AI Planning

11. Abstractions

It's a Long Way to the Goal, But How Long Exactly?
Part III: *Willfully Ignoring Some of Those Distinctions*

Álvaro Torralba, Cosmina Croitoru

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Winter Term 2018/2019

Thanks to Prof. Jörg Hoffmann for slide sources

Agenda

Motivation

→ Abstractions are a method to relax planning tasks, and thus automatically compute heuristic functions h.

→ Every h yields good performance only in some domains!

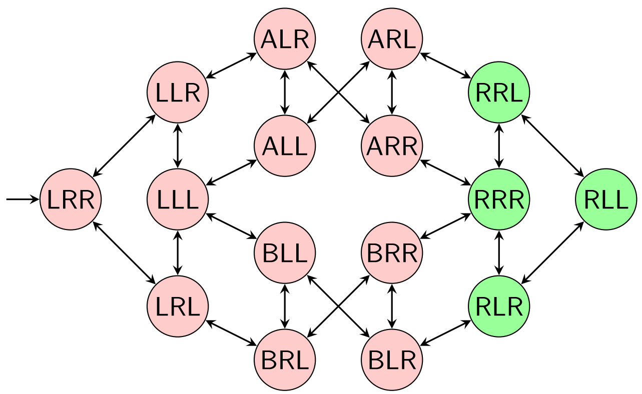
We cover the 4 different methods currently known:

- Critical path heuristics: Done. → **Chapter 8**
- Delete relaxation: Basically done. → **Chapters 9, 10**
- Abstractions: → **This Chapter, and Chapters 12 & 13**
- Landmarks: → **Chapter 14**

→ Abstractions are among the most successful methods for computing lower-bound estimators! See Conclusion sections of **Chapters 12 and 13**, as well as **Chapter 19**.

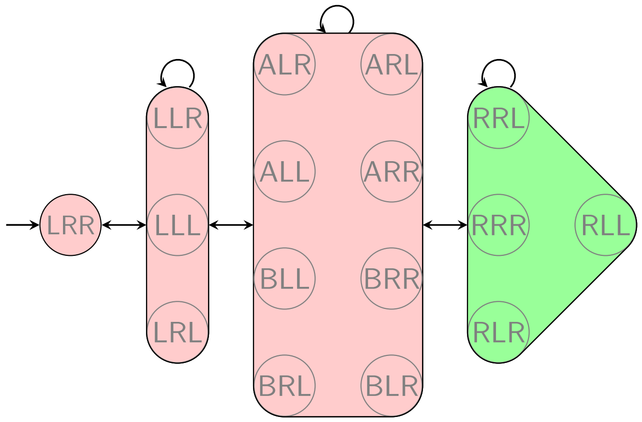
Abstractions in a Nutshell: Example

Concrete transition system: (of "Logistics mal anders", see later)



Abstractions in a Nutshell: Example

Abstract transition system: (of "Logistics mal anders", see later)



Abstractions in a Nutshell: Wrap-Up

→ Abstracting a transition system means dropping some distinctions between states, while preserving all transitions and goal states.

- An abstraction of a transition system Θ is defined by a function α (the abstraction mapping), mapping states to abstract states (also block states).
- If α maps states s and t to the same abstract state, then s and t are not distinguished anymore (they are equivalent under α).
- The abstract transition system Θ^α on the image of α is defined by homomorphically mapping over all goal states and transitions from Θ, and thus preserving all solutions.
- The abstract remaining cost, i.e., remaining cost in Θ^α, is an estimate h^α for remaining cost in Θ. As we preserve all solutions, h^α is admissible.

Our Program for Abstraction Heuristics

We take a look at abstractions and their use for generating admissible heuristic functions:

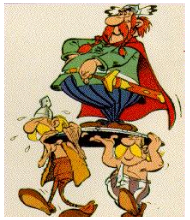
- In **This Chapter**, we introduce abstractions and abstraction heuristics and study some of their most important properties. We disregard how to actually construct abstractions in practice.
- In **Chapter 12**, we will discuss a particular class of abstraction heuristics and its practical handling in detail, namely pattern database heuristics.
- In **Chapter 13**, we will discuss another particular class of abstraction heuristics and its practical handling in detail, namely merge-and-shrink abstractions.

→ We handle all these methods in FDR, where they are most natural. We do not mention STRIPS at all (which is a special case anyway).

Our Agenda for This Chapter

2. **Abstraction Basics:** Formal definition of abstractions and their associated structures; proving their basic properties.
3. **Practical vs. Pathological Abstractions:** We briefly illuminate basic practical issues, through a number of examples illustrating "how not to do it".
4. **A Prominent Example: The 15-Puzzle:** Abstractions in AI were invented in the context of the 15-Puzzle, so we include this here as a more interesting illustration than the usual "trucks & packages".
5. **Additive Abstractions:** We introduce a simple criterion allowing to admissibly sum up several abstraction heuristics.
6. **Abstraction Refinements:** Abstractions often are constructed by modifying other abstractions, and we briefly introduce the basic concepts here.

Questionnaire



- V: $M : \{MajHome, Bar, Pool, Shield\}$; $S_1, S_2 : \{MajHome, Bar, Pool\}$.
- Initial state I: $M = Bar$, $S_1 = MajHome$, $S_2 = MajHome$.
- Goal G: $M = MajHome$, $S_1 = MajHome$, $S_2 = MajHome$.
- Actions A:
  $lift(x)$: pre $S_1 = x$, $S_2 = x$, $M = x$; eff $M = Shield$
  $drop(x)$: pre $S_1 = x$, $S_2 = x$, $M = Shield$; eff $M = x$
  $go(i, x, y)$: pre $S_i = x$; eff $S_i = y$

Question!

Say α projects onto $\{M\}$, i.e., $\alpha(s) = \alpha(t)$ iff s and t agree on M. What is $h^\alpha(I)$? And what if α projects onto $\{S_1, S_2\}$?

→ α projects onto $\{S_1, S_2\}$: $\alpha(I) = \alpha(M = Bar, S_1 = MajHome, S_2 = MajHome) = \alpha(M = MajHome, S_1 = MajHome, S_2 = MajHome)$, so $h^\alpha(I) = 0$.

→ α projects onto $\{M\}$: Θ^α has "block states" for M values $MajHome$, Bar, $Pool$, $Shield$. We can use $lift(Bar)$ to get from Bar to $Shield$, and then directly $drop(MajHome)$ to get from $Shield$ to $MajHome$. So $h^\alpha(I) = 2$.

→ Note: This is a pattern database abstraction (→ Chapter 12).

What Do We Abstract?

Here, i.e., this Chapter: Arbitrary transition systems.

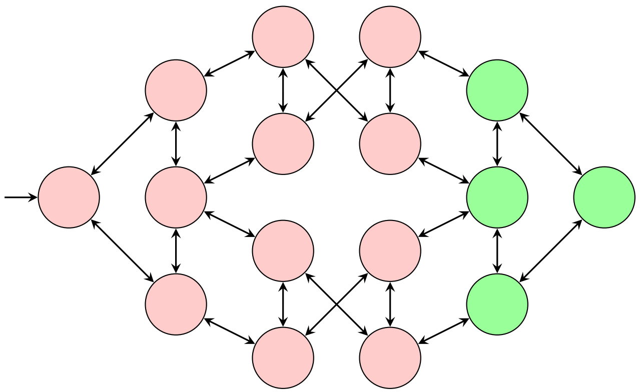
Reminder: → Chapter 2

A transition system is a 6-tuple $\Theta = (S, L, c, T, I, S^G)$ where S is the set of states, L are the transition labels, c maps each label to its cost, $T \subseteq S \times L \times S$ are the transitions, I is the initial state, and S^G is the set of goal states.

Later, i.e., Chapters 12 and 13: FDR state spaces.

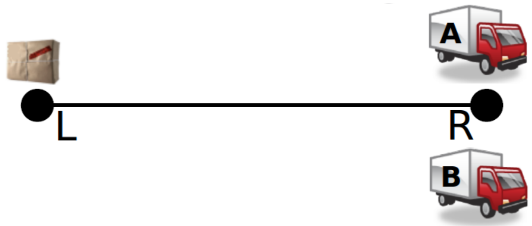
- Abstraction of an FDR task Π = abstraction of its state space Θ_Π.
- The results in this Chapter apply to arbitrary Θ.
- The results of Chapters 12 and 13 are specific to FDR. They exploit the compact representation of $\Theta = \Theta_\Pi$ via Π in order to build the abstract state space effectively.

This is How We'll Depict Transition Systems



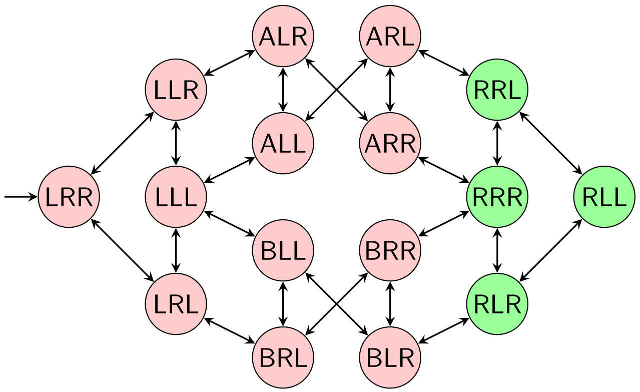
→ To reduce clutter, the figures usually omit arc labels, and collapse transitions between identical states.

"Logistics mal anders": *One* Package, *Two* Trucks



- $V = \{p, t_A, t_B\}$ with $D_p = \{L, R, A, B\}$ and $D_{t_A} = D_{t_B} = \{L, R\}$.
- $A = \{pickup(x, y) \mid x \in \{A, B\}, y \in \{L, R\}\}$
  $\cup \{drop(x, y) \mid x \in \{A, B\}, y \in \{L, R\}\}$
  $\cup \{move(x, y, y') \mid x \in \{A, B\}, y, y' \in \{L, R\}, y \neq y'\}$, *with*
  - $pre_{pickup(x,y)}$: $t_x = y, p = y$; $eff_{pickup(x,y)}$: $p = x$;
  - $pre_{drop(x,y)}$: $t_x = y, p = x$; $eff_{drop(x,y)}$: $p = y$;
  - $pre_{move(x,y,y')}$: $t_x = y$; $eff_{move(x,y,y')}$: $t_x = y'$.
- I: $p = L$, $t_A = R$, $t_B = R$. G: $p = R$.

The State Space of "Logistics mal anders"



- State $p = x$, $t_A = y$, $t_B = z$ is depicted as xyz.
- Transition labels not shown. For example, the transition from LLL to ALL has the label $pickup(A, L)$.

Abstractions

Definition (Abstraction). *Let $\Theta = (S, L, c, T, I, S^G)$ be a transition system. An abstraction of Θ is a surjective function $\alpha : S \mapsto S^\alpha$, also referred to as the abstraction mapping. The abstract state space induced by α, written Θ^α, is the transition system $\Theta^\alpha = (S^\alpha, L, c, T^\alpha, I^\alpha, S^{\alpha G})$ defined by:*

1. $I^\alpha = \alpha(I)$.
2. $S^{\alpha G} = \{\alpha(s) \mid s \in S^G\}$. */* preserve goal states */*
3. $T^\alpha = \{(\alpha(s), l, \alpha(t)) \mid (s, l, t) \in T\}$.*/* preserve transitions */*

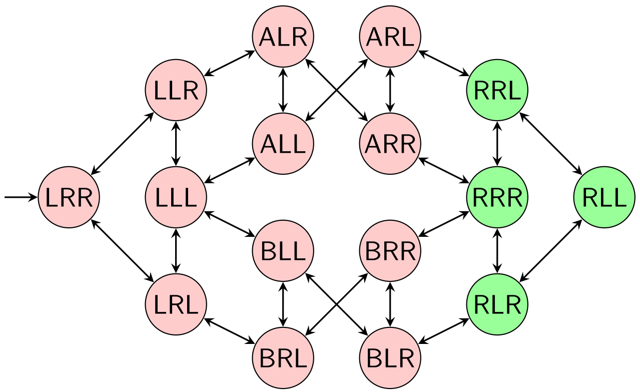
The size of the abstraction is the number $|S^\alpha|$ of abstract states.

$\rightarrow$ Θ is called the concrete state space. Similarly: concrete/abstract transition system, concrete/abstract transition, etc.

$\rightarrow$ Why do we require α to be surjective? So that Θ^α does not contain superfluous states.

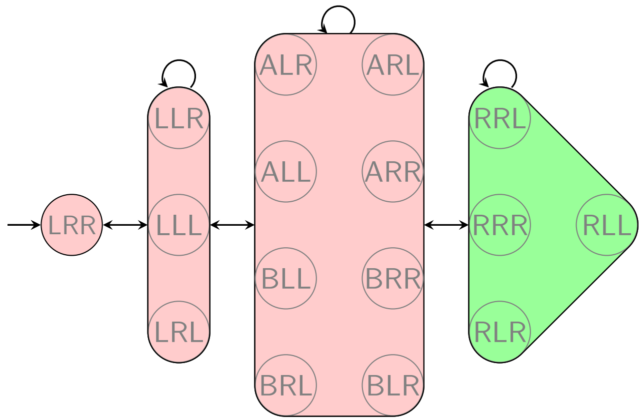
Abstractions: "Logistics mal anders"

Concrete transition system:



Abstractions: "Logistics mal anders"

Abstract transition system:



$\rightarrow$ A transition between concrete states is "spurious" if it exists in the abstract but not in the concrete state space. Example here? We can go in a single step from LRR to LLL.

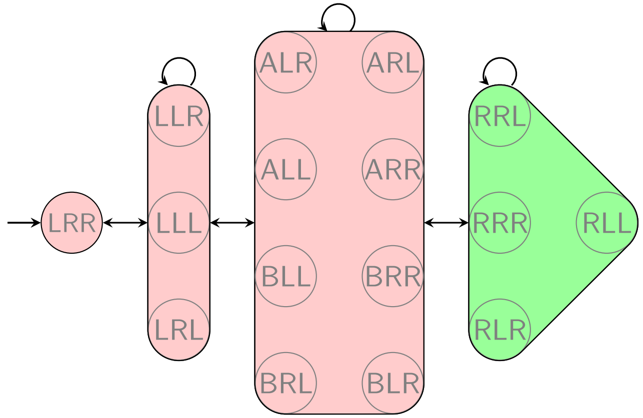
Abstraction Heuristics

Definition (Abstraction Heuristic). *Let $\Theta = (S, L, c, T, I, S^G)$ be a transition system, and let α be an abstraction of Θ. The abstraction heuristic induced by α, written h^α, is the heuristic function $h^\alpha : S \mapsto \mathbb{R}_0^+ \cup \{\infty\}$ which maps each state $s \in S$ to $h^*_{\Theta^\alpha}(\alpha(s))$, i.e., to the remaining cost of $\alpha(s)$ in Θ^α.*

→ The abstract remaining cost (remaining cost in Θ^α) is used as the heuristic estimate for remaining cost in Θ.

→ $h^\alpha(s) = \infty$ if no goal state of Θ^α is reachable from $\alpha(s)$.

Abstraction Heuristics: "Logistics mal anders"



$$h^\alpha(\{p = L, t_A = R, t_B = R\}) = 3 \neq h^*(\{p = L, t_A = R, t_B = R\}) = 4$$

Abstraction Heuristics: Properties

Proposition (h^α is Admissible). *Let Θ be a transition system, and let α be an abstraction of Θ. Then h^α is consistent and goal-aware, and thus also admissible and safe.*

Proof. Let $\Theta = (S, L, c, T, I, S^G)$ and $\Theta^\alpha = (S^\alpha, L, c, T^\alpha, I^\alpha, S^{\alpha G})$.

For goal-awareness, we need to show that $h^\alpha(s) = 0$ for all $s \in S^G$. So let $s \in S^G$. Then $\alpha(s) \in S^{\alpha G}$ by definition of abstractions, and hence $h^\alpha(s) = h^*_{\Theta^\alpha}(\alpha(s)) = 0$.

For consistency, we need to show that whenever $(s, a, t) \in T$, $h^\alpha(s) \leq h^\alpha(t) + c(a)$. By definition, $h^\alpha(s) = h^*_{\Theta^\alpha}(\alpha(s))$ and $h^\alpha(t) = h^*_{\Theta^\alpha}(\alpha(t))$, so we need to show that $h^*_{\Theta^\alpha}(\alpha(s)) \leq h^*_{\Theta^\alpha}(\alpha(t)) + c(a)$. Since (s, a, t) is a concrete transition, by definition of abstractions we have an abstract transition $(\alpha(s), a, \alpha(t))$ in Θ^α. But then, $h^*_{\Theta^\alpha}(\alpha(s)) \leq h^*_{\Theta^\alpha}(\alpha(t)) + c(a)$ holds simply because h^* is consistent. (In our notation here: $h^*_{\Theta^\alpha}$ is consistent in Θ^α).

Abstractions as Quotient Systems

Alternate views: (a) transition systems Θ^α vs. (b) quotient system $\Theta/{\sim^\alpha}$

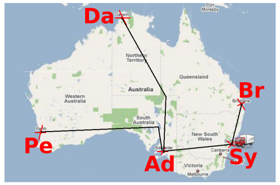
- (b) is intuitive, and useful to characterize certain classes of abstractions (see **Chapter 13**).
- (a) is used in implementation (abstract states may be large).

Definition (Induced Equivalence Relation). *Let $\Theta = (S, L, c, T, I, S^G)$ be a transition system, and let $\alpha : S \mapsto S'$ be a surjective function. Then by $\sim^\alpha$ we denote the induced equivalence relation on Θ, defined by $s \sim^\alpha t$ iff $\alpha(s) = \alpha(t)$.*

The quotient system $\Theta/{\sim^\alpha}$ is the transition system $(S/{\sim^\alpha}, L, c, T/{\sim^\alpha}, I/{\sim^\alpha}, S^G/{\sim^\alpha})$ where: the states $[s] \in S/{\sim^\alpha}$ are the equivalence classes under $\sim^\alpha$; $([s], l, [t])$ is a transition in $T/{\sim^\alpha}$ iff (s, l, t) is a transition in T; the initial state is $I/{\sim^\alpha} = [I]$; the goal states are $S^G/{\sim^\alpha} = \{[s] \mid s \in S^G\}$.

Proposition. *Let $\Theta = (S, L, c, T, I, S^G)$ be a transition system, and let $\alpha : S \mapsto S'$ be an abstraction of Θ. Then $\Theta/{\sim^\alpha}$ is isomorphic to Θ^α.* (Direct from definition.)

Questionnaire



- Variables: $at : \{Sy, Ad, Br, Pe, Ad\}$; $v(x) : \{T, F\}$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- Actions: $drive(x, y)$ where x, y have a road.
- Costs: $Sy \leftrightarrow Br : 1$, $Sy \leftrightarrow Ad : 1.5$, $Ad \leftrightarrow Pe : 3.5$, $Ad \leftrightarrow Da : 4$.
- Initial state: $at = Sy, v(Sy) = T, v(x) = F$ for $x \neq Sy$.
- Goal: $at = Sy, v(x) = T$ for all x.

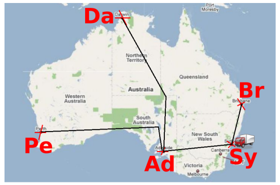
Question!

Say α projects this planning task onto $\{at, v(Pe), v(Da)\}$, i.e., $\alpha(s) = \alpha(t)$ iff they agree on these variables. What is $h^\alpha(I)$?

(A): 10 (B): 12.5
(C): 18 (D): 20

→ In the abstract state space induced by α, any solution must visit Perth and Darwin, then return to Sydney. The optimal sequence doing so has cost 18, so (C) is correct.

Questionnaire, ctd.



- Variables: $at : \{Sy, Ad, Br, Pe, Ad\}$; $v(x) : \{T, F\}$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- Actions: $drive(x, y)$ where x, y have a road.
- Costs: $Sy \leftrightarrow Br : 1$, $Sy \leftrightarrow Ad : 1.5$, $Ad \leftrightarrow Pe : 3.5$, $Ad \leftrightarrow Da : 4$.
- Initial state: $at = Sy, v(Sy) = T, v(x) = F$ for $x \neq Sy$.
- Goal: $at = Sy, v(x) = T$ for all x.

Question!

Say α projects this task onto $\{v(Pe), v(Da)\}$. What is $h^\alpha(I)$?

(A): 2 (B): 7.5
(C): 12.5 (D): 14

→ We can drive to Perth and Darwin without achieving the truck precondition. The only actions driving to these cities cost 3.5 respectively 4, so (B) is correct.

Which Abstractions Should We Use in Practice?

Conflicting Objectives

The eternal trade-off between accuracy and efficiency:

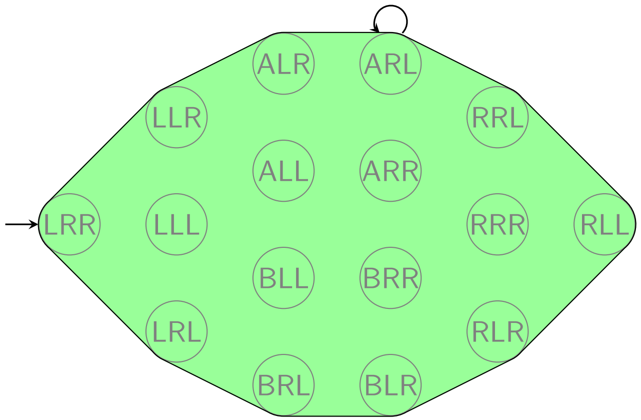
- We want to obtain an informative heuristic.
- We want to obtain a small computational overhead.

→ The abstraction function α is a very powerful parameter, allowing to travel the whole way between both extremes (see next slides).

→ What do we mean by "small computational overhead"?

- **Fast computation of α:** For a given state s, the abstract state $\alpha(s)$ must be efficiently computable.
- **Few abstract states:** For a given abstract state $\alpha(s)$, the abstract remaining cost $h^\alpha(s) = h^*_{\Theta^\alpha}(\alpha(s))$ must be efficiently computable.

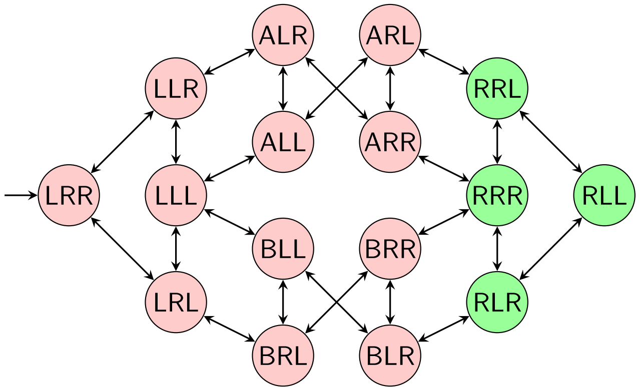
Pathological Case 1: One-State Abstraction



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

+ Trivial to compute α, just one abstract state.
– Completely uninformative h^α.

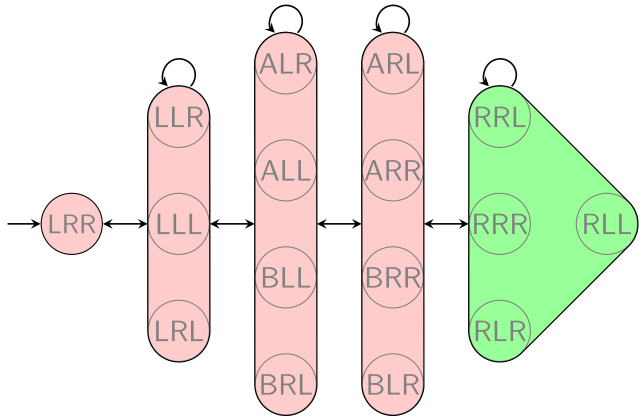
Pathological Case 2: Identity Abstraction



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

+ $h^\alpha = h^*$, trivial to compute α.
– Abstract state space = concrete state space.

Pathological Case 3: Perfect Abstraction



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

+ $h^\alpha = h^*$, usually very few abstract states.
– Computing α entails solving the optimal planning problem.

So, How to Obtain *Non*-Pathological Abstractions?

Covered in this course:

- Pattern database heuristics [Culberson and Schaeffer (1998); Edelkamp (2001); Haslum *et al.* (2007)]. → **Chapter 12**
- Merge-and-shrink abstractions [Dräger *et al.* (2006); Helmert *et al.* (2007); Katz *et al.* (2012); Helmert *et al.* (2014)]. → **Chapter 13**

Not covered in this course:

- Domain Abstractions, obtained by aggregating values within state variable domains [Hernádvölgyi and Holte (2000)]. Generalizes pattern database heuristics.
- Cartesian Abstractions, where abstract states are characterized by cross-products of state-variable-domain-subsets [Seipp and Helmert (2013)]. Generalizes domain abstractions.
- Structural patterns, where abstractions are implicitly represented [Katz and Domshlak (2008)].

The 15-Puzzle

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

—

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

→ Abstractions, in the context of AI, were first introduced in the form of pattern database heuristics for the 15-Puzzle. We now briefly review this from an FDR-planning perspective.

FDR-Style Encoding and Abstraction

The 15-Puzzle

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

$\rightarrow$ In other words, FDR state variables $= \{b, t_1, \dots, t_{15}\}$.

A 15-Puzzle Abstraction

One possible abstraction mapping α ignores the location of tiles $8, \dots, 15$. Two states are distinguished iff they differ in the position of the blank or one of the tiles $1, \dots, 7$:

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

The heuristic values for this abstraction roughly (see slide 33) correspond to the cost of moving tiles $1, \dots, 7$ to their goal positions.

Concrete vs. Abstract State Space

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

—

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

Concrete State Space: $16^{16} \approx 1.8 * 10^{19}$ states.

Concrete vs. Abstract State Space

	2		6
5	7		
3	4	1	

—

1	2	3	4
5	6	7	

Abstract State Space: $16^{8} \approx 4.2 * 10^{9}$ states.

The Abstract State Space in Detail

1	2	3	4
5	6	7	

Goal States

- Θ has the unique goal state $\langle 16, 1, 2, \dots, 15\rangle$.
- Θ^α has the unique goal state $\langle 16, 1, 2, \dots, 7\rangle$.

Transitions: Let x and y be neighboring positions in the 4×4 grid

- Θ 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 $i \in \{1, \dots, 15\}$.

  $\rightarrow$ In other words, FDR actions: pre $b = x, t_i = y$ eff $b = y, t_i = x$.
- Θ^α 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 $i \in \{1, \dots, 7\}$.

  $\rightarrow$ FDR: For $i \in \{1, \dots, 7\}$: pre $b = x, t_i = y$ eff $b = y, t_i = x$.
- Moreover, Θ^α has a transition from $\langle x, t_1, \dots, t_7\rangle$ to $\langle y, t_1, \dots, t_7\rangle$: These come from moves of a tile $j \in \{8, \dots, 15\}$.

  $\rightarrow$ FDR: pre $b = x$ eff $b = y$.

And How to Compute the Heuristic?

Computation of α

In this example, can α can be efficiently computed?

$\rightarrow$ Sure, just *project* the given 16-tuple onto its first 8 components.

$\rightarrow$ This heuristic is an example of a pattern database heuristic (where α is a projection).

Computation of Abstract Remaining Costs

To compute abstract remaining costs efficiently during search, most common algorithms precompute all abstract remaining costs prior to search, by a regression search on Θ^α. The distances are then stored in a lookup table.

$\rightarrow$ During search, computing $h^*_{\Theta^\alpha}(\alpha(s))$ is just a table lookup.

Multiple Abstractions

$\rightarrow$ There is a huge number of possible choices for α. This choice governs the informedness of the resulting heuristic function.

Example 15-Puzzle

The mapping to tiles $1, \dots, 7$ was arbitrary. We can use any subset of the tiles.

$\rightarrow$ There is no need to commit to a single α. We can *combine several* α.

Example 15-Puzzle

With the same amount of memory required for the lookup table for tiles $1, \dots, 7$ (16^8 states), we could store the lookup tables for 16 different abstractions to six tiles (16^7 states).

How to *Admissibly* Combine Multiple Abstractions?

Maximizing over 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 these.
- Thus, we can always compute several abstractions and maximize over the individual abstract goal distances.

Better idea: Summing over several abstractions!

- In some cases, the abstraction heuristics are additive (cf. Chapter 7): We can take their sum and still remain admissible.
- Summation often leads to much higher estimates than maximization, so it is important to understand when abstractions are additive.

Additive Abstractions: Example 15-Puzze

1	2	3	4
5	6	7	

			8
9	10	11	12
13	14		15

- 1st abstraction: Ignore location of $8, \dots, 15$.
- 2nd abstraction: Ignore location of $1, \dots, 7$.

$\rightarrow$ The sum of the abstraction heuristics is not admissible.

Additive Abstractions: Example 15-Puzze

1	2	3	4
5	6	7	

			8
9	10	11	12
13	14		15

- 1st abstraction: Ignore location of $8, \ldots, 15$ and blank.
- 2nd abstraction: Ignore location of $1, \ldots, 7$ and blank.

$\rightarrow$ The sum of the abstraction heuristics is admissible.

Orthogonal Abstractions

Terminology: If $s = t$ in (s, l, t), then the transition is called a self-loop.

Definition (Affecting Transition Labels). *Let α be an abstraction of Θ, and let l be one of the labels in Θ. We say that l affects α if Θ^α has at least one non-self-loop transition labeled by l, i.e., if there exists a transition $(\alpha(s), l, \alpha(t))$ with $\alpha(s) \neq \alpha(t)$.*

$\rightarrow$ Here is a simple sufficient criterion for additivity:

Definition (Orthogonal Abstractions). *Let α_1 and α_2 be abstractions of Θ. We say that α_1 and α_2 are orthogonal if no label of Θ affects both α_1 and α_2.*

Orthogonal Abstractions: Example 15-Puzze

Reminder: A label affects α if it labels a non-self loop transition in Θ^α. We say that α_1 and α_2 are orthogonal if no label of Θ affects both α_1 and α_2.

	2		6
5	7		
3	4	1	

9		12	
		14	13
			11
15	10	8	

	2		6
5	7		
3	4	1	

9		12	
		14	13
			11
15	10	8	

$\rightarrow$ Are the left-hand side abstraction mappings α_{left} and α_{right} orthogonal? No. E.g., consider the action that moves the blank upwards here, mapping the current state s to state t. This transition is not a self-loop in either of the two abstractions: $\alpha_{\text{left}}(s) \neq \alpha_{\text{left}}(t)$ and $\alpha_{\text{right}}(s) \neq \alpha_{\text{right}}(t)$.

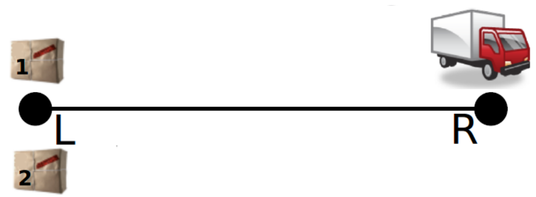
$\rightarrow$ Are the right-hand side abstraction mappings α_{left} and α_{right} orthogonal? Yes. Say a is any action that affects α_{left}. Then a moves a tile t_i for $i \in \{1, \ldots, 7\}$. Neither that t_i nor the blank are accounted for in α_{right} so a labels only self-loops there. Same vice versa.

Orthogonality and Additivity

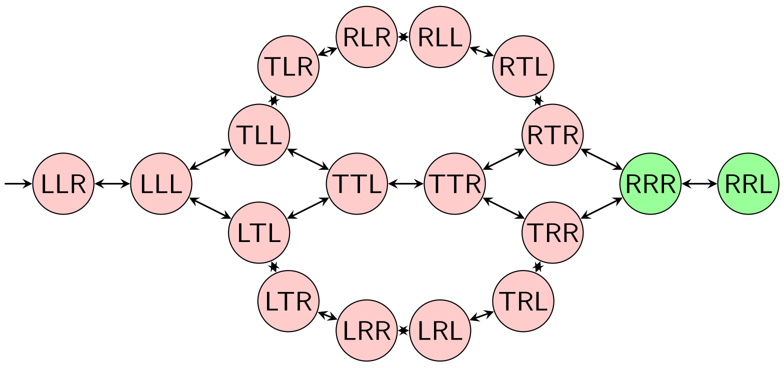
Theorem (Orthogonal Abstractions are Additive). *Let $\alpha_1, \ldots, \alpha_n$ be pairwise orthogonal abstractions for the same transition system Θ. Then $\sum_{i=1}^{n} h^{\alpha_i}$ is consistent and goal-aware, and thus also admissible and safe.*

$\rightarrow$ Intuition for admissibility: "Self-loops don't count." Every transition in an optimal solution path affects at most one of the abstractions, and thus is counted in at most one of the abstraction heuristics.

To illustrate the proof idea, we use yet another variant of "Logistics":

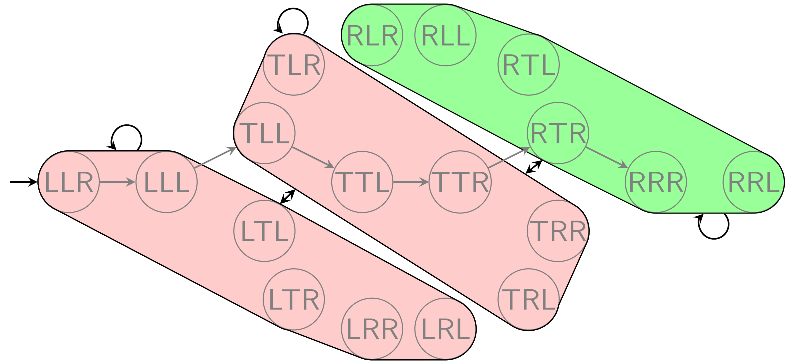


Orthogonality and Additivity: Illustration



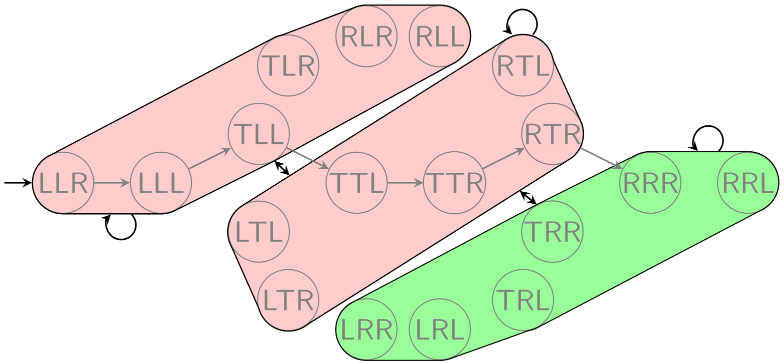
State space Θ. State variables: package 1, package 2, truck.

Orthogonality and Additivity: Illustration



Abstraction α_1.
Mapping: Only consider position of package 1.

Orthogonality and Additivity: Illustration



Abstraction α_2. (orthogonal to α_1)
Mapping: Only consider position of package 2.

Orthogonality and Additivity: Proof

Proof. Let $\Theta = (S, L, c, T, I, S^G)$.

For goal-awareness, we need to show that $\sum_{i=1}^{n} h^{\alpha_i}(s) = 0$ for all $s \in S^G$. So let $s \in S^G$. Then, for all i, $h^{\alpha_i}(s) = 0$ because h^{α_i} is goal aware.

For consistency, consider any state transition $(s, a, t) \in T$ in the concrete state space. We need to show that $\sum_{i=1}^{n} h^{\alpha_i}(s) \leq \sum_{i=1}^{n} h^{\alpha_i}(t) + c(a)$.

Because the abstraction mappings are orthogonal, $\alpha_i(s) \neq \alpha_i(t)$ for at most one $i \in \{1, \dots, n\}$. (Assume the opposite were true, and there were $i \neq j \in \{1, \dots, n\}$ s.t. $\alpha_i(s) \neq \alpha_i(t)$ and $\alpha_j(s) \neq \alpha_j(t)$. Then a labels a non-self-loop transition in both Θ^{α_i} and Θ^{α_j}, and thus α_i and α_j are not orthogonal, in contradiction.)

Orthogonality and Additivity: Proof, ctd.

Situation: Consider a concrete state transition $(s, a, t) \in T$. We need to show that $\sum_{i=1}^{n} h^{\alpha_i}(s) \leq \sum_{i=1}^{n} h^{\alpha_i}(t) + c(a)$.
We know that $\alpha_i(s) \neq \alpha_i(t)$ for at most one $i \in \{1, \dots, n\}$.

Case 1: $\alpha_i(s) = \alpha_i(t)$ for all i. **Then:**

$$\begin{aligned}\sum_{i=1}^{n} h^{\alpha_i}(s) &= \sum_{i=1}^{n} h^*_{\Theta^{\alpha_i}}(\alpha_i(s)) \\ &= \sum_{i=1}^{n} h^*_{\Theta^{\alpha_i}}(\alpha_i(t)) \text{ [because } \alpha_i(s) = \alpha_i(t)\text{]} \\ &= \sum_{i=1}^{n} h^{\alpha_i}(t) \\ &\leq \sum_{i=1}^{n} h^{\alpha_i}(t) + c(a).\end{aligned}$$

Case 2: $\alpha_k(s) \neq \alpha_k(t)$, and $\alpha_i(s) = \alpha_i(t)$ for $i \neq k$. **Then:**

$$\begin{aligned}\sum_{i=1}^{n} h^{\alpha_i}(s) &= \sum_{i \neq k} h^*_{\Theta^{\alpha_i}}(\alpha_i(s)) + h^{\alpha_k}(s) \\ &= \sum_{i \neq k} h^*_{\Theta^{\alpha_i}}(\alpha_i(t)) + h^{\alpha_k}(s) \text{ [}\alpha_i(s) = \alpha_i(t) \text{ for } i \neq k\text{]} \\ &\leq \sum_{i \neq k} h^*_{\Theta^{\alpha_i}}(\alpha_i(t)) + h^{\alpha_k}(t) + c(a) \text{ [}h^{\alpha_k} \text{ is consistent]} \\ &= \sum_{i=1}^{n} h^{\alpha_i}(t) + c(a).\end{aligned}$$

Questionnaire

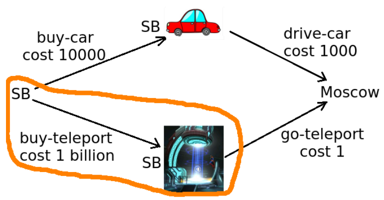
→ Are optimal abstract plans just abstractions of optimal real plans?

The situation: Assume an FDR planning task Π, a state s, and an optimal plan $\vec{a}$ for s in Π. Say α is an abstraction, and say we obtain $\vec{a}^\alpha$ from $\vec{a}$ by *removing all actions that do not affect* α.

Question!

Is $\vec{a}^\alpha$ necessarily an optimal abstract plan, i.e., $\sum_{a \in \vec{a}^\alpha} c(a) = h^\alpha(s)$?

→ No! Spurious transitions may lead to "shortcuts" that do not correspond to an optimal real plan, or to any plan at all. **Example:**



Car: cost 10000 (buy) + 1000 (go); teleport: cost 1 billion (buy) + 1 (go). If α does not distinguish between I and the state where we have the teleport, then I has a spurious cost-1 transition to Moscow, and the only optimal abstract plan uses that transition.

Abstractions of Abstractions

Proposition (Transitivity of Abstractions). *Let Θ be a transition system. If α is an abstraction of Θ and α' is an abstraction of Θ^α, then $\alpha' \circ \alpha$ is an abstraction of Θ.*

Proof. All we need to prove is that $\alpha' \circ \alpha$ is surjective. This follows directly from surjectivity of α and α'.

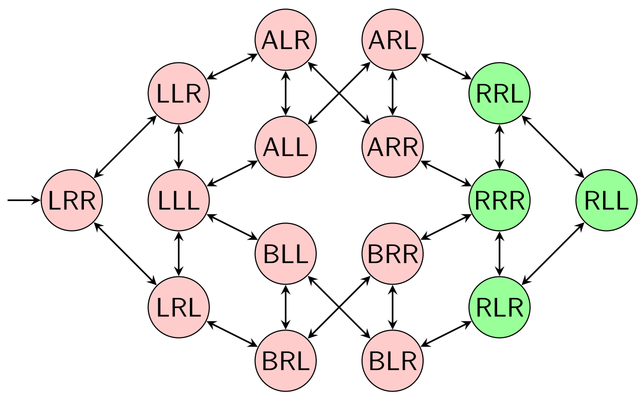
Terminology: Let Θ be a transition system, α an abstraction of Θ, and α' an abstraction of Θ^α. Then:

- $\alpha' \circ \alpha$ is called a coarsening of α.
- α is called a refinement of $\alpha' \circ \alpha$.

→ Abstractions are often obtained by incrementally refining or coarsening some initial abstraction until a termination criterion applies.

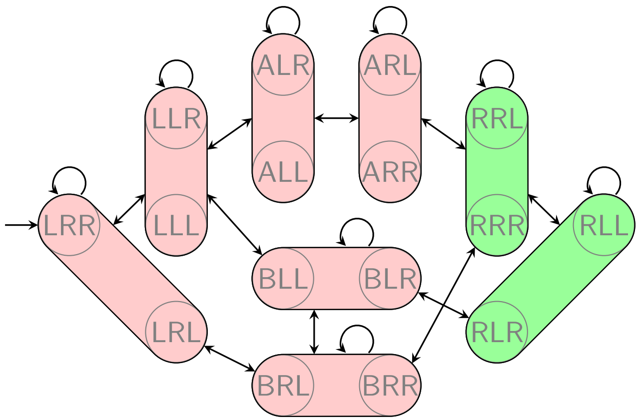
→ E.g., merge-and-shrink (Chapter 13), and abstraction refinement in Verification.

Abstractions of Abstractions: Illustration



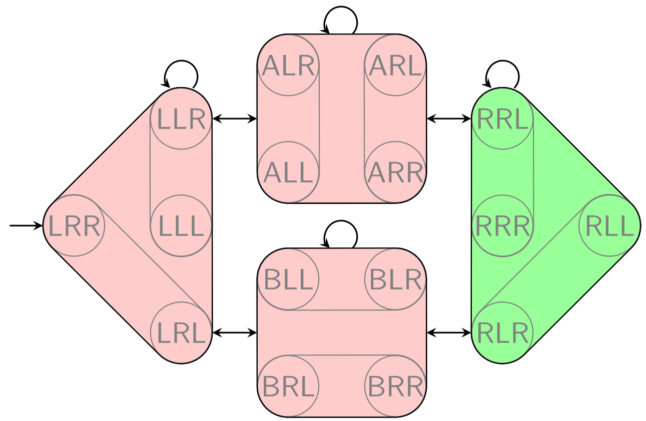
Transition system Θ.

Abstractions of Abstractions: Illustration



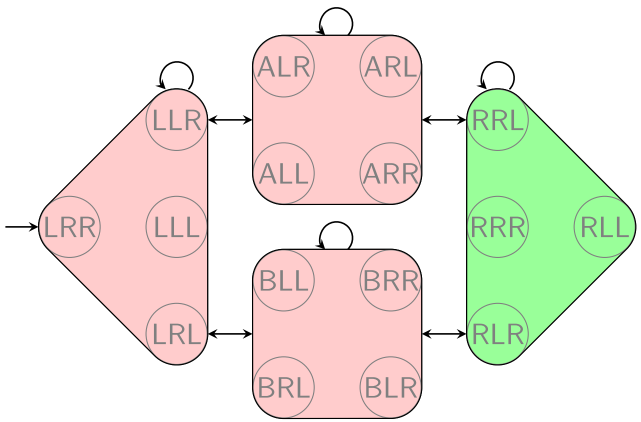
Transition system Θ^{α} as an abstraction of Θ.

Abstractions of Abstractions: Illustration



Transition system $\Theta^{\alpha' \circ \alpha}$ as an abstraction of Θ^{α}.

Abstractions of Abstractions: Illustration



Transition system $\Theta^{\alpha' \circ \alpha}$ as an abstraction of Θ.

Refinements Improve the Heuristic

Theorem (Refinements Improve the Heuristic). *Let h^{α} and $h^{\alpha''}$ be abstraction heuristics of Θ, such that α is a refinement of α''. Then h^{α} dominates $h^{\alpha''}$, i.e., $h^{\alpha''} \leq h^{\alpha}$.*

Proof. Since α is a refinement of α'', there exists a mapping α' such that $\alpha'' = \alpha' \circ \alpha$. For any state s, we get

$$
\begin{aligned}
h^{\alpha''}(s) &= h^*_{\Theta^{\alpha''}}(\alpha''(s)) \\
&= h^*_{\Theta^{\alpha''}}(\alpha'(\alpha(s))) \\
&= h^{\alpha'}(\alpha(s)) \\
&\leq h^*_{\Theta^{\alpha}}(\alpha(s)) \\
&= h^{\alpha}(s),
\end{aligned}
$$

where the inequality holds because $h^{\alpha'}$ is an admissible heuristic in the transition system Θ^{α}.

$\rightarrow$ If we start from abstraction α and then abstract less, we can only improve the lower bound (h values), relative to h^{α}.

Summary

- An abstraction α is a surjective function on a transition system Θ (e.g., of a planning task).
- The abstract state space Θ^α inherits the initial state, goal states, and transitions from Θ; it is isomorphic to the quotient system $\Theta/\sim^\alpha$ of Θ under the equivalence relation $\sim^\alpha$ induced by α.
- Remaining cost in Θ^α is the abstraction heuristic h^α, which is safe, goal-aware, admissible, and consistent.
- The heuristics of orthogonal abstractions are additive, i.e., their sum is admissible (cf. Chapter 7).
- A coarsening of an abstraction α is an abstraction α'' of α, i.e., $\alpha'' = \alpha' \circ \alpha$; in this situation, α is a refinement of α'', and $h^\alpha \geq h^{\alpha''}$.
- Practically useful abstractions yield informative heuristics at a small computational overhead.

  The state of the art to accomplish this are pattern databases $\rightarrow$ Chapter 12, and merge-and-shrink abstractions $\rightarrow$ Chapter 13.

References I

Joseph C. Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

Klaus Dräger, Bernd Finkbeiner, and Andreas Podelski. Directed model checking with distance-preserving abstractions. In Antti Valmari, editor, *Proceedings of the 13th International SPIN Workshop (SPIN 2006)*, volume 3925 of *Lecture Notes in Computer Science*, pages 19–34. Springer-Verlag, 2006.

Stefan Edelkamp. Planning with pattern databases. In A. Cesta and D. Borrajo, editors, *Proceedings of the 6th European Conference on Planning (ECP'01)*, pages 13–24. Springer-Verlag, 2001.

Patrik Haslum, Adi Botea, Malte Helmert, Blai Bonet, and Sven Koenig. Domain-independent construction of pattern database heuristics for cost-optimal planning. In Adele Howe and Robert C. Holte, editors, *Proceedings of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI'07)*, pages 1007–1012, Vancouver, BC, Canada, July 2007. AAAI Press.

References II

Malte Helmert, Patrik Haslum, and Jörg Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In Mark Boddy, Maria Fox, and Sylvie Thiebaux, editors, *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, pages 176–183, Providence, Rhode Island, USA, 2007. Morgan Kaufmann.

Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the Association for Computing Machinery*, 61(3), 2014.

István T. Hernádvölgyi and Robert C. Holte. Experiments with automatically created memory-based heuristics. In Berthe Y. Choueiry and Toby Walsh, editors, *Proceedings of the 4th International Symposium on Abstraction, Reformulation and Approximation (SARA 2000)*, volume 1864 of *Lecture Notes in Artificial Intelligence*, pages 281–290. Springer-Verlag, 2000.

Michael Katz and Carmel Domshlak. Structural patterns heuristics via fork decomposition. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric Hansen, editors, *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, pages 182–189. AAAI Press, 2008.

References III

Michael Katz, Jörg Hoffmann, and Malte Helmert. How to relax a bisimulation? In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 101–109. AAAI Press, 2012.

Jendrik Seipp and Malte Helmert. Counterexample-guided Cartesian abstraction refinement. In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, pages 347–351, Rome, Italy, 2013. AAAI Press.