Planning and Optimization

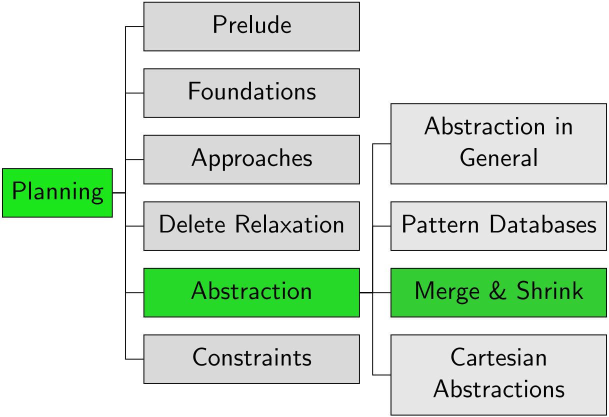
E12. Merge-and-Shrink: Merge Strategies & Outlook

Malte Helmert and Gabriele Röger

Universität Basel

November 19, 2025

Content of the Course



Merge Strategies

Reminder: Generic Algorithm Template

Generic Merge & Shrink Algorithm for planning task Π

$F := F(\Pi)$
while $|F| > 1$:
  select $type \in \{\text{merge}, \text{shrink}\}$
  if $type = \text{merge}$:
    select $\mathcal{T}_1, \mathcal{T}_2 \in F$
    $F := (F \setminus \{\mathcal{T}_1, \mathcal{T}_2\}) \cup \{\mathcal{T}_1 \otimes \mathcal{T}_2\}$
  if $type = \text{shrink}$:
    select $\mathcal{T} \in F$
    choose an abstraction mapping β on $\mathcal{T}$
    $F := (F \setminus \{\mathcal{T}\}) \cup \{\mathcal{T}^{\beta}\}$
return the remaining factor $\mathcal{T}^{\alpha}$ in F

Remaining Question:

- Which abstractions to select for merging? $\rightsquigarrow$ merge strategy

Linear vs. Non-linear Merge Strategies

Linear Merge Strategy

In each iteration after the first, choose the abstraction computed in the previous iteration as $\mathcal{T}_1$.

Rationale: only maintains one “complex” abstraction at a time

- Fully defined by an ordering of atomic projections/variables.
- Each merge-and-shrink heuristic computed with a non-linear merge strategy can also be computed with a linear merge strategy.
- However, linear merging can require a super-polynomial blow-up of the final representation size.
- Recent research turned from linear to non-linear strategies, also because better label reduction techniques (later in this chapter) enabled a more efficient computation.

Classes of Merge Strategies

We can distinguish two major types of merge strategies:

- precomputed merge strategies fix a unique merge order up-front.
  One-time effort but cannot react to other transformations applied to the factors.
- stateless merge strategies only consider the current FTS and decide what factors to merge.
  Typically computing a score for each pair of factors and naturally non-linear; easy to implement but cannot capture dependencies between more than two factors.

Hybrid strategies combine ideas from precomputed and stateless strategies.

Example Linear Precomputed Merge Strategy

Idea: Use similar causal graph criteria as for growing patterns.

Example: Strategy of h_{HHH}

h_{HHH}: Ordering of atomic projections

- Start with a goal variable.
- Add variables that appear in preconditions of operators affecting previous variables.
- If that is not possible, add a goal variable.

Rationale: increases h quickly

Example Non-linear Precomputed Merge Strategy

Idea: Build clusters of variables with strong interactions and first merge variables within each cluster.

Example: MIASM ("maximum intermediate abstraction size minimizing merging strategy")

MIASM strategy

- Measure interaction by ratio of unnecessary states in the merged system (= states not traversed by any abstract plan).
- Best-first search to identify interesting variable sets.
- Disjoint variable sets chosen by a greedy algorithm for maximum weighted set packing.

Rationale: increase power of pruning (later in this chapter)

Example Non-linear Stateless Merge Strategy

Idea: Preferrably merge transition systems that must synchronize on labels that occur close to a goal state.

Example: DFP (named after Dräger, Finkbeiner and Podelski)

DFP strategy

- $\mathit{labelrank}(\ell, \mathcal{T}) = \min\{h^*(t) \mid \langle s, \ell, t\rangle \text{ transition in } \mathcal{T}\}$
- $\mathit{score}(\mathcal{T}, \mathcal{T}') = \min\{\max\{\mathit{labelrank}(\ell, \mathcal{T}), \mathit{labelrank}(\ell, \mathcal{T}')\} \mid \ell \text{ label in } \mathcal{T} \text{ and } \mathcal{T}'\}$
- Select two transition systems with minimum score.

Rationale: abstraction fine-grained in the goal region, which is likely to be searched by A^*.

Example Hybrid Merge Strategy

Idea: first combine the variables within each strongly connected component of the causal graph.

Example: SCC framework

SCC strategy

- Compute strongly connected components of causal graph
- Secondary strategies for order in which
  - the SCCs are considered (e.g. topologic order),
  - the factors within an SCC are merged, and
  - the resulting product systems are merged.

Rationale: reflect strong interactions of variables well

State of the art: SCC+DFP or a stateless MIASM variant

Outlook: Label Reduction and Pruning

Further Transformations

State-of-the-art Merge & Shrink uses two further transformations:

- Label reduction
- Pruning

Label Reduction

- Do no longer distinguish certain labels, similar to abstraction that does not distinguish certain states.

Label Reduction

- Do no longer distinguish certain labels, similar to abstraction that does not distinguish certain states.
- A label reduction $\langle \lambda, c' \rangle$ for a FTS F with label set L is given by a function $\lambda : L \to L'$, where L' is an arbitrary set of labels, and a label cost function c' on L' such that for all $\ell \in L$, $c'(\lambda(\ell)) \leq c(\ell)$.
  The label-reduced TSs have L' and c' for the labels and cost, and in each transition the original label ℓ is replaced with $\lambda(\ell)$.

Label Reduction

- Do no longer distinguish certain labels, similar to abstraction that does not distinguish certain states.
- A label reduction $\langle \lambda, c' \rangle$ for a FTS F with label set L is given by a function $\lambda : L \to L'$, where L' is an arbitrary set of labels, and a label cost function c' on L' such that for all $\ell \in L$, $c'(\lambda(\ell)) \leq c(\ell)$.
  The label-reduced TSs have L' and c' for the labels and cost, and in each transition the original label ℓ is replaced with $\lambda(\ell)$.
- Label reduction is a conservative transformation.

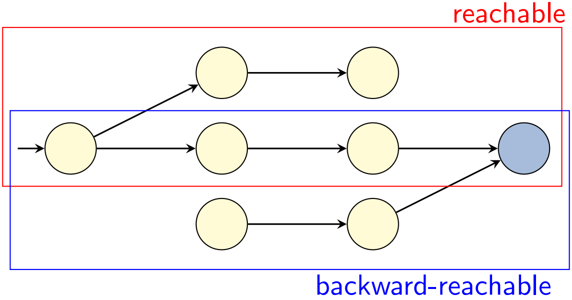
Label Reduction

- Do no longer distinguish certain labels, similar to abstraction that does not distinguish certain states.
- A label reduction $\langle \lambda, c' \rangle$ for a FTS F with label set L is given by a function $\lambda : L \to L'$, where L' is an arbitrary set of labels, and a label cost function c' on L' such that for all $\ell \in L$, $c'(\lambda(\ell)) \leq c(\ell)$.
  The label-reduced TSs have L' and c' for the labels and cost, and in each transition the original label ℓ is replaced with $\lambda(\ell)$.
- Label reduction is a conservative transformation.
- There are also clear criteria when label reduction is exact.

Label Reduction

- Do no longer distinguish certain labels, similar to abstraction that does not distinguish certain states.
- A label reduction $\langle \lambda, c' \rangle$ for a FTS F with label set L is given by a function $\lambda : L \to L'$, where L' is an arbitrary set of labels, and a label cost function c' on L' such that for all $\ell \in L$, $c'(\lambda(\ell)) \leq c(\ell)$.
  The label-reduced TSs have L' and c' for the labels and cost, and in each transition the original label ℓ is replaced with $\lambda(\ell)$.
- Label reduction is a conservative transformation.
- There are also clear criteria when label reduction is exact.
- Reduces the time and memory requirement for merge and shrink steps and enables coarser bisimulation abstractions.

Alive States



- state s is reachable if we can reach it from the initial state
- state s is backward-reachable if we can reach the goal from s
- state s is alive if it is reachable and backward-reachable
  $\rightarrow$ only alive states can be traversed by a solution
- a state s is dead if it is not alive.

Pruning States (1)

- If in a factor, state s is dead/not backward-reachable then all states that “cover” s in a synchronized product are dead/not backward-reachable in the synchronized product.
- Removing such states and all adjacent transitions in a factor does not remove any solutions from the synchronized product.
- This pruning leads to states in the original state space for which the merge-and-shrink abstraction does not define an abstract state.
  $\rightarrow$ use heuristic estimate ∞

Pruning States (2)

- Keeping exactly all backward-reachable states we still obtain safe, consistent, goal-aware and admissible (with conservative transformations) or perfect heuristics (with exact transformations).
- Pruning unreachable, backward-reachable states can render the heuristic unsafe because pruned states lead to infinite estimates.
- However, all reachable states in the original state space will have admissible estimates, so we can use the heuristic like an admissible one in a forward state-space search such as A^*(but not in other contexts like such as orbit search).
  We usually prune all dead states to keep the factors small.

Summary

Summary

- There is a wide range of merge strategies. We only covered some important ones.
- Label reduction is crucial for the performance of the merge-and-shrink algorithm, especially when using bisimilarity for shrinking.
- Pruning is used to keep the size of the factors small. It depends on the intended application how aggressive the pruning can be.

Literature

Literature (1)

References on merge-and-shrink abstractions:

Klaus Dräger, Bernd Finkbeiner and Andreas Podelski.
Directed Model Checking with Distance-Preserving Abstractions.
Proc. SPIN 2006, pp. 19–34, 2006.
Introduces merge-and-shrink abstractions (for model checking) and DFP merging strategy.

Malte Helmert, Patrik Haslum and Jörg Hoffmann.
Flexible Abstraction Heuristics for Optimal Sequential Planning.
Proc. ICAPS 2007, pp. 176–183, 2007.
Introduces merge-and-shrink abstractions for planning.

Literature (2)

- Raz Nissim, Jörg Hoffmann and Malte Helmert.
  Computing Perfect Heuristics in Polynomial Time:
  On Bisimulation and Merge-and-Shrink Abstractions
  in Optimal Planning.
  Proc. IJCAI 2011, pp. 1983–1990, 2011.
  Introduces bisimulation-based shrinking.

- Malte Helmert, Patrik Haslum, Jörg Hoffmann
  and Raz Nissim.
  Merge-and-Shrink Abstraction: A Method
  for Generating Lower Bounds in Factored State Spaces.
  Journal of the ACM 61 (3), pp. 16:1–63, 2014.
  Detailed journal version of the previous two publications.

Literature (3)

Silvan Sievers, Martin Wehrle and Malte Helmert.
Generalized Label Reduction for Merge-and-Shrink Heuristics.
Proc. AAAI 2014, pp. 2358–2366, 2014.
Introduces modern version of label reduction.
(There was a more complicated version before.)

Gaojian Fan, Martin Müller and Robert Holte.
Non-linear merging strategies for merge-and-shrink
based on variable interactions.
Proc. SoCS 2014, pp. 53–61, 2014.
Introduces UMC and MIASM merging strategies

Literature (4)

Malte Helmert, Gabriele Röger and Silvan Sievers.
On the Expressive Power of Non-Linear Merge-and-Shrink Representations.
Proc. ICAPS 2015, pp. 106–114, 2015.
Shows that linear merging can require a super-polynomial blow-up in representation size.

Silvan Sievers and Malte Helmert.
Merge-and-Shrink: A Compositional Theory of Transformations of Factored Transition Systems.
JAIR 71, pp. 781–883, 2021.
Detailed theoretical analysis of task transformations as sequence of transformations.

Literature (5)

Silvan Sievers, Florian Pommerening , Thomas Keller and Malte Helmert.
Cost-Partitioned Merge-and-Shrink Heuristics for Optimal Classical Planning.
Proc. IJCAI 2020, pp. 4152–4160, 2020.
Extends saturated cost partitioning to merge-and-shrink.