Planning and Optimization

E13. Merge-and-Shrink: Pruning and Usage in Practise

Malte Helmert and Gabriele Röger

Universität Basel

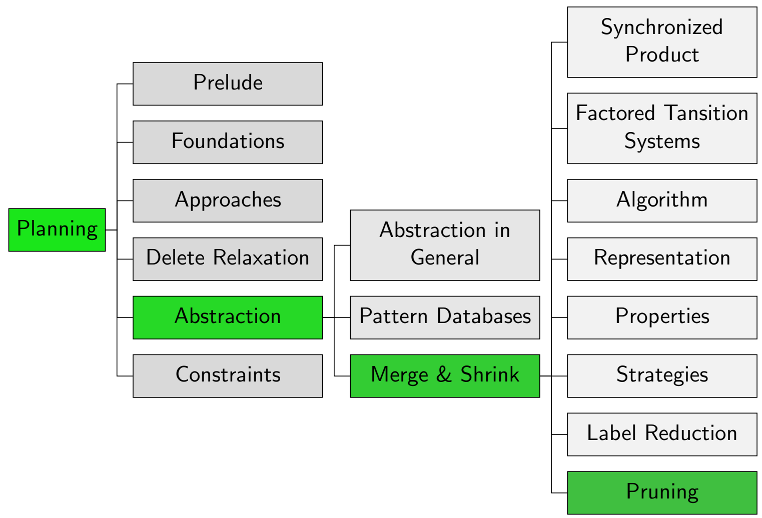
November 25, 2024

Planning and Optimization

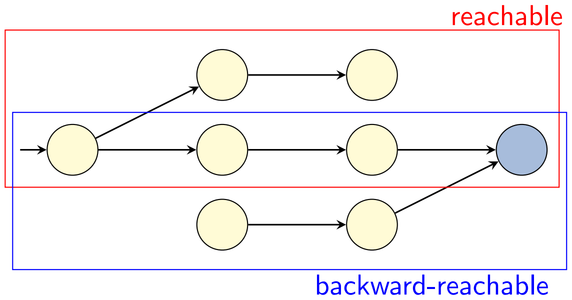
November 25, 2024 — E13. Merge-and-Shrink: Pruning and Usage in Practise

E13.1 Pruning

Content of the Course



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.

E13.2 Merge-and-Shrink in Practise

Merge-and-Shrink

- Merge-and-Shrink is a general framework.
- The full framework also covers label reduction and pruning.
- For all transformations, we need to select a strategy.
  merge, shrink, label reduction, pruning strategy
- The general strategy orchestrates the tranformations.
  How can this look like in practise?

Merge-and-Shrink in Fast Downward

Input: Factored transition system F, merge strategy MS, shrink strategy SS, prune strategy PS, label reduction strategy LRS, size limit $N \in \mathbb{N}$.
Output: Trans. system $\mathcal{T}$ and mapping σ from states of $\bigotimes F$ to states of $\mathcal{T}$.

▷ *Copy input factored transition system, compute Σ to represent the identity state mapping on $\bigotimes F'$, set λ to the identity label mapping.*
$\langle F', \Sigma, \lambda\rangle \leftarrow \langle F, \{\pi_{\mathcal{T}} \mid \mathcal{T} \in F'\}, \mathbf{id}\rangle$

for $\mathcal{T} \in F$ **do**
  ▷ *Prune atomic factor $\mathcal{T}$ with PS.*
  $\langle F', \Sigma, \lambda\rangle \leftarrow \textsc{ComposeTransformation}(\textsc{Prune}(F', \mathcal{T}))$
end for
...

Merge-and-Shrink in Fast Downward (cont'd)

while $|F'| > 1$ **do**
  ▷ *With MS, select two factors from F to be merged in this iteration.*
  $\mathcal{T}_1, \mathcal{T}_2, \leftarrow \textsc{Select}(F')$

  ▷ *With LRS, apply a label reduction to F'.*
  $\langle F', \Sigma, \lambda\rangle \leftarrow \textsc{ComposeTransformation}(\textsc{LabelReduction}(F'))$

  ▷ *With SS, shrink $\mathcal{T}_1$ and $\mathcal{T}_2$ so that the size of their product respects N.*
  $\langle F', \Sigma, \lambda\rangle \leftarrow \textsc{ComposeTransformation}(\textsc{Shrink}(F', \mathcal{T}_1, \mathcal{T}_2, N))$

  ▷ *With LRS, apply a label reduction to F'.*
  $\langle F', \Sigma, \lambda\rangle \leftarrow \textsc{ComposeTransformation}(\textsc{LabelReduction}(F'))$

  ▷ *Apply the merge transformation.*
  $\langle F', \Sigma, \lambda\rangle \leftarrow \textsc{ComposeTransformation}(\textsc{Merge}(F', \mathcal{T}_1, \mathcal{T}_2))$

  ▷ *With PS, prune the product factor $\mathcal{T}^{\otimes}$ of $\mathcal{T}_1$ and $\mathcal{T}_2$.*
  $\langle F', \Sigma, \lambda\rangle \leftarrow \textsc{ComposeTransformation}(\textsc{Prune}(F', \mathcal{T}^{\otimes}))$
end while
return single elements $\mathcal{T} \in F'$ and $\sigma \in \Sigma$

Stopping Early

- Merge-and-shrink has significant precomputation time before we can start the search.
- We typically stop the algorithm after a preset time (e.g. half of the time that is overall available).
- The factored transition system then still contains several factors. Each of them induces an individual heuristic.
- We can combine them by taking the maximum or use a generalization of operator cost partitioning (cf. Ch. F7/8) to labels to obtain better estimates.
- Cost partitioning benefits from additional snapshots of factors from several iterations of merge-and-shrink.

State of the art: snapshots and saturated cost partitioning (Ch.F8)

E13.3 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.

E13.4 Summary

Summary

- Pruning is a transformation that is used to keep the size of the factors small. It depends on the intended application how aggressive the pruning can be.
- In practise, it is beneficial to set a time limit for merge-and-shrink. The factors can be considered as individual admissible heuristics.