Planning and Optimization

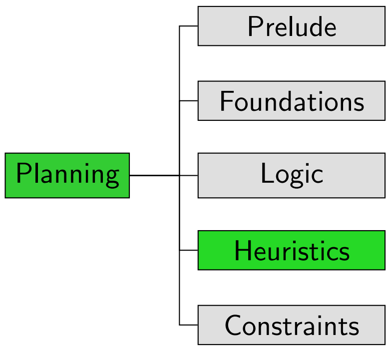
E12. Merge-and-Shrink: Merging Strategies and Label Reduction

Malte Helmert and Gabriele Röger

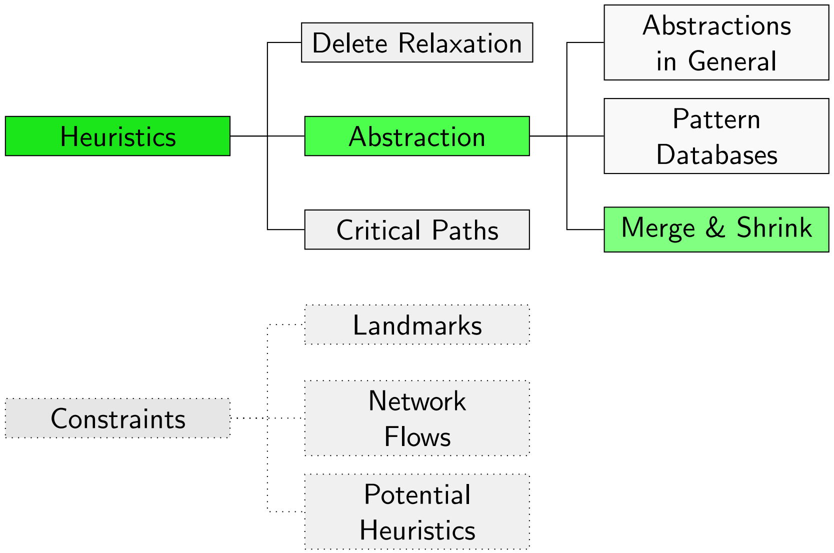
Universität Basel

November 21, 2022

Content of this Course



Content of this Course: Heuristics



E12.1 Merging Strategies

Reminder: Generic Algorithm Template

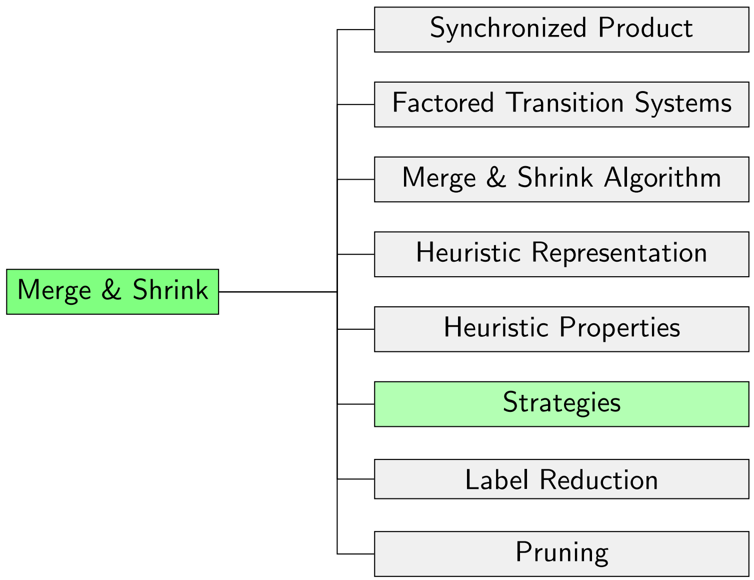
Generic Merge & Shrink Algorithm for planning task Π

$F := F(\Pi)$
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? ⇝ merging strategy

Merge-and-Shrink



Linear Merging Strategies

Linear Merging 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.

Linear Merging Strategies: Choosing the Ordering

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.

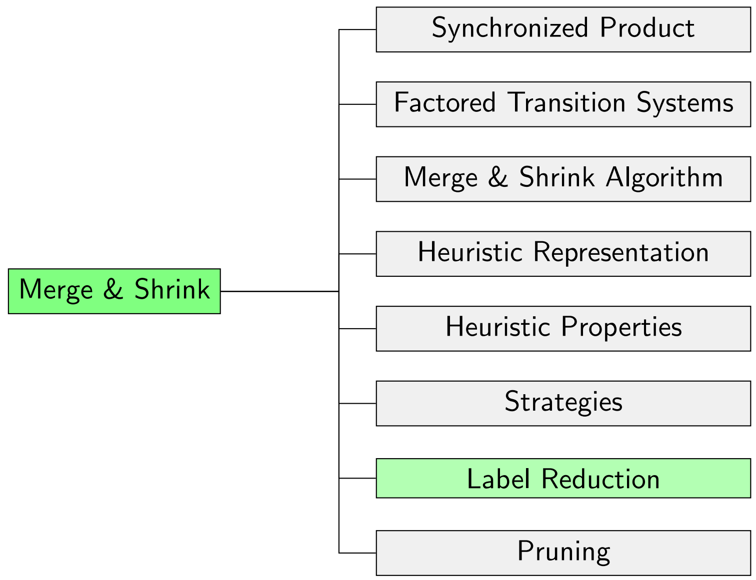
Rationale: increases h quickly

Non-linear Merging Strategies

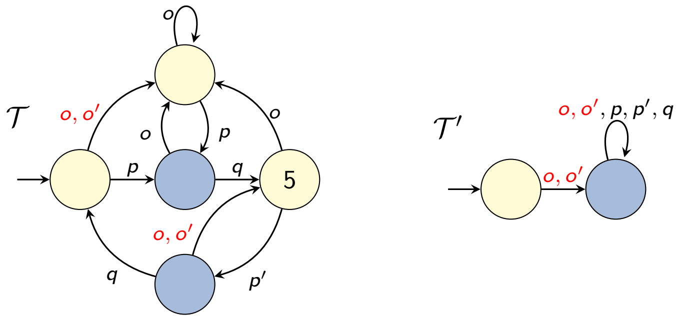
- Non-linear merging strategies only recently gained more interest in the planning community.
- One reason: Better label reduction techniques (later in this chapter) enabled a more efficient computation.
- Examples:
  - DFP: preferrably merge transition systems that must synchronize on labels that occur close to a goal state.
  - UMC and MIASM: Build clusters of variables with strong interactions and first merge variables within each cluster.
- Each merge-and-shrink heuristic computed with a non-linear merging strategy can also be computed with a linear merging strategy.
- However, linear merging can require a super-polynomial blow-up of the final representation size.

E12.2 Label Reduction

Merge-and-Shrink



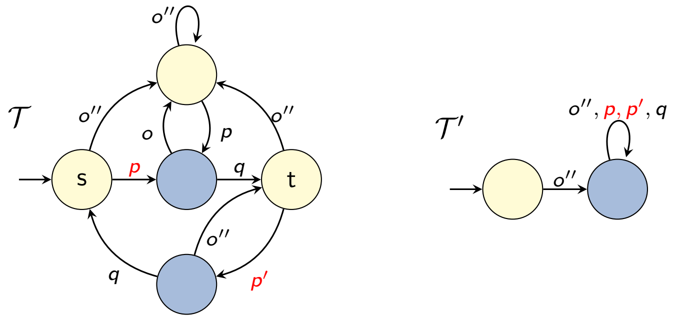
Label Reduction: Motivation (1)



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

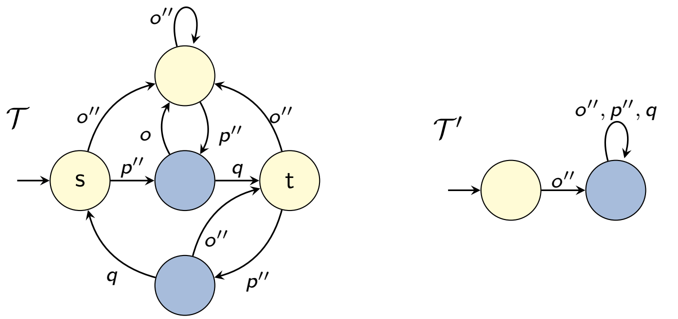
Label Reduction: Motivation (2)



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.)

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)

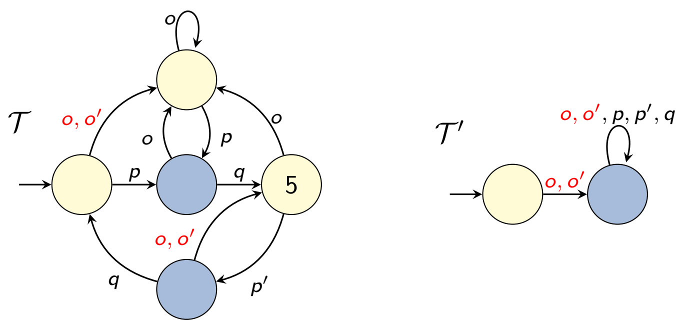
Let F be a factored transition systems with cost function c and label set L that contains no dead labels.

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 $\mathcal{T} \in F$.*

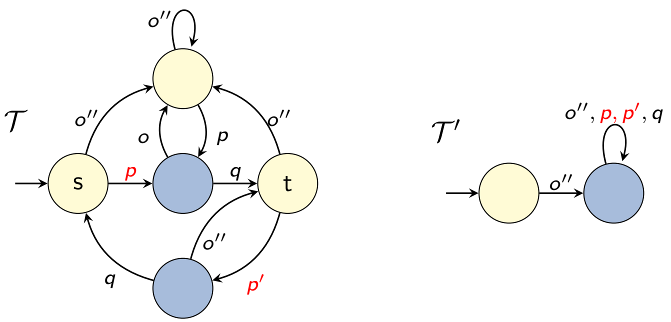
(Proof omitted.)

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)

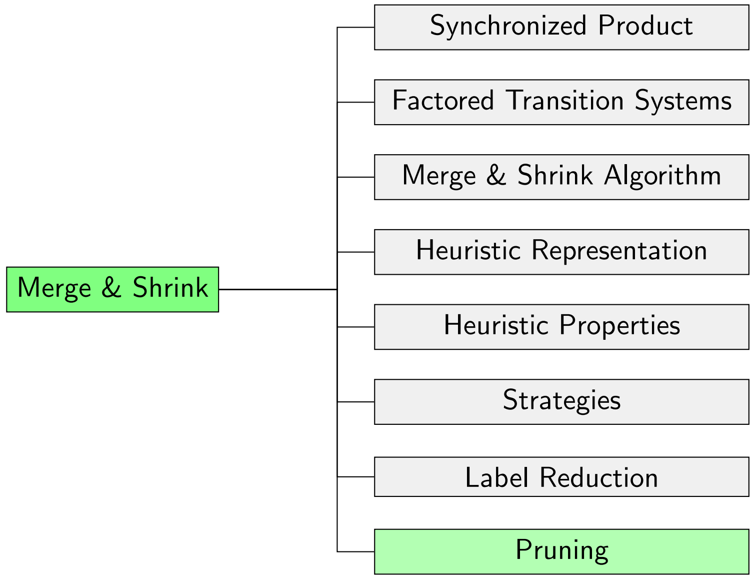
eq_i := set of label equivalence classes of $\mathcal{T}_i \in F$

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

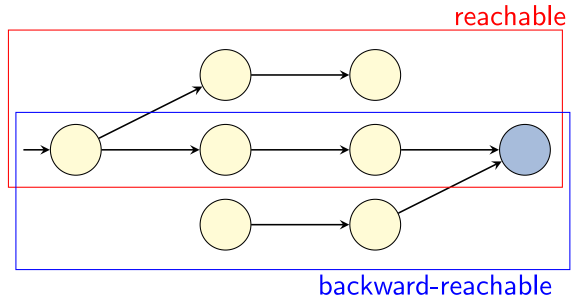
$eq := \{[\ell]_{\sim_c} \mid \ell \in L, \ell' \sim_c \ell'' \text{ iff } c(\ell') = c(\ell'')\}$
for $j \in \{1, \dots, |F|\} \setminus \{i\}$
    Refine eq with eq_j
// two labels are in the same set of eq iff they have
// the same cost and are locally equivalent in all $\mathcal{T}_j \neq \mathcal{T}_i$.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

E12.3 Pruning

Merge-and-Shrink



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 inadmissible 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.

E12.4 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. 106–1014, 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.

E12.5 Summary

Summary

- There is a wide range of merging 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.