

Automatic Planning

Jörg Hoffmann

University of Innsbruck
DERI

Masterstudium Informatik Vertiefungsmodul (WM 9)

- ▶ **Patterns and Databases**
- ▶ Domain Abstractions
- ▶ Pattern Databases
- ▶ Disjoint Pattern Databases
- ▶ Pattern Databases in STRIPS

The “Pattern Database” Relaxation: Idea

- ▶ Ignore some aspects of world/system states
- ▶ That is, take into account only a *pattern* of the real structure
- ▶ More formally, define a mapping abs of the real problem into an **abstract** problem with a smaller state space (in a way so that paths between states can only get shorter \Rightarrow admissibility)
- ▶ To generate the heuristic value, solve the abstract problem optimally
- ▶ Do this just once before search starts, and store state values in a lookup table \Rightarrow **database**

Example 8-Puzzle

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & & 6 \\ \hline 7 & 8 & 9 \\ \hline \end{array} \xrightarrow{\text{abs}} \begin{array}{|c|c|c|} \hline x & x & x \\ \hline 4 & & x \\ \hline 7 & x & x \\ \hline \end{array}$$

- ▶ Replace all but a subset of the tiles with x
- ▶ That is, do not distinguish between these tiles
- ▶ Every solution is also an abstract solution

Example 8-Puzzle

How to get from left to right?

1	2	3	goal \Rightarrow	1	2	3
6		9		4		6
4	7	8		7	8	9

x	x	x	goal \Rightarrow	x	x	x
x		x		4		x
4	7	x		7	x	x

Example 8-Puzzle

There are different abstraction functions:

x	x	x	goal \Rightarrow	x	x	x
x		9		x		x
x	x	8		x	8	9

x	x	x	goal \Rightarrow	x	x	x
x		x		4		x
4	x	8		x	8	x

Example 8-Puzzle

Abstraction functions can be more complex:

x	x	x	abs	x	x	x	
y		9		→	x		y
x	y	8			y	8	9

x	x	x	abs	x	x	x	
y		y		→	x		y
x	y	y			y	y	y

Example 8-Puzzle

A pattern database:

	X	X
X	X	X
X	X	X

2

X		X
X	X	X
X	X	X

1

X	X	
X	X	X
X	X	X

2

X	X	X
	X	X
X	X	X

1

X	X	X
X		X
X	X	X

0

X	X	X
X	X	
X	X	X

1

X	X	X
X	X	X
	X	X

2

X	X	X
X	X	X
X		X

1

X	X	X
X	X	X
X	X	

2

The “Pattern Database” Relaxation: Observations

There are many possible abstraction functions.

- ▶ Different abstraction functions yield different heuristics
- ▶ The less coarse the abstraction, the more precise the heuristic
- ▶ Extreme cases: everything/nothing abstracted away – *what happens in these cases, respectively?*
- ▶ We can combine different heuristics by taking the *maximum*, preserving admissibility

The “Pattern Database” Relaxation: Observations

We don't map the real problem into a computationally simpler problem, but only reduce the (instance) size parameters.

- ▶ Abstract state space can be large, no point in solving this in every search state
- ▶ Instead, create entire abstract state space (pattern database) as pre-process
- ▶ We can keep several databases in memory

- ▶ Patterns and Databases
- ▶ **Domain Abstractions**
- ▶ Pattern Databases
- ▶ Disjoint Pattern Databases
- ▶ Pattern Databases in STRIPS

Multiple-valued STRIPS (MVSTRIPS)

- ▶ Set V of variables v with finite domain $dom(v)$; to simplify presentation, we assume $dom(v) = dom(v') =: dom$ for all $v, v' \in V$ (extension to different domains trivial)
- ▶ States are value assignments to the variables
- ▶ Actions a have a precondition
 $pre(a) = \{v_1 = c_1, \dots, v_k = c_k\}$
- ▶ ... and an effect $eff(a) = \{v'_1 := c'_1, \dots, v'_{k'} := c'_{k'}\}$
- ▶ Goal formula $G = \{v_1 = c_1, \dots, v_m = c_m\}$

Example 8-Puzzle

1	2	3	goal \rightarrow	1	2	3
6		9		4		6
4	7	8		7	8	9

(Different encoding than previously: variables are fields, values are tiles.)

- ▶ Variables: $cont(p_i)$ for $1 \leq i \leq 9$,
 $dom = \{blank, t_1, t_2, t_3, t_4, t_6, t_7, t_8, t_9\}$
- ▶ Actions: e.g., $mv(t_6, p_4, p_5) : cont(p_4) = t_6 \wedge cont(p_5) = blank \rightarrow cont(p_4) := blank, cont(p_5) := t_6$
- ▶ Initial, goal: $cont(p_1) = t_1, \dots$

(Variable) Domain Abstractions

Definition

Let (V, A, I, G) be an MVSTRIPS task with variable domain dom . A **domain abstraction** is a function $^{abs} : dom \mapsto absdom$ where $|absdom| \leq |dom|$.

- ▶ We map dom into a set $absdom$ that allows less distinctions
- ▶ abs is extended to conditions/effects/actions/tasks by applying it to the value identifiers inside the more complex constructs
- ▶ E.g. $\{v_1 = c_1, \dots, v_k = c_k\}^{abs} = \{v_1 = c_1^{abs}, \dots, v_k = c_k^{abs}\}$

Example 8-Puzzle

1	2	3	$\xrightarrow{\text{abs}}$	x	x	x
6		9		x		x
4	7	8		4	7	x

- ▶ $dom = \{blank, t_1, t_2, t_3, t_4, t_6, t_7, t_8, t_9\}$,
 $absdom = \{blank, x, t_4, t_7\}$
- ▶ $\{blank, t_1, t_2, t_3, t_4, t_6, t_7, t_8, t_9\}^{abs} =$
 $\{blank, x, x, x, t_4, x, t_7, x, x\}$
- ▶ Actions: e.g., $mv(t_6, p_4, p_5)^{abs} : cont(p_4) = x \wedge$
 $cont(p_5) = blank \rightarrow cont(p_4) := blank, cont(p_5) := x$
- ▶ Initial, goal: $cont(p_1) = x, \dots$

Proposition

Let (V, A, I, G) be an MVSTRIPS task, abs a domain abstraction. Let s be a state in the task, a an action applicable to s . Then a^{abs} is applicable in s^{abs} , and

$$result(s, \langle a \rangle)^{abs} = result(s^{abs}, \langle a^{abs} \rangle)$$

Proof.

Blackboard. □

So abs is a **homomorphism**.

Proposition

Let (V, A, I, G) be an MVSTRIPS task, abs a domain abstraction. Let s_0 and s_n be states in the task. Let n be the length of a shortest path between s_0 and s_n . Let n^{abs} be the length of a shortest path between s_0^{abs} and s_n^{abs} . Then $n^{abs} \leq n$.

Proof.

Do you see why?



State Space Homomorphisms

It may happen that there is a shorter path than $\langle a_1^{abs}, \dots, a_n^{abs} \rangle$.
Denote by s_0, s_1, \dots, s_n the states on the sequence.

E.g. shortcuts $s_j^{abs} = result(s_i^{abs}, \langle a^{abs} \rangle)$, $j > i + 1$, for an action a not in the sequence

$s_0, mv(6, p_4, p_5), s_1, mv(4, p_7, p_4), s_2, mv(7, p_8, p_7),$
 $s_3, mv(8, p_9, p_8), s_4, mv(9, p_6, p_9), s_5, mv(6, p_5, p_6), s_6$

1	2	3		1	2	3
6		9	goal	4		6
4	7	8		7	8	9

x	x	x		x	x	x
x		x	goal	4		x
4	7	x		7	x	x

Where does this happen here?

- ▶ Patterns and Databases
- ▶ Domain Abstractions
- ▶ **Pattern Databases**
- ▶ Disjoint Pattern Databases
- ▶ Pattern Databases in STRIPS

Definition

Let (V, A, l, G) be an MVSTRIPS task, abs a domain abstraction. For a state s in the state space of the task, we define $h^{abs}(s)$ as the (forward) solution distance $sd(s^{abs})$ of s^{abs} in $(V, A^{abs}, l^{abs}, G^{abs})$. The set of all $(s^{abs}, h^{abs}(s))$ pairs is called a *pattern database*.

- ▶ We have seen that h^{abs} is an admissible heuristic function
- ▶ The pattern database is a table-representation of that function
- ▶ The definition is targeted to forward search; a dual definition is valid for backward search
- ▶ A forward database can be re-used for several ???
A backward database can be re-used for several ???

“Solve” ($V, A^{abs}, I^{abs}, G^{abs}$):

- ▶ Do a standard breadth-first forward search from I^{abs}
- ▶ Using a closed list, i.e. generating each state only once
- ▶ When explicit state space is built, compute the solution distances
- ▶ Multiple-source shortest paths problem: simple generalization of Dijkstra

Building Pattern Databases

We can avoid the need to visit the abstract state space twice:
Do a standard breadth-first *backward* search from G^{abs} , using a closed list.

Problem: the generated sub-goals correspond to *sets* of world states.

- ▶ Solution 1: enumerate the represented sets of states instead of just $at(P, Right)$, say $at(P, Right) \wedge at(T, Left)$, $at(P, Right) \wedge at(T, Right)$ – larger search space
- ▶ Solution 2: during search, take $h^{abs}(s)$ to be the *minimum* over the sd of all abstract sub-goals that are satisfied by s^{abs} – more complicated indexing

Unique-State Sub-goals

Sometimes the sub-goals always identify a unique single world state – backward building unproblematic.

How about the 8-puzzle?

1	2	3	abs →	x	x	x
4		6		4		x
7	8	9		7	x	x

Do you see a general rule?

What about a syntactic sufficient criterion?

- ▶ Patterns and Databases
- ▶ Domain Abstractions
- ▶ Pattern Databases
- ▶ **Disjoint Pattern Databases**
- ▶ Pattern Databases in STRIPS

Add'ing vs. Max'ing

- ▶ We can keep several databases, and *max* over *h* values
- ▶ It would be much better to *add* the *h* values
- ▶ *Adding h values is not admissible in general*
- ▶ Well ...
- ▶ Can we identify cases where it *is* admissible?
- ▶ Can we do something about the other cases?

Example 8-Puzzle

Is adding admissible here? (two abstractions of our running example)

x	x	x	goal \Rightarrow	x	x	x
x		9		x		x
x	x	8		x	8	9

x	x	x	goal \Rightarrow	x	x	x
x		x		4		x
4	x	8		x	8	x

Definition

Let (V, A, I, G) be an MVSTRIPS task. Let abs1 and abs2 be domain abstractions. abs1 and abs2 are said to be **disjoint** if, for every state s , $h^{abs1}(s) + h^{abs2}(s) \leq sd(s)$.

- ▶ We use the word “disjoint” also for the respective pattern databases
- ▶ Finding out, in general, if two pattern databases are disjoint, is hard (I didn’t find an easy proof quickly but this just can’t be polynomial)
- ▶ We can identify a simple sufficient criterion
- ▶ “disjoint” also called “additive” in literature

Definition

Let (V, A, I, G) be an MVSTRIPS task. An action $a \in A$ is said to be *void*, if $\forall (v := c) \in \text{eff}(a) : \exists (v = c) \in \text{pre}(a)$.

- ▶ Void actions do not change the state
- ▶ You wouldn't expect a void action in a real task – but in an *abstracted* task!
- ▶ Example: if the abstraction maps both tile t and the blank to a symbol x , then moving t is void
- ▶ $mv(t, p_j, p_{j'})^{abs} = ???$

Proposition

Let (V, A, I, G) be an MVSTRIPS task. Let $abs1$ and $abs2$ be domain abstractions. If there is no $a \in A$ that is non-void in both $(V, A^{abs1}, I^{abs1}, G^{abs1})$ and $(V, A^{abs2}, I^{abs2}, G^{abs2})$, then $abs1$ and $abs2$ are disjoint.

Proof.

Blackboard.



Example 8-Puzzle

Are these two disjoint?

1	2	3	abs →	1	2	3
4		6		4		x
7	8	9		x	x	x

1	2	3	abs →	x	x	x
4		6		x		6
7	8	9		7	8	9

Example 8-Puzzle

And these two?

1	2	3	abs →	1	2	3
4		6		4	x	x
7	8	9		x	x	x

1	2	3	abs →	x	x	x
4		6		x	x	6
7	8	9		7	8	9

“Disjoining” Pattern Databases

- ▶ If we have two pattern databases that share the non-void actions $A_{1,2}$, then an option is to *count* $a \in A_{1,2}$ *only in one of the pattern databses*
- ▶ When building a database, set edge costs for non-counted actions to 0
- ▶ With several databases, count every action only in one of the databases in which it is non-void
- ▶ The resulting h values can be added safely. *Proof?*

- ▶ Patterns and Databases
- ▶ Domain Abstractions
- ▶ Pattern Databases
- ▶ Disjoint Pattern Databases
- ▶ **Pattern Databases in STRIPS**

Pattern Databases in STRIPS

- ▶ In STRIPS we have only Boolean variables: $\{0, 1\}$
- ▶ There are just two different domain abstractions:
 - ▶ *What are they?*
 - ▶ *What heuristics do they induce?*
- ▶ We will have to use a different abstraction technique
- ▶ A natural adaptation for linear encoding exists (this is mentioned in [Hoffmann et al, ICAPS'06])
- ▶ I used to teach that but it's complicated
- ▶ Let's do something simpler

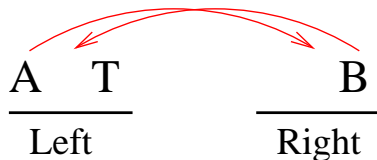
Definition

Let (P, A, I, G) be a STRIPS task, and let $\mathcal{P} \subseteq P$. The *abstraction of (P, A, I, G) to \mathcal{P}* is the task (P', A', I', G') where $P' = P \cap \mathcal{P}$, $I' = I \cap \mathcal{P}$, $G' = G \cap \mathcal{P}$, and

$$A' = \{(pre \cap \mathcal{P}, add \cap \mathcal{P}, del \cap \mathcal{P}) \mid (pre, add, del) \in A\}$$

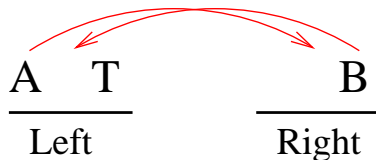
- ▶ We simply remove everything outside \mathcal{P}
- ▶ Introduced by [Edelkamp, ECP'01]
- ▶ Intuition: replace **all** values of some variables with x

Example Logistics



- ▶ Facts: $at(A, Left)$, $at(A, Right)$, $in(A, T)$, $at(B, Left)$, $at(B, Right)$, $in(B, T)$, $at(T, Left)$, $at(T, Right)$
- ▶ $\mathcal{P} := at(A, Left)$, $at(A, Right)$, $in(A, T)$
- ▶ “Abstract Facts”: $at(A, Left)$, $at(A, Right)$, $in(A, T)$, x , x , x , x , x
- ▶ All values of $position(B)$ and $position(T)$ ignored!

Example Logistics



- ▶ Facts: $at(A, Left)$, $at(A, Right)$, $in(A, T)$, $at(B, Left)$, $at(B, Right)$, $in(B, T)$, $at(T, Left)$, $at(T, Right)$
- ▶ $\mathcal{P} := at(A, Left)$, $at(A, Right)$, $in(A, T)$
- ▶ $\Rightarrow drive(T, Left, Right) = ?$
- ▶ $\Rightarrow load(B, T, Right) = ?$
- ▶ $\Rightarrow load(A, T, Right) = ?$
- ▶ $\Rightarrow unload(A, T, Right) = ?$

Definition

Let (P, A, I, G) be a STRIPS task, let $\mathcal{P} \subseteq P$, and let (P', A', I', G') be the abstraction of (P, A, I, G) to \mathcal{P} . Let s be a state in (P, A, I, G) . By $h^{\mathcal{P}}(s)$, we denote the length of a shortest path in (P', A', I', G') from $s \cap \mathcal{P}$ to a state s_g where $s_g \supseteq G'$.

Proposition

Let (P, A, I, G) be a STRIPS task and let $\mathcal{P} \subseteq P$. Let s be a state in (P, A, I, G) . Then $h^{\mathcal{P}}(s) \leq sd(s)$.

Proof.

Exercise. □

Edelkamp's Abstraction, ctd.

- ▶ We can in principle ignore any set of facts, not necessarily all values of one multi-valued variable
- ▶ If we ignore only a subset of values, then that abstracts more than if we only replace that subset with x : *why?*
- ▶ All existing methods detect the multi-valued variables underlying a STRIPS task, and abstract a subset of those
- ▶ “Real” variable domain abstraction is more fine-grained, but is not yet being used
- ▶ Potentially promising
- ▶ Some people argue that disjointness is easier to obtain using Edelkamp:

Proposition

Let (P, A, I, G) be a STRIPS task, and let $\mathcal{P}_1, \mathcal{P}_2 \subseteq P$ such that $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$, and for all $a \in A$ either $(add(a) \cup del(a)) \cap \mathcal{P}_1 = \emptyset$ or $(add(a) \cup del(a)) \cap \mathcal{P}_2 = \emptyset$. Let s be a state in (P, A, I, G) . Then $h_1^{\mathcal{P}}(s) + h_2^{\mathcal{P}}(s) \leq sd(s)$.

Proof.

Exercise. □

- ▶ Actually it suffices if for all $a \in A$ either $add(a) \cap \mathcal{P}_1 = \emptyset$ or $add(a) \cap \mathcal{P}_2 = \emptyset$
- ▶ Made simpler for the sake of the exercise ...

- ▶ J. Culberson and J. Schaeffer, *Pattern Databases*, Computational Intelligence, 1998
- ▶ I. Hernadvölgyi and R. Holte, *PSVN: A Vector Representation for Production Systems*, Technical Report, 1999
- ▶ R. Korf and A. Felner, *Disjoint Pattern Database Heuristics*, Artificial Intelligence, 2002
- ▶ A. Felner, R. Korf, and S. Hanan, *Additive Pattern Database Heuristics*, JAIR, 2004
- ▶ S. Edelkamp, *Planning with Pattern Databases*, ECP 2001
- ▶ P. Haslum, A. Botea, M. Helmert, B. Bonet, and S. Koenig, *Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning*, AAAI 2007
- ▶ J. Hoffmann, A. Sabharwal, and C. Domshlak, *Friends or Foes? An AI Planning Perspective on Abstraction and Search*, ICAPS 2006