

# Foundations of Artificial Intelligence

## F6. Automated Planning: Abstraction Heuristics

Malte Helmert

University of Basel

May 6, 2026

# Foundations of Artificial Intelligence

May 6, 2026 — F6. Automated Planning: Abstraction Heuristics

## F6.1 Abstraction Heuristics

## F6.2 Pattern Databases

## F6.3 Summary

# Automated Planning: Overview

## Chapter overview: automated planning

- ▶ F1. Introduction
- ▶ F2. Planning Formalisms
- ▶ F3. Delete Relaxation
- ▶ F4. Delete Relaxation Heuristics
- ▶ F5. Abstraction
- ▶ F6. Abstraction Heuristics

# F6.1 Abstraction Heuristics

# Abstraction Heuristic

Given an abstraction function  $\alpha$  for a state space  $\mathcal{S}$ , use **abstract solution cost** (solution cost of  $\alpha(s)$  in  $\mathcal{S}^\alpha$ ) as heuristic for **concrete solution cost** (solution cost of  $s$  in  $\mathcal{S}$ ).

## Definition (abstraction heuristic)

The **abstraction heuristic** for abstraction  $\alpha$  maps each state  $s$  to its abstract solution cost

$$h^\alpha(s) = h_{\mathcal{S}^\alpha}^*(\alpha(s)),$$

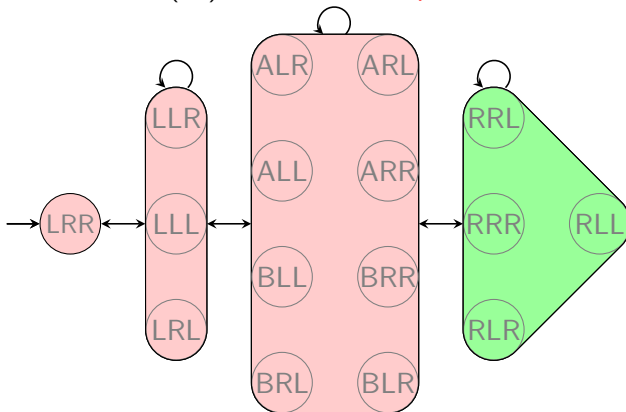
where  $h_{\mathcal{S}^\alpha}^*$  is the perfect heuristic in  $\mathcal{S}^\alpha$ .

**German:** abstrakte/konkrete Zielabstände, Abstraktionsheuristik



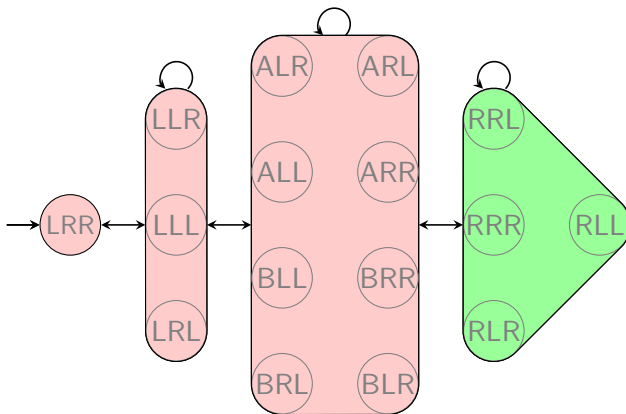
# Abstraction: Example

(an) abstract state space



**Remark:** Most arcs correspond to several (parallel) transitions with different labels.

# Abstraction Heuristic: Example



$$h^\alpha(\{p \mapsto L, t_A \mapsto R, t_B \mapsto R\}) = 3$$

# Abstraction Heuristics: Discussion

- ▶ Every abstraction heuristic is **admissible** and **consistent**.  
(proof idea?)
- ▶ The choice of the **abstraction function**  $\alpha$  is very important.
  - ▶ **Every**  $\alpha$  yields an admissible and consistent heuristic.
  - ▶ But most  $\alpha$  lead to poor heuristics.
- ▶ An effective  $\alpha$  must yield an **informative heuristic** ...
- ▶ ... as well as being **efficiently computable**.
- ▶ **How to find a suitable  $\alpha$ ?**

# Automatic Computation of Suitable Abstractions

## Main Problem with Abstraction Heuristics

How to find a good abstraction?

Several successful methods:

- ▶ **pattern databases (PDBs)**  $\rightsquigarrow$  this course  
(Culberson & Schaeffer, 1996)
- ▶ **merge-and-shrink** abstractions  
(Dräger, Finkbeiner & Podelski, 2006)
- ▶ **Cartesian** abstractions (Seipp & Helmert, 2013)
- ▶ **domain** abstractions (Kreft et al., 2023)

**German:** Pattern Databases, Merge-and-Shrink-Abstraktionen,  
Kartesische Abstraktionen, Domänenabstraktionen

## F6.2 Pattern Databases

# Pattern Databases: Background

- ▶ The most common abstraction heuristics are **pattern database heuristics**.
- ▶ originally introduced for the **15-puzzle** (Culberson & Schaeffer, 1996) and for **Rubik's Cube** (Korf, 1997)
- ▶ introduced for **automated planning** by Edelkamp (2001)
- ▶ for many search problems the **best known** heuristics
- ▶ many many research papers studying
  - ▶ theoretical properties
  - ▶ efficient implementation and application
  - ▶ pattern selection
  - ▶ ...

## Pattern Databases: Projections

A PDB heuristic for a planning task is an abstraction heuristic where

- ▶ some aspects (= state variables) of the task are preserved **with perfect precision** while
- ▶ all other aspects are not preserved **at all**.

formalized as **projections** to a **pattern**  $P \subseteq V$ :

$$\pi_P(s) = \{v \mapsto s(v) \mid v \in P\}$$

example:

- ▶  $s = \{p \mapsto L, t_A \mapsto R, t_B \mapsto R\}$
- ▶ **projection** on  $P = \{p\}$  (= ignore trucks):  
 $\pi_P(s) = \{p \mapsto L\}$
- ▶ **projection** on  $P = \{p, t_A\}$  (= ignore truck  $B$ ):  
 $\pi_P(s) = \{p \mapsto L, t_A \mapsto R\}$

German: Projektionen

# Pattern Databases: Definition

## Definition (pattern database heuristic)

Let  $P$  be a subset of the variables of a planning task.

The abstraction heuristic induced by the **projection**  $\pi_P$  on  $P$  is called **pattern database heuristic** (**PDB heuristic**) with **pattern**  $P$ .

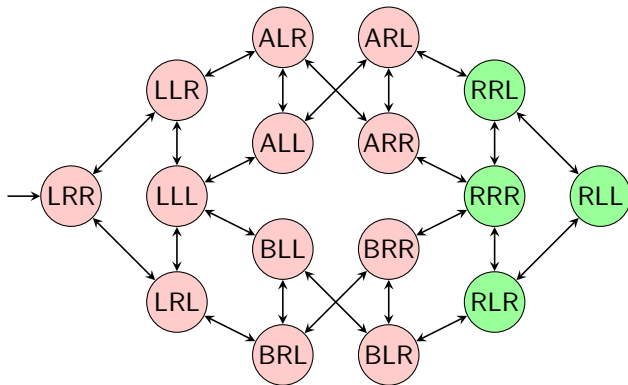
abbreviated notation:  $h^P$  for  $h^{\pi_P}$

German: Pattern-Database-Heuristik

remark:

- ▶ “pattern databases” in analogy to **endgame databases** (which have been successfully applied in 2-person-games)

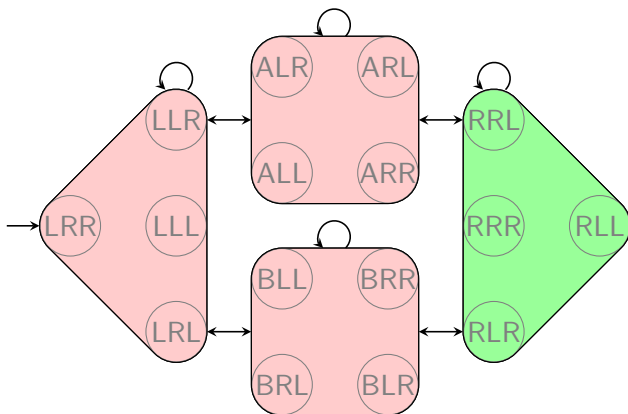
## Example: Concrete State Space



- ▶ state variable *package*: {L, R, A, B}
- ▶ state variable *truck A*: {L, R}
- ▶ state variable *truck B*: {L, R}

# Example: Projection (1)

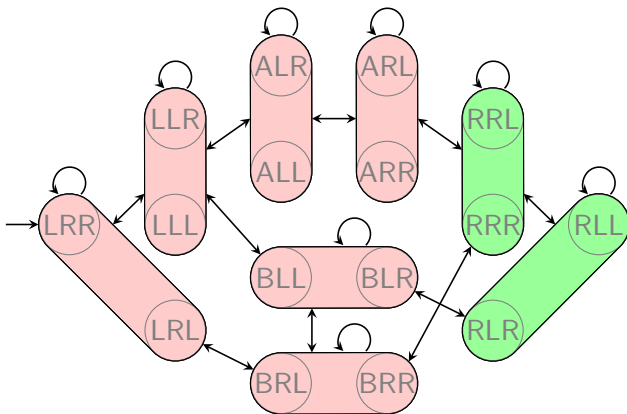
abstraction induced by  $\pi_{\{package\}}$ :



$$h^{\{package\}}(\text{LRR}) = 2$$

## Example: Projection (2)

abstraction induced by  $\pi_{\{package, truck A\}}$ :



$$h^{\{package, truck A\}}(LRR) = 2$$

# Pattern Databases in Practice

practical aspects which we do not discuss in detail:

- ▶ How to automatically find **good patterns**?
- ▶ How to combine **multiple** PDB heuristics?
- ▶ How to **implement** PDB heuristics efficiently?
  - ▶ good implementations efficiently handle **abstract** state spaces with  $10^7$ ,  $10^8$  or more abstract states
  - ▶ effort independent of the size of the **concrete** state space
  - ▶ usually all heuristic values are precomputed
    - ↪ space complexity = number of abstract states

## F6.3 Summary

# Summary

- ▶ basic idea of **abstraction heuristics**: estimate solution cost by considering a **smaller** planning task.
- ▶ formally: **abstraction function**  $\alpha$  maps states to **abstract states** and thus defines which states can be distinguished by the resulting heuristic.
- ▶ induces **abstract state space** whose solution costs are used as heuristic
- ▶ **Pattern database heuristics** are abstraction heuristics based on **projections** onto state variable subsets (**patterns**): states are distinguishable iff they differ on the pattern.