

Foundations of Artificial Intelligence

37. Automated Planning: Abstraction

Malte Helmert

University of Basel

May 6, 2019

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 and the next chapter: SAS⁺ encoding instead of STRIPS (see Chapter 34)
- difference: state variables v not binary, but with **finite domain** $\text{dom}(v)$
- 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
 - $\text{pre}(a)$: its **preconditions** (partial assignment)
 - $\text{eff}(a)$: its **effects** (partial assignment)
 - $\text{cost}(a) \in \mathbb{N}_0$: its **cost**

German: SAS⁺-Planungsaufgabe

State Space of SAS⁺ Planning Task

Definition (state space induced by SAS⁺ planning task)

Let $\Pi = \langle V, \text{dom}, I, G, A \rangle$ be a SAS⁺ planning task.

Then Π **induces** the **state space** $\mathcal{S}(\Pi) = \langle S, A, \text{cost}, T, s_0, S_\star \rangle$:

- **set of states**: total assignments of V according to dom
- **actions**: actions A defined as in Π
- **action costs**: cost as defined in Π
- **transitions**: $s \xrightarrow{a} s'$ for states s, s' and action a iff
 - $\text{pre}(a)$ complies with s (precondition satisfied)
 - s' complies with $\text{eff}(a)$ 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

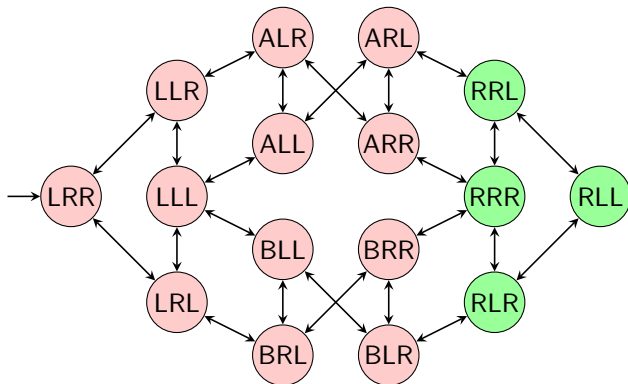
Example (one package, two trucks)

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

- $V = \{p, t_A, t_B\}$
- $\text{dom}(p) = \{L, R, A, B\}$ and $\text{dom}(t_A) = \text{dom}(t_B) = \{L, R\}$
- $I = \{p \mapsto L, t_A \mapsto R, t_B \mapsto R\}$ and $G = \{p \mapsto R\}$
- $A = \{ \text{pickup}_{i,j} \mid i \in \{A, B\}, j \in \{L, R\} \}$
 $\cup \{ \text{drop}_{i,j} \mid i \in \{A, B\}, j \in \{L, R\} \}$
 $\cup \{ \text{move}_{i,j,j'} \mid i \in \{A, B\}, j, j' \in \{L, R\}, j \neq j' \}$ with:
 - $\text{pickup}_{i,j}$ has preconditions $\{t_i \mapsto j, p \mapsto j\}$, effects $\{p \mapsto i\}$
 - $\text{drop}_{i,j}$ has preconditions $\{t_i \mapsto j, p \mapsto i\}$, effects $\{p \mapsto j\}$
 - $\text{move}_{i,j,j'}$ has preconditions $\{t_i \mapsto j\}$, effects $\{t_i \mapsto j'\}$
 - All actions have cost 1.

pickup corresponds to **load**, and **drop** to **unload** from Chapter 35 (renamed to avoid confusion in the following abbreviations)

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 $pickup_{A,L}$

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** α that determines which states can be distinguished in the abstraction.
- Based on \mathcal{S} and α , we compute the **abstract state space** \mathcal{S}^α which is “similar” to \mathcal{S} but smaller.

German: Abstraktionsfunktion, abstrakter Zustandsraum

Abstraction Heuristic

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

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 $\alpha : S \rightarrow S'$ be a surjective function.

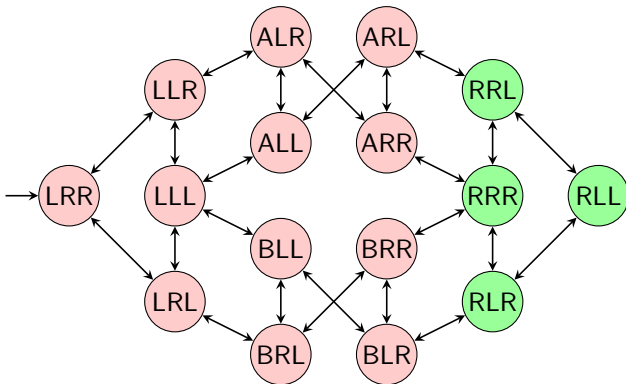
The **abstraction of \mathcal{S} induced by α** , denoted as \mathcal{S}^α , is the state space $\mathcal{S}^\alpha = \langle S', A, cost, T', s'_0, S'_\star \rangle$ with:

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

German: induzierte Abstraktion

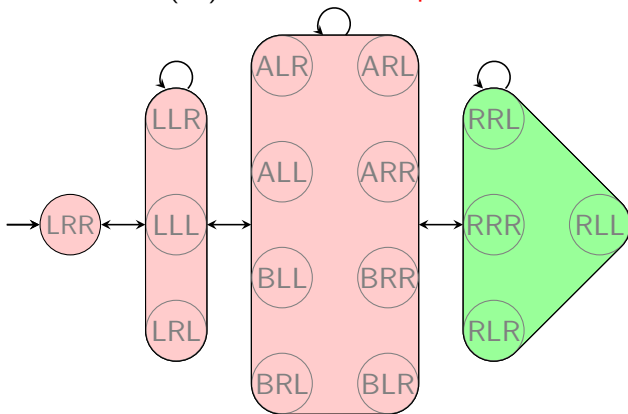
Abstraction: Example

concrete state space



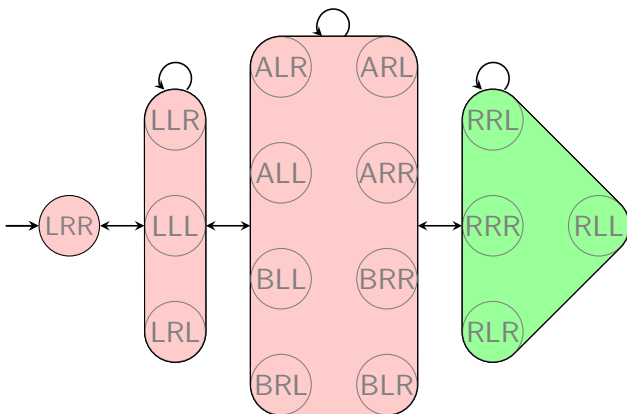
Abstraction: Example

(an) **abstract state space**



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

Abstraction Heuristic: Example

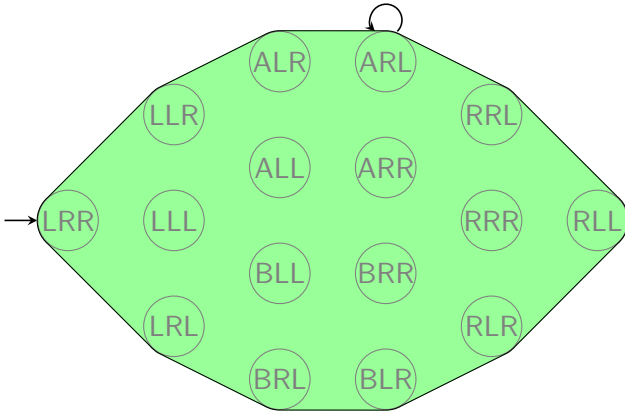


$$h^\alpha(\{p \mapsto L, t_A \mapsto R, t_B \mapsto R\}) = 3$$

Abstraction Heuristics: Discussion

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

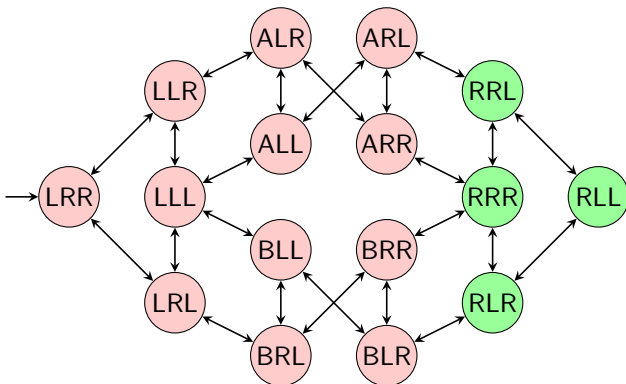
Usually a Bad Idea: Single-State Abstraction



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

- + compactly representable and α easy to compute
- very uninformed heuristic

Usually a Bad Idea: Identity Abstraction



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

+ perfect heuristic and α easy to compute

- too many abstract states \rightsquigarrow computation of h^α 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** π_P on P is called **pattern database heuristic** (**PDB heuristic**) with **pattern** P .

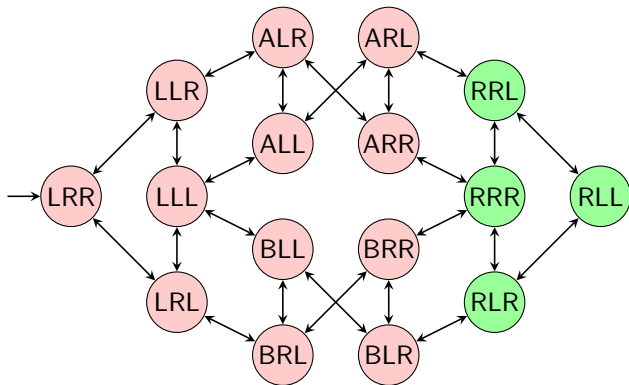
abbreviated notation: h^P for h^{π_P}

German: Musterdatenbank-Heuristik

remark:

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

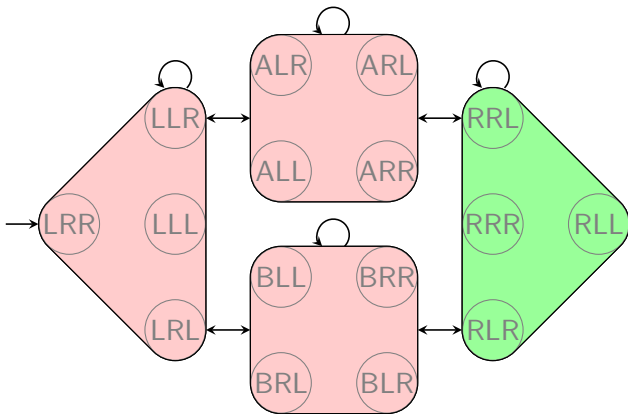
Example: Concrete State Space



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

Example: Projection (1)

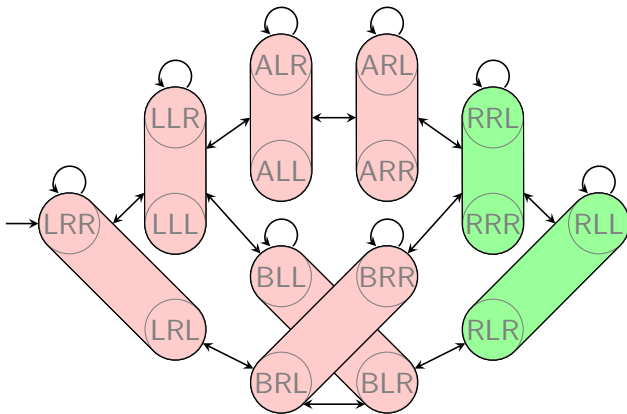
abstraction induced by $\pi_{\{package\}}$:



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

Example: Projection (2)

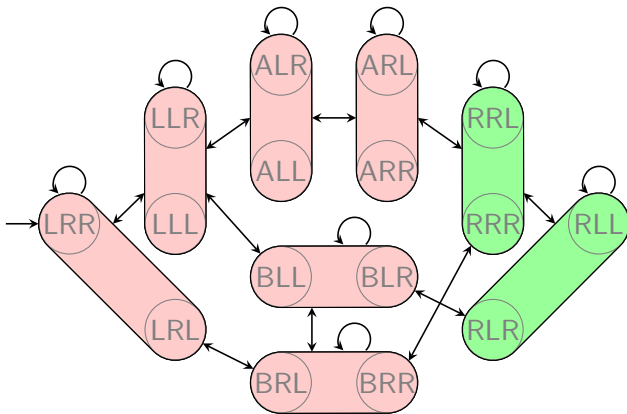
abstraction induced by $\pi_{\{package, truck A\}}$:



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

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

Summary

Summary

- basic idea of **abstraction heuristics**: estimate solution cost by considering a **smaller** planning task.
- formally: **abstraction function** α 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.