Planning and Optimization

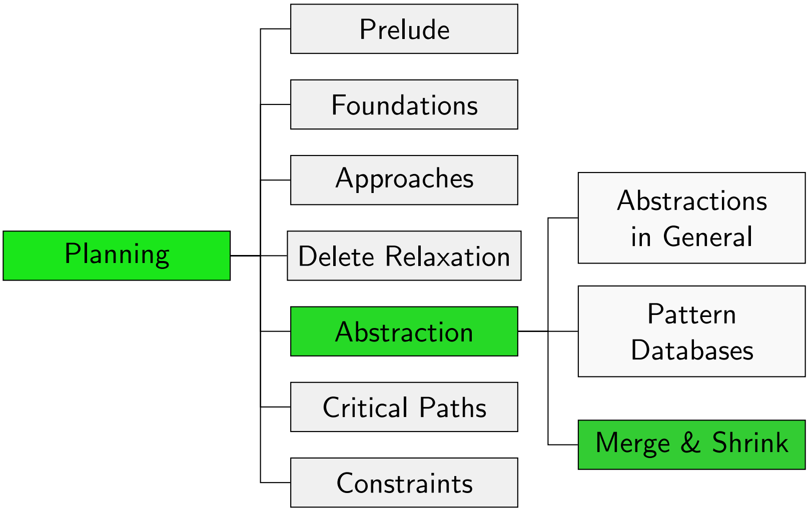
E12. Merge-and-Shrink: Merge Strategies and Label Reduction

Malte Helmert and Gabriele Röger

Universität Basel

November 20, 2023

Content of this Course



Merge Strategies

Reminder: Generic Algorithm Template

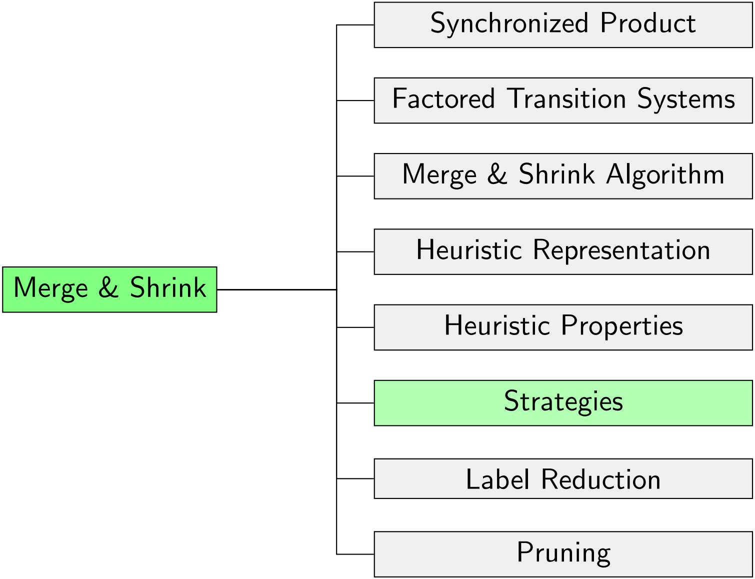
Generic Merge & Shrink Algorithm for planning task Π

$F := F(\Pi)$
while $|F| > 1$:
  select $\mathit{type} \in \{\text{merge}, \text{shrink}\}$
  if $\mathit{type} = \text{merge}$:
    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 $\mathit{type} = \text{shrink}$:
    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

Merge-and-Shrink



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 (cf. next 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

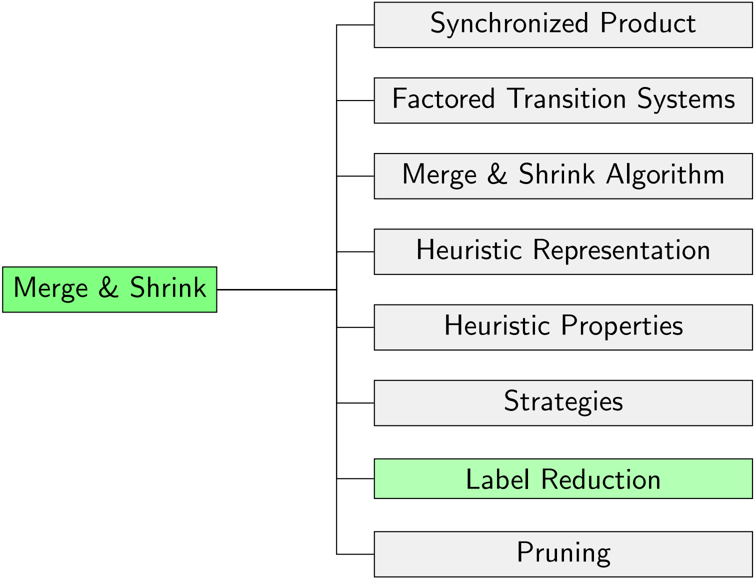
- Compute strongly connected components of causal graph
- Secondary strategies for order in which
  - the SCCs are considered (e.g. topologic order),
  - the factors within an SCC are merged, and
  - the resulting product systems are merged.

Rationale: reflect strong interactions of variables well

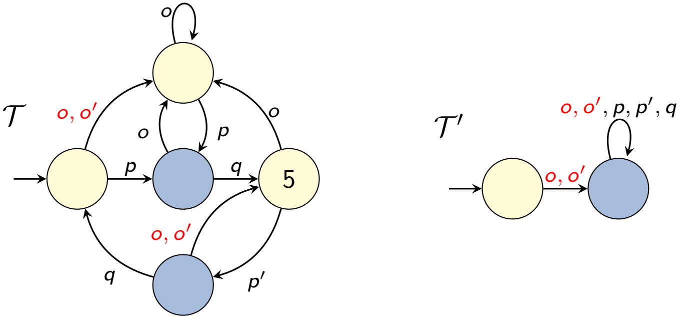
State of the art: SCC+DFP or a stateless MIASM variant

Label Reduction

Merge-and-Shrink



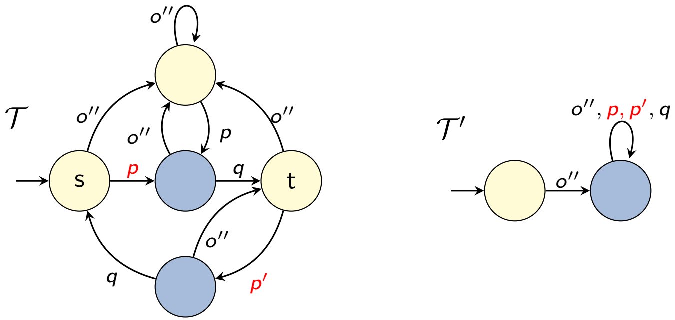
Label Reduction: Motivation (1)



Whenever there is a transition with label o' there is also a transition with label o. If o' is not cheaper than o, we can always use the transition with o.

Idea: Replace o and o' with label o'' with cost of o

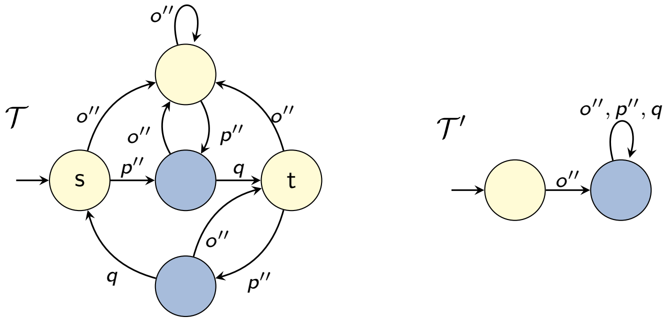
Label Reduction: Motivation (2)



States s and t are not bisimilar due to labels p and p'. In $\mathcal{T}'$ they label the same (parallel) transitions. If p and p' have the same cost, in such a situation there is no need for distinguishing them.

Idea: Replace p and p' with label p'' with same cost.

Label Reduction: Motivation (3)



Label reductions reduce the time and memory requirement for merge and shrink steps and enable coarser bisimulation abstractions.

When is label reduction a conservative transformation?

Label Reduction: Definition

Definition (Label Reduction)

Let F be a factored transition system with label set L and label cost function c. A label reduction $\langle \lambda, c' \rangle$ for F 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 $\ell \in L$, $c'(\lambda(\ell)) \leq c(\ell)$.

For $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle \in F$ the label-reduced transition system is $\mathcal{T}^{\langle \lambda, c' \rangle} = \langle S, L', c', \{\langle s, \lambda(\ell), t \rangle \mid \langle s, \ell, t \rangle \in T\}, s_0, S_\star \rangle$.

The label-reduced FTS is $F^{\langle \lambda, c' \rangle} = \{\mathcal{T}^{\langle \lambda, c' \rangle} \mid \mathcal{T} \in F\}$.

$L' \cap L \neq \emptyset$ and $L' = L$ are allowed.

Label Reduction is Conservative

Theorem (Label Reduction is Safe)

Let F be a factored transition systems and $\langle \lambda, c' \rangle$ be a label-reduction for F.
The transformation $\langle F, id, \lambda, F^{\langle \lambda, c' \rangle} \rangle$ is conservative.

(Proof omitted.)

Label Reduction is Conservative

Theorem (Label Reduction is Safe)

Let F be a factored transition systems and $\langle \lambda, c' \rangle$ be a label-reduction for F.
The transformation $\langle F, id, \lambda, F^{\langle \lambda, c' \rangle} \rangle$ is conservative.

(Proof omitted.)

We can use label reduction as an additional possible step in merge-and-shrink.

More Terminology

Let F be a factored transition systems with labels L. Let $\ell, \ell' \in L$ be labels and let $\mathcal{T} \in F$.

- Label ℓ is alive in F if all $\mathcal{T}' \in F$ have some transition labelled with ℓ. Otherwise, ℓ is dead.
- Label ℓ locally subsumes label ℓ' in $\mathcal{T}$ if for all transitions $\langle s, \ell', t\rangle$ of $\mathcal{T}$ there is also a transition $\langle s, \ell, t\rangle$ in $\mathcal{T}$.
- ℓ globally subsumes ℓ' if it locally subsumes ℓ' in all $\mathcal{T}' \in F$.
- ℓ and ℓ' are locally equivalent in $\mathcal{T}$ if they label the same transitions in $\mathcal{T}$, i.e. ℓ locally subsumes ℓ' in $\mathcal{T}$ and vice versa.
- ℓ and ℓ' are $\mathcal{T}$-combinable if they are locally equivalent in all transition systems $\mathcal{T}' \in F \setminus \{\mathcal{T}\}$.

Exact Label Reduction

Theorem (Criteria for Exact Label Reduction)

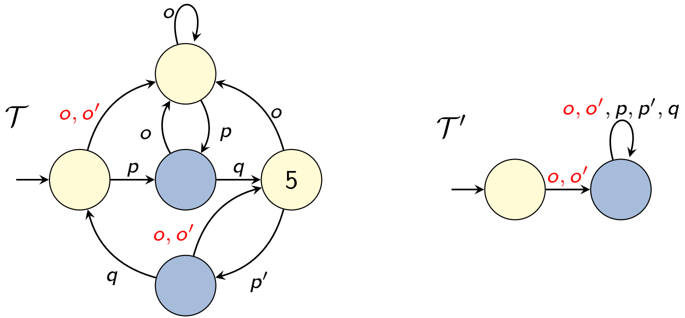
Let F be a factored transition systems with cost function c and label set L that contains no dead labels.

Let $\langle \lambda, c' \rangle$ be a label-reduction for F such that λ combines labels ℓ_1 and ℓ_2 and leaves other labels unchanged. The transformation from F to $F^{\langle \lambda, c' \rangle}$ is exact iff $c(\ell_1) = c(\ell_2)$, $c'(\lambda(\ell)) = c(\ell)$ for all $\ell \in L$, and

- *ℓ_1 globally subsumes ℓ_2, or*
- *ℓ_2 globally subsumes ℓ_1, or*
- *ℓ_1 and ℓ_2 are $\mathcal{T}$-combinable for some $\mathcal{T} \in F$.*

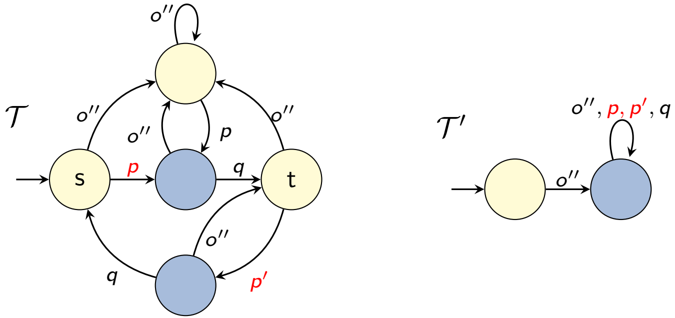
(Proof omitted.)

Back to Example (1)



Label o globally subsumes label o'.

Back to Example (2)



Labels p and p' are $\mathcal{T}$-combinable.

Computation of Exact Label Reduction (1)

- For given labels ℓ_1, ℓ_2, the criteria can be tested in low-order polynomial time.
- Finding globally subsumed labels involves finding subset relationsships in a set family.
  $\rightsquigarrow$ no linear-time algorithms known
- The following algorithm exploits only $\mathcal{T}$-combinability.

Computation of Exact Label Reduction (2)

eq_i := set of label equivalence classes of $\mathcal{T}_i \in F$

Label-reduction based on $\mathcal{T}_i$-combinability

$eq := \{[\ell]_{\sim_c} \mid \ell \in L, \ell' \sim_c \ell'' \text{ iff } c(\ell') = c(\ell'')\}$
for $j \in \{1, \ldots, |F|\} \setminus \{i\}$
    Refine eq with eq_j
// two labels are in the same set of eq iff they have
// the same cost and are locally equivalent in all $\mathcal{T}_j \neq \mathcal{T}_i$.
$\lambda = \text{id}$
for $B \in eq$
    ℓ_{new} := new label
    $c'(\ell_{\text{new}})$:= cost of labels in B
    for $\ell \in B$
        $\lambda(\ell) = \ell_{\text{new}}$

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.