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

1. Introduction
2. Abstraction Basics
3. Practical vs. Pathological Abstractions
4. A Prominent Example: The 15-Puzzle
5. Additive Abstractions
6. Abstraction Refinements
7. Conclusion
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
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 \( \Theta \) is defined by a function \( \alpha \) (the abstraction mapping), mapping states to abstract states (also block states).

- If \( \alpha \) maps states \( s \) and \( t \) to the same abstract state, then \( s \) and \( t \) are not distinguished anymore (they are equivalent under \( \alpha \)).

- The abstract transition system \( \Theta^\alpha \) on the image of \( \alpha \) is defined by homomorphically mapping over all goal states and transitions from \( \Theta \), and thus preserving all solutions.

- The abstract remaining cost, i.e., remaining cost in \( \Theta^\alpha \), is an estimate \( h^\alpha \) for remaining cost in \( \Theta \). As we preserve all solutions, \( h^\alpha \) 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
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 $\Pi = \text{abstraction of its state space } \Theta_\Pi$.
- The results in this Chapter apply to arbitrary $\Theta$.
- The results of Chapters 12 and 13 are specific to FDR. They exploit the compact representation of $\Theta = \Theta_\Pi$ via $\Pi$ in order to build the abstract state space effectively.
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 $D_{t_A} = D_{t_B} = \{L, R\}$.

- $A = \{\text{pickup}(x, y) \mid x \in \{A, B\}, y \in \{L, R\}\}$
  $\cup \{\text{drop}(x, y) \mid x \in \{A, B\}, y \in \{L, R\}\}$
  $\cup \{\text{move}(x, y, y') \mid x \in \{A, B\}, y, y' \in \{L, R\}, y \neq y'\}$, with
  - $\text{pre}_{\text{pickup}}(x, y): \ t_x = y, p = y$; $\text{eff}_{\text{pickup}}(x, y): \ p = x$;
  - $\text{pre}_{\text{drop}}(x, y): \ t_x = y, p = x$; $\text{eff}_{\text{drop}}(x, y): \ p = y$;
  - $\text{pre}_{\text{move}}(x, y, y'): \ t_x = y$; $\text{eff}_{\text{move}}(x, y, y'): \ t_x = y'$.

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

(i) $I^\alpha = \alpha(I)$.

(ii) $S^{\alpha G} = \{\alpha(s) \mid s \in S^G\}$. /* preserve goal states */

(iii) $T^\alpha = \{(\alpha(s), l, \alpha(t)) \mid (s, l, t) \in T\}$. /* preserve transitions */

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

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

Why do we require $\alpha$ to be surjective?
Abstractions: “Logistics mal anders”

Concrete transition system:
→ A transition between concrete states is “spurious” if it exists in the abstract but not in the concrete state space. Example here?
Definition (Abstraction Heuristic). Let $\Theta = (S, L, c, T, I, S^G)$ be a transition system, and let $\alpha$ be an abstraction of $\Theta$. The abstraction heuristic induced by $\alpha$, written $h^{\alpha}$, 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 $\Theta^\alpha$.

→ The abstract remaining cost (remaining cost in $\Theta^\alpha$) is used as the heuristic estimate for remaining cost in $\Theta$.

→ $h^{\alpha}(s) = \infty$ if no goal state of $\Theta^\alpha$ 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
Abstractions as Quotient Systems

Alternate views: (a) transition systems \( \Theta^\alpha \) 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 \( \Theta \), 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 \( \Theta \). Then \( \Theta/\sim^\alpha \) is isomorphic to \( \Theta^\alpha \).
(Direct from definition.)
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 \( \alpha \) 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
Questionnaire, ctd.

- Variables: $at : \{Sy, Ad, Br, Pe, Ad\}$; $v(x) : \{T, F\}$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- Actions: $\text{drive}(x, y)$ 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 $\alpha$ projects this task onto $\{v(Pe), v(Da)\}$. What is $h^\alpha(I)$?

- (A): 2
- (B): 7.5
- (C): 12.5
- (D): 14
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 $\alpha$ 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”? 

Pathological Case 1:
Pathological Case 2:
Pathological Case 3:
So, How to Obtain \textit{Non}-Pathological Abstractions?

\textbf{Covered in this course:}

- \textbf{Pattern database heuristics} [Culberson and Schaeffer (1998); Edelkamp (2001); Haslum \textit{et al.} (2007)]. \rightarrow \textbf{Chapter 12}


\textbf{Not covered in this course:}

- \textbf{Domain Abstractions}, obtained by aggregating values within state variable domains [Hernádvölgyi and Holte (2000)]. Generalizes pattern database heuristics.

- \textbf{Cartesian Abstractions}, where abstract states are characterized by cross-products of state-variable-domain-subsets [Seipp and Helmert (2013)]. Generalizes domain abstractions.

- \textbf{Structural patterns}, where abstractions are implicitly represented [Katz and Domshlak (2008)].
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.
The 15-Puzzle

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

→ In other words, FDR state variables = \( \{b, t_1, \ldots, t_{15}\} \).

A 15-Puzzle Abstraction

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

\[
\alpha(\langle b, t_1, \ldots, t_{15} \rangle) :=
\]

The heuristic values for this abstraction roughly (see slide 33) correspond to the cost of moving tiles 1, \ldots, 7 to their goal positions.
Concrete vs. Abstract State Space

Concrete State Space:
Concrete vs. Abstract State Space

Abstract State Space:
Goal States

- $\Theta$ has the unique goal state $\langle 16, 1, 2, \ldots, 15 \rangle$.
- $\Theta^\alpha$ has the unique goal state.

Transitions: Let $x$ and $y$ be neighboring positions in the $4 \times 4$ grid

- $\Theta$ has a transition from $\langle x, t_1, \ldots, t_{i-1}, y, t_{i+1}, \ldots, t_{15} \rangle$ to $\langle y, t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_{15} \rangle$ for all $i \in \{1, \ldots, 15\}$.
- In other words, FDR actions: pre $b = x$, $t_i = y$, eff $b = y$, $t_i = x$.
And How to Compute the Heuristic?

**Computation of $\alpha$**

In this example, can $\alpha$ can be efficiently computed?

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

→ This heuristic is an example of a *pattern database heuristic* (where $\alpha$ 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 $\Theta^\alpha$. The distances are then stored in a *lookup table*.

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

→ There is a huge number of possible choices for \( \alpha \). This choice governs the informedness of the resulting heuristic function.

Example 15-Puzzle

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

→ There is no need to commit to a single \( \alpha \). We can combine several \( \alpha \).
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-Puzzle

- **1st abstraction**: Ignore location of 8, ..., 15.
- **2nd abstraction**: Ignore location of 1, ..., 7.

→ The sum of the abstraction heuristics is
Additive Abstractions: Example 15-Puzzle

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

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

→ The sum of the abstraction heuristics is
Orthogonal Abstractions

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

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

→ Here is a simple sufficient criterion for additivity:

**Definition (Orthogonal Abstractions).** Let \( \alpha_1 \) and \( \alpha_2 \) be abstractions of \( \Theta \). We say that \( \alpha_1 \) and \( \alpha_2 \) are orthogonal if no label of \( \Theta \) affects both \( \alpha_1 \) and \( \alpha_2 \).
### Orthogonal Abstractions: Example 15-Puzzle

**Reminder:** A label affects $\alpha$ if it labels a non-self loop transition in $\Theta^\alpha$. We say that $\alpha_1$ and $\alpha_2$ are orthogonal if no label of $\Theta$ affects both $\alpha_1$ and $\alpha_2$.

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

→ Are the left-hand side abstraction mappings $\alpha_{\text{left}}$ and $\alpha_{\text{right}}$ orthogonal?

→ Are the right-hand side abstraction mappings $\alpha_{\text{left}}$ and $\alpha_{\text{right}}$ orthogonal?
Orthogonality and Additivity

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

→ 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 $\Theta$. State variables: package 1, package 2, truck.
Orthogonality and Additivity: Illustration

Abstraction $\alpha_1$.
Mapping: Only consider position of package 1.
Orthogonality and Additivity: Illustration

Abstraction $\alpha_2$. (orthogonal to $\alpha_1$)

Mapping: Only consider position of package 2.
Orthogonality and Additivity: Proof
Orthogonality and Additivity: Proof, ctd.
→ Are optimal abstract plans just abstractions of optimal real plans?

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

**Question!**

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

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

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

**Terminology:** Let $\Theta$ be a transition system, $\alpha$ an abstraction of $\Theta$, and $\alpha'$ an abstraction of $\Theta^\alpha$. Then:

- $\alpha' \circ \alpha$ is called a **coarsening** of $\alpha$.
- $\alpha$ 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 $\Theta$. 
Abstractions of Abstractions: Illustration

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

Transition system $\Theta^{\alpha' \circ \alpha}$ as an abstraction of $\Theta$. 
Theorem (Refinements Improve the Heuristic). Let $h^\alpha$ and $h^{\alpha''}$ be abstraction heuristics of $\Theta$, such that $\alpha$ is a refinement of $\alpha''$. Then $h^\alpha$ dominates $h^{\alpha''}$, i.e., $h^{\alpha''} \leq h^\alpha$.

Proof. Since $\alpha$ is a refinement of $\alpha''$, there exists a mapping $\alpha'$ such that $\alpha'' = \alpha' \circ \alpha$. For any state $s$, we get
\[
\begin{align*}
    h^{\alpha''}(s) &= h^*_{\Theta^{\alpha''}}(\alpha''(s)) \\
                     &= h^*_{\Theta^{\alpha''}}(\alpha'(\alpha(s))) \\
                     &= h^{\alpha'}(\alpha(s)) \\
                     &\leq h^*_{\Theta^{\alpha}}(\alpha(s)) \\
                     &= h^{\alpha}(s),
\end{align*}
\]
where the inequality holds because $h^{\alpha'}$ is an admissible heuristic in the transition system $\Theta^{\alpha}$.

→ If we start from abstraction $\alpha$ and then abstract less, we can only improve the lower bound ($h$ values), relative to $h^\alpha$. 
Summary

- An abstraction $\alpha$ is a surjective function on a transition system $\Theta$ (e.g., of a planning task).

- The abstract state space $\Theta^\alpha$ inherits the initial state, goal states, and transitions from $\Theta$; it is isomorphic to the quotient system $\Theta/\sim^\alpha$ of $\Theta$ under the equivalence relation $\sim^\alpha$ induced by $\alpha$.

- Remaining cost in $\Theta^\alpha$ is the abstraction heuristic $h^\alpha$, 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 $\alpha$ is an abstraction $\alpha''$ of $\alpha$, i.e., $\alpha'' = \alpha' \circ \alpha$; in this situation, $\alpha$ is a refinement of $\alpha''$, 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


