in \tau(\Theta_i)$ for all $1 \leq i \leq n$; finally, again by definition of products we have $\langle s_1, \ldots, s_n \rangle \xrightarrow{\tau(l)} \langle t_1, \ldots, t_n \rangle \in \bigotimes X'$. With α set to the identity function, this proves that label reduction is transition-safe. (Label reduction does not change the set of goal states.) Due to the condition on τ in the definition of label reduction, the transformation is also cost-safe. In summary, label reduction is safe.

Exact Label Reduction

Previous papers that study label reduction in the merge-and-shrink framework (Nissim, Hoffmann, and Helmert 2011a; Helmert et al. 2014) focus on the question which conditions are required to make label reduction exact. In particular, exact label reduction is a critical ingredient in the polynomial-time perfect heuristics obtained in some planning domains (Nissim, Hoffmann, and Helmert 2011a).

Helmert et al. (2014) discuss conditions for exactness of label reduction that are sufficient and in a certain sense necessary, thus seemingly closing the topic of exact label reduction. However, these results do not directly apply to our theory, as they rely on the limitations of the previous theory. We revisit the topic here, proving a sufficient and necessary condition for exact label reduction that generalizes the previous result.

It is obvious that a label reduction is cost-exact iff it only combines labels of the same cost (i. e., $\tau(l) = \tau(l')$ implies $c(l) = c(l')$), and of course we must always set $c(\tau(l)) := c(l)$ to be cost-exact. It remains to discuss under which conditions label reduction is transition-exact. We start by introducing some additional terminology.

Definition 1. *Let X be a set of transition systems with labels L. Let $l, l' \in L$ be labels, and let $\Theta \in X$.*

Label l is alive *in X if all transition systems $\Theta' \in X$ have some transition $s \xrightarrow{l} t \in \Theta'$. Otherwise, l is* dead.

Label l locally subsumes *label l' in Θ if for all $s \xrightarrow{l'} t \in \Theta$ we also have $s \xrightarrow{l} t$. Label l* globally subsumes *label l' in X if l locally subsumes l' in all $\Theta' \in X$.*

Labels l and l' are locally equivalent *in Θ if they label the same transitions in Θ, i. e., if l and l' locally subsume each other in Θ.*

Labels l and l' are Θ-combinable *in X if they are locally equivalent in all transition systems $\Theta' \in X \setminus \{\Theta\}$. (It does not matter whether or not they are locally equivalent in Θ.)*

It is easy to see that dead labels induce no transitions in $\bigotimes X$. Consequently, it is an exact transformation to remove all dead labels (and their transitions) from X. Hence, it suffices to consider the case where X has no dead labels.

Moreover, we can restrict attention to label reductions τ that combine two labels l_1 and l_2 into some new label l_{12} ($\tau(l_1) = \tau(l_2) = l_{12}$) while leaving all other labels unchanged ($\tau(l') = l'$ for all $l' \notin \{l_1, l_2\}$). Other label reductions can be represented as chains of such "minimal" label reductions. We are now ready to state our major result.

Theorem 1. *Let X be a set of transition systems without dead labels. Consider a label reduction on X which combines labels l_1 and l_2 and leaves other labels unchanged.*

This label reduction is exact iff $c(l_1) = c(l_2)$ and

1. *l_1 globally subsumes l_2, or*
2. *l_2 globally subsumes l_1, or*
3. *l_1 and l_2 are Θ-combinable for some $\Theta \in X$.*

Proof. Let τ be the described label mapping, let $X = \{\Theta_1, \ldots, \Theta_n\}$ and let $X' = \{\tau(\Theta_1), \ldots, \tau(\Theta_n)\}$ be the result of label reduction. Let $l_{12} := \tau(l_1) = \tau(l_2)$.

Clearly, the label reduction is cost-exact iff $c(l_1) = c(l_2)$. We need to show that it is transition-exact iff 1., 2., or 3. holds. We prove this in three parts:

(A) If neither 1. nor 2. nor 3. holds, then the label reduction is not exact.

(B) If 1. or 2. holds, then the label reduction is exact.

(C) If 3. holds, then the label reduction is exact.

Label reduction is always transition-safe and leaves the set of goal states unchanged, so we only need to consider the second condition in the definition of transition-exactness.

On (A): We must show that no function α satisfies the criterion of transition-exactness. It is sufficient to consider the case where α is a bijection because $\bigotimes X$ and $\bigotimes X'$ have the same number of states, so non-bijective α cannot possibly work. Renaming states does not affect the notion of exactness, so we can further limit attention to α being the identity function without loss of generality.

We say that a transition system $\Theta \in X$ has an l_1-*only* transition if there exists a transition $s \xrightarrow{l_1} t \in \Theta$ with $s \xrightarrow{l_2} t \notin \Theta$. Symmetrically, it has an l_2-*only* transition if there exists a transition $s \xrightarrow{l_2} t \in \Theta$ with $s \xrightarrow{l_1} t \notin \Theta$.

We try to find two transition systems $\Theta_i, \Theta_j \in X$ with $i \neq j$ such that there is an l_1-only transition $s_i \xrightarrow{l_1} t_i \in \Theta_i$ and an l_2-only transition $s_j \xrightarrow{l_2} t_j \in \Theta_j$. Then $\Theta_i \otimes \Theta_j$ does not contain a transition $\langle s_i, s_j \rangle \xrightarrow{l} \langle t_i, t_j \rangle$ for either $l = l_1$ or $l = l_2$, but $\tau(\Theta_i) \otimes \tau(\Theta_j)$ does contain the transition $\langle s_i, s_j \rangle \xrightarrow{l_{12}} \langle t_i, t_j \rangle$. By induction over the remaining transition systems, it is then easy to show that $\bigotimes X'$ contains a transition that does not correspond to a transition in $\bigotimes X$, proving inexactness. (Here, we use that there are no dead labels: the argument fails if l_1 and l_2 are dead.) It remains to show that l_1-only and l_2-only transitions in different transition systems of X exist.

Because 1. does not hold, there exists an l_2-only transition in some transition system $\Theta \in X$. Because 2. does not hold, there exists an l_1-only transition in some transition system $\Theta' \in X$. If Θ and Θ' are different transition systems, we have found the required transitions and are done.

So let us assume that $\Theta = \Theta'$. Because 3. does not hold, there exist at least two transition systems where l_1 and l_2

are not locally equivalent, so there is at least one transition system $\Theta'' \neq \Theta$ where they are not locally equivalent. This means that Θ'' must have an l_1-only transition or an l_2-only transition. In the former case, we select the l_1-only transition in Θ'' and the l_2-only transition in Θ. Otherwise, we select the l_2-only transition in Θ'' and the l_1-only transition in Θ' $(= \Theta)$.

On (B): Consider Case 1., where l_1 globally subsumes l_2. Case 2. is identical with l_1 and l_2 swapped. As the function α in the definition of transition-exactness, we choose the identity mapping. Then the condition for transition-exactness we need to verify simplifies to: for all $s \xrightarrow{l'} t \in \bigotimes X'$, there exists a label $l \in \tau^{-1}(l')$ with $s \xrightarrow{l} t \in \bigotimes X$. For $l' \neq l_{12}$, this is trivial because $\bigotimes X$ and $\bigotimes X'$ are exactly identical regarding labels other than l_1, l_2 and l_{12}. So consider the case $l' = l_{12}$. Let $s = \langle s_1, \ldots, s_n \rangle$ and let $t = \langle t_1, \ldots, t_n \rangle$. From $s \xrightarrow{l_{12}} t \in \bigotimes X'$ we get $s_i \xrightarrow{l_{12}} t_i \in \tau(\Theta_i)$ for all $1 \leq i \leq n$, and hence $s_i \xrightarrow{l_1} t_i \in \Theta_i$ or $s_i \xrightarrow{l_2} t_i \in \Theta_i$ for all $1 \leq i \leq n$. Because l_1 globally subsumes l_2, this implies $s_i \xrightarrow{l_1} t_i \in \Theta_i$ for all $1 \leq i \leq n$, and hence $s \xrightarrow{l_1} t \in \bigotimes X$, concluding this part of the proof.

On (C): As in (B), we set α to the identity function and only need to consider transitions $s \xrightarrow{l_{12}} t \in \bigotimes X'$. Let $s = \langle s_1, \ldots, s_n \rangle$ and let $t = \langle t_1, \ldots, t_n \rangle$. Again, we obtain that for all $1 \leq i \leq n$, $s_i \xrightarrow{l_{12}} t_i$ and hence $s_i \xrightarrow{l_1} t_i$ or $s_i \xrightarrow{l_2} t_i$. Choose $l \in \{l_1, l_2\}$ such that $s_j \xrightarrow{l} t_j$, where $j \in \{1, \ldots, n\}$ is chosen in such a way that l_1 and l_2 are Θ_j-combinable in X. (Such a transition system Θ_j exists because we are in Case 3.) By the definition of Θ-combinable, l_1 and l_2 are locally equivalent for all transition systems in X other than Θ_j, and hence ($s_i \xrightarrow{l_1} t_i$ or $s_i \xrightarrow{l_2} t_i$) implies ($s_i \xrightarrow{l_1} t_i$ *and* $s_i \xrightarrow{l_2} t_i$) for all $i \neq j$. This shows that $s_i \xrightarrow{l} t_i \in \Theta_i$ for all $1 \leq i \leq n$, and hence $s \xrightarrow{l} t \in \bigotimes X$, concluding the final part of the proof. □

We conclude the section with a brief discussion of the conditions in Theorem 1. Although all conditions can be checked in low-order polynomial time, there is a practical difference in complexity. Finding Θ-combinable labels essentially consists in computing the local equivalence relations of all $\Theta \in X$, which is possible in linear time in the representation size of X. In contrast, finding globally subsumed labels involves finding subset relationships in a set family, for which to the best of our knowledge no linear-time algorithms are known.

A comparison to the results of Helmert et al. (2014) shows that the Θ-combinability condition strictly generalizes the previous conditions on exactness. Hence, the new theory permits a larger number of exact label reductions even if we only use Θ-combinability and do not consider global subsumption of labels. For this reason, coupled with efficiency concerns, we only perform exact label reductions based on Θ-combinability in our experiments, which we describe next.

Experiments

As discussed in the preceding sections, the new theory of label reduction is significantly more general and at the same time much less complicated than previous work. However, we have yet to establish that it is useful for practical implementations of merge-and-shrink heuristics.

Firstly, we need to show that label reduction is actually a practically useful element of the merge-and-shrink toolbox. Although previous papers on merge-and-shrink heuristics already mentioned significant performance improvements due to label reduction, these are not a central focus of any previous experiment, and we think it is important to give solid quantitative evidence in favour of label reduction.

Secondly, while the semantic (rather than syntax-based, as in previous work) basis for exact label reduction has the advantage of being much more flexible and easier to implement than previous label reduction theory, it does carry a nontrivial computational overhead. If this overhead were so large that implementations based on the new theory performed significantly worse than ones based on the older theory, the usefulness of the new theory would be diminished.

Thirdly, a major drawback of previous label-reduction approaches are the limitations and difficulties in using them for *non-linear* merge strategies. Consequently, we are not aware of any implementations of non-linear merge strategies in the planning literature. The new theory removes these weaknesses, so it is appropriate to test it with non-linear merging.

In this section, we report on experiments that address these three aspects.

Experiment Description

Our experiments were conducted with the Fast Downward planning system (Helmert 2006), which already features the merge-and-shrink framework including the previous label reduction approach. We evaluate on all benchmarks from the International Planning Competitions for optimal planning (up to 2011) that only use language features supported by the merge-and-shrink framework (44 domains and 1396 instances in total). The experiments were performed on Intel Xeon E5-2660 CPUs running at 2.2 GHz, using a time bound of 30 minutes and a memory bound of 2 GB per run.

All planning algorithms we evaluate employ an A* search with a merge-and-shrink heuristic, which we varied along three dimensions: *label reduction method*, *merge strategy* and *shrink strategy*.

Label Reduction Methods We consider the case without label reduction (*none*), the *old* label reduction method based on the syntactic descriptions of operators (Nissim, Hoffmann, and Helmert 2011a; Helmert et al. 2014) and the *new* concept of label reduction described in this paper.

Our implementation of the new method only performs exact label reduction, combining labels whenever the Θ-combinability condition in Theorem 1 applies. Specifically, the computation proceeds as follows: whenever label reduction makes sense (after each merge or shrink step), we compute the local equivalence relations for labels in each transition system, then use these to test for Θ-combinable labels in each transition system Θ. If such labels exist, they are combined in all transition systems, and the local equivalence relations are recomputed. The process repeats until no further Θ-combinable labels exist for any transition system Θ. Local equivalence relations are cached so that they are only

recomputed from scratch if the given transition system has changed since the last computation.

Merge Strategies We consider two merge strategies. Firstly, in order to represent the state of the art, we report results for the (linear) *reverse-level* (*RL*) strategy used in previous work (Nissim, Hoffmann, and Helmert 2011a; 2011b).

However, to more fully utilize the potential of the new label reduction approach, we also evaluate it on a *non-linear* merge strategy, for which the previous label reduction approach is comparatively ill-suited and no implementations were previously available. Therefore, as a case study, we implemented the originally proposed non-linear strategy by Dräger, Finkbeiner, and Podelski (2006) from model checking, which we call the *DFP* merge strategy in the following.

Roughly speaking, the DFP merge strategy is based on the idea of preferably merging transition systems which must synchronize on labels that occur close to a goal state. We refer to the original paper by Dräger et al. (2006) for details. We remind the reader that the work of Dräger et al. preceded the concept of label reduction, so the combination of non-linear merge strategies with label reduction is novel.

Shrink Strategies We report results on shrink strategies based on bisimulation (Nissim, Hoffmann, and Helmert 2011a; Helmert et al. 2014), which set the current state of the art. Specifically, we consider a shrink strategy based on *greedy bisimulation* with no limit on transition system size (*G-N∞*) as well as shrink strategies based on (exact) *bisimulation* with different size limits N for the intermediate transition system size (*B-N10k*, *B-N50k*, *B-N100k*, *B-N200k*, *B-N∞*). For example, with $N = 10000$ (strategy B-N10k), shrinking is performed to guarantee that no intermediate transition system has more than 10,000 abstract states, while with $N = \infty$ (strategy B-N∞) there is no size bound, so that a perfect heuristic is constructed.

The *threshold* parameter (Helmert et al. 2014) was set to N for the strategies with bounded transition system size and to 1 for the unbounded ones (G-N∞ and B-N∞), following Nissim, Hoffmann, and Helmert (2011a). This configuration space includes the shrink strategies used in the *merge-and-shrink* planner that participated in IPC 2011 (Nissim, Hoffmann, and Helmert 2011b).

Experimental Results

Table 1 provides a result overview for coverage, i. e., the number of instances solved by each planner configuration within our resource bounds. The top half of the table presents results for the linear merge strategy (RL), the bottom half presents results for the non-linear DFP strategy.

Usefulness of Label Reduction Table 1 shows that planner configurations with label reduction dramatically outperform the corresponding ones without. (For readers less familiar with optimal planning, we point out that these tasks tend to scale exponentially in difficulty, so that even small improvements in coverage tend to be very hard to obtain.)

Table 2 shows detailed coverage results for the individual planning domains in the benchmark set for the best-

merge/shrink strategy	none	old	new
RL-G-N∞	417	485	465
RL-B-N10k	590	624	617
RL-B-N50k	577	618	634
RL-B-N100k	560	599	**639**
RL-B-N200k	544	590	630
RL-B-N∞	257	302	302
DFP-G-N∞	415	—	465
DFP-B-N10k	597	—	622
DFP-B-N50k	565	—	**644**
DFP-B-N100k	551	—	632
DFP-B-N200k	522	—	625
DFP-B-N∞	253	—	302

Table 1: Total coverage for several merge-and-shrink configurations, using no label reduction (*none*), the previous (*old*) or the *new* label reduction. See the text for descriptions of the merge and shrink strategies. Best results for each merge strategy in bold.

	RL-B-100K			DFP-B-50K	
	none	old	new	none	new
mprime (35)	8	+6	**+15**	6	**+17**
miconic (150)	60	**+13**	**+13**	58	**+14**
gripper (20)	7	**+13**	**+13**	7	**+11**
freecell (80)	6	−2	**+13**	9	**+11**
mystery (30)	8	+1	**+8**	8	**+8**
zenotravel (20)	9	**+3**	**+3**	10	**+2**
pipesworld-tankage (50)	8	+2	**+3**	12	**+2**
nomystery-opt11-strips (20)	17	**+1**	**+1**	16	**+2**
woodworking-opt08-strips (30)	11	−1	**+1**	11	**+2**
blocks (35)	**25**	−3	−3	25	**+2**
grid (5)	1	**+2**	**+2**	1	**+1**
floortile-opt11-strips (20)	5	**+1**	**+1**	4	**+1**
rovers (40)	7	**+1**	**+1**	7	**+1**
satellite (36)	5	**+1**	**+1**	5	**+1**
scanalyzer-08-strips (30)	12	**+1**	**+1**	12	**+1**
scanalyzer-opt11-strips (20)	9	**+1**	**+1**	9	**+1**
woodworking-opt11-strips (20)	6	−1	**+1**	6	**+1**
pipesworld-notankage (50)	**14**	**±0**	**±0**	14	**+1**
sokoban-opt08-strips (30)	24	±0	**+2**	**25**	**±0**
trucks-strips (30)	6	±0	**+2**	**6**	**±0**
transport-opt11-strips (20)	6	**+1**	**+1**	**6**	**±0**
driverlog (20)	**13**	−1	−1	**12**	**±0**
Sum (**791**)	267	+39	+79	269	+79
Remaining domains (**605**)	**293**	**±0**	**±0**	**296**	**±0**
Sum (**1396**)	560	599	**639**	565	**644**

Table 2: Per-domain coverage. Columns 2–4 compare no (*none*), *old* and *new* label reduction for the linear merge strategy *reverse level* (RL) in its best configuration (RL-B-100K). Columns 5–6 compare no (*none*) and *new* label reduction for the non-linear DFP merge strategy in its best configuration (DFP-B-50K). For *old* and *new*, the columns show increase/decrease in coverage compared to *none*. Domains where label reduction showed no increase/decrease in coverage are omitted. The best results for the given merge strategy are highlighted in bold.

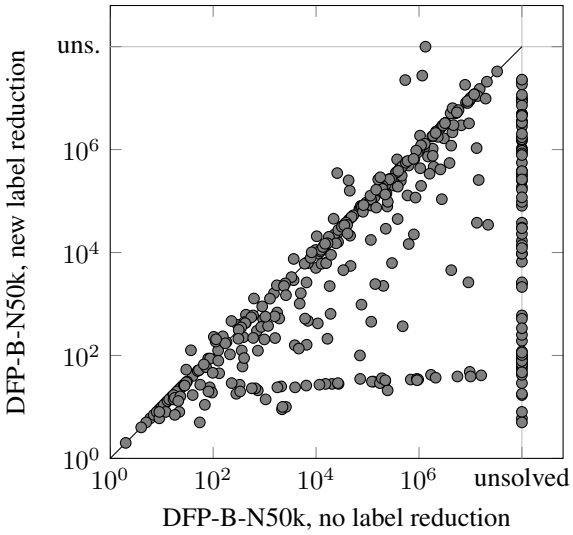

Figure 2: Number of expanded states for DFP-B-N50k: no label reduction vs. new label reduction.

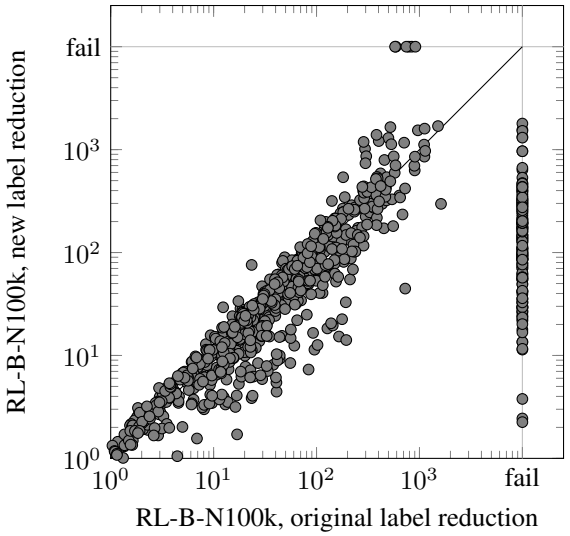

Figure 3: Construction time (in seconds) for RL-B-N100k: old label reduction vs. new label reduction. Almost all failures are due to running out of memory.

performing shrink strategies for each merge strategy. The table shows that label reduction is very useful across the board, over a wide range of domains.

For the linear RL merge strategy, the new label reduction approach increases coverage in 19 domains compared to the baseline where no labels are reduced, while decreasing coverage in 2 domains. For the non-linear DFP merge strategy, label reduction increases coverage in 18 domains and decreases it in none.

To provide another detailed view, Figure 2 shows the number of expanded states with the strongest configuration, DFP-B-N50k, with and without label reduction. The figure plots the results without label reduction against the results with our new label reduction approach, over all instances in the benchmark suite. The figure clearly shows the significant impact that label reduction has on performance in many cases.

Old vs. New Label Reduction Method Focusing on the comparison between the old and new label reduction method with a linear merge strategy (top half of Table 1), we see that despite the larger effort involved in determining reducible labels, the results are in fact quite a bit *better* with new label reduction compared to the old technique. In particular, the best overall result of 639 solved tasks (RL-B-100k) is considerably higher than the best result with the previous state of the art (624 solved tasks with RL-B-10K and the old label reduction method).

There are two shrink strategies that show the opposite trend, namely the ones that tend to compute the simplest abstractions among the six strategies we consider: greedy bisimulation (RL-G-N∞) and exact bisimulation with the smallest size bound (RL-B-N10k). One possible explanation for this behaviour is that for the shrink strategies that compute more complex abstractions, the additional label reductions afforded by the new method are critical for computing the merge-and-shrink abstraction within the given limits for time and especially memory. With the shrink strategies that compute simpler abstractions, on the other hand, memory for computing the abstraction is less of a concern, and the new label reduction method suffers from the higher computational cost for determining combinable labels.

This interpretation is supported by Figure 3, which compares the time to construct the abstraction heuristic for the old and new label reduction method for the strategy RL-B-N100k. The new strategy tends to construct abstractions faster and runs out of memory far less frequently. Figure 4 compares state expansions for the same configurations, showing that the heuristics are similarly informative in both cases, and it is mainly the ability to complete the computation of the abstraction (see Figure 3) that makes the difference between the old and new label reduction here.

In the case of perfect bisimulations (RL-B-N∞), there is no difference in coverage between the two label reduction methods for a different reason: unless the given planning task exhibits significant amounts of symmetry, unrestricted bisimulation tends to exhaust the available memory very quickly, and hence the perfect abstraction heuristic is either computed quickly or not at all. In all cases not solved by the perfect bisimulation approaches, this is due to running out of memory while computing the abstraction.

Non-Linear Merge Strategy Shifting attention to the results for the non-linear DFP merge strategy (bottom half of Table 1), we see that the results with the new label reduction method are excellent. In particular, the best configuration (DFP-B-N50k) solves 644 tasks, again setting a new best result (compared to 639 solved by RL-B-N100k, also with our new label reduction method).

Generally speaking, the non-linear merge strategy appears to benefit even more from label reduction than the linear one

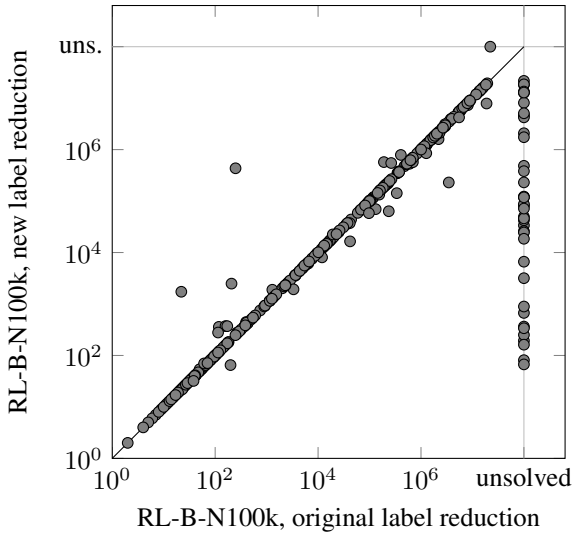

Figure 4: Number of expanded states for RL-B-N100k: old label reduction vs. new label reduction.

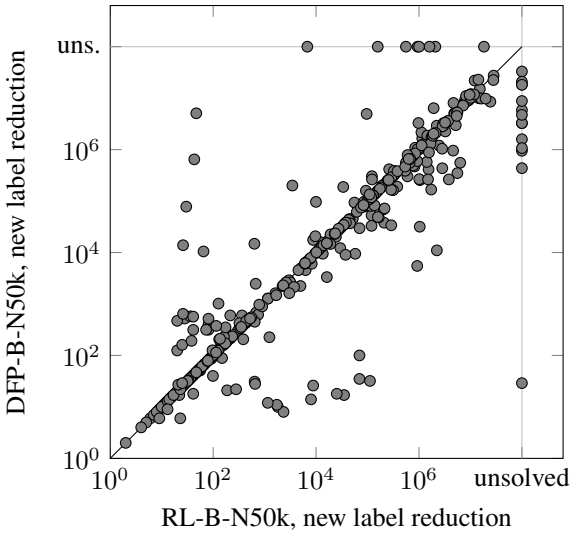

Figure 5: Number of expanded states for RL-B-N50k vs. DFP-B-N50k, both using new label reduction.

on average. One possible explanation for this observation is that non-linear merge strategies involve more complex products (merges) than linear ones, and hence benefit more from label reduction collapsing multiple parallel transitions into one. In linear merge strategies, at least one of the merged transition systems is always atomic, and atomic transition systems tend to have a comparatively low density of transitions. An alternative possibility is that label reduction interacts favourably with the DFP merge strategy, which – unlike merge strategies previously considered in planning – takes the labels into account directly in order to decide which transition systems to merge next.

Figure 5 compares the number of state expansions for the linear and non-linear merge strategy on an otherwise identical configuration (shrink strategy B-N50k, new label reduction). The comparison shows that the two merge strategies are quite complementary, with both strategies greatly outperforming each other on a significant number of instances.

Conclusions

We have introduced a general theory of label reduction that addresses several drawbacks in the previous development of this topic. Compared to the previous theory, the new theory of label reduction is easier to understand, easier to reason about, and more general.

Under the new theory, label reduction can always be safely applied. Moreover, we have provided efficiently checkable necessary and sufficient criteria for label reduction to be *exact*, i. e., preserve all relevant information. The new theory allows identifying more cases where exact label reduction is possible, leading to improved performance of merge-and-shrink heuristics based on label reduction.

Unlike the previous theory of label reduction, the new theory allows for a straight-forward application of non-linear merge strategies. We conducted the first experiments of this kind by adapting the originally proposed non-linear merge strategy from model checking to planning. In the future, we hope that the development of strong non-linear merge strategies can further increase the scalability of merge-and-shrink heuristics.

Another possible direction for future work is the exploration of *inexact* label reduction. Inexact label reduction is a general abstraction method just like shrinking, and similar intuitions to those that have guided the development of state-of-the-art shrink strategies could be used to develop useful inexact label reduction methods. For example, one might try to abstract a factored transition system by combining labels that only occur far away from goal states, similarly to the way that current shrink strategies prefer to combine abstract states that are far away from the goal.

Acknowledgments

We thank the anonymous reviewers for their comments, which helped improve the paper. This work was supported by the Swiss National Science Foundation (SNSF) as part of the project "Abstraction Heuristics for Planning and Combinatorial Search" (AHPACS).

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS^+ planning. *Computational Intelligence* 11(4):625–655.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2006. Directed model checking with distance-preserving abstractions. In Valmari, A., ed., *Proceedings of the 13th International SPIN Workshop (SPIN 2006)*, volume 3925 of *Lecture Notes in Computer Science*, 19–34. Springer-Verlag.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2009. Directed model checking with distance-preserving abstractions. *In-*

ternational Journal on Software Tools for Technology Transfer 11(1):27–37.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM*. In press.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In Boddy, M.; Fox, M.; and Thiébaux, S., eds., *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, 176–183. AAAI Press.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2008. Explicit-state abstraction: A new method for generating heuristic functions. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 1547–1550. AAAI Press.

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

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011a. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 1983–1990.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011b. The Merge-and-Shrink planner: Bisimulation-based abstraction for optimal planning. In *IPC 2011 planner abstracts*, 106–107.