# Foundations of Artificial Intelligence

## 37. Automated Planning: Abstraction

Malte Helmert and Thomas Keller

University of Basel

May 4, 2020

# Automated Planning: Overview

Chapter overview: automated planning

- 33. Introduction
- 34. Planning Formalisms
- 35.–36. Planning Heuristics: Delete Relaxation
- 37. Planning Heuristics: Abstraction
- 38.–39. Planning Heuristics: Landmarks

# Planning Heuristics

We consider three basic ideas for general heuristics:

- Delete Relaxation
- Abstraction $\rightsquigarrow$ this chapter
- Landmarks

# Planning Heuristics

We consider three basic ideas for general heuristics:

- Delete Relaxation
- Abstraction $\rightsquigarrow$ this chapter
- Landmarks

**Abstraction: Idea**

Estimate solution costs by considering a smaller planning task.

# $SAS^+$

# $SAS^+$ Encoding

- in this chapter: $SAS^+$ encoding instead of STRIPS (see Chapter 34)
- difference: state variables $v$ not binary, but with finite domain $\mathrm{dom}(v)$
- accordingly, preconditions, effects, goals specified as partial assignments
- everything else equal to STRIPS

(In practice, planning systems convert automatically between STRIPS and $SAS^+$.)

# SAS$^+$ Planning Task

**Definition (SAS$^+$ planning task)**

A SAS$^+$ planning task is a 5-tuple $\Pi = \langle V, \text{dom}, I, G, A \rangle$ with the following components:

- $V$: finite set of state variables
- dom: domain; $\text{dom}(v)$ finite and non-empty for all $v \in V$
  - states: total assignments for $V$ according to dom
- $I$: the initial state (state = total assignment)
- $G$: goals (partial assignment)
- $A$: finite set of actions $a$ with
  - $pre(a)$: its preconditions (partial assignment)
  - $eff(a)$: its effects (partial assignment)
  - $cost(a) \in \mathbb{N}_0$: its cost

German: SAS$^+$-Planungsaufgabe

# State Space of SAS$^+$ Planning Task

## Definition (state space induced by SAS$^+$ planning task)

Let $\Pi = \langle V, \mathrm{dom}, I, G, A \rangle$ be a SAS$^+$ planning task.
Then $\Pi$ induces the state space $\mathcal{S}(\Pi) = \langle S, A, cost, T, s_0, S_\star \rangle$:

- set of states: total assignments of $V$ according to dom
- actions: actions $A$ defined as in $\Pi$
- action costs: $cost$ as defined in $\Pi$
- transitions: $s \xrightarrow{a} s'$ for states $s, s'$ and action $a$ iff
  - $pre(a)$ complies with $s$ (precondition satisfied)
  - $s'$ complies with $eff(a)$ for all variables mentioned in $eff$; complies with $s$ for all other variables (effects are applied)
- initial state: $s_0 = I$
- goal states: $s \in S_\star$ for state $s$ iff $G$ complies with $s$

German: durch SAS$^+$-Planungsaufgabe induzierter Zustandsraum

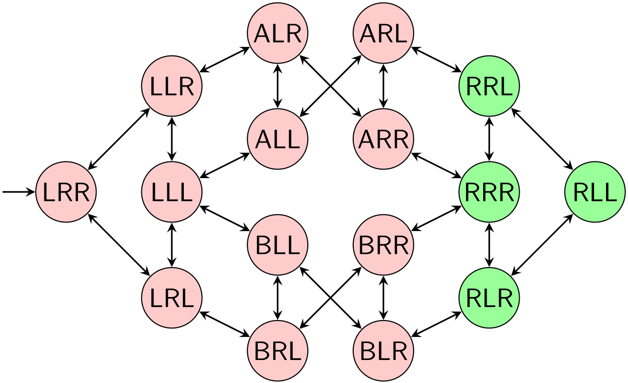
# Example: Logistics Task with One Package, Two Trucks

## Example (one package, two trucks)

Consider the SAS$^+$ planning task $\langle V, \mathrm{dom}, I, G, A\rangle$ with:

- $V = \{p, t_\mathrm{A}, t_\mathrm{B}\}$
- $\mathrm{dom}(p) = \{\mathrm{L}, \mathrm{R}, \mathrm{A}, \mathrm{B}\}$ and $\mathrm{dom}(t_\mathrm{A}) = \mathrm{dom}(t_\mathrm{B}) = \{\mathrm{L}, \mathrm{R}\}$
- $I = \{p \mapsto \mathrm{L}, t_\mathrm{A} \mapsto \mathrm{R}, t_\mathrm{B} \mapsto \mathrm{R}\}$ and $G = \{p \mapsto \mathrm{R}\}$
- $A = \{load_{i,j} \mid i \in \{\mathrm{A}, \mathrm{B}\}, j \in \{\mathrm{L}, \mathrm{R}\}\}$
  $\cup \{unload_{i,j} \mid i \in \{\mathrm{A}, \mathrm{B}\}, j \in \{\mathrm{L}, \mathrm{R}\}\}$
  $\cup \{move_{i,j,j'} \mid i \in \{\mathrm{A}, \mathrm{B}\}, j, j' \in \{\mathrm{L}, \mathrm{R}\}, j \neq j'\}$ with:
  - $load_{i,j}$ has preconditions $\{t_i \mapsto j, p \mapsto j\}$, effects $\{p \mapsto i\}$
  - $unload_{i,j}$ has preconditions $\{t_i \mapsto j, p \mapsto i\}$, effects $\{p \mapsto j\}$
  - $move_{i,j,j'}$ has preconditions $\{t_i \mapsto j\}$, effects $\{t_i \mapsto j'\}$
  - All actions have cost 1.

# State Space for Example Task




- state $\{p \mapsto i, t_A \mapsto j, t_B \mapsto k\}$ denoted as $ijk$
- annotations of edges not shown for simplicity
- for example, edge from LLL to ALL has annotation $load_{A,L}$

# Abstractions

# State Space Abstraction

State space abstractions drop distinctions between certain states, but preserve the state space behavior as well as possible.

- An abstraction of a state space $\mathcal{S}$ is defined by an abstraction function $\alpha$ that determines which states can be distinguished in the abstraction.
- Based on $\mathcal{S}$ and $\alpha$, we compute the abstract state space $\mathcal{S}^\alpha$ which is "similar" to $\mathcal{S}$ but smaller.

German: Abstraktionsfunktion, abstrakter Zustandsraum

## Abstraction Heuristic

Use abstract solution costs (solution costs in $\mathcal{S}^\alpha$) as heuristic values for concrete solution costs (solution costs in $\mathcal{S}$).
$\rightsquigarrow$ abstraction heuristic $h^\alpha$

German: abstrakte/konkrete Zielabstände, Abstraktionsheuristik

# Induced Abstraction

## Definition (induced abstraction)

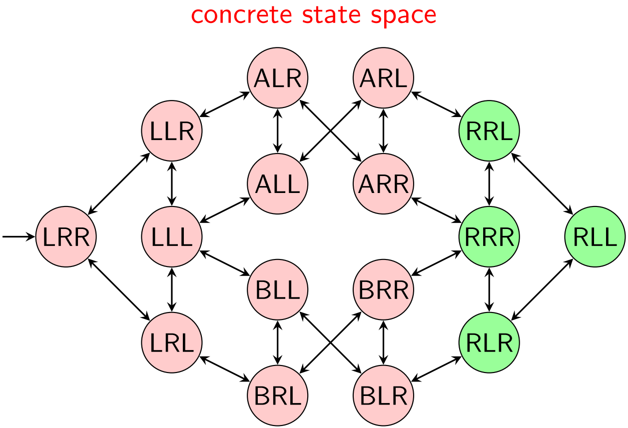
Let $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$ be a state space, and let $\alpha : S \to S'$ be a surjective function.

The abstraction of $\mathcal{S}$ induced by $\alpha$, denoted as $\mathcal{S}^\alpha$, is the state space $\mathcal{S}^\alpha = \langle S', A, cost, T', s'_0, S'_\star \rangle$ with:

- $T' = \{\langle \alpha(s), a, \alpha(t) \rangle \mid \langle s, a, t \rangle \in T\}$
- $s'_0 = \alpha(s_0)$
- $S'_\star = \{\alpha(s) \mid s \in S_\star\}$

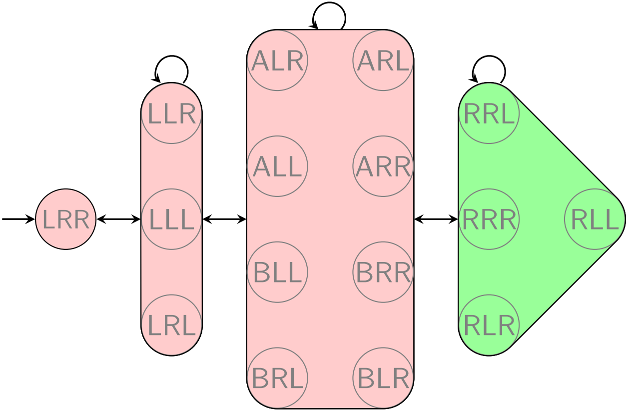
German: induzierte Abstraktion

# Abstraction: Example



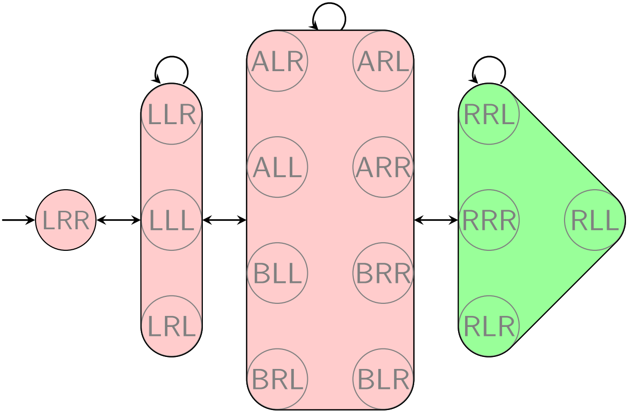
# Abstraction: Example

(an) abstract state space




remark: Most edges correspond to several (parallel) transitions with different annotations.

# Abstraction Heuristic: Example

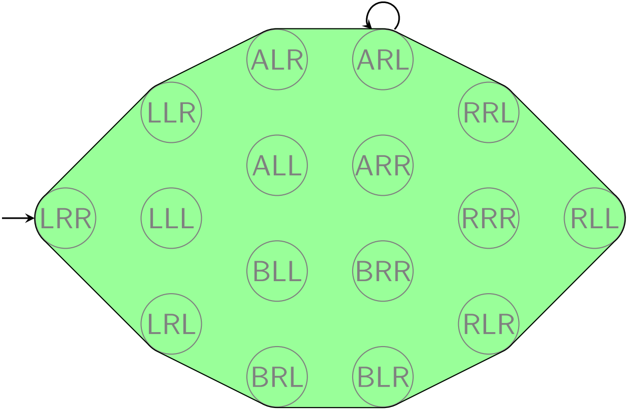



$$h^{\alpha}(\{p \mapsto \mathsf{L}, t_{\mathsf{A}} \mapsto \mathsf{R}, t_{\mathsf{B}} \mapsto \mathsf{R}\}) = 3$$

# 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 $\alpha$?

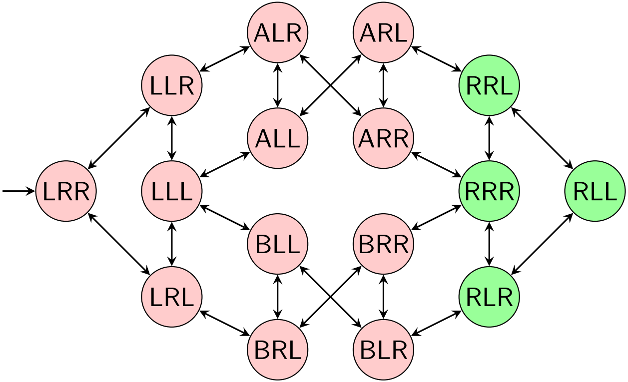
# Usually a Bad Idea: Single-State Abstraction




one state abstraction: $\alpha(s) :=$ const

- $+$ compactly representable and $\alpha$ easy to compute
- $-$ very uninformed heuristic

# Usually a Bad Idea: Identity Abstraction




identity abstraction: $\alpha(s) := s$

- $+$ perfect heuristic and $\alpha$ easy to compute
- $-$ too many abstract states $\rightsquigarrow$ computation of $h^\alpha$ too hard

# 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)

German: Musterdatenbanken, Merge-and-Shrink-Abstraktionen, Kartesische Abstraktionen

# 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; example:

- $s = \{v_1 \mapsto d_1, v_2 \mapsto d_2, v_3 \mapsto d_3\}$
- projection on $P = \{v_1\}$ (= ignore $v_2, v_3$):
  $\alpha(s) = s|_P = \{v_1 \mapsto d_1\}$
- projection on $P = \{v_1, v_3\}$ (= ignore $v_2$):
  $\alpha(s) = s|_P = \{v_1 \mapsto d_1, v_3 \mapsto d_3\}$

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 $P$.

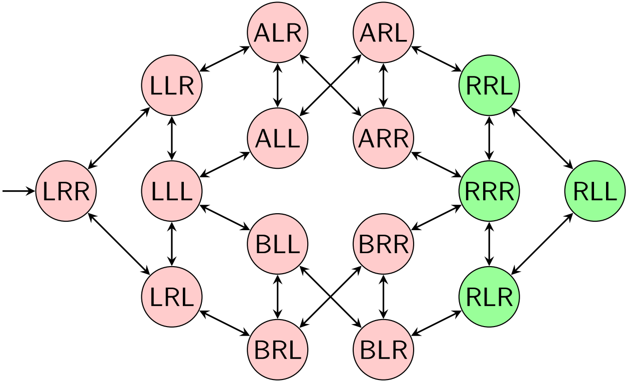
abbreviated notation: $h^P$ for $h^{\pi_P}$

German: Musterdatenbank-Heuristik

remark:

- "pattern databases" in analogy to endgame databases (which have been successfully applied in 2-person-games)

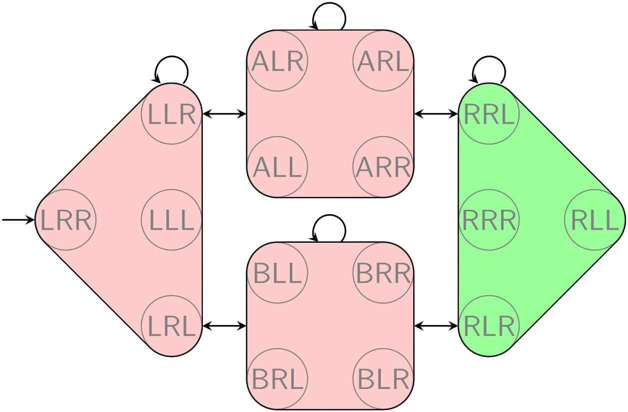
# Example: Concrete State Space




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

# Example: Projection (1)

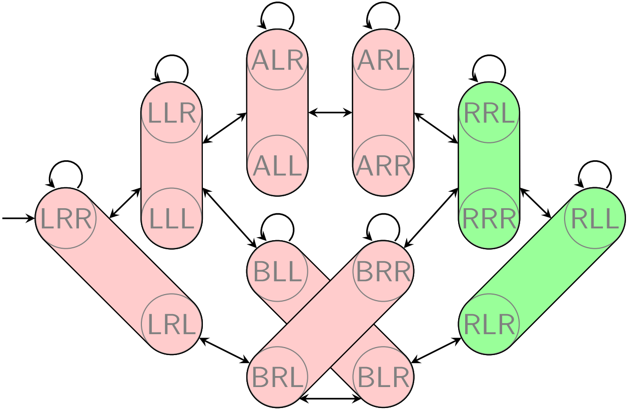
abstraction induced by $\pi_{\{package\}}$:




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

# Example: Projection (2)

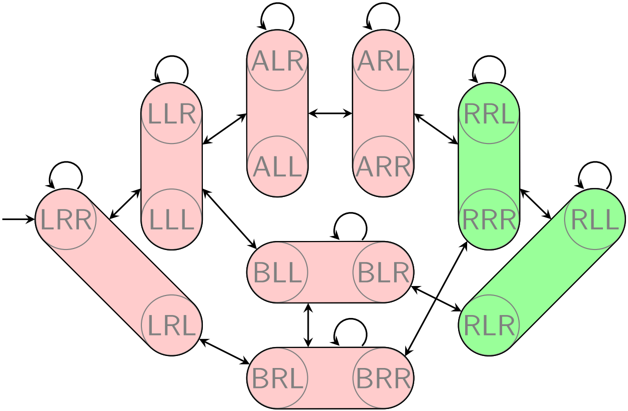
abstraction induced by $\pi_{\{package, truck\ A\}}$:




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

# Example: Projection (2)

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




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

# 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
    $\rightsquigarrow$ space complexity = number of abstract states

# 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.