Principles of AI Planning

10. Planning as search: abstractions

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Bernhard Nebel and Robert Mattmüller

November 27th, 2017

Abstractions: informally

Coming up with heuristics in a principled way

General procedure for obtaining a heuristic

Solve an easier version of the problem.

Two common methods:

- relaxation: consider less constrained version of the problem
- abstraction: consider smaller version of real problem

In previous chapters, we have studied relaxation, which has been very successfully applied to satisficing planning.

Now, we study abstraction, which is one of the most prominent techniques for optimal planning.

Abstracting a transition system

Abstracting a transition system means dropping some distinctions between states, while preserving the transition behaviour as much as possible.

- An abstraction of a transition system $\mathcal{T}$ is defined by an abstraction mapping α that defines which states of $\mathcal{T}$ should be distinguished and which ones should not.
- From $\mathcal{T}$ and α, we compute an abstract transition system $\mathcal{T}'$ which is similar to $\mathcal{T}$, but smaller.
- The abstract goal distances (goal distances in $\mathcal{T}'$) are used as heuristic estimates for goal distances in $\mathcal{T}$.

Abstracting a transition system: example

Example (15-puzzle)

(from Wikimedia Commons, Attribution: Micha L. Rieser)

Abstractions: informally
Introduction
Practical requirements
Multiple abstractions
Outlook
Abstractions: formally
Summary

Abstracting a transition system: example

Example (15-puzzle)

A 15-puzzle state is given by a permutation $\langle b, t_1, \dots, t_{15} \rangle$ of $\{1, \dots, 16\}$, where b denotes the blank position and the other components denote the positions of the 15 tiles.

One possible abstraction mapping ignores the precise location of tiles 8–15, i. e., two states are distinguished iff they differ in the position of the blank or one of the tiles 1–7:

$$\alpha(\langle b, t_1, \dots, t_{15} \rangle) = \langle b, t_1, \dots, t_7 \rangle$$

The heuristic values for this abstraction correspond to the cost of moving tiles 1–7 to their goal positions.

Abstractions: informally
Introduction
Practical requirements
Multiple abstractions
Outlook
Abstractions: formally
Summary

Abstraction example: 15-puzzle

9	2	12	6
5	7	14	13
3	4	1	11
15	10	8	

$\longrightarrow$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

real state space

- $16! = 20922789888000 \approx 2 \cdot 10^{13}$ states
- $\frac{16!}{2} = 10461394944000 \approx 10^{13}$ reachable states

Abstractions: informally
Introduction
Practical requirements
Multiple abstractions
Outlook
Abstractions: formally
Summary

Abstraction example: 15-puzzle

	2		6
5	7		
3	4	1	

$\longrightarrow$

1	2	3	4
5	6	7	

abstract state space

- $16 \cdot 15 \cdot \ldots \cdot 9 = 518918400 \approx 5 \cdot 10^8$ states
- $16 \cdot 15 \cdot \ldots \cdot 9 = 518918400 \approx 5 \cdot 10^8$ reachable states

Computing the abstract transition system

Given $\mathcal{T}$ and α, how do we compute $\mathcal{T}'$?

Requirement

We want to obtain an admissible heuristic.
Hence, $h^*(\alpha(s))$ (in the abstract state space $\mathcal{T}'$) should never overestimate $h^*(s)$ (in the concrete state space $\mathcal{T}$).

An easy way to achieve this is to ensure that all solutions in $\mathcal{T}$ also exist in $\mathcal{T}'$:

- If s is a goal state in $\mathcal{T}$, then $\alpha(s)$ is a goal state in $\mathcal{T}'$.
- If $\mathcal{T}$ has a transition from s to t, then $\mathcal{T}'$ has a transition from $\alpha(s)$ to $\alpha(t)$.

Computing the abstract transition system: example

Example (15-puzzle)

In the running example:

- $\mathcal{T}$ has the unique goal state $\langle 16,1,2,\dots,15\rangle$.
  - $\rightsquigarrow$ $\mathcal{T}'$ has the unique goal state $\langle 16,1,2,\dots,7\rangle$.
- Let x and y be neighboring positions in the 4×4 grid. $\mathcal{T}$ has a transition from $\langle x,t_1,\dots,t_{i-1},y,t_{i+1},\dots,t_{15}\rangle$ to $\langle y,t_1,\dots,t_{i-1},x,t_{i+1},\dots,t_{15}\rangle$ for all $i \in \{1,\dots,15\}$.
  - $\rightsquigarrow$ $\mathcal{T}'$ has a transition from $\langle x,t_1,\dots,t_{i-1},y,t_{i+1},\dots,t_7\rangle$ to $\langle y,t_1,\dots,t_{i-1},x,t_{i+1},\dots,t_7\rangle$ for all $i \in \{1,\dots,7\}$.
  - $\rightsquigarrow$ Moreover, $\mathcal{T}'$ has a transition from $\langle x,t_1,\dots,t_7\rangle$ to $\langle y,t_1,\dots,t_7\rangle$ if $y \notin \{t_1,\dots,t_7\}$.

Practical requirements for abstractions

To be useful in practice, an abstraction heuristic must be efficiently computable. This gives us two requirements for α:

- For a given state s, the abstract state $\alpha(s)$ must be efficiently computable.
- For a given abstract state $\alpha(s)$, the abstract goal distance $h^*(\alpha(s))$ must be efficiently computable.

There are different ways of achieving these requirements:

- pattern database heuristics (Culberson & Schaeffer, 1996)
- merge-and-shrink abstractions (Dräger, Finkbeiner & Podelski, 2006)
- structural patterns (Katz & Domshlak, 2008)
- Cartesian abstractions (Ball, Podelski & Rajamani, 2001; Seipp & Helmert, 2013)

Practical requirements for abstractions: example

Example (15-puzzle)

In our running example, α can be very efficiently computed: just project the given 16-tuple to its first 8 components.

To compute abstract goal distances efficiently during search, most common algorithms precompute all abstract goal distances prior to search by performing a backward breadth-first search from the goal state(s). The distances are then stored in a table (requires about 495 MB of RAM). During search, computing $h^*(\alpha(s))$ is just a table lookup.

This heuristic is an example of a pattern database heuristic.

Multiple abstractions

- One important practical question is how to come up with a suitable abstraction mapping α.
- Indeed, there is usually a huge number of possibilities, and it is important to pick good abstractions (i. e., ones that lead to informative heuristics).
- However, it is generally not necessary to commit to a single abstraction.

Combining multiple abstractions

Maximizing several abstractions:

- Each abstraction mapping gives rise to an admissible heuristic.
- By computing the maximum of several admissible heuristics, we obtain another admissible heuristic which dominates the component heuristics.
- Thus, we can always compute several abstractions and maximize over the individual abstract goal distances.

Adding several abstractions:

- In some cases, we can even compute the sum of individual estimates and still stay admissible.
- Summation often leads to much higher estimates than maximization, so it is important to understand when it is admissible.

Maximizing several abstractions: example

Example (15-puzzle)

- mapping to tiles 1–7 was arbitrary
  $\rightsquigarrow$ can use any subset of tiles
- with the same amount of memory required for the tables for the mapping to tiles 1–7, we could store the tables for nine different abstractions to six tiles and the blank
- use maximum of individual estimates

Adding several abstractions: example

9	2	12	6
5	7	14	13
3	4	1	11
15	10	8	

9	2	12	6
5	7	14	13
3	4	1	11
15	10	8	

- 1st abstraction: ignore precise location of 8–15
- 2nd abstraction: ignore precise location of 1–7

$\rightsquigarrow$ Is the sum of the abstraction heuristics admissible?

Adding several abstractions: example

	2		6
5	7		
3	4	1	

9		12	
		14	13
			11
15	10	8	

- 1st abstraction: ignore precise location of 8–15
- 2nd abstraction: ignore precise location of 1–7

⇝ The sum of the abstraction heuristics is not admissible.

Adding several abstractions: example

	2		6
5	7		
3	4	1	

9		12	
		14	13
			11
15	10	8	

- 1st abstraction: ignore precise location of 8–15 and blank
- 2nd abstraction: ignore precise location of 1–7 and blank

⇝ The sum of the abstraction heuristics is admissible.

Our plan for the next lectures

In the following, we take a deeper look at abstractions and their use for admissible heuristics.

- In the rest of this chapter, we formally introduce abstractions and abstraction heuristics and study some of their most important properties.
- In the following chapter, we discuss one particular class of abstraction heuristics in detail, namely pattern database heuristics.

Abstractions: formally

Transition systems

Reminder from Chapter 2:

Definition (transition system)

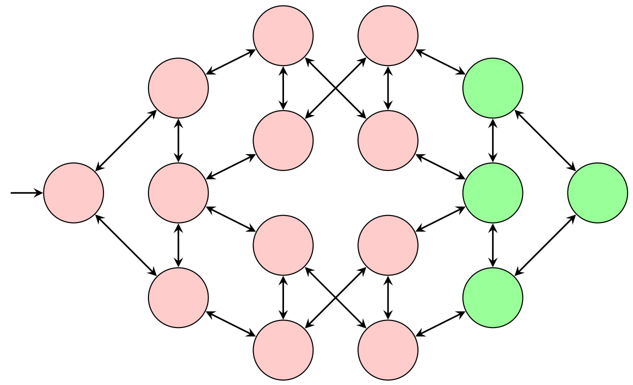
A transition system is a 5-tuple $\mathcal{T} = \langle S, L, T, s_0, S_\star \rangle$ where

- S is a finite set of states,
- L is a finite set of (transition) labels,
- $T \subseteq S \times L \times S$ is the transition relation,
- $s_0 \in S$ is the initial state, and
- $S_\star \subseteq S$ is the set of goal states.

We say that $\mathcal{T}$ has the transition $\langle s, \ell, s' \rangle$ if $\langle s, \ell, s' \rangle \in T$.

We also write this $s \xrightarrow{\ell} s'$, or $s \to s'$ when not interested in ℓ.

Transition systems: example



Note: To reduce clutter, our figures usually omit arc labels and collapse transitions between identical states. However, these are important for the formal definition of the transition system.

Transition systems of FDR planning tasks

Definition (induced transition system of an FDR planning task)

Let $\Pi = \langle V, I, O, \gamma \rangle$ be an FDR planning task.
The induced transition system of Π, in symbols $\mathcal{T}(\Pi)$, is the transition system $\mathcal{T}(\Pi) = \langle S, L, T, s_0, S_\star \rangle$, where

- S is the set of states over V,
- $L = O$,
- $T = \{\langle s, o, t \rangle \in S \times L \times S \mid app_o(s) = t\}$,
- $s_0 = I$, and
- $S_\star = \{s \in S \mid s \models \gamma\}$.

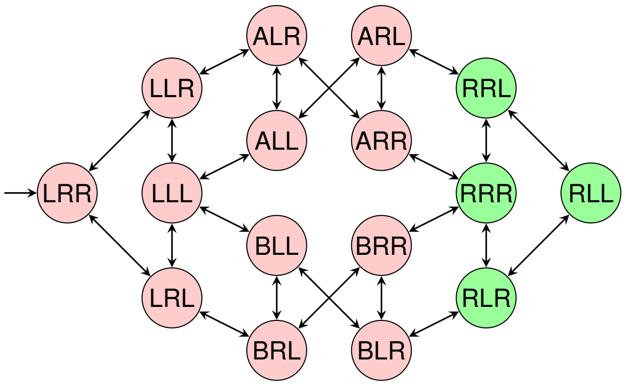
Example task: one package, two trucks

Example (one package, two trucks)

Consider the following FDR planning task $\langle V, I, O, \gamma \rangle$:

- $V = \{p, t_A, t_B\}$ with
  - $\mathcal{D}_p = \{\mathrm{L}, \mathrm{R}, \mathrm{A}, \mathrm{B}\}$
  - $\mathcal{D}_{t_A} = \mathcal{D}_{t_B} = \{\mathrm{L}, \mathrm{R}\}$
- $I = \{p \mapsto \mathrm{L}, t_A \mapsto \mathrm{R}, t_B \mapsto \mathrm{R}\}$
- $O = \{\mathrm{pickup}_{i,j} \mid i \in \{\mathrm{A}, \mathrm{B}\}, j \in \{\mathrm{L}, \mathrm{R}\}\}$
  $\cup \{\mathrm{drop}_{i,j} \mid i \in \{\mathrm{A}, \mathrm{B}\}, j \in \{\mathrm{L}, \mathrm{R}\}\}$
  $\cup \{\mathrm{move}_{i,j,j'} \mid i \in \{\mathrm{A}, \mathrm{B}\}, j, j' \in \{\mathrm{L}, \mathrm{R}\}, j \neq j'\}$, where
  - $\mathrm{pickup}_{i,j} = \langle t_i = j \wedge p = j, p := i \rangle$
  - $\mathrm{drop}_{i,j} = \langle t_i = j \wedge p = i, p := j \rangle$
  - $\mathrm{move}_{i,j,j'} = \langle t_i = j, t_i := j' \rangle$
- $\gamma = (p = \mathrm{R})$

Transition system of example task



- State $\{p \mapsto i, t_A \mapsto j, t_B \mapsto k\}$ is depicted as *ijk*.
- Transition labels are again not shown. For example, the transition from LLL to ALL has the label $\text{pickup}_{A,L}$.

Abstractions

Definition (abstraction, abstraction mapping)

Let $\mathcal{T} = \langle S, L, T, s_0, S_\star \rangle$ and $\mathcal{T}' = \langle S', L', T', s'_0, S'_\star \rangle$ be transition systems with the same label set $L = L'$, and let $\alpha : S \to S'$ be a surjective function.

We say that $\mathcal{T}'$ is an abstraction of $\mathcal{T}$ with abstraction mapping α (or: abstraction function α) if

- $\alpha(s_0) = s'_0$,
- for all $s \in S_\star$, we have $\alpha(s) \in S'_\star$, and
- for all $\langle s, \ell, t \rangle \in T$, we have $\langle \alpha(s), \ell, \alpha(t) \rangle \in T'$.

Abstractions: terminology

Let $\mathcal{T}$ and $\mathcal{T}'$ be transition systems and α a function such that $\mathcal{T}'$ is an abstraction of $\mathcal{T}$ with abstraction mapping α.

- $\mathcal{T}$ is called the concrete transition system.
- $\mathcal{T}'$ is called the abstract transition system.
- Similarly: concrete/abstract state space, concrete/abstract transition, etc.

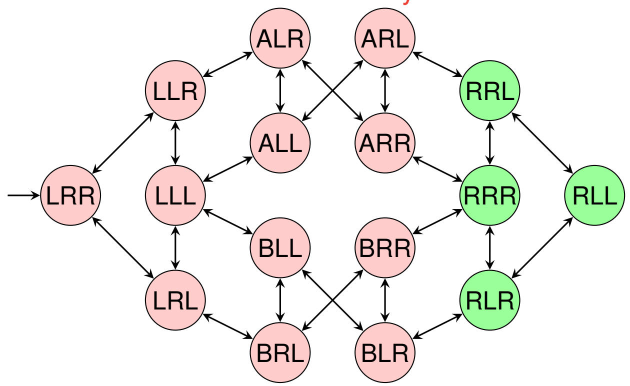
We say that:

- $\mathcal{T}'$ is an abstraction of $\mathcal{T}$ (without mentioning α)
- α is an abstraction mapping on $\mathcal{T}$ (without mentioning $\mathcal{T}'$)

Note: For a given $\mathcal{T}$ and α, there can be multiple abstractions $\mathcal{T}'$, and for a given $\mathcal{T}$ and $\mathcal{T}'$, there can be multiple abstraction mappings α.

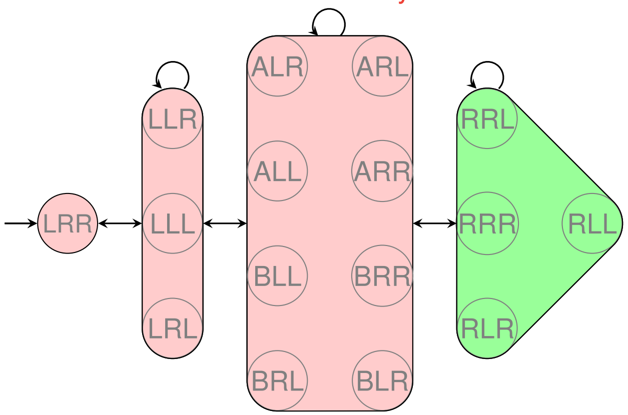
Abstraction: example

concrete transition system



Abstraction: example

abstract transition system



Note: Most arcs represent many parallel transitions.

Induced abstractions

Definition (induced abstractions)

Let $\mathscr{T} = \langle S, L, T, s_0, S_\star \rangle$ be a transition system, and let $\alpha : S \to S'$ be a surjective function.

The abstraction (of $\mathscr{T}$) induced by α, in symbols $\mathscr{T}^\alpha$, is the transition system $\mathscr{T}^\alpha = \langle S', L, T', s'_0, S'_\star \rangle$ defined by:

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

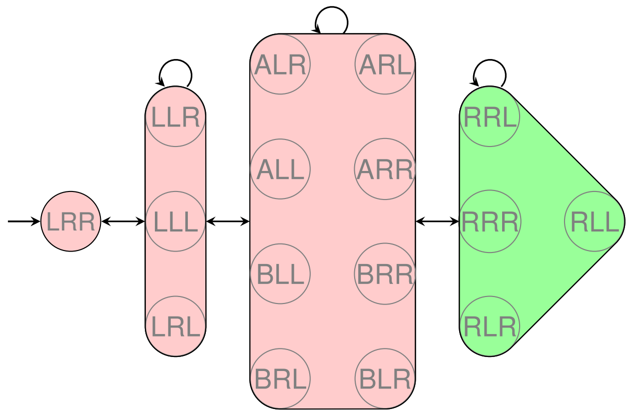
Note: It is easy to see that $\mathscr{T}^\alpha$ is an abstraction of $\mathscr{T}$. It is the "smallest" abstraction of $\mathscr{T}$ with abstraction mapping α.

Induced abstractions: terminology

Let $\mathscr{T}$ and $\mathscr{T}'$ be transition systems and α be a function such that $\mathscr{T}' = \mathscr{T}^\alpha$ (i. e., $\mathscr{T}'$ is the abstraction of $\mathscr{T}$ induced by α).

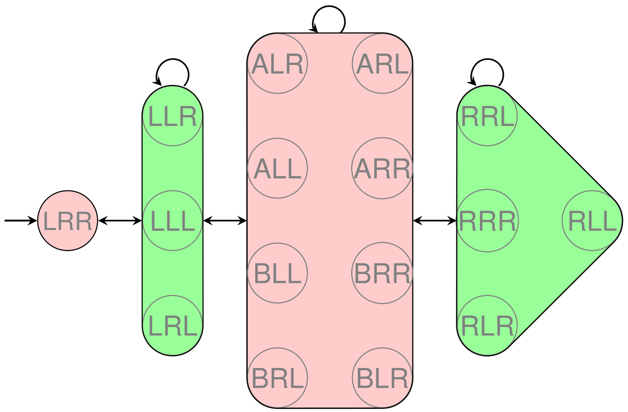
- α is called a strict homomorphism from $\mathscr{T}$ to $\mathscr{T}'$, and $\mathscr{T}'$ is called a strictly homomorphic abstraction of $\mathscr{T}$.
- If α is bijective, it is called an isomorphism between $\mathscr{T}$ and $\mathscr{T}'$, and the two transition systems are called isomorphic.

Strictly homomorphic abstractions: example



This abstraction is a strictly homomorphic abstraction of the concrete transition system $\mathscr{T}$.

Strictly homomorphic abstractions: example



If we add any goal states or transitions, it is still an abstraction of $\mathscr{T}$, but no longer a strictly homomorphic one.

Abstraction heuristics

Definition (abstr. heur. induced by an abstraction)

Let Π be an FDR planning task with state space S, and let $\mathscr{A}$ be an abstraction of $\mathscr{T}(\Pi)$ with abstraction mapping α.

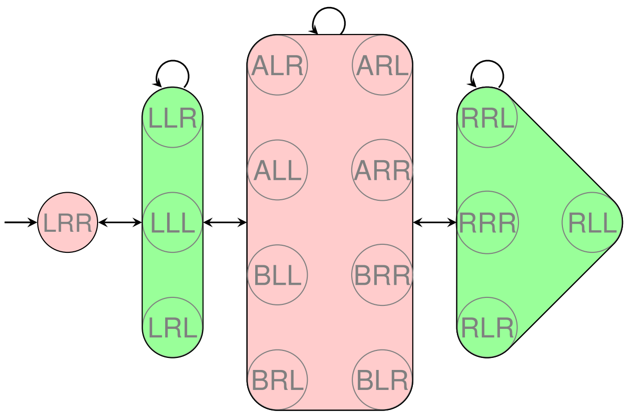
The abstraction heuristic induced by $\mathscr{A}$ and α, $h^{\mathscr{A},\alpha}$, is the heuristic function $h^{\mathscr{A},\alpha} : S \to \mathbb{N}_0 \cup \{\infty\}$ which maps each state $s \in S$ to $h^*_{\mathscr{A}}(\alpha(s))$ (the goal distance of $\alpha(s)$ in $\mathscr{A}$).

Note: $h^{\mathscr{A},\alpha}(s) = \infty$ if no goal state of $\mathscr{A}$ is reachable from $\alpha(s)$

Definition (abstr. heur. induced by strict homomorphism)

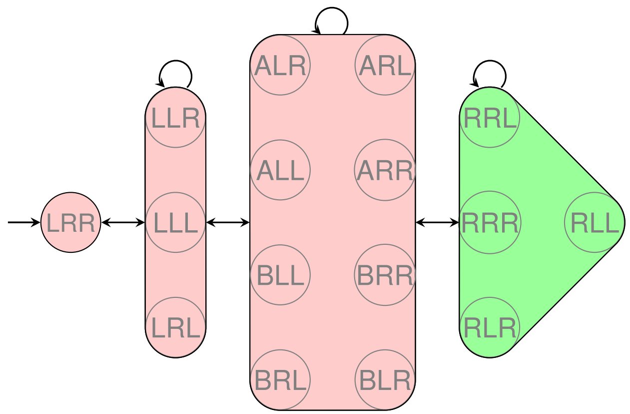
Let Π be an FDR planning task and α a strict homomorphism on $\mathscr{T}(\Pi)$. The abstraction heuristic induced by α, h^{α}, is the abstraction heuristic induced by $\mathscr{T}(\Pi)^{\alpha}$ and α, i. e., $h^{\alpha} := h^{\mathscr{T}(\Pi)^{\alpha},\alpha}$.

Abstraction heuristics: example



$h^{\mathscr{A},\alpha}(\{p \mapsto \mathrm{L}, t_{\mathrm{A}} \mapsto \mathrm{R}, t_{\mathrm{B}} \mapsto \mathrm{R}\}) = 1$

Abstraction heuristics: example



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

Consistency of abstraction heuristics

Theorem (consistency and admissibility of $h^{\mathscr{A},\alpha}$)

Let Π be an FDR planning task, and let $\mathscr{A}$ be an abstraction of $\mathscr{T}(\Pi)$ with abstraction mapping α.
Then $h^{\mathscr{A},\alpha}$ is safe, goal-aware, admissible and consistent.

Proof.

We prove goal-awareness and consistency;
the other properties follow from these two.

Let $\mathscr{T} = \mathscr{T}(\Pi) = \langle S, L, T, s_0, S_\star \rangle$ and $\mathscr{A} = \langle S', L', T', s'_0, S'_\star \rangle$.

Goal-awareness: We need to show that $h^{\mathscr{A},\alpha}(s) = 0$ for all $s \in S_\star$, so let $s \in S_\star$. Then $\alpha(s) \in S'_\star$ by the definition of abstractions and abstraction mappings, and hence $h^{\mathscr{A},\alpha}(s) = h^*_{\mathscr{A}}(\alpha(s)) = 0$.

Consistency of abstraction heuristics (ctd.)

Proof (ctd.)

Consistency: Let $s, t \in S$ such that t is a successor of s. We need to prove that $h^{\mathscr{A},\alpha}(s) \leq h^{\mathscr{A},\alpha}(t) + 1$.
Since t is a successor of s, there exists an operator o with $app_o(s) = t$ and hence $\langle s, o, t \rangle \in T$.
By the definition of abstractions and abstraction mappings, we get $\langle \alpha(s), o, \alpha(t) \rangle \in T' \rightsquigarrow \alpha(t)$ is a successor of $\alpha(s)$ in $\mathscr{A}$.
Therefore, $h^{\mathscr{A},\alpha}(s) = h^*_{\mathscr{A}}(\alpha(s)) \leq h^*_{\mathscr{A}}(\alpha(t)) + 1 = h^{\mathscr{A},\alpha}(t) + 1$,
where the inequality holds because the shortest path from $\alpha(s)$ to the goal in $\mathscr{A}$ cannot be longer than the shortest path from $\alpha(s)$ to the goal via $\alpha(t)$. □

Orthogonality of abstraction mappings

Definition (orthogonal abstraction mappings)

Let α_1 and α_2 be abstraction mappings on $\mathscr{T}$.

We say that α_1 and α_2 are orthogonal if for all transitions $\langle s, \ell, t \rangle$ of $\mathscr{T}$, we have $\alpha_i(s) = \alpha_i(t)$ for at least one $i \in \{1, 2\}$.

Affecting transition labels

Definition (affecting transition labels)

Let $\mathscr{T}$ be a transition system, and let ℓ be one of its labels.
We say that ℓ affects $\mathscr{T}$ if $\mathscr{T}$ has a transition $\langle s, \ell, t \rangle$ with $s \neq t$.

Theorem (affecting labels vs. orthogonality)

Let $\mathscr{A}_1$ be an abstraction of $\mathscr{T}$ with abstraction mapping α_1.
Let $\mathscr{A}_2$ be an abstraction of $\mathscr{T}$ with abstraction mapping α_2.

If no label of $\mathscr{T}$ affects both $\mathscr{A}_1$ and $\mathscr{A}_2$, then α_1 and α_2 are orthogonal.

(Easy proof omitted.)

Orthogonal abstraction mappings: example

	2		6
5	7		
3	4	1	

9		12	
		14	13
			11
15	10	8	

Are the abstraction mappings orthogonal?

Orthogonal abstraction mappings: example

	2		6
5	7		
3	4	1	

9		12	
		14	13
			11
15	10	8	

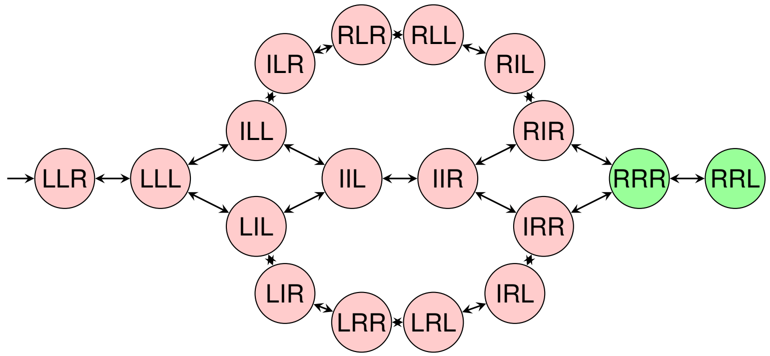
Are the abstraction mappings orthogonal?

Orthogonality and additivity

Theorem (additivity for orthogonal abstraction mappings)

Let $h^{\mathscr{A}_1,\alpha_1},\dots,h^{\mathscr{A}_n,\alpha_n}$ be abstraction heuristics for the same planning task Π such that α_i and α_j are orthogonal for all $i \neq j$. Then $\sum_{i=1}^{n} h^{\mathscr{A}_i,\alpha_i}$ is a safe, goal-aware, admissible and consistent heuristic for Π.

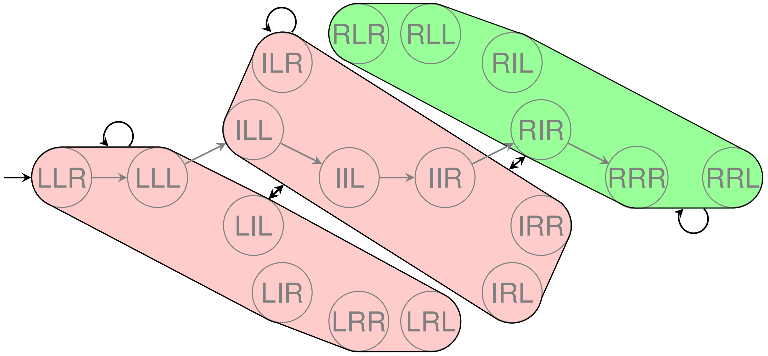
Orthogonality and additivity: example



transition system $\mathscr{T}$

state variables: first package, second package, truck

Orthogonality and additivity: example

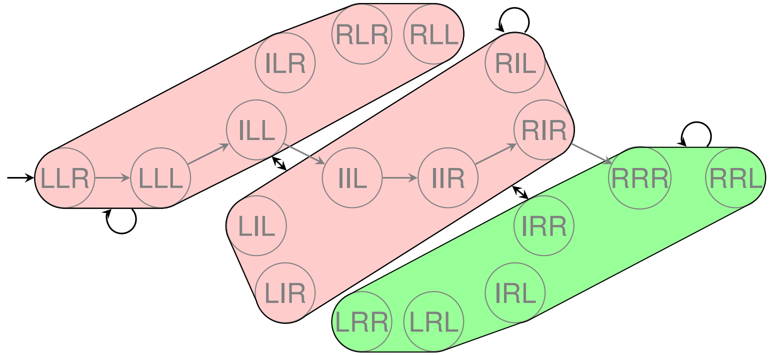


abstraction $\mathscr{A}_1$

mapping: only consider state of first package

Orthogonality and additivity: example

Abstractions: informally
Abstractions: formally
Transition systems
Abstractions
Abstraction heuristics
Additivity
Refinements
Equivalence
Practice
Summary



abstraction $\mathscr{A}_2$ (orthogonal to $\mathscr{A}_1$)

mapping: only consider state of second package

Orthogonality and additivity: proof

Abstractions: informally
Abstractions: formally
Transition systems
Abstractions
Abstraction heuristics
Additivity
Refinements
Equivalence
Practice
Summary

Proof.

We prove goal-awareness and consistency;
the other properties follow from these two.

Let $\mathcal{T} = \mathcal{T}(\Pi) = \langle S, L, T, s_0, S_\star \rangle$.

Goal-awareness: For goal states $s \in S_\star$,
$\sum_{i=1}^n h^{\mathscr{A}_i,\alpha_i}(s) = \sum_{i=1}^n 0 = 0$ because all individual abstractions are goal-aware.

Orthogonality and additivity: proof (ctd.)

Abstractions: informally
Abstractions: formally
Transition systems
Abstractions
Abstraction heuristics
Additivity
Refinements
Equivalence
Practice
Summary

Proof (ctd.)

Consistency: Let $s, t \in S$ such that t is a successor of s.
Let $L := \sum_{i=1}^n h^{\mathscr{A}_i,\alpha_i}(s)$ and $R := \sum_{i=1}^n h^{\mathscr{A}_i,\alpha_i}(t)$.
We need to prove that $L \leq R + 1$.

Since t is a successor of s, there exists an operator o with $app_o(s) = t$ and hence $\langle s, o, t \rangle \in T$.
Because the abstraction mappings are orthogonal, $\alpha_i(s) \neq \alpha_i(t)$ for at most one $i \in \{1, \dots, n\}$.

Case 1: $\alpha_i(s) = \alpha_i(t)$ for all $i \in \{1, \dots, n\}$.

Then
$$\begin{aligned} L &= \sum_{i=1}^n h^{\mathscr{A}_i,\alpha_i}(s) \\ &= \sum_{i=1}^n h^*_{\mathscr{A}_i}(\alpha_i(s)) \\ &= \sum_{i=1}^n h^*_{\mathscr{A}_i}(\alpha_i(t)) \\ &= \sum_{i=1}^n h^{\mathscr{A}_i,\alpha_i}(t) \\ &= R \leq R + 1. \end{aligned}$$

Orthogonality and additivity: proof (ctd.)

Proof (ctd.)

Case 2: $\alpha_i(s) \neq \alpha_i(t)$ for exactly one $i \in \{1, \ldots, n\}$.

Let $k \in \{1, \ldots, n\}$ such that $\alpha_k(s) \neq \alpha_k(t)$.

Then
$$\begin{aligned} L &= \textstyle\sum_{i=1}^{n} h^{\mathscr{A}_i,\alpha_i}(s) \\ &= \textstyle\sum_{i\in\{1,\ldots,n\}\setminus\{k\}} h^*_{\mathscr{A}_i}(\alpha_i(s)) + h^{\mathscr{A}_k,\alpha_k}(s) \\ &\leq \textstyle\sum_{i\in\{1,\ldots,n\}\setminus\{k\}} h^*_{\mathscr{A}_i}(\alpha_i(t)) + h^{\mathscr{A}_k,\alpha_k}(t) + 1 \\ &= \textstyle\sum_{i=1}^{n} h^{\mathscr{A}_i,\alpha_i}(t) + 1 \\ &= R + 1, \end{aligned}$$
where the inequality holds because $\alpha_i(s) = \alpha_i(t)$ for all $i \neq k$ and $h^{\mathscr{A}_k,\alpha_k}$ is consistent. □

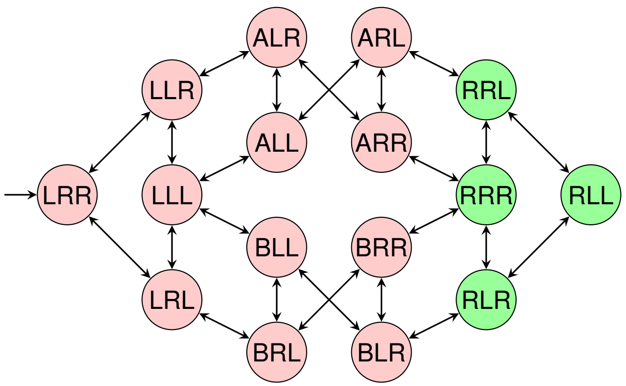
Abstractions of abstractions

Theorem (transitivity of abstractions)

Let $\mathscr{T}$, $\mathscr{T}'$ and $\mathscr{T}''$ be transition systems.

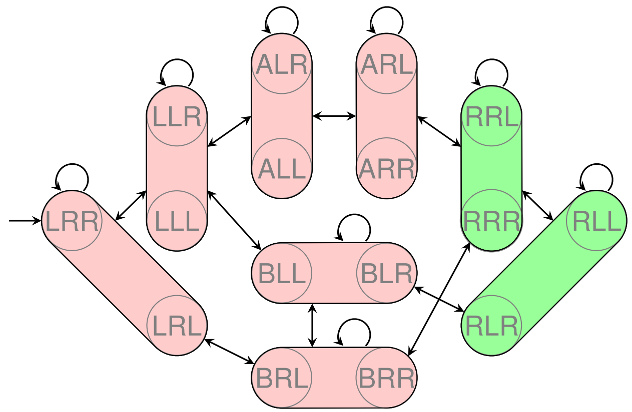
- *If $\mathscr{T}'$ is an abstraction of $\mathscr{T}$ and $\mathscr{T}''$ is an abstraction of $\mathscr{T}'$, then $\mathscr{T}''$ is an abstraction of $\mathscr{T}$.*
- *If $\mathscr{T}'$ is a strictly homomorphic abstraction of $\mathscr{T}$ and $\mathscr{T}''$ is a strictly homomorphic abstraction of $\mathscr{T}'$, then $\mathscr{T}''$ is a strictly homomorphic abstraction of $\mathscr{T}$.*

Abstractions of abstractions: example



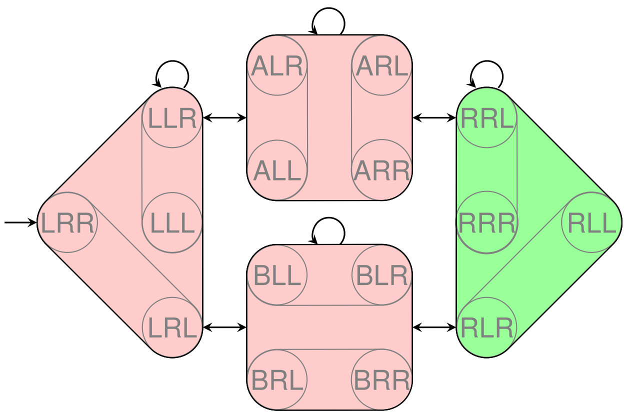
transition system $\mathscr{T}$

Abstractions of abstractions: example



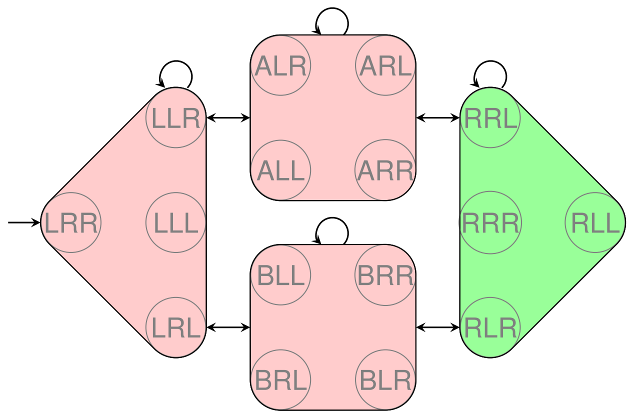
Transition system $\mathscr{T}'$ as an abstraction of $\mathscr{T}$

Abstractions of abstractions: example



Transition system $\mathcal{T}''$ as an abstraction of $\mathcal{T}'$

Abstractions of abstractions: example



Transition system $\mathcal{T}''$ as an abstraction of $\mathcal{T}$

Abstractions of abstractions (proof)

Proof.

Let $\mathcal{T} = \langle S, L, T, s_0, S_\star \rangle$, let $\mathcal{T}' = \langle S', L, T', s'_0, S'_\star \rangle$ be an abstraction of $\mathcal{T}$ with abstraction mapping α, and let $\mathcal{T}'' = \langle S'', L, T'', s''_0, S''_\star \rangle$ be an abstraction of $\mathcal{T}'$ with abstraction mapping α'.

We show that $\mathcal{T}''$ is an abstraction of $\mathcal{T}$ with abstraction mapping $\beta := \alpha' \circ \alpha$, i. e., that

1. $\beta(s_0) = s''_0$,
2. for all $s \in S_\star$, we have $\beta(s) \in S''_\star$, and
3. for all $\langle s, \ell, t \rangle \in T$, we have $\langle \beta(s), \ell, \beta(t) \rangle \in T''$.

Moreover, we show that if α and α' are strict homomorphisms, then β is also a strict homomorphism.

...

Abstractions of abstractions: proof

Proof (ctd.)

1. $\beta(s_0) = s''_0$

Because $\mathcal{T}'$ is an abstraction of $\mathcal{T}$ with mapping α, we have $\alpha(s_0) = s'_0$. Because $\mathcal{T}''$ is an abstraction of $\mathcal{T}'$ with mapping α', we have $\alpha'(s'_0) = s''_0$.
Hence $\beta(s_0) = \alpha'(\alpha(s_0)) = \alpha'(s'_0) = s''_0$.

...

Abstractions of abstractions: proof (ctd.)

Proof (ctd.)

2. For all $s \in S_\star$, we have $\beta(s) \in S''_\star$:

Let $s \in S_\star$. Because $\mathcal{T}'$ is an abstraction of $\mathcal{T}$ with mapping α, we have $\alpha(s) \in S'_\star$. Because $\mathcal{T}''$ is an abstraction of $\mathcal{T}'$ with mapping α' and $\alpha(s) \in S'_\star$, we have $\alpha'(\alpha(s)) \in S''_\star$. Hence $\beta(s) = \alpha'(\alpha(s)) \in S''_\star$.

Strict homomorphism if α and α' strict homomorphisms:

Let $s'' \in S''_\star$. Because α' is a strict homomorphism, there exists a state $s' \in S'_\star$ such that $\alpha'(s') = s''$. Because α is a strict homomorphism, there exists a state $s \in S_\star$ such that $\alpha(s) = s'$. Thus $s'' = \alpha'(\alpha(s)) = \beta(s)$ for some $s \in S_\star$.

...

Abstractions of abstractions: proof (ctd.)

Abstractions: informally
Abstractions: formally
Transition systems
Abstractions
Abstraction heuristics
Additivity
Refinements
Equivalence
Practice
Summary

Proof (ctd.)

3. For all $\langle s, \ell, t \rangle \in T$, we have $\langle \beta(s), \ell, \beta(t) \rangle \in T''$

Let $\langle s, \ell, t \rangle \in T$. Because $\mathcal{T}'$ is an abstraction of $\mathcal{T}$ with mapping α, we have $\langle \alpha(s), \ell, \alpha(t) \rangle \in T'$. Because $\mathcal{T}''$ is an abstraction of $\mathcal{T}'$ with mapping α' and $\langle \alpha(s), \ell, \alpha(t) \rangle \in T'$, we have $\langle \alpha'(\alpha(s)), \ell, \alpha'(\alpha(t)) \rangle \in T''$.
Hence $\langle \beta(s), \ell, \beta(t) \rangle = \langle \alpha'(\alpha(s)), \ell, \alpha'(\alpha(t)) \rangle \in T''$.

Strict homomorphism if α and α' strict homomorphisms:

Let $\langle s'', \ell, t'' \rangle \in T''$. Because α' is a strict homomorphism, there exists a transition $\langle s', \ell, t' \rangle \in T'$ such that $\alpha'(s') = s''$ and $\alpha'(t') = t''$. Because α is a strict homomorphism, there exists a transition $\langle s, \ell, t \rangle \in T$ such that $\alpha(s) = s'$ and $\alpha(t) = t'$.
Thus $\langle s'', \ell, t'' \rangle = \langle \alpha'(\alpha(s)), \ell, \alpha'(\alpha(t)) \rangle = \langle \beta(s), \ell, \beta(t) \rangle$ for some $\langle s, \ell, t \rangle \in T$. □

Coarsenings and refinements

Abstractions: informally
Abstractions: formally
Transition systems
Abstractions
Abstraction heuristics
Additivity
Refinements
Equivalence
Practice
Summary

Terminology: Let $\mathcal{T}$ be a transition system, let $\mathcal{T}'$ be an abstraction of $\mathcal{T}$ with abstraction mapping α, and let $\mathcal{T}''$ be an abstraction of $\mathcal{T}'$ with abstraction mapping α'.

Then:

- $\langle \mathcal{T}'', \alpha' \circ \alpha \rangle$ is called a coarsening of $\langle \mathcal{T}', \alpha \rangle$, and
- $\langle \mathcal{T}', \alpha \rangle$ is called a refinement of $\langle \mathcal{T}'', \alpha' \circ \alpha \rangle$.

Heuristic quality of refinements

Abstractions: informally
Abstractions: formally
Transition systems
Abstractions
Abstraction heuristics
Additivity
Refinements
Equivalence
Practice
Summary

Theorem (heuristic quality of refinements)

Let $h^{\mathcal{A},\alpha}$ and $h^{\mathcal{B},\beta}$ be abstraction heuristics for the same planning task Π such that $\langle \mathcal{A}, \alpha \rangle$ is a refinement of $\langle \mathcal{B}, \beta \rangle$. Then $h^{\mathcal{A},\alpha}$ dominates $h^{\mathcal{B},\beta}$.

In other words, $h^{\mathcal{A},\alpha}(s) \geq h^{\mathcal{B},\beta}(s)$ for all states s of Π.

Proof.

Since $\langle \mathcal{A}, \alpha \rangle$ is a refinement of $\langle \mathcal{B}, \beta \rangle$, there exists a mapping α' such that $\beta = \alpha' \circ \alpha$ and $\mathcal{B}$ is an abstraction of $\mathcal{A}$ with abstraction mapping α'.
For any state s of Π, we get $h^{\mathcal{B},\beta}(s) = h^*_{\mathcal{B}}(\beta(s)) = h^*_{\mathcal{B}}(\alpha'(\alpha(s))) = h^{\mathcal{B},\alpha'}(\alpha(s)) \leq h^*_{\mathcal{A}}(\alpha(s)) = h^{\mathcal{A},\alpha}(s)$, where the inequality holds because $h^{\mathcal{B},\alpha'}$ is an admissible heuristic in the transition system $\mathcal{A}$.

Isomorphic transition systems

Definition (isomorphic transition systems)

Let $\mathcal{T} = \langle S, L, T, s_0, S_\star \rangle$ and $\mathcal{T}' = \langle S', L', T', s'_0, S'_\star \rangle$ be transition systems.
We say that $\mathcal{T}$ is isomorphic to $\mathcal{T}'$, in symbols $\mathcal{T} \sim \mathcal{T}'$, if there exist bijective functions $\varphi : S \to S'$ and $\psi : L \to L'$ such that:

- $\varphi(s_0) = s'_0$,
- $s \in S_\star$ iff $\varphi(s) \in S'_\star$, and
- $\langle s, \ell, t \rangle \in T$ iff $\langle \varphi(s), \psi(\ell), \varphi(t) \rangle \in T'$.

Graph-equivalent transition systems

Definition (graph-equivalent transition systems)

Let $\mathcal{T} = \langle S, L, T, s_0, S_\star \rangle$ and $\mathcal{T}' = \langle S', L', T', s'_0, S'_\star \rangle$ be transition systems.
We say that $\mathcal{T}$ is graph-equivalent to $\mathcal{T}'$, in symbols $\mathcal{T} \overset{G}{\sim} \mathcal{T}'$, if there exists a bijective function $\varphi : S \to S'$ such that:

- $\varphi(s_0) = s'_0$,
- $s \in S_\star$ iff $\varphi(s) \in S'_\star$, and
- $\langle s, \ell, t \rangle \in T$ for some $\ell \in L$ iff $\langle \varphi(s), \ell', \varphi(t) \rangle \in T'$ for some $\ell' \in L'$.

Note: There is no requirement that the labels of $\mathcal{T}$ and $\mathcal{T}'$ correspond in any way. For example, it is permitted that all transitions of $\mathcal{T}$ have different labels and all transitions of $\mathcal{T}'$ have the same label.

Isomorphism vs. graph equivalence

- $(\sim)$ and $(\overset{G}{\sim})$ are equivalence relations.
- Two isomorphic transition systems are interchangeable for all practical intents and purposes.
- Two graph-equivalent transition systems are interchangeable for most intents and purposes.
  In particular, their state distances are identical, so they define the same abstraction heuristic for corresponding abstraction functions.
- Isomorphism implies graph equivalence, but not vice versa.

Using abstraction heuristics in practice

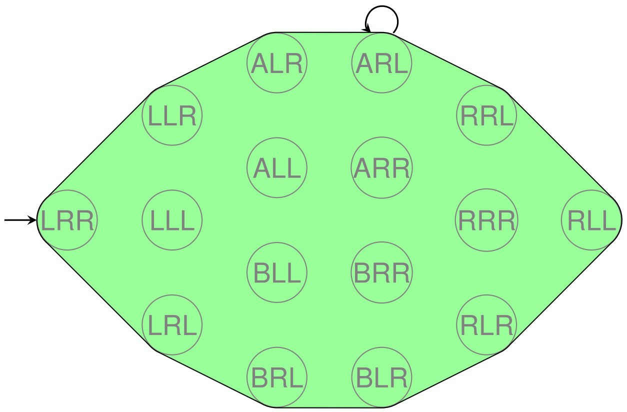
In practice, there are conflicting goals for abstractions:

- we want to obtain an informative heuristic, but
- want to keep its representation small.

Abstractions have small representations if they have

- few abstract states and
- a succinct encoding for α.

Counterexample: one-state abstraction

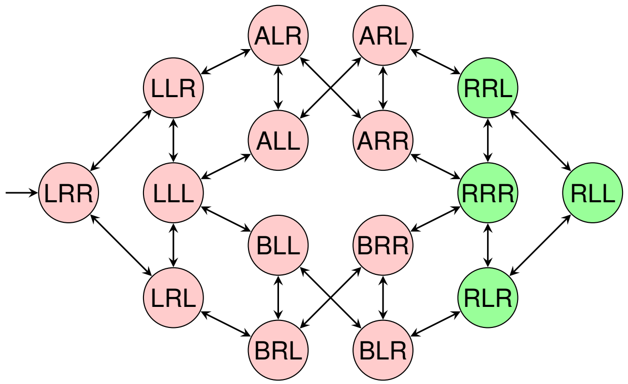


One-state abstraction: $\alpha(s) :=$ const.

+ very few abstract states and succinct encoding for α
– completely uninformative heuristic

Counterexample: identity abstraction

Abstractions: informally
Abstractions: formally
Transition systems
Abstractions
Abstraction heuristics
Additivity
Refinements
Equivalence
Practice
Summary

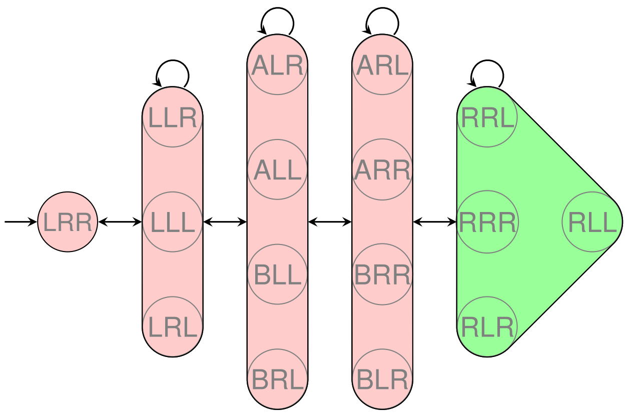


Identity abstraction: $\alpha(s) := s$.

+ perfect heuristic and succinct encoding for α
– too many abstract states

Counterexample: perfect abstraction

Abstractions: informally
Abstractions: formally
Transition systems
Abstractions
Abstraction heuristics
Additivity
Refinements
Equivalence
Practice
Summary



Perfect abstraction: $\alpha(s) := h^*(s)$.

+ perfect heuristic and usually few abstract states
– usually no succinct encoding for α

Automatically deriving good abstraction heuristics

Abstractions: informally
Abstractions: formally
Transition systems
Abstractions
Abstraction heuristics
Additivity
Refinements
Equivalence
Practice
Summary

Abstraction heuristics for planning: main research problem

Automatically derive effective abstraction heuristics for planning tasks.

⇝ we will study one state-of-the-art approach in the next chapter.

Summary

- An abstraction relates a transition system $\mathcal{T}$ (e. g. of a planning task) to another (usually smaller) transition system $\mathcal{T}'$ via an abstraction mapping α.
- Abstraction preserves all important aspects of $\mathcal{T}$: initial state, goal states and (labeled) transitions.
- Hence, they can be used to define heuristics for the original system $\mathcal{T}$: estimate the goal distance of s in $\mathcal{T}$ by the optimal goal distance of $\alpha(s)$ in $\mathcal{T}'$.
- Such abstraction heuristics are safe, goal-aware, admissible and consistent.

Summary (ctd.)

- Strictly homomorphic abstractions are desirable as they do not include “unnecessary” abstract goal states or transitions (which could lower heuristic values).
- Any surjection from the states of $\mathcal{T}$ to any set induces a strictly homomorphic abstraction in a natural way.
- Multiple abstraction heuristics can be added without losing properties like admissibility if the underlying abstraction mappings are orthogonal.
- One sufficient condition for orthogonality is that abstractions are affected by disjoint sets of labels.

Summary (ctd.)

- The process of abstraction is transitive: an abstraction can be abstracted further to yield another abstraction.
- Based on this notion, we can define abstractions that are coarsenings or refinements of others.
- A refinement can never lead to a worse heuristic.
- Practically useful abstractions are those which give informative heuristics, yet have a small representation.