

Principles of AI Planning

10. Planning as search: abstractions

Albert-Ludwigs-Universität Freiburg



Bernhard Nebel and Robert Mattmüller
November 30th, 2016

1 Abstractions: informally



- Introduction
- Practical requirements
- Multiple abstractions
- Outlook

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

November 30th, 2016

B. Nebel, R. Mattmüller – AI Planning

3 / 62

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**.

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

November 30th, 2016

B. Nebel, R. Mattmüller – AI Planning

4 / 62

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} .

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

November 30th, 2016

B. Nebel, R. Mattmüller – AI Planning

5 / 62

Abstracting a transition system: example

Example (15-puzzle)

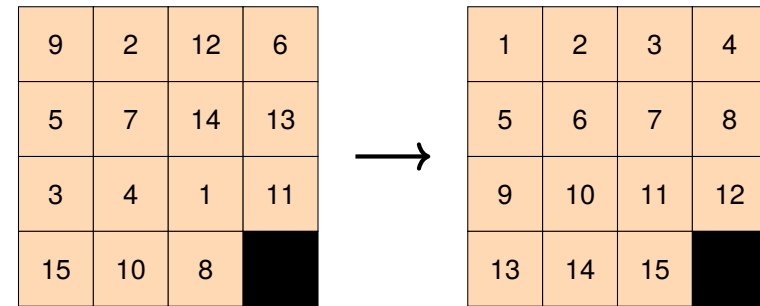
A **15-puzzle** state is given by a permutation $\langle b, t_1, \dots, t_{15} \rangle$ of $\{1, \dots, 16\}$, where b denotes the blank position and the other components denote the positions of the 15 tiles.

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.

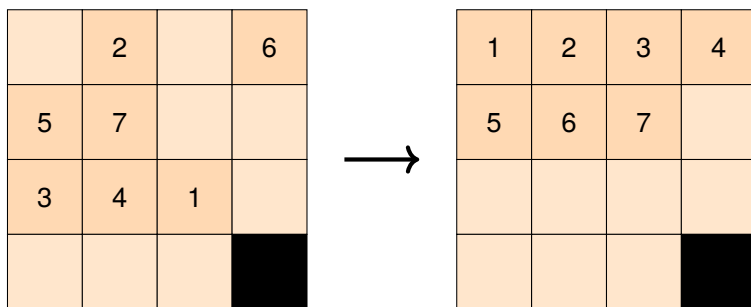
Abstraction example: 15-puzzle



real state space

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

Abstraction example: 15-puzzle



abstract state space

- $16 \cdot 15 \cdot \dots \cdot 9 = 518918400 \approx 5 \cdot 10^8$ states
- $16 \cdot 15 \cdot \dots \cdot 9 = 518918400 \approx 5 \cdot 10^8$ 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 $y \notin \{t_1, \dots, t_7\}$.

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

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)

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

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**.

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

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**.

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

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
↪ 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
- ↪ 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	

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

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

2 Abstractions: formally

- Transition systems
- Abstractions
- Abstraction heuristics
- Additive abstraction heuristics
- Coarsenings and refinements
- Equivalent transition systems
- Abstraction heuristics in practice

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

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

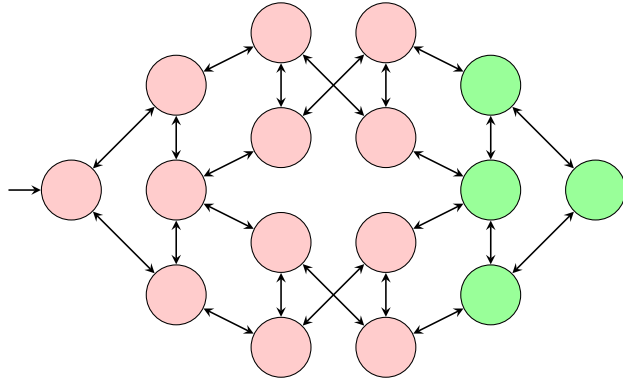
- S is a finite set of **states**,
- L is a finite set of (transition) **labels**,
- $T \subseteq S \times L \times S$ is the **transition relation**,
- $s_0 \in S$ is the **initial state**, and
- $S_\star \subseteq S$ is the set of **goal states**.

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

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

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

Transition systems: example



Abstractions:
informally

Abstractions:
formally

Transition systems

Abstractions
Abstraction
heuristics
Additivity
Refinements
Equivalence
Practice

Summary

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
- $S_\star = \{ s \in S \mid s \models \gamma \}$.

Abstractions:
informally

Abstractions:
formally

Transition systems

Abstractions
Abstraction
heuristics
Additivity
Refinements
Equivalence
Practice

Summary

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

Abstractions:
informally

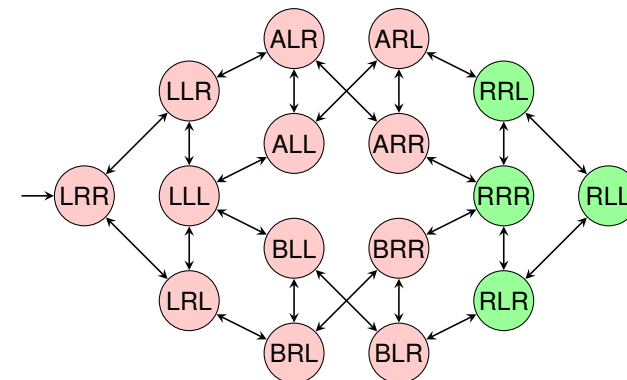
Abstractions:
formally

Transition systems

Abstractions
Abstraction
heuristics
Additivity
Refinements
Equivalence
Practice

Summary

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:
informally

Abstractions:
formally

Transition systems

Abstractions
Abstraction
heuristics
Additivity
Refinements
Equivalence
Practice

Summary

Definition (abstraction, abstraction mapping)

Let $\mathcal{T} = \langle S, L, T, s_0, S_* \rangle$ and $\mathcal{T}' = \langle S', L', T', s'_0, S'_* \rangle$ be transition systems with the same label set $L = L'$, and let $\alpha : S \rightarrow S'$ 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 $s \in S_*$, we have $\alpha(s) \in S'_*$, and
- for all $\langle s, l, t \rangle \in T$, we have $\langle \alpha(s), l, \alpha(t) \rangle \in T'$.

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 .

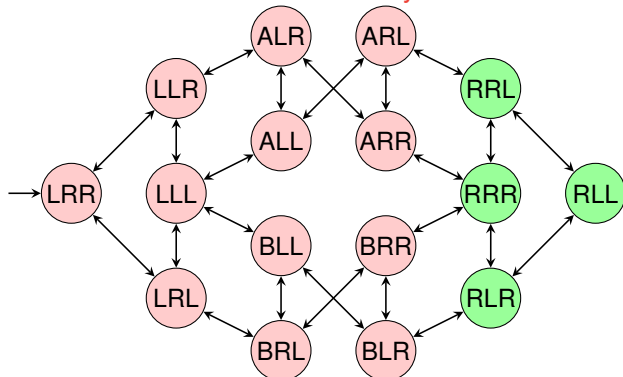
- \mathcal{T} is called the **concrete transition system**.
- \mathcal{T}' is called the **abstract transition system**.
- Similarly: **concrete/abstract state space**, **concrete/abstract transition**, etc.

We say that:

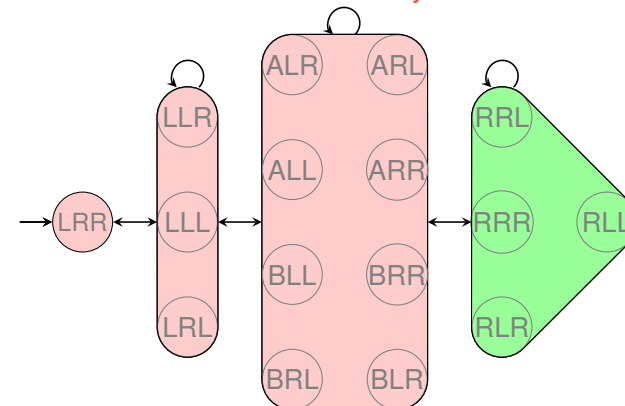
- \mathcal{T}' is an **abstraction of \mathcal{T}** (without mentioning α)
- α is an **abstraction mapping on \mathcal{T}** (without mentioning \mathcal{T}')

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 .

concrete transition system



abstract transition system



Note: Most arcs represent many parallel transitions.

Induced abstractions

Definition (induced abstractions)

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

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

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

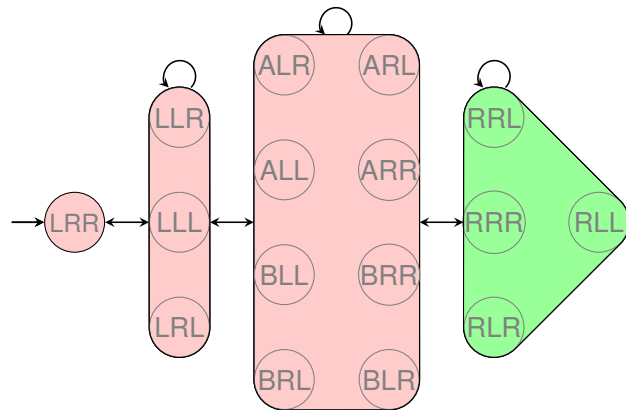
Note: It is easy to see that \mathcal{T}^α is an abstraction of \mathcal{T} . It is the “smallest” abstraction of \mathcal{T} with abstraction mapping α .

Induced abstractions: terminology

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

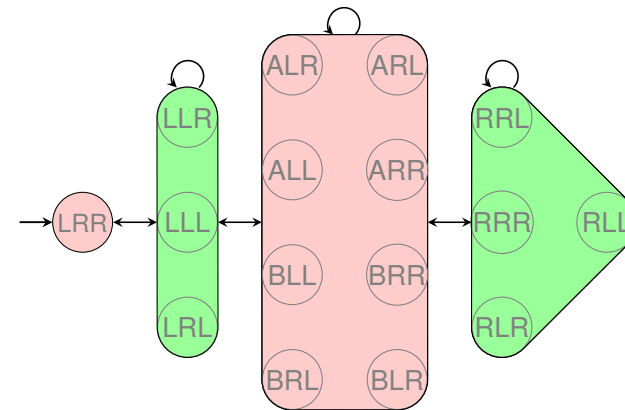
- α is called a **strict homomorphism** from \mathcal{T} to \mathcal{T}' , and \mathcal{T}' is called a **strictly homomorphic abstraction** of \mathcal{T} .
- If α is bijective, it is called an **isomorphism** between \mathcal{T} and \mathcal{T}' , and the two transition systems are called **isomorphic**.

Strictly homomorphic abstractions: example



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

Strictly homomorphic abstractions: example



If we add any goal states or transitions, it is still an abstraction of \mathcal{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 \mathcal{A} be an abstraction of $\mathcal{T}(\Pi)$ with abstraction mapping α .

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

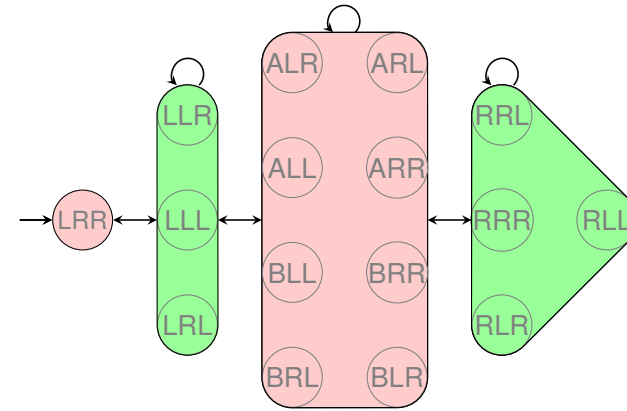
Note: $h^{\mathcal{A},\alpha}(s) = \infty$ if no goal state of \mathcal{A} is reachable from $\alpha(s)$

Definition (abstr. heur. induced by strict homomorphism)

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

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

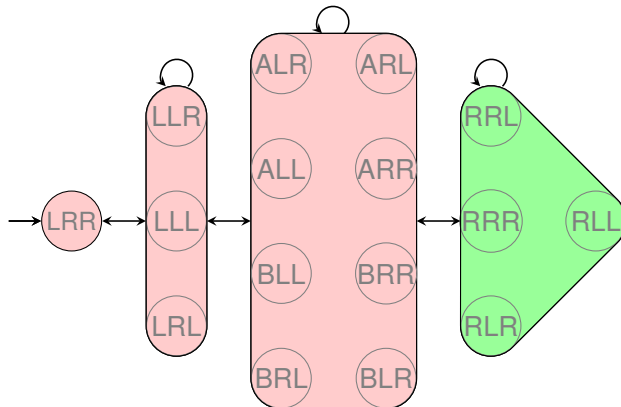
Abstraction heuristics: example



$$h^{\mathcal{A},\alpha}(\{p \mapsto L, t_A \mapsto R, t_B \mapsto R\}) = 1$$

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

Abstraction heuristics: example



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

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

Consistency of abstraction heuristics

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

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

Then $h^{\mathcal{A},\alpha}$ is safe, goal-aware, admissible and consistent.

Proof.

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

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

Goal-awareness: We need to show that $h^{\mathcal{A},\alpha}(s) = 0$ 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 $h^{\mathcal{A},\alpha}(s) = h_{\mathcal{A}}^*(\alpha(s)) = 0$.

- 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 . We need to prove that $h^{\mathcal{A}, \alpha}(s) \leq h^{\mathcal{A}, \alpha}(t) + 1$.

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 \mathcal{A} . Therefore, $h^{\mathcal{A}, \alpha}(s) = h_{\mathcal{A}}^*(\alpha(s)) \leq h_{\mathcal{A}}^*(\alpha(t)) + 1 = h^{\mathcal{A}, \alpha}(t) + 1$, where the inequality holds because the shortest path from $\alpha(s)$ to the goal in \mathcal{A} cannot be longer than the shortest path from $\alpha(s)$ to the goal via $\alpha(t)$. \square

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

Definition (orthogonal abstraction mappings)

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

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

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

Definition (affecting transition labels)

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

Theorem (affecting labels vs. orthogonality)

Let \mathcal{A}_1 be an abstraction of \mathcal{T} with abstraction mapping α_1 . Let \mathcal{A}_2 be an abstraction of \mathcal{T} with abstraction mapping α_2 . If no label of \mathcal{T} affects both \mathcal{A}_1 and \mathcal{A}_2 , then α_1 and α_2 are orthogonal.

(Easy proof omitted.)

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

	2		6
5	7		
3	4	1	

9		12	
		14	13
			11
15	10	8	

Are the abstraction mappings orthogonal?

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

Orthogonal abstraction mappings: example



	2		6
5	7		
3	4	1	

9		12	
		14	13
			11
15	10	8	

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

Are the abstraction mappings orthogonal?

Orthogonality and additivity

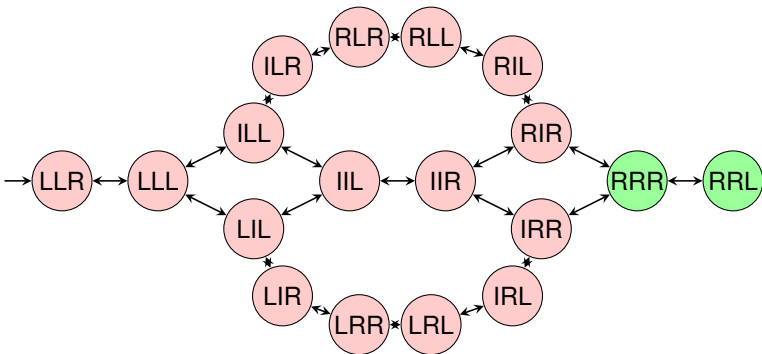


Theorem (additivity for orthogonal abstraction mappings)

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

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

Orthogonality and additivity: example

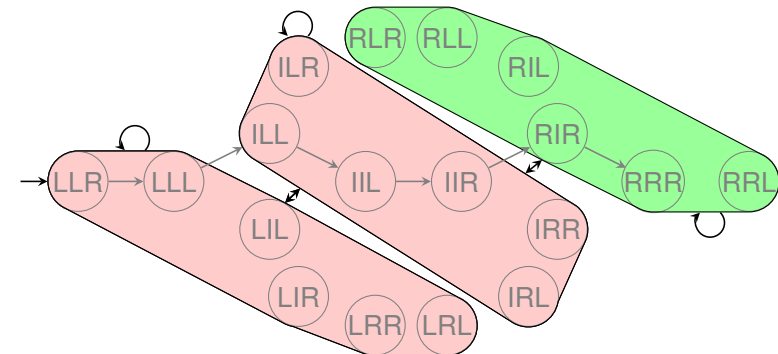


- Abstractions: informally
- Abstractions: formally
- Transition systems
- Abstractions
- Abstraction heuristics
- Additivity**
- Refinements
- Equivalence
- Practice
- Summary

transition system \mathcal{T}

state variables: first package, second package, truck

Orthogonality and additivity: example

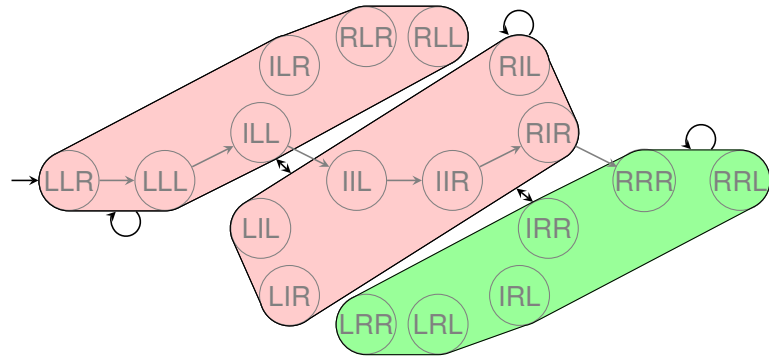


- Abstractions: informally
- Abstractions: formally
- Transition systems
- Abstractions
- Abstraction heuristics
- Additivity**
- Refinements
- Equivalence
- Practice
- Summary

abstraction \mathcal{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 \mathcal{A}_2 (orthogonal to \mathcal{A}_1)
mapping: only consider state of second package

Orthogonality and additivity: proof

Proof.

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

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

Goal-awareness: For goal states $s \in S_*$,

$\sum_{i=1}^n h^{\mathcal{A}_i, \alpha_i}(s) = \sum_{i=1}^n 0 = 0$ because all individual abstractions
are goal-aware.

Abstractions:
informally

Abstractions:
formally

Transition systems

Abstractions

Abstraction
heuristics

Additivity

Refinements

Equivalence

Practice

Summary

Orthogonality and additivity: proof (ctd.)

Proof (ctd.)

Consistency: Let $s, t \in S$ such that t is a successor of s .

Let $L := \sum_{i=1}^n h^{\mathcal{A}_i, \alpha_i}(s)$ and $R := \sum_{i=1}^n h^{\mathcal{A}_i, \alpha_i}(t)$.

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 $i \in \{1, \dots, n\}$.

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

Abstractions:
informally

Abstractions:
formally

Transition systems

Abstractions

Abstraction
heuristics

Additivity

Refinements

Equivalence

Practice

Summary

Orthogonality and additivity: proof (ctd.)

Proof (ctd.)

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

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

$$\begin{aligned} \text{Then } L &= \sum_{i=1}^n h^{\mathcal{A}_i, \alpha_i}(s) \\ &= \sum_{i \in \{1, \dots, n\} \setminus \{k\}} h^{\mathcal{A}_i, \alpha_i}(s) + h^{\mathcal{A}_k, \alpha_k}(s) \\ &\leq \sum_{i \in \{1, \dots, n\} \setminus \{k\}} h^{\mathcal{A}_i, \alpha_i}(t) + h^{\mathcal{A}_k, \alpha_k}(t) + 1 \\ &= \sum_{i=1}^n h^{\mathcal{A}_i, \alpha_i}(t) + 1 \\ &= R + 1, \end{aligned}$$

where the inequality holds because $\alpha_i(s) = \alpha_i(t)$ for all $i \neq k$ and
 $h^{\mathcal{A}_k, \alpha_k}$ is consistent. \square

Abstractions:
informally

Abstractions:
formally

Transition systems

Abstractions

Abstraction
heuristics

Additivity

Refinements

Equivalence

Practice

Summary

Abstractions of abstractions



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

Theorem (transitivity of abstractions)

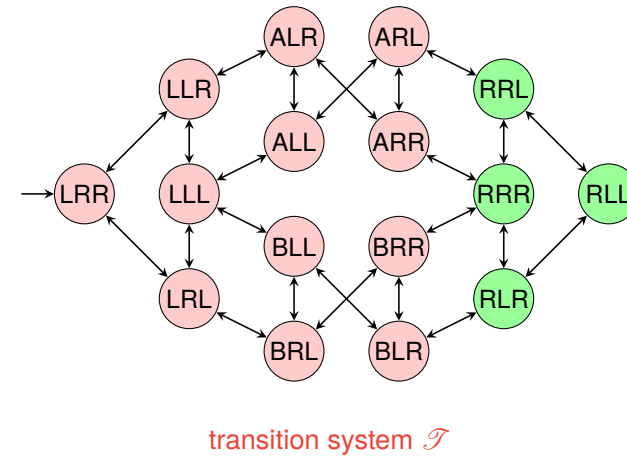
Let \mathcal{T} , \mathcal{T}' and \mathcal{T}'' be transition systems.

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

Abstractions of abstractions: example



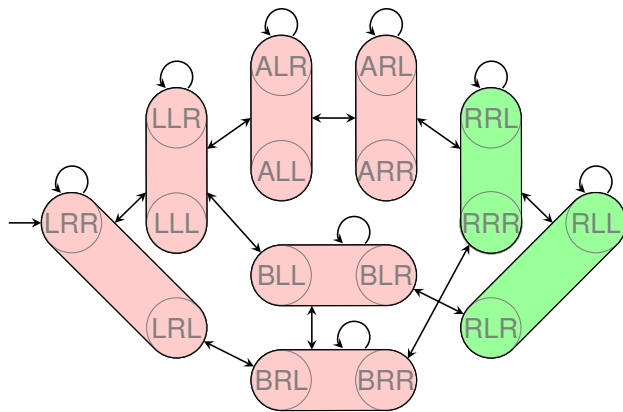
- Abstractions: informally
- Abstractions: formally
- Transition systems
- Abstractions
- Abstraction heuristics
- Additivity
- Refinements
- Equivalence
- Practice
- Summary



Abstractions of abstractions: example



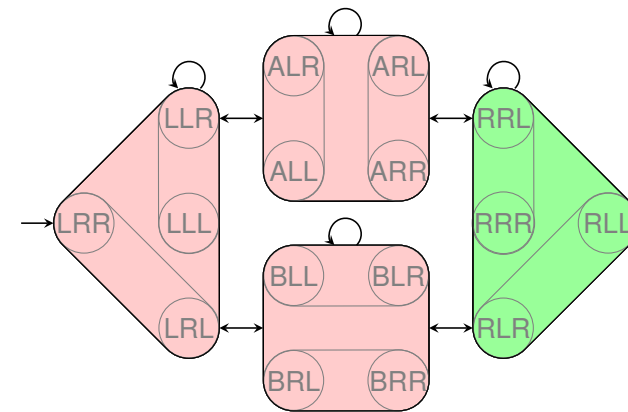
- Abstractions: informally
- Abstractions: formally
- Transition systems
- Abstractions
- Abstraction heuristics
- Additivity
- Refinements
- Equivalence
- Practice
- Summary

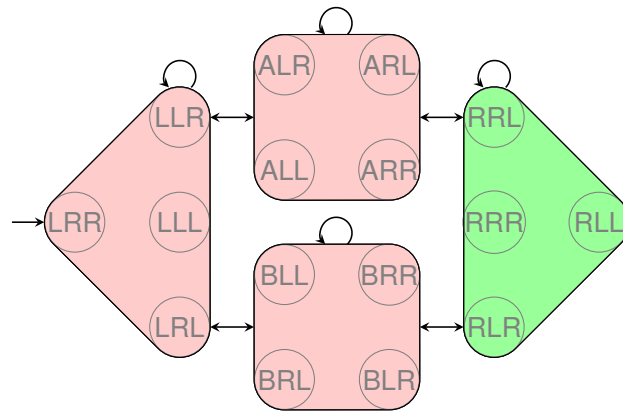


Abstractions of abstractions: example



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





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

Proof.

Let $\mathcal{T} = \langle S, L, T, s_0, S_* \rangle$, let $\mathcal{T}' = \langle S', L, T', s'_0, S'_* \rangle$ be an abstraction of \mathcal{T} with abstraction mapping α , and let $\mathcal{T}'' = \langle S'', L, T'', s''_0, S''_* \rangle$ 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 $s \in S_*$, we have $\beta(s) \in S''_*$, and
- 3 for all $\langle s, l, t \rangle \in T$, we have $\langle \beta(s), l, \beta(t) \rangle \in T''$.

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

...

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

...

Proof (ctd.)

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

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

Strict homomorphism if α and α' strict homomorphisms:

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

...

Proof (ctd.)

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

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

Hence $\langle \beta(s), l, \beta(t) \rangle = \langle \alpha'(\alpha(s)), l, \alpha'(\alpha(t)) \rangle \in T''$.

Strict homomorphism if α and α' strict homomorphisms:

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

Thus $\langle s'', l, t'' \rangle = \langle \alpha'(\alpha(s)), l, \alpha'(\alpha(t)) \rangle = \langle \beta(s), l, \beta(t) \rangle$ for some $\langle s, l, t \rangle \in T$. □

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 $\langle \mathcal{T}'', \alpha' \circ \alpha \rangle$.

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} .

Abstractions:
informally

Abstractions:
formally

Transition systems

Abstractions

Abstraction

heuristics

Additivity

Refinements

Equivalence

Practice

Summary

Definition (isomorphic transition systems)

Let $\mathcal{T} = \langle S, L, T, s_0, S_* \rangle$ and $\mathcal{T}' = \langle S', L', T', s'_0, S'_* \rangle$ 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 $\varphi : S \rightarrow S'$ and $\psi : L \rightarrow L'$ such that:

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

Abstractions:
informally

Abstractions:
formally

Transition systems

Abstractions

Abstraction

heuristics

Additivity

Refinements

Equivalence

Practice

Summary

Graph-equivalent transition systems

Definition (graph-equivalent transition systems)

Let $\mathcal{T} = \langle S, L, T, s_0, S_* \rangle$ and $\mathcal{T}' = \langle S', L', T', s'_0, S'_* \rangle$ be transition systems.

We say that \mathcal{T} is **graph-equivalent** to \mathcal{T}' , in symbols $\mathcal{T} \stackrel{G}{\sim} \mathcal{T}'$, if there exists a bijective function $\varphi : S \rightarrow S'$ such that:

- $\varphi(s_0) = s'_0$,
- $s \in S_*$ iff $\varphi(s) \in S'_*$, 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.

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

Isomorphism vs. graph equivalence

- (\sim) and $(\stackrel{G}{\sim})$ 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.

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

Using abstraction heuristics in practice

In practice, there are conflicting goals for abstractions:

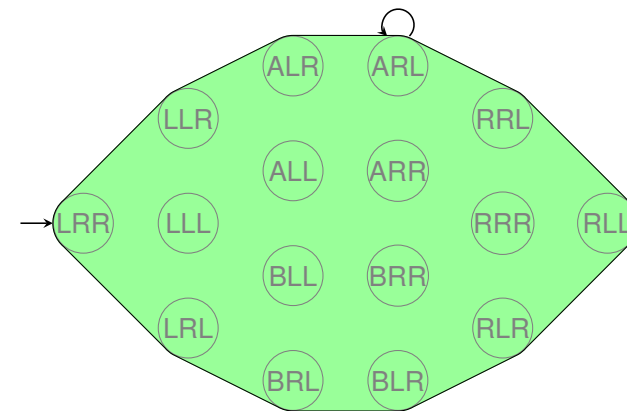
- we want to obtain an **informative heuristic**, but
- want to keep its **representation small**.

Abstractions have small representations if they have

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

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

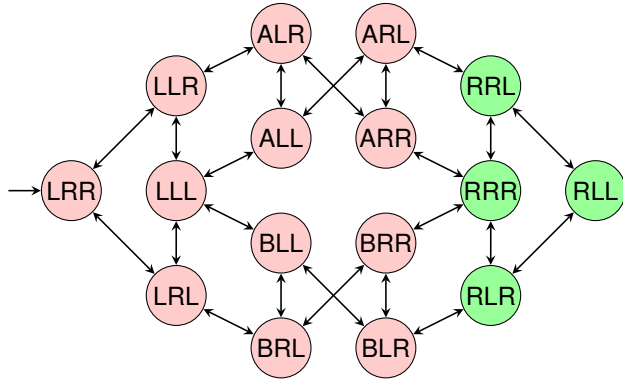
Counterexample: one-state abstraction



- One-state abstraction: $\alpha(s) := \text{const.}$**
- + very few abstract states and succinct encoding for α
 - completely uninformative heuristic

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

Counterexample: identity abstraction



Abstractions:
informally

Abstractions:
formally

Transition systems

Abstractions

Abstraction
heuristics

Additivity

Refinements

Equivalence

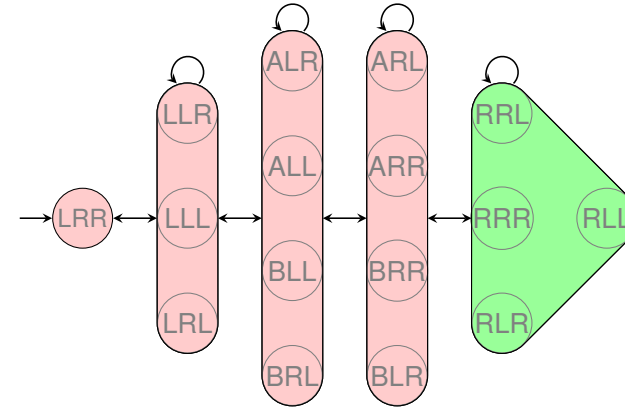
Practice

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

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.

Abstractions:
informally

Abstractions:
formally

Transition systems

Abstractions

Abstraction
heuristics

Additivity

Refinements

Equivalence

Practice

Summary

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**.

Abstractions:
informally

Abstractions:
formally

Summary

- **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.

Abstractions:
informally

Abstractions:
formally

Summary

- 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**.

Abstractions:
informally

Abstractions:
formally

Summary