Foundations of Artificial Intelligence

38. Automated Planning: Merge-and-Shrink Abstractions

Martin Wehrle

Universität Basel

May 13, 2016

Foundations of Artificial Intelligence

May 13, 2016 — 38. Automated Planning: Merge-and-Shrink Abstractions

Automated Planning: Overview

Chapter overview: planning

- 33. Introduction
- 34. Planning Formalisms
- 35.–36. Planning Heuristics: Delete Relaxation
- 37.–38. Planning Heuristics: Abstraction
  - 37. Abstraction and Pattern Databases
  - 38. Merge-and-Shrink Abstractions
- 39.–40. Planning Heuristics: Landmarks

38.1 Merge-and-Shrink: Motivation

Beyond Pattern Databases (1)

- Despite their popularity, PDBs have fundamental restrictions: the patterns must be kept small.
- We have to pay a price regarding heuristic accuracy:
  - consider generalization of example from previous chapter: N trucks, M locations (one package)
  - Consider an arbitrary pattern that does not contain all variables in V.
  - $h(s_0) \leq 2 \rightsquigarrow$ no better than atomic projection to the package

Beyond Pattern Databases (2)

Merge-and-shrink abstractions (M&S) are a proper generalization of PDBs.

- They can represent PDBs (with polynomial overhead).
- They can sometimes represent abstractions compactly where this is impossible with PDBs.

German: Merge-and-Shrink-Abstraktionen

Merge-and-Shrink: Difference to PDBs

M&S Abstractions vs. Pattern Databases

While PDBs represent a few state variables perfectly in the abstract state space, M&S abstractions represent all state variables, but in a potentially lossy fashion.

38.2 Synchronized Product

M&S Abstraction: Basic Idea

basic ideas of M&S:

1. Information about two abstract state spaces $\mathcal{A}$ and $\mathcal{A}'$ for the same concrete state space can be combined through a simple operation on graphs: synchronized product $\mathcal{A} \otimes \mathcal{A}'$.
2. The concrete state space $\mathcal{S}$ of a SAS$^+$ planning task can be reconstructed based on the atomic projections:

$$\bigotimes_{v \in V} \mathcal{S}^{\pi_{\{v\}}} \text{ is isomorphic to } \mathcal{S}$$

$\rightsquigarrow$ build finer abstractions from coarse abstractions

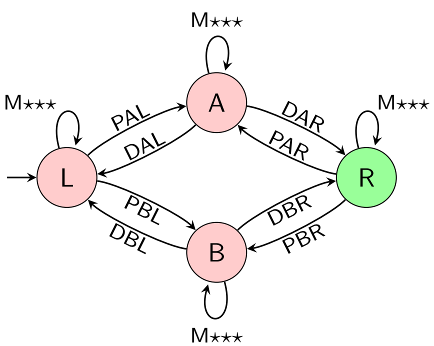
German: synchrones Produkt

Example: Abbreviations

- For the synchronized product, the edge annotations in the state spaces (the "action names") are important.
- Therefore we will include them in the following figures.
- We use abbreviations as follows:
  - MALR: move truck A from left to right
  - DAR: drop package from truck A at right location
  - PBL: pick up package with truck B at left location
- Often there are several parallel edges. We abbreviate them with commas and wild cards as in the following examples:
  - PAL, DAL: parallel edges for actions PAL and DAL
  - MA⋆⋆: parallel edges for actions MALR and MARL

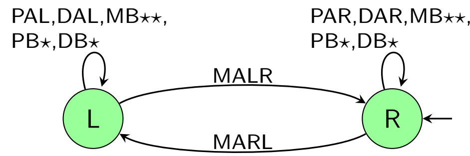
Example: Atomic Projection onto the Package

$\mathcal{S}^{\pi_{\{package\}}}$:



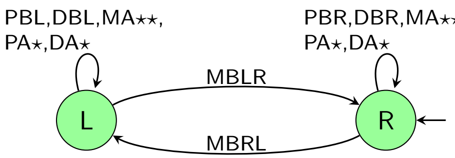
Example: Atomic Projection onto Truck A

$\mathcal{S}^{\pi_{\{truck\ A\}}}$:



Example: Atomic Projection onto Truck B

$\mathcal{S}^{\pi_{\{truck\ B\}}}$:



Synchronized Product of State Spaces

Definition (synchronized product of state spaces)

For $i \in \{1, 2\}$, let $\mathcal{S}^i = \langle S^i, A, cost, T^i, s_0^i, S_\star^i \rangle$ be state spaces (with equal sets of actions and costs).

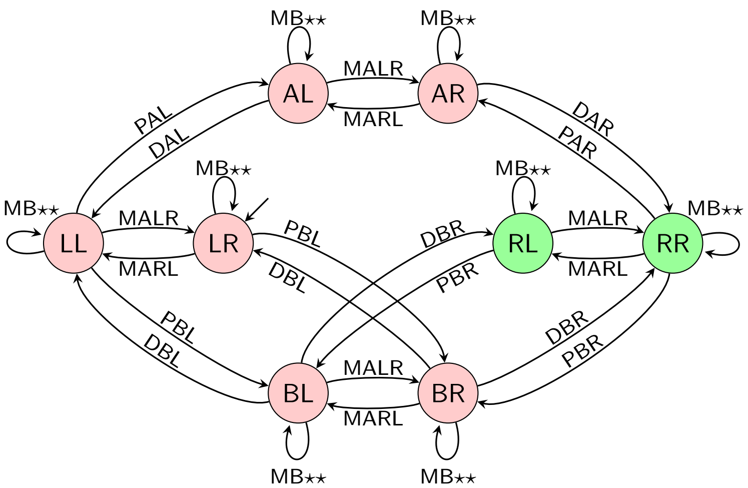
The synchronized product of $\mathcal{S}^1$ and $\mathcal{S}^2$, denoted by $\mathcal{S}^1 \otimes \mathcal{S}^2$, is the state space $\mathcal{S}^\otimes = \langle S^\otimes, A, cost, T^\otimes, s_0^\otimes, S_\star^\otimes \rangle$ with

- $S^\otimes := S^1 \times S^2$
- $T^\otimes := \{\langle \langle s_1, s_2 \rangle, a, \langle t_1, t_2 \rangle \rangle \mid \langle s_1, a, t_1 \rangle \in T^1 \wedge \langle s_2, a, t_2 \rangle \in T^2\}$
- $s_0^\otimes := \langle s_0^1, s_0^2 \rangle$
- $S_\star^\otimes := S_\star^1 \times S_\star^2$

German: synchrones Produkt von Zustandsräumen

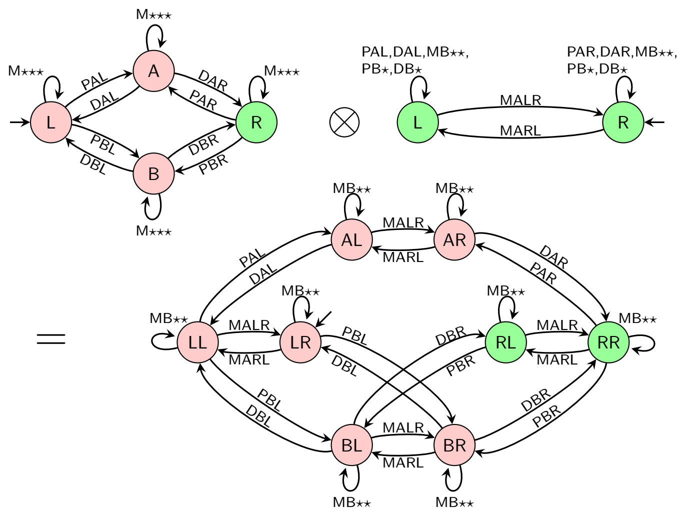
Example: Synchronized Product

$\mathcal{S}^{\pi_{\{package\}}} \otimes \mathcal{S}^{\pi_{\{truck\ A\}}}$:



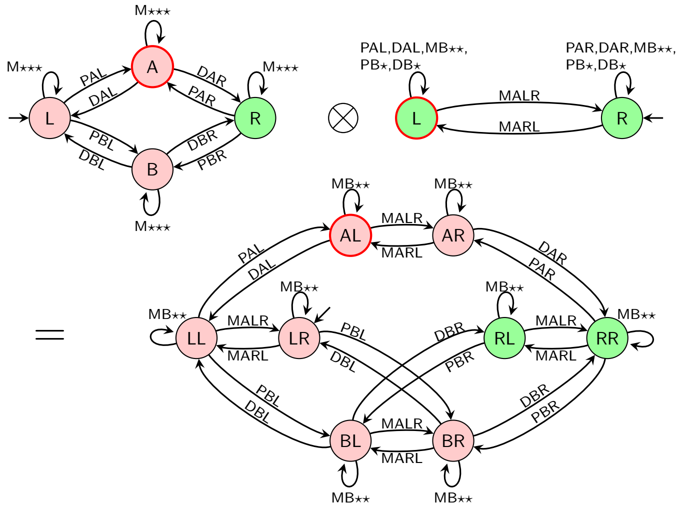
Example: Computation of Synchronized Product

$\mathcal{S}^{\pi_{\{package\}}} \otimes \mathcal{S}^{\pi_{\{truck\ A\}}}$:



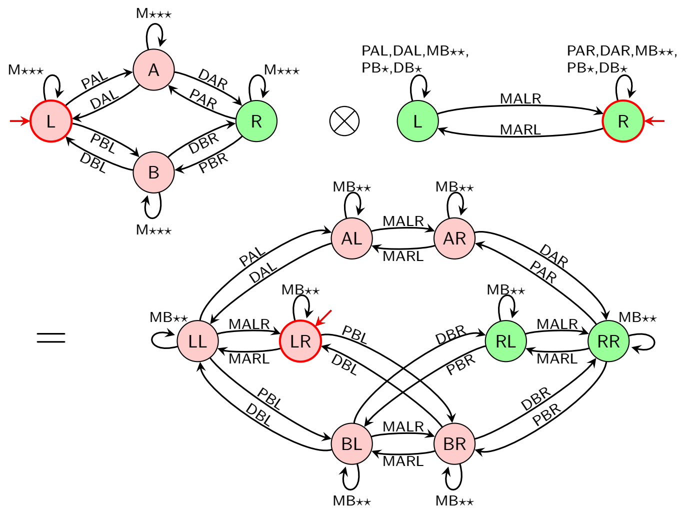
Example: Computation of Synchronized Product

$\mathcal{S}^{\pi\{\textit{package}\}} \otimes \mathcal{S}^{\pi\{\textit{truck A}\}}$: $S^{\otimes} = S^1 \times S^2$



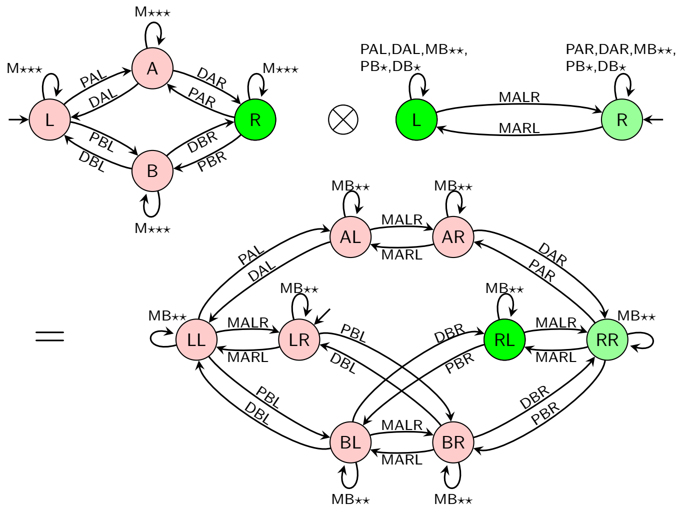
Example: Computation of Synchronized Product

$\mathcal{S}^{\pi\{\textit{package}\}} \otimes \mathcal{S}^{\pi\{\textit{truck A}\}}$: $s_0^{\otimes} = \langle s_0^1, s_0^2 \rangle$



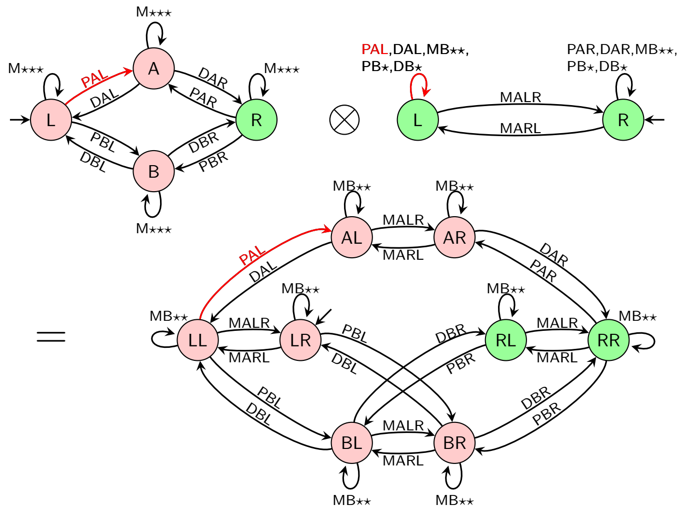
Example: Computation of Synchronized Product

$\mathcal{S}^{\pi\{\textit{package}\}} \otimes \mathcal{S}^{\pi\{\textit{truck A}\}}$: $S_{\star}^{\otimes} = S_{\star}^1 \times S_{\star}^2$



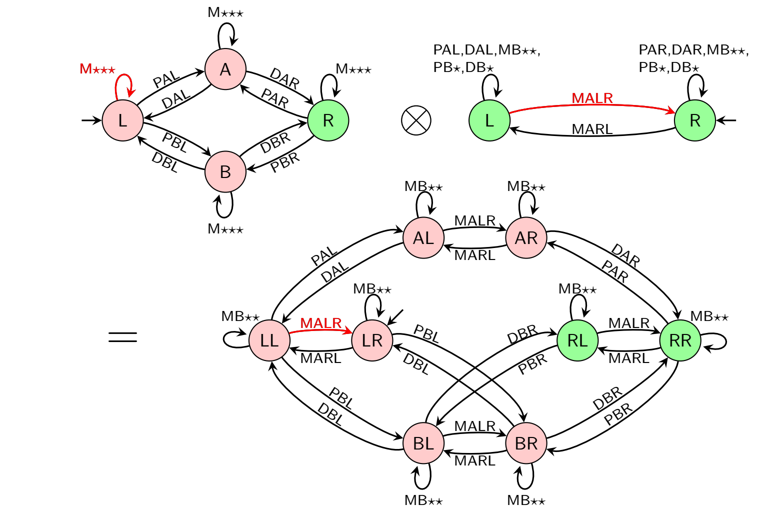
Example: Computation of Synchronized Product

$\mathcal{S}^{\pi\{\textit{package}\}} \otimes \mathcal{S}^{\pi\{\textit{truck A}\}}$: $T^{\otimes} = \{\langle\langle s_1, s_2\rangle, a, \langle t_1, t_2\rangle\rangle \mid \dots\}$



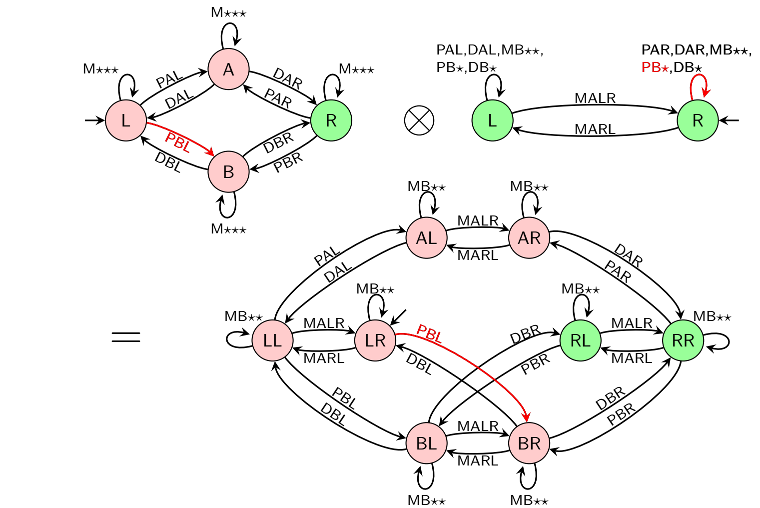
Example: Computation of Synchronized Product

$\mathcal{S}^{\pi_{\{package\}}} \otimes \mathcal{S}^{\pi_{\{truck\ A\}}}$: $T^{\otimes} = \{\langle\langle s_1, s_2\rangle, a, \langle t_1, t_2\rangle\rangle \mid \dots\}$



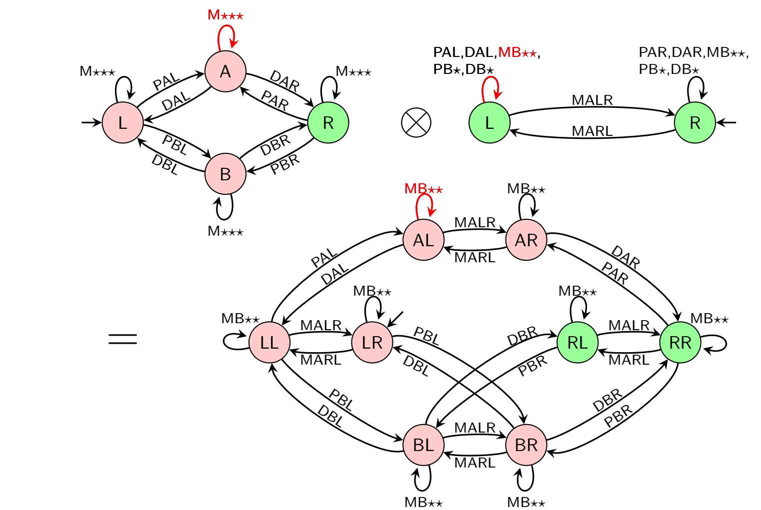
Example: Computation of Synchronized Product

$\mathcal{S}^{\pi_{\{package\}}} \otimes \mathcal{S}^{\pi_{\{truck\ A\}}}$: $T^{\otimes} = \{\langle\langle s_1, s_2\rangle, a, \langle t_1, t_2\rangle\rangle \mid \dots\}$



Example: Computation of Synchronized Product

$\mathcal{S}^{\pi_{\{package\}}} \otimes \mathcal{S}^{\pi_{\{truck\ A\}}}$: $T^{\otimes} = \{\langle\langle s_1, s_2\rangle, a, \langle t_1, t_2\rangle\rangle \mid \dots\}$



38.3 Merge-and-Shrink

M&S Abstractions: Basic Ideas (Continued)

basic ideas of M&S:

1. Information about two abstract state spaces $\mathcal{A}$ and $\mathcal{A}'$ for the same concrete state space can be combined through a simple operation on graphs: synchronized product $\mathcal{A} \otimes \mathcal{A}'$.
2. The concrete state space $\mathcal{S}$ of a SAS$^+$ planning task can be reconstructed based on the atomic projections:

$$\bigotimes_{v \in V} \mathcal{S}^{\pi_{\{v\}}} \text{ is isomorphic to } \mathcal{S}$$

   $\rightsquigarrow$ build finer abstractions from coarse abstractions
3. If intermediate results are too big: reduce the size by combining some of the abstract states

Computation of M&S Abstractions

Generic Algorithm for the Computation of M&S Abstractions

abs := $\{\mathcal{S}^{\pi_{\{v\}}} \mid v \in V\}$ [abstraction set for atomic projections]
while $|abs| > 1$:
  select $\mathcal{S}^1$, $\mathcal{S}^2$ from *abs*
  shrink $\mathcal{S}^1$ and/or $\mathcal{S}^2$ until size$(\mathcal{S}^1) \cdot$ size$(\mathcal{S}^2) \leq K$
  abs := $(abs \setminus \{\mathcal{S}^1, \mathcal{S}^2\}) \cup \{\mathcal{S}^1 \otimes \mathcal{S}^2\}$ [merge step]
return the remaining abstraction in *abs*

K: parameter that bounds the maximal number of abstract states

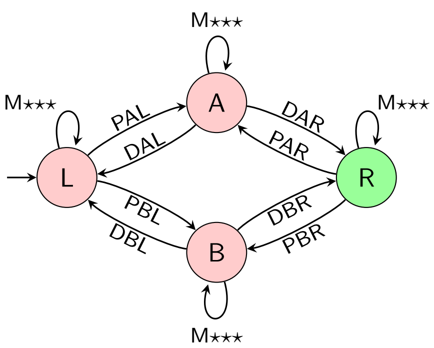
practical implementations need to decide:

- Which abstractions to merge? $\rightsquigarrow$ merge strategy
- How to shrink abstractions? $\rightsquigarrow$ shrink strategy
- How to choose K?

German: Merge-Strategie, Shrink-Strategie

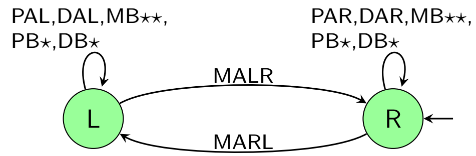
Initial Step: Atomic Projection onto the Package

$\mathcal{S}^{\pi_{\{package\}}}$:



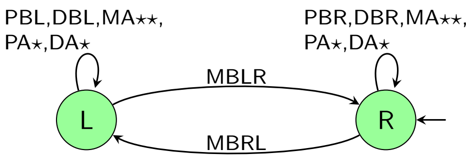
Initial Step: Atomic Projection onto Truck A

$\mathcal{S}^{\pi_{\{truck\ A\}}}$:



Initial Step: Atomic Projection onto Truck B

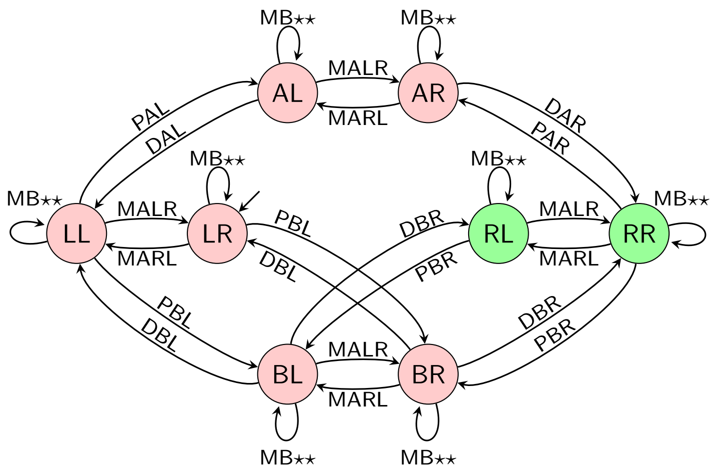
$\mathcal{S}^{\pi_{\{truck\ B\}}}$:



current abstraction set: $abs = \{\mathcal{S}^{\pi_{\{package\}}}, \mathcal{S}^{\pi_{\{truck\ A\}}}, \mathcal{S}^{\pi_{\{truck\ B\}}}\}$

First Merge Step

$\mathcal{S}^1 := \mathcal{S}^{\pi_{\{package\}}} \otimes \mathcal{S}^{\pi_{\{truck\ A\}}}$:



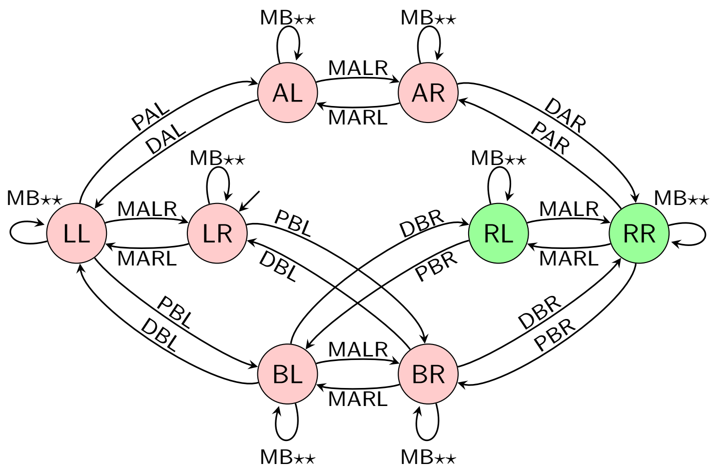
current abstraction set: $abs = \{\mathcal{S}^1, \mathcal{S}^{\pi_{\{truck\ B\}}}\}$

Do We Need to Shrink?

- If enough memory is available, we can now compute $\mathcal{S}^1 \otimes \mathcal{S}^{\pi_{\{truck\ B\}}}$, resulting in the concrete state space.
- To illustrate the general idea, we assume that insufficient memory is available for this product.
- In more detail: assume that after every merge step we must reduce to four states to fit into memory.

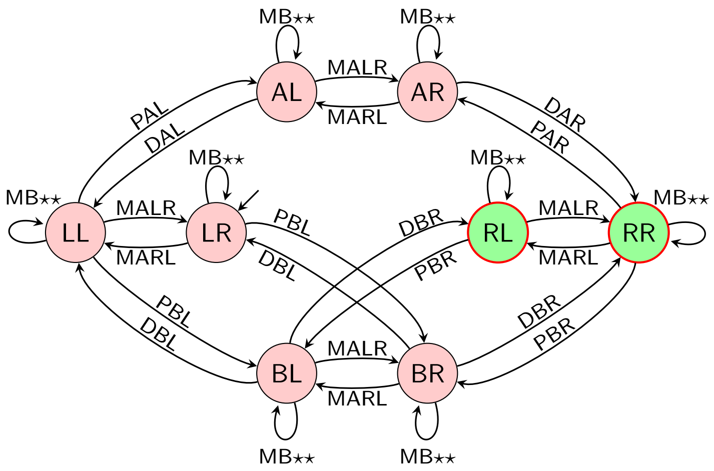
First Shrink Step

$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



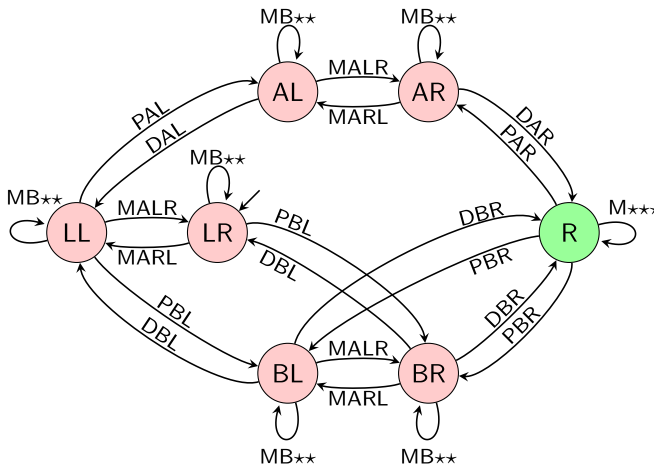
First Shrink Step

$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



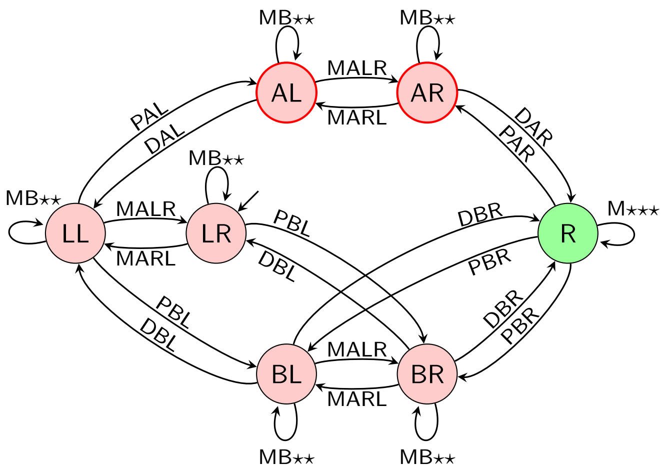
First Shrink Step

$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



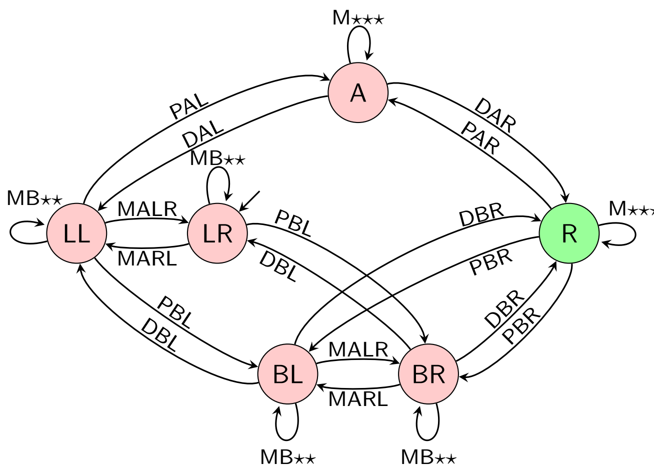
First Shrink Step

$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



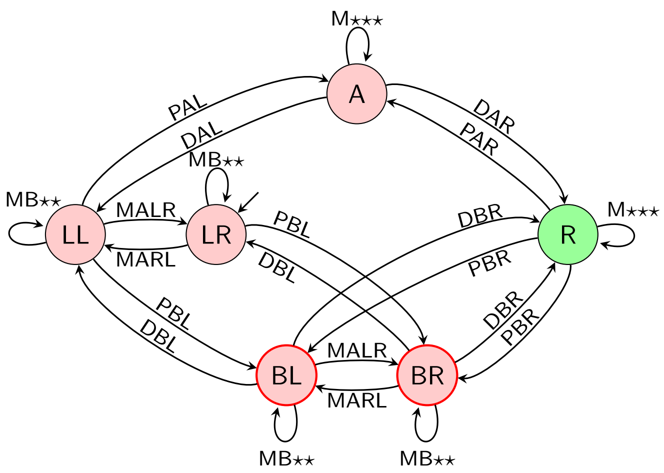
First Shrink Step

$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



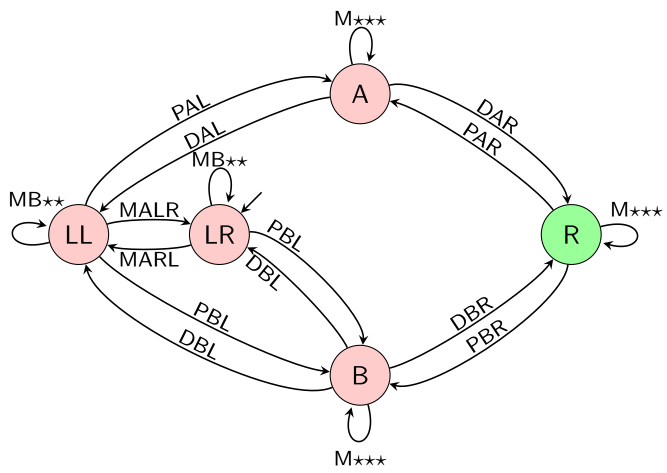
First Shrink Step

$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



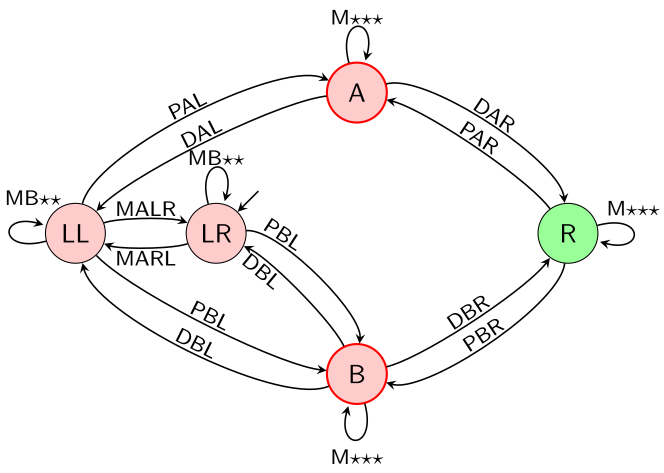
First Shrink Step

$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



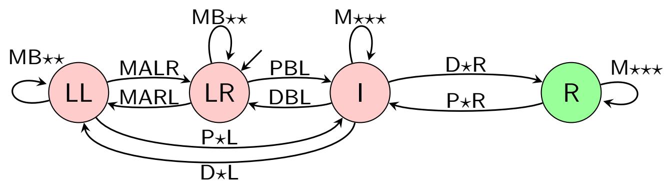
First Shrink Step

$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



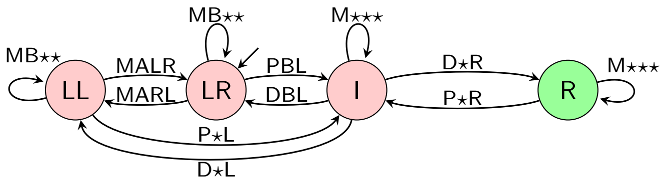
First Shrink Step

$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



First Shrink Step

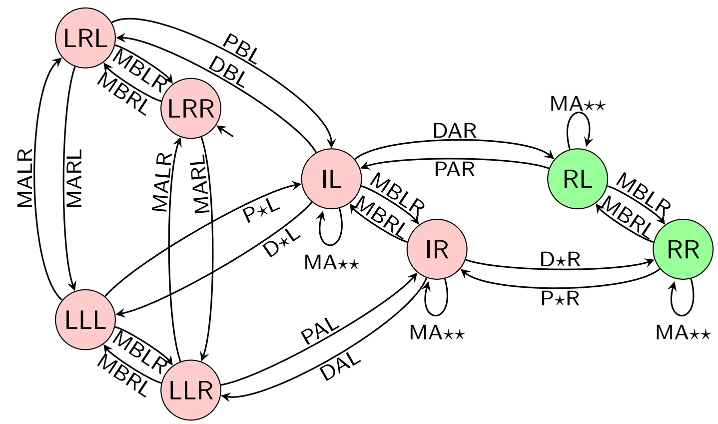
$\mathcal{S}^2 :=$ some abstraction of $\mathcal{S}^1$



current abstraction set: $abs = \{\mathcal{S}^2, \mathcal{S}^{\pi_{\{truck\ B\}}}\}$

Second Merge Step

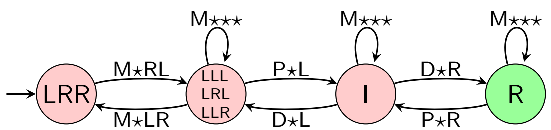
$\mathcal{S}^3 := \mathcal{S}^2 \otimes \mathcal{S}^{\pi_{\{truck\ B\}}}$:



current abstraction set: $\{\mathcal{S}^3\}$

Second Shrink Step

- To illustrate the idea (the generic algorithm would terminate at this point), we further shrink $\mathcal{S}^3$ to $\mathcal{S}^4$:



- We get a heuristic value of 3 for the initial state
  ⇝ better than any PDB heuristic
  that does not already contain all variables in the pattern.
- The example can be generalized to more locations and trucks, and the size bound of 4 (after the merge step) still suffices.

Merge-and-Shrink Abstractions in Practice

practical aspects that are not discussed further in this course:

- How to choose the size bound?
- Which merge strategies are suitable?
- Which shrink strategies are suitable?
- How to implement merge-and-shrink efficiently?
  - suitable data structures and algorithms important!

38.4 Summary

Summary (1)

- merge-and-shrink abstractions: instead of considering few variables with perfect precision in the abstraction, consider all variables in a lossy fashion
- synchronized product: graph operation that combines two (abstract) transition systems into a new one
- planning task can be reconstructed completely from the synchronized product of the atomic abstractions

Summary (2)

- merge-and-shrink:
  - start with all atomic abstractions, then repeatedly:
  - replace two abstractions by their synchronized product (merge)
  - reduce the size of the abstraction (if too big) in order to fit into memory (shrink)
- in practice: good merge and shrink strategies important