Planning and Optimization

C14. Merge-and-Shrink Abstractions: Generic Algorithm

Malte Helmert and Gabriele Röger

Universität Basel

November 14, 2016

Planning and Optimization

November 14, 2016 — C14. Merge-and-Shrink Abstractions: Generic Algorithm

C14.1 Generic Algorithm

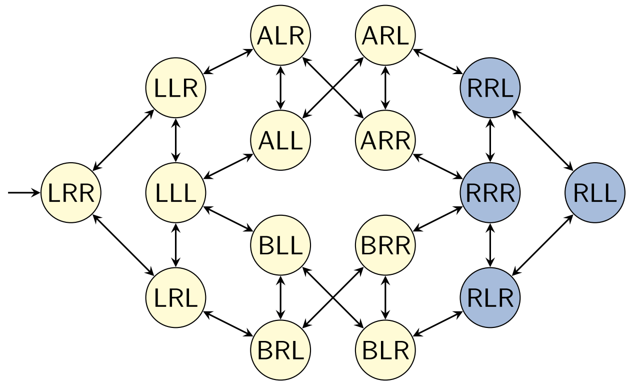
Generic Merge-and-shrink Abstractions: Outline

Using the results from the previous chapter, we can develop the ideas of a generic abstraction computation procedure that takes all state variables into account:

- Initialization step: Compute all abstract transition systems for atomic projections to form the initial abstraction collection.
- Merge steps: Combine two abstract systems in the collection by replacing them with their synchronized product. (Stop once only one transition system is left.)
- Shrink steps: If the abstractions in the collection are too large to compute their synchronized product, make them smaller by abstracting them further (applying an arbitrary abstraction to them).

We explain these steps with our running example.

Back to the Running Example

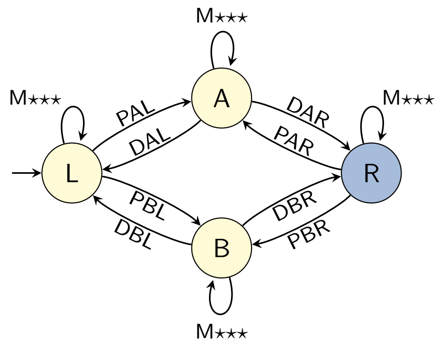


Logistics problem with one package, two trucks, two locations:

- state variable package: $\{L, R, A, B\}$
- state variable truck A: $\{L, R\}$
- state variable truck B: $\{L, R\}$

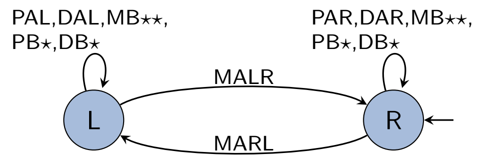
Initialization Step: Atomic Projection for Package

$\mathcal{T}^{\pi_{\{\text{package}\}}}$:



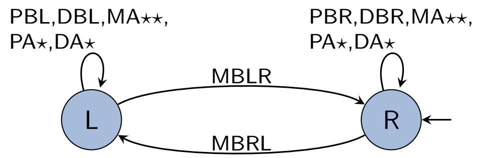
Initialization Step: Atomic Projection for Truck A

$\mathcal{T}^{\pi_{\{\text{truck A}\}}}$:



Initialization Step: Atomic Projection for Truck B

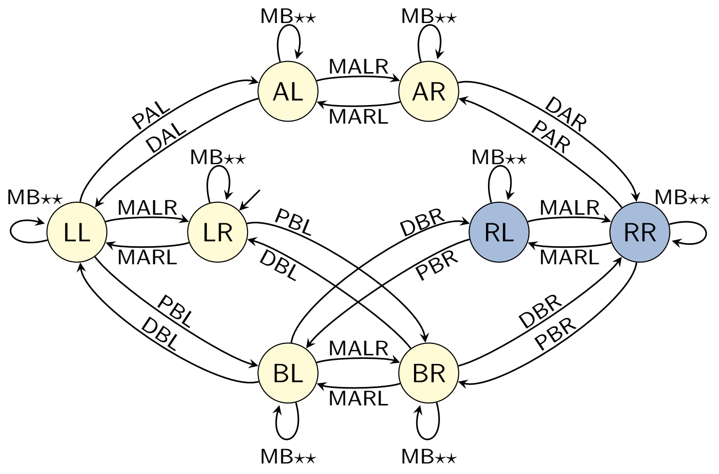
$\mathcal{T}^{\pi_{\{\text{truck B}\}}}$:



current collection: $\{\mathcal{T}^{\pi_{\{\text{package}\}}}, \mathcal{T}^{\pi_{\{\text{truck A}\}}}, \mathcal{T}^{\pi_{\{\text{truck B}\}}}\}$

First Merge Step

$\mathcal{T}_1 := \mathcal{T}^{\pi_{\{\text{package}\}}} \otimes \mathcal{T}^{\pi_{\{\text{truck A}\}}}$:



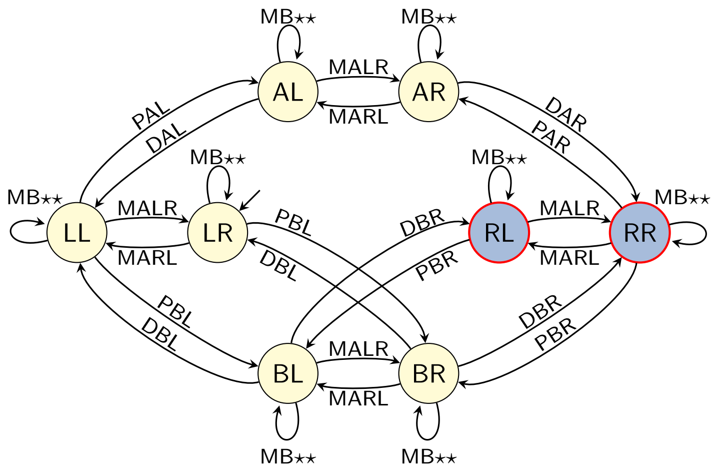
current collection: $\{\mathcal{T}_1, \mathcal{T}^{\pi_{\{\text{truck B}\}}}\}$

Need to Simplify?

- If we have sufficient memory available, we can now compute $\mathcal{T}_1 \otimes \mathcal{T}^{\pi_{\{\text{truck B}\}}}$, which would recover the complete transition system of the task.
- However, to illustrate the general idea, let us assume that we do not have sufficient memory for this product.
- More specifically, we will assume that after each product operation we need to reduce the result transition system to four states to obey memory constraints.
- So we need to reduce $\mathcal{T}_1$ to four states. We have a lot of leeway in deciding how exactly to abstract $\mathcal{T}_1$.
- In this example, we simply use an abstraction that leads to a good result in the end.

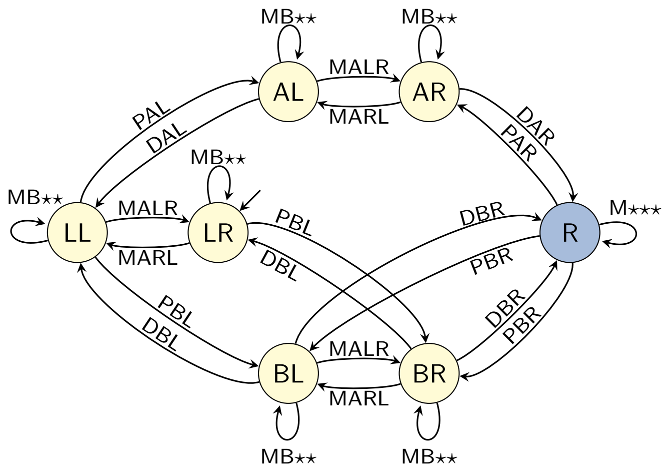
First Shrink Step

$\mathcal{T}_2 :=$ some abstraction of $\mathcal{T}_1$



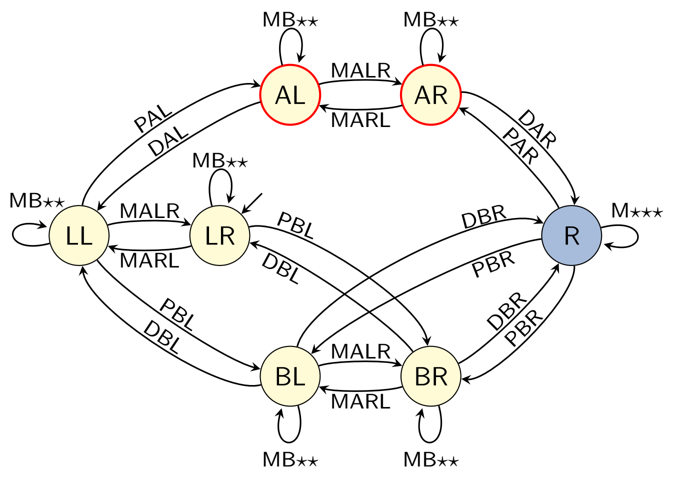
First Shrink Step

$\mathcal{T}_2 :=$ some abstraction of $\mathcal{T}_1$



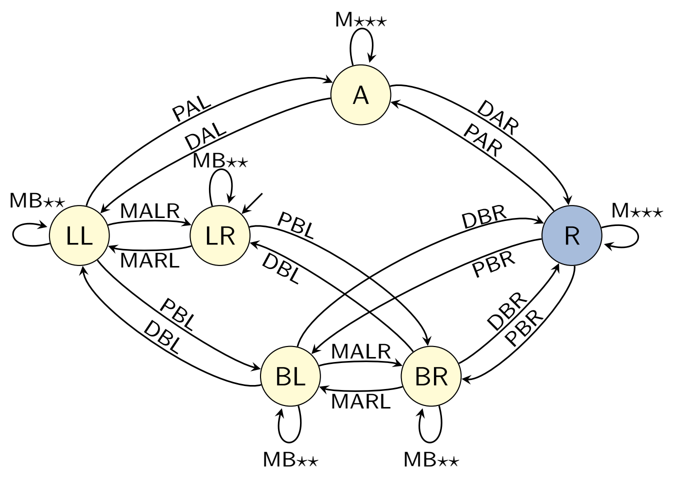
First Shrink Step

$\mathcal{T}_2 :=$ some abstraction of $\mathcal{T}_1$



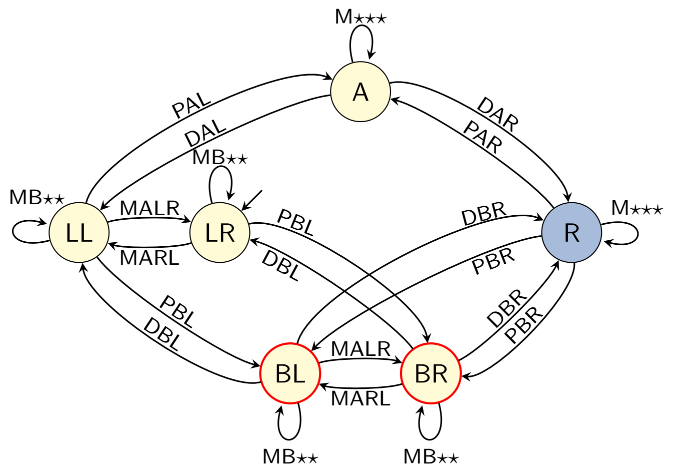
First Shrink Step

$\mathcal{T}_2 :=$ some abstraction of $\mathcal{T}_1$



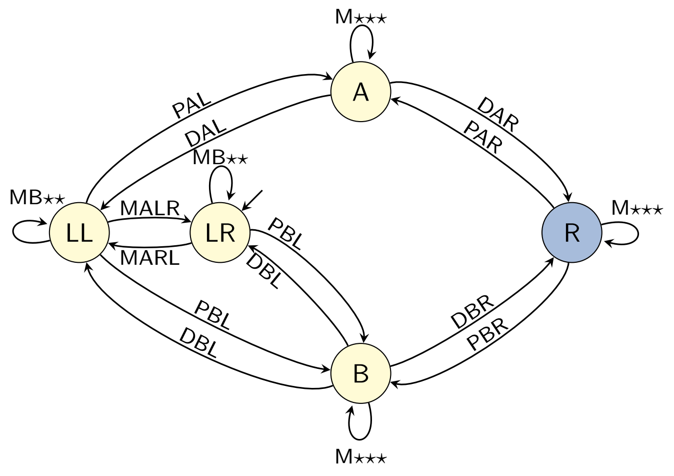
First Shrink Step

$\mathcal{T}_2 :=$ some abstraction of $\mathcal{T}_1$



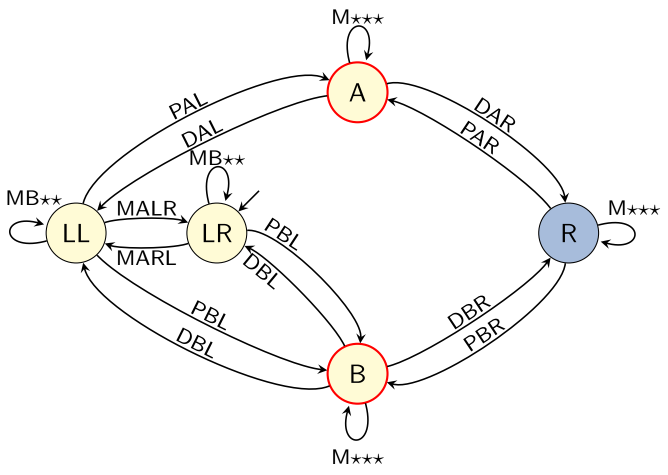
First Shrink Step

$\mathcal{T}_2 :=$ some abstraction of $\mathcal{T}_1$



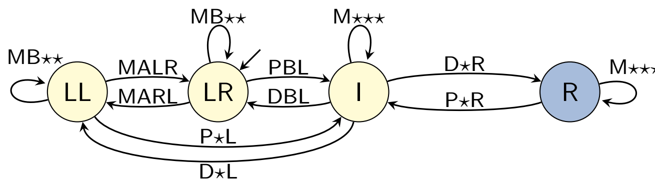
First Shrink Step

$\mathcal{T}_2 :=$ some abstraction of $\mathcal{T}_1$



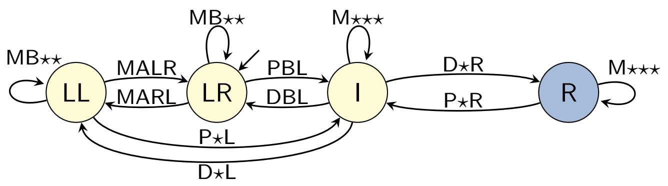
First Shrink Step

$\mathcal{T}_2 :=$ some abstraction of $\mathcal{T}_1$



First Shrink Step

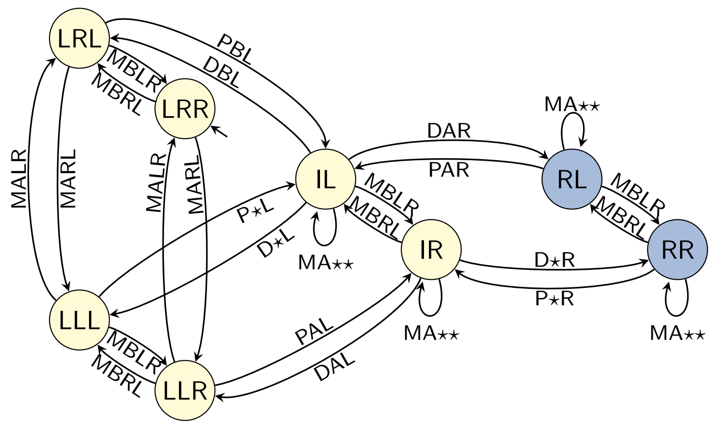
$\mathcal{T}_2 :=$ some abstraction of $\mathcal{T}_1$



current collection: $\{\mathcal{T}_2, \mathcal{T}^{\pi_{\{\text{truck B}\}}}\}$

Second Merge Step

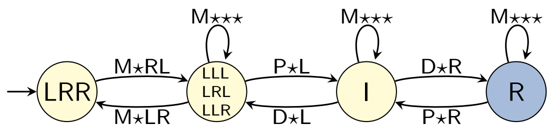
$\mathcal{T}_3 := \mathcal{T}_2 \otimes \mathcal{T}^{\pi_{\{\text{truck B}\}}}$:



current collection: $\{\mathcal{T}_3\}$

Another Shrink Step?

- Normally we could stop now and use the distances in the final abstract transition system as our heuristic function.
- However, if there were further state variables to integrate, we would simplify further, e.g. leading to the following abstraction (again with four states):



- We get a heuristic value of 3 for the initial state, better than any PDB heuristic that is a proper abstraction.
- The example generalizes to more locations and trucks, even if we stick to the size limit of 4 (after merging).

Generic Algorithm Template

Generic Abstraction Computation Algorithm

```
$abs := \{\mathcal{T}^{\pi_{\{v\}}} \mid v \in V\}$
**while** $abs$ contains more than one abstract transition system:
        select $\mathcal{A}_1$, $\mathcal{A}_2$ from $abs$
        shrink $\mathcal{A}_1$ and/or $\mathcal{A}_2$ until $size(\mathcal{A}_1) \cdot size(\mathcal{A}_2) \leq N$
        $abs := abs \setminus \{\mathcal{A}_1, \mathcal{A}_2\} \cup \{\mathcal{A}_1 \otimes \mathcal{A}_2\}$
**return** the remaining abstract transition system in $abs$
```

N: parameter bounding number of abstract states

Questions for practical implementation:

- Which abstractions to select? ⇝ merging strategy
- How to shrink an abstraction? ⇝ shrinking strategy
- How to choose N? ⇝ usually: as high as memory allows
- How to represent the corresponding abstraction?

C14.2 Summary

Summary

- Projections perfectly reflect a few state variables. Merge-and-shrink abstractions are a generalization that can reflect all state variables, but in a potentially lossy way.
- The merge steps combine two abstract transition systems by replacing them with their synchronized product.
- The shrink steps make an abstract system smaller by abstracting it further.