University of Basel

Non-Orthogonal Factored Transition Systems for Merge-and-Shrink

Luka Obser <luka.obser@unibas.ch>

Department of Mathematics and Computer Science, University of Basel

November 6, 2023

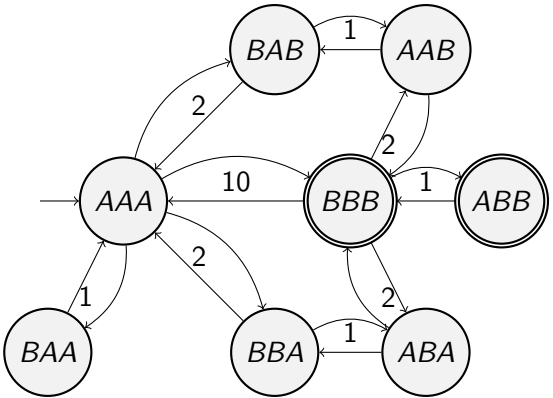
Contents

Planning Tasks

Planning Tasks

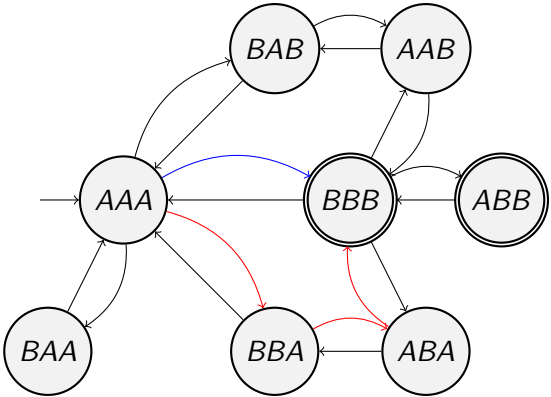
Transition Systems

- *BBA*: Car and cat are in location B, dog is in location A.
- Action are driving an empty car to A or B, driving the cat or the dog to A or B, and driving both to A or B.
- Driving without an animal costs 1, driving with one animal costs 2, and driving with both costs 10.



Solving Transition Systems

- Red: Plan of cost 5. Bring the cat, go back for the dog and bring it too.
- Blue: Plan of cost 10. Take both cat and dog on the first trip.

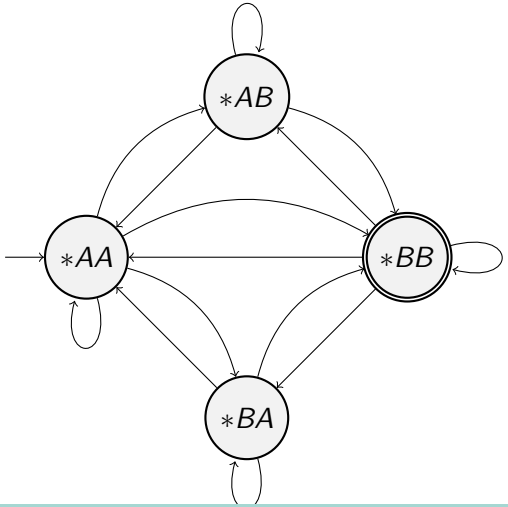


Heuristics

- Explicitly searching the full transition system is very expensive.
- If we knew how close a state is to a goal state, we could find the best path easily.
- Heuristics estimate the distance of a state to a goal state.
- Admissible heuristics never overestimate the distance of a state to a goal state.
- A^* search using an admissible heuristic finds optimal solutions.

Abstraction Heuristics

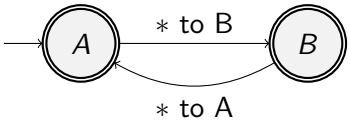
- Solving a simplified version (abstraction) of a transition system yields an admissible heuristic.
- Abstractions combine states of the original system into one in the abstract system.
- How can we generate good abstractions automatically?

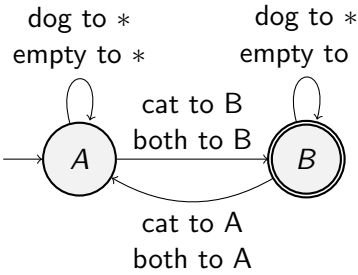


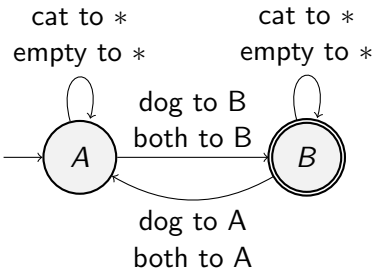
Factored Transition Systems

Definition

A **factored transition system** is a tuple $F = \langle \Theta^1, \ldots, \Theta^n \rangle$ of transition systems where each transition system has the same set of labels and the same label cost function.

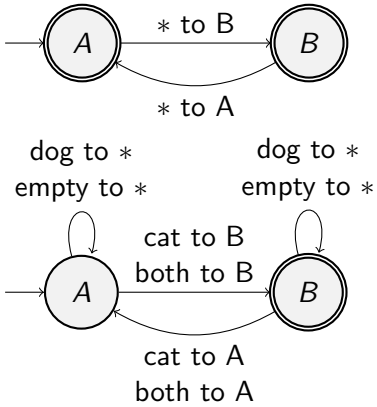






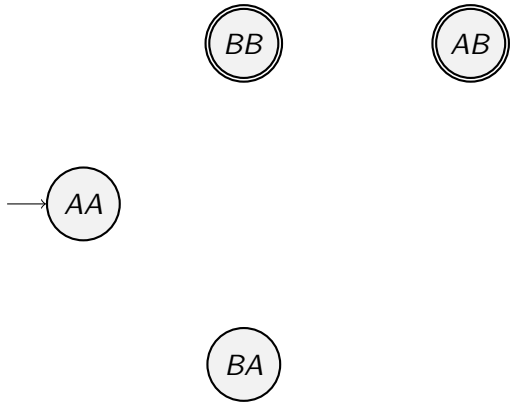
Merge-and-Shrink Heuristics

- We begin with the atomic factors.



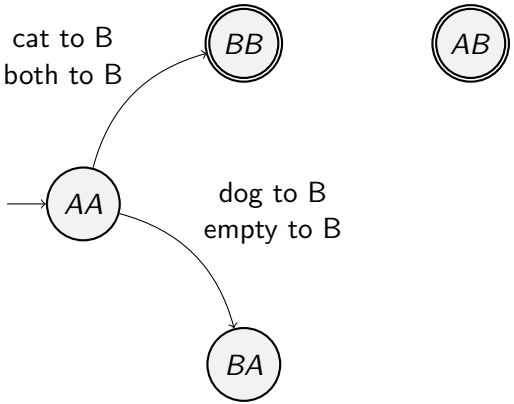
Merge-and-Shrink Heuristics

- We begin with the atomic factors.
- We combine (merge) two of them according to some merge strategy.



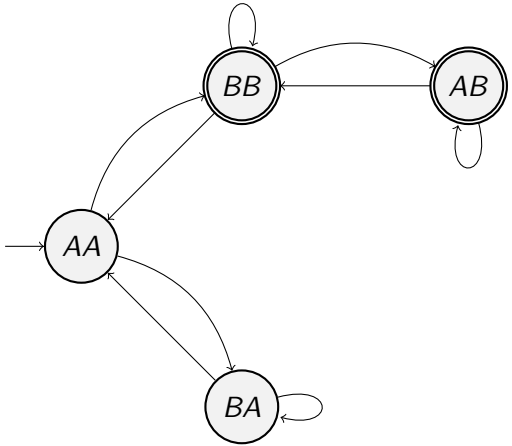
Merge-and-Shrink Heuristics

- We begin with the atomic factors.
- We combine (merge) two of them according to some merge strategy.



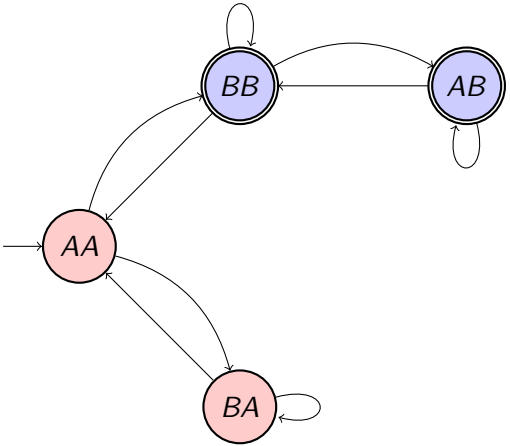
Merge-and-Shrink Heuristics

- We begin with the atomic factors.
- We combine (merge) two of them according to some merge strategy.



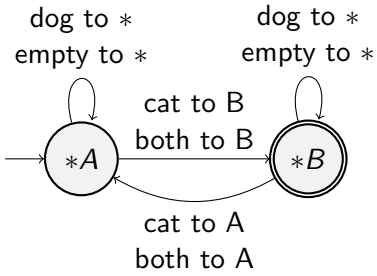
Merge-and-Shrink Heuristics

- We begin with the atomic factors.
- We combine (merge) two of them according to some merge strategy.
- We abstract (shrink) a factor according to some shrink strategy.



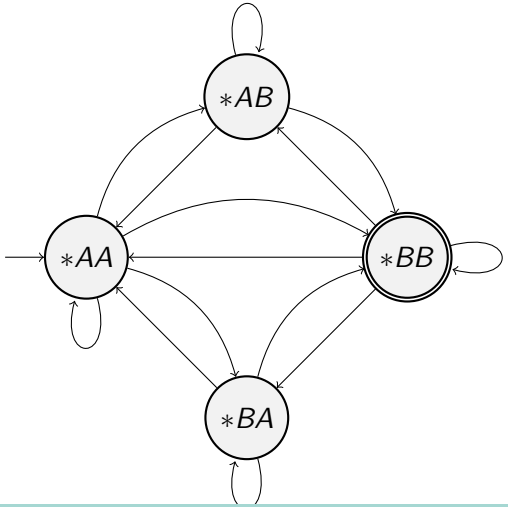
Merge-and-Shrink Heuristics

- We begin with the atomic factors.
- We combine (merge) two of them according to some merge strategy.
- We abstract (shrink) a factor according to some shrink strategy.



Abstraction Heuristics

- We begin with the atomic factors.
- We combine (merge) two of them according to some merge strategy.
- We abstract (shrink) a factor according to some shrink strategy.
- We repeat these until we terminate according to some general strategy.



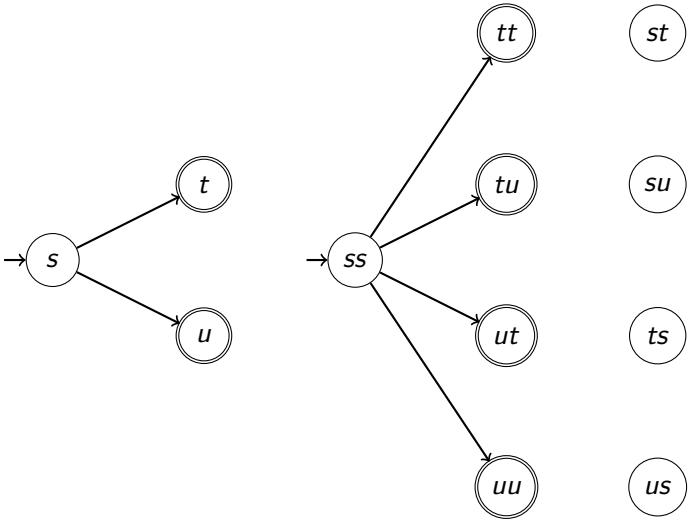
Cost Partitioning

- What if we are left with more than one factor?
- By distributing the cost of the operators among multiple abstractions we can add up the heuristic values obtained from each one admissibly!
- This may be better than taking the maximum.

Cloning a Factor

- We might want to merge a factor with two different ones but not all three of them.
- What if we simply "clone" a factor? Could a combination of merging and shrinking yield an inadmissible heuristic?
- Merging clones with each other creates spurious states and transitions.

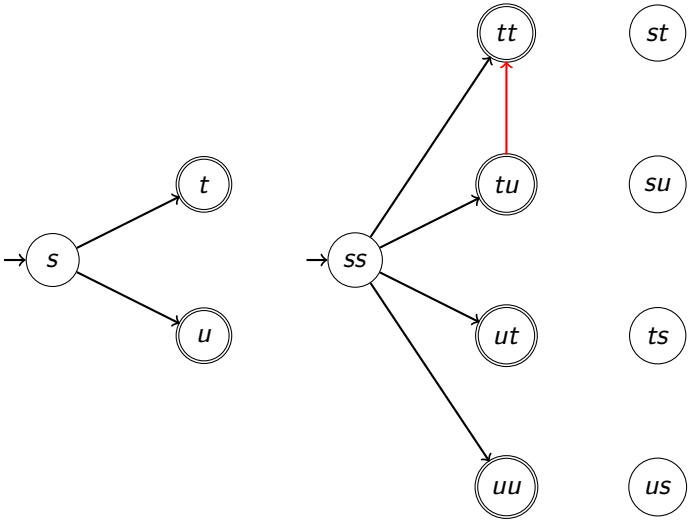
Cloning a Factor



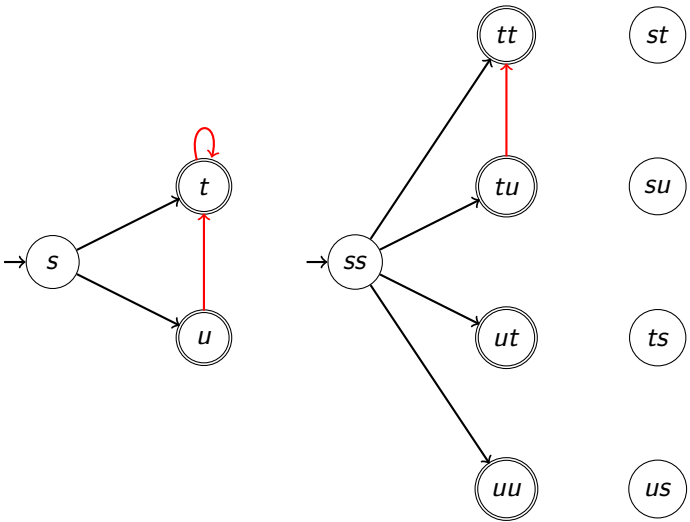
Cloning a Factor

- How problematic are the spurious states and transitions?
- Not very. Heuristic based on transition systems with spurious states and transitions are still admissible.
- Paths including spurious states and transitions have non-spurious counterparts that use the same order of labels.
- Spurious states are only reachable from the initial state by labels which also lead to all non-spurious counterparts.
- Transitions from spurious states only exist if their non-spurious counterparts have transitions with the same label.

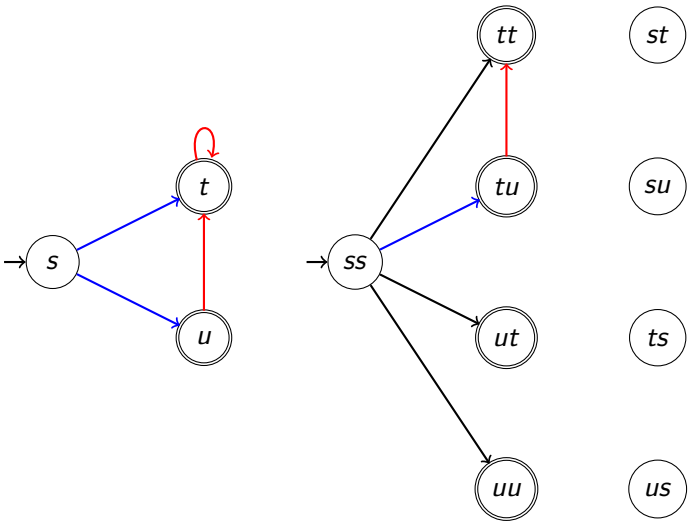
Cloning a Factor



Cloning a Factor



Cloning a Factor



Adhoc Cloning

- When should we clone a factor? Only ever useful if we merge the clone shortly after.
- Which factor do we clone? Use the merge strategy to determine candidates.
- Score based merge selection assigns scores to potential merges and tiebreaks if we have multiple equally good merges.
- What if we perform all of the equally good merges?

Adhoc Cloning

- When should we clone a factor? Only ever useful if we merge the clone shortly after.
- Which factor do we clone? Use the merge strategy to determine candidates.
- Score based merge selection assigns scores to potential merges and tiebreaks if we have multiple equally good merges.
- What if we perform all of the equally good merges?

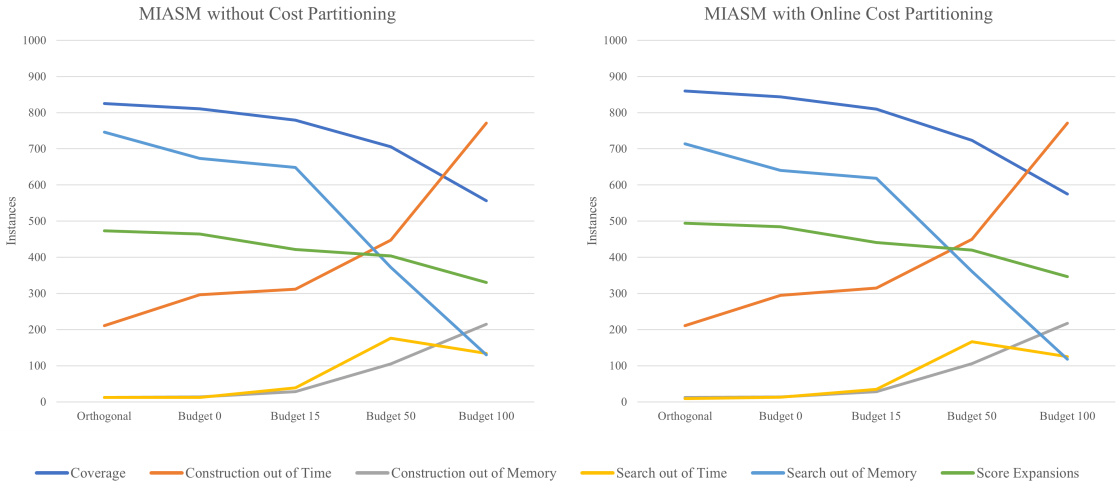
Adhoc Cloning

- When should we clone a factor? Only ever useful if we merge the clone shortly after.
- Which factor do we clone? Use the merge strategy to determine candidates.
- Score based merge selection assigns scores to potential merges and tiebreaks if we have multiple equally good merges.
- What if we perform all of the equally good merges?

Adhoc Cloning

- When should we clone a factor? Only ever useful if we merge the clone shortly after.
- Which factor do we clone? Use the merge strategy to determine candidates.
- Score based merge selection assigns scores to potential merges and tiebreaks if we have multiple equally good merges.
- What if we perform all of the equally good merges?

Adhoc Cloning



Adhoc Cloning

- The more we clone, the worse the performance. How come?
- Merge selection too greedy. Clones are not being utilized in a planned manner.
- If we do not prohibit the same merge it might use up our full budget.

Adhoc Cloning

- The more we clone, the worse the performance. How come?
- Merge selection too greedy. Clones are not being utilized in a planned manner.
- If we do not prohibit the same merge it might use up our full budget.

Adhoc Cloning

- The more we clone, the worse the performance. How come?
- Merge selection too greedy. Clones are not being utilized in a planned manner.
- If we do not prohibit the same merge it might use up our full budget.

Precomputed Cloning

- How can we use cloning in a less greedy way?
- Create multiple small transition systems containing overlapping variables and combine them using cost partitioning.
- How do we determine which variables to combine?
- Use neighborhoods in the causal graph of the task. Vary depth of subgraphs, which edges to include, and combine clusters to adhere to cloning limits.

Precomputed Cloning

- How can we use cloning in a less greedy way?
- Create multiple small transition systems containing overlapping variables and combine them using cost partitioning.
- How do we determine which variables to combine?
- Use neighborhoods in the causal graph of the task. Vary depth of subgraphs, which edges to include, and combine clusters to adhere to cloning limits.

Precomputed Cloning

- How can we use cloning in a less greedy way?
- Create multiple small transition systems containing overlapping variables and combine them using cost partitioning.
- How do we determine which variables to combine?
- Use neighborhoods in the causal graph of the task. Vary depth of subgraphs, which edges to include, and combine clusters to adhere to cloning limits.

Precomputed Cloning

- How can we use cloning in a less greedy way?
- Create multiple small transition systems containing overlapping variables and combine them using cost partitioning.
- How do we determine which variables to combine?
- Use neighborhoods in the causal graph of the task. Vary depth of subgraphs, which edges to include, and combine clusters to adhere to cloning limits.

Precomputed Cloning vs. State-of-the-Art SCC

	SCC	Precomputed Cloning
Coverage	901	831
Construction out of Time	27	253
Construction out of Memory	17	16
Search out of Time	508	15
Search out of Memory	356	692
Score Expansions	505.08	408.09
Score Search Time	740.12	709.74
Score Total Time	637.19	586.74

Conclusion

- Cloning allows for more flexibility in creating heuristics using the merge-and-shrink framework.
- Adhoc cloning can be improved with better scoring functions or new merge strategies.
- Precomputed cloning is a more promising approach. Choosing good subsets of state variables is an interesting topic for future work.

Questions?

luka.obser@unibas.ch