

Foundations of Artificial Intelligence

37. Automated Planning: Abstraction and Pattern Databases

Martin Wehrle

Universität Basel

May 13, 2016

Foundations of Artificial Intelligence

May 13, 2016 — 37. Automated Planning: Abstraction and Pattern Databases

37.1 SAS⁺

37.2 Abstractions

37.3 Pattern Databases

37.4 Summary

Planning Heuristics

We consider **three basic ideas** for general heuristics:

- ▶ Delete Relaxation
- ▶ **Abstraction** \rightsquigarrow this and next chapter
- ▶ Landmarks

Abstraction: Idea

Estimate solution costs by considering a **smaller** planning task.

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

37.1 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** $\mathcal{S}(\Pi) = \langle S, A, \text{cost}, T, s_0, S_* \rangle$:

- ▶ **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**: $s \in S_*$ for state s iff G complies with s

German: durch SAS⁺-Planungsaufgabe induzierter Zustandsraum

Example: Logistics Task with One Package, Two Trucks

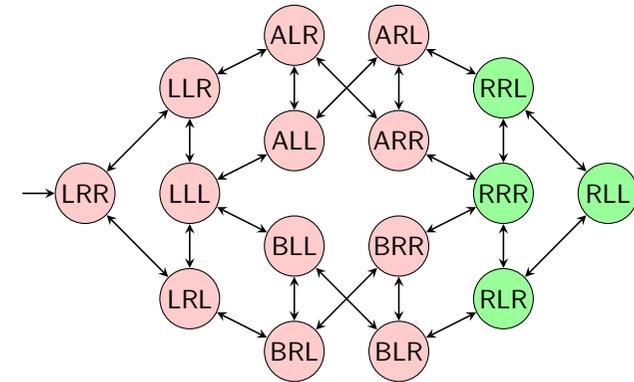
Example (one package, two trucks)

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 $G = \{p \mapsto R\}$
- ▶ $A = \{pickup_{i,j} \mid i \in \{A, B\}, j \in \{L, R\}\} \cup \{drop_{i,j} \mid i \in \{A, B\}, j \in \{L, R\}\} \cup \{move_{i,j,j'} \mid i \in \{A, B\}, j, j' \in \{L, R\}, j \neq j'\}$ with:
 - ▶ $pickup_{i,j}$ has preconditions $\{t_i \mapsto j, p \mapsto j\}$, effects $\{p \mapsto i\}$
 - ▶ $drop_{i,j}$ 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.

$pickup$ 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 $pickup_{A,L}$

37.2 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 goal distances** (goal distances in \mathcal{S}^α) as heuristic values for **concrete goal distances** (goal distances in \mathcal{S}).

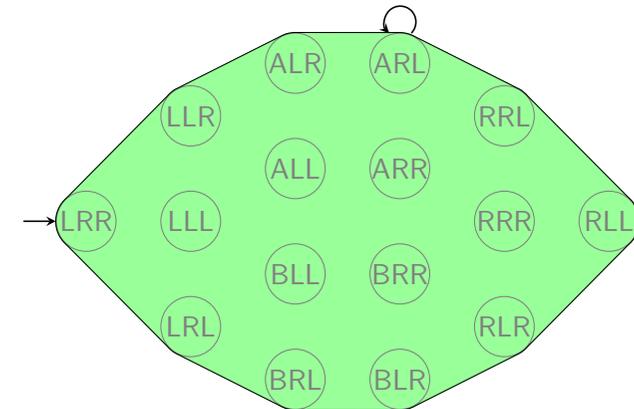
\rightsquigarrow **abstraction heuristic** h^α

German: abstrakte/konkrete Zielabstände, Abstraktionsheuristik

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 ?**

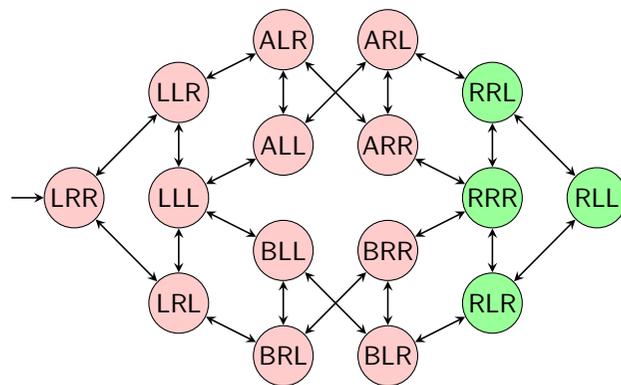
Usually a Bad Idea: Single-State Abstraction



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?

We introduce two successful methods:

- ▶ **pattern databases (PDBs)** (Culberson & Schaeffer, 1996)
- ▶ **merge-and-shrink abstractions** (Dräger, Finkbeiner & Podelski, 2006)

German: Musterdatenbanken, Merge-and-Shrink-Abstraktionen

37.3 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)

37.4 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 if they differ on the pattern.