Planning and Optimization

D9. M&S: Maintaining the Abstraction and Shrinking Strategies

Malte Helmert and Gabriele Röger

Universität Basel

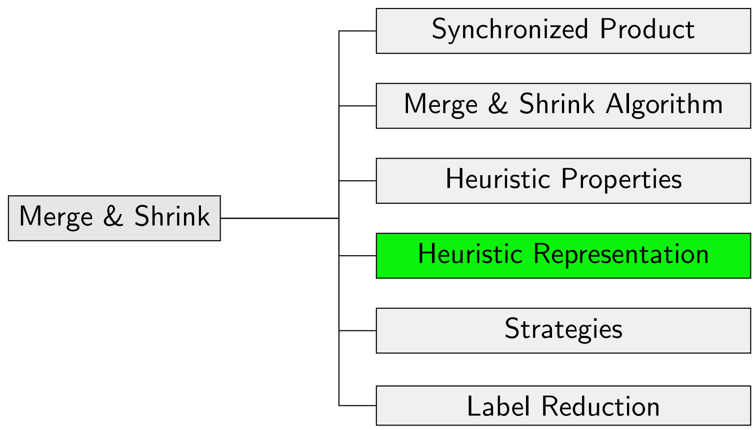
November 15, 2017

Planning and Optimization

November 15, 2017 — D9. M&S: Maintaining the Abstraction and Shrinking Strategies

D9.1 Maintaining the Abstraction

Heuristic Representation



Generic Algorithm Template

Generic Merge & Shrink Algorithm

$abs := \{\mathcal{T}^{\pi_{\{v\}}} \mid v \in V\}$
while *abs* contains more than one abstract transition system:
  select $\mathcal{A}_1$, $\mathcal{A}_2$ from *abs*
  shrink $\mathcal{A}_1$ and/or $\mathcal{A}_2$ until $size(\mathcal{A}_1) \cdot size(\mathcal{A}_2) \leq N$
  $abs := abs \setminus \{\mathcal{A}_1, \mathcal{A}_2\} \cup \{\mathcal{A}_1 \otimes \mathcal{A}_2\}$
return the remaining abstract transition system in *abs*

N: parameter bounding number of abstract states

- The algorithm computes an abstract transition system.
- For the heuristic evaluation, we need an abstraction.
- How to maintain and represent the corresponding abstraction?

The Need for Succinct Abstractions

- One major difficulty for non-PDB abstraction heuristics is to succinctly represent the abstraction.
- For pattern databases, this is easy because the abstractions – projections – are very structured.
- For less rigidly structured abstractions, we need another idea.

How to Represent the Abstraction? (1)

Idea: the computation of the abstraction follows the sequence of product computations

- For the atomic abstractions $\pi_{\{v\}}$, we generate a one-dimensional table that denotes which value in dom(v) corresponds to which abstract state in $\mathcal{T}^{\pi_{\{v\}}}$.
- During the merge (product) step $\mathcal{A} := \mathcal{A}_1 \otimes \mathcal{A}_2$, we generate a two-dimensional table that denotes which pair of states of $\mathcal{A}_1$ and $\mathcal{A}_2$ corresponds to which state of $\mathcal{A}$.
- During the shrink (abstraction) steps, we make sure to keep the table in sync with the abstraction choices.

How to Represent the Abstraction? (2)

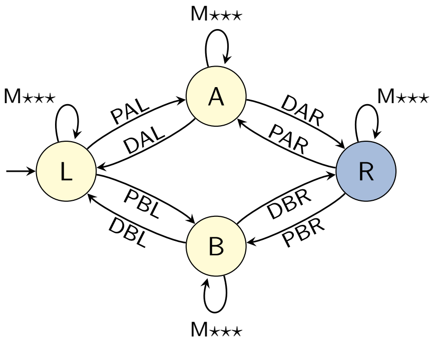
Idea: the computation of the abstraction mapping follows the sequence of product computations

- Once we have computed the final abstract transition system, we compute all abstract goal distances and store them in a one-dimensional table.
- At this point, we can throw away all the abstract transition systems – we just need to keep the tables.
- During search, we do a sequence of table lookups to navigate from the atomic abstraction states to the final abstract state and heuristic value
  $\rightsquigarrow$ $2|V|$ lookups, $O(|V|)$ time

Again, we illustrate the process with our running example.

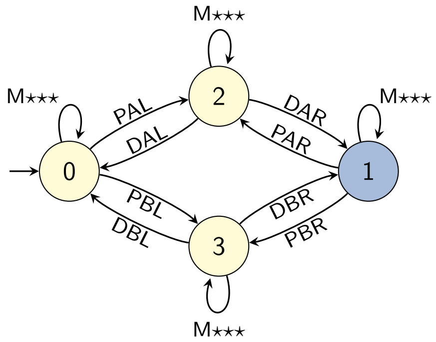
Abstraction Example: Atomic Abstractions

Computing abstractions for the transition systems of atomic abstractions is simple. Just number the states (domain values) consecutively and generate a table of references to the states:



Abstraction Example: Atomic Abstractions

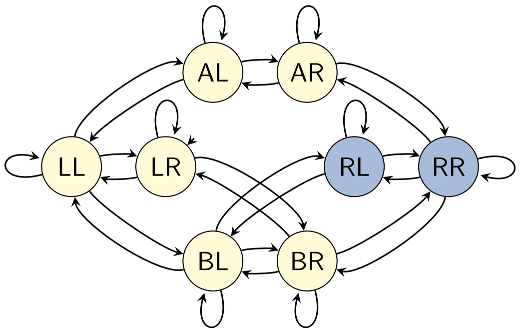
Computing abstractions for the transition systems of atomic abstractions is simple. Just number the states (domain values) consecutively and generate a table of references to the states:



L	R	A	B
0	1	2	3

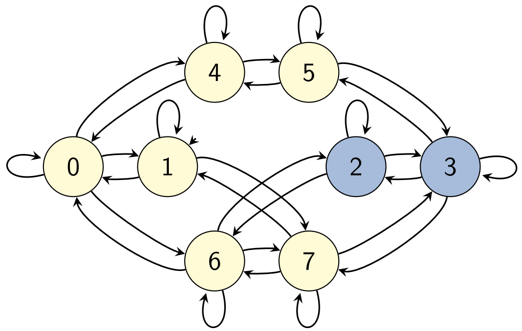
Abstraction Example: Merge Step

For product transition systems $\mathcal{A}_1 \otimes \mathcal{A}_2$, we again number the product states consecutively and generate a table that links state pairs of $\mathcal{A}_1$ and $\mathcal{A}_2$ to states of $\mathcal{A}$:



Abstraction Example: Merge Step

For product transition systems $\mathcal{A}_1 \otimes \mathcal{A}_2$, we again number the product states consecutively and generate a table that links state pairs of $\mathcal{A}_1$ and $\mathcal{A}_2$ to states of $\mathcal{A}$:



	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	0	1
$s_1 = 1$	2	3
$s_1 = 2$	4	5
$s_1 = 3$	6	7

Maintaining the Abstraction when Shrinking

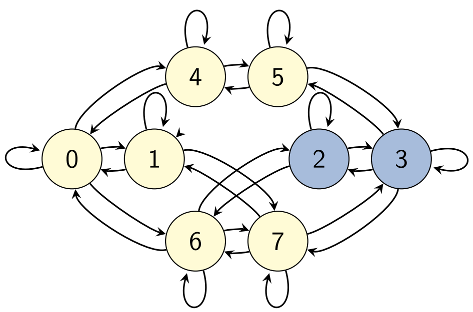
- The hard part in representing the abstraction is to keep it consistent when shrinking.
- In theory, this is easy to do:
  - When combining states i and j, arbitrarily use one of them (say i) as the number of the new state.
  - Find all table entries in the table for this abstraction which map to the other state j and change them to i.
- However, doing a table scan each time two states are combined is very inefficient.
- Fortunately, there also is an efficient implementation which takes constant time per combination.

Maintaining the Abstraction Efficiently

- Associate each abstract state with a linked list, representing all table entries that map to this state.
- Before starting the shrink operation, initialize the lists by scanning through the table, then discard the table.
- While shrinking, when combining i and j, splice the list elements of j into the list elements of i.
  - For linked lists, this is a constant-time operation.
- Once shrinking is completed, renumber all abstract states so that there are no gaps in the numbering.
- Finally, regenerate the mapping table from the linked list information.

Abstraction Example: Shrink Step

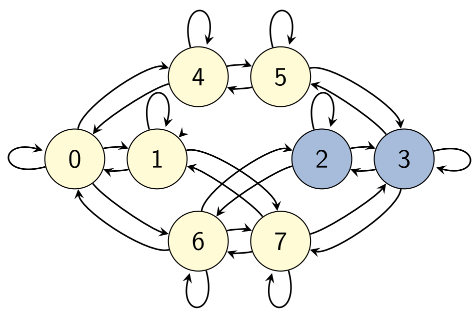
Representation before shrinking:



	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	0	1
$s_1 = 1$	2	3
$s_1 = 2$	4	5
$s_1 = 3$	6	7

Abstraction Example: Shrink Step

1. Convert table to linked lists and discard it.

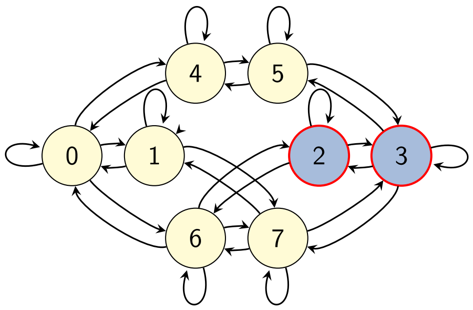


$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0)\}$
$list_3 = \{(1,1)\}$
$list_4 = \{(2,0)\}$
$list_5 = \{(2,1)\}$
$list_6 = \{(3,0)\}$
$list_7 = \{(3,1)\}$

	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	0	1
$s_1 = 1$	2	3
$s_1 = 2$	4	5
$s_1 = 3$	6	7

Abstraction Example: Shrink Step

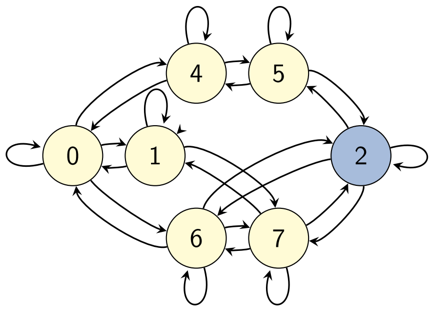
2. When combining i and j, splice $list_j$ into $list_i$.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0)\}$
$list_3 = \{(1,1)\}$
$list_4 = \{(2,0)\}$
$list_5 = \{(2,1)\}$
$list_6 = \{(3,0)\}$
$list_7 = \{(3,1)\}$

Abstraction Example: Shrink Step

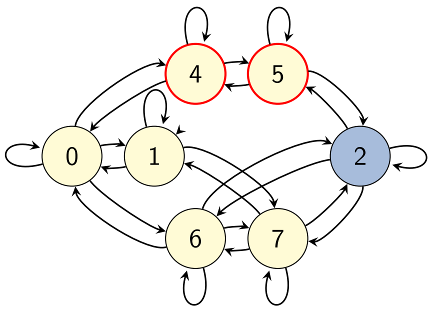
2. When combining i and j, splice $list_j$ into $list_i$.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \emptyset$
$list_4 = \{(2,0)\}$
$list_5 = \{(2,1)\}$
$list_6 = \{(3,0)\}$
$list_7 = \{(3,1)\}$

Abstraction Example: Shrink Step

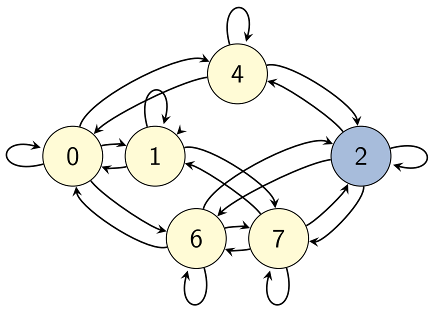
2. When combining i and j, splice $list_j$ into $list_i$.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \emptyset$
$list_4 = \{(2,0)\}$
$list_5 = \{(2,1)\}$
$list_6 = \{(3,0)\}$
$list_7 = \{(3,1)\}$

Abstraction Example: Shrink Step

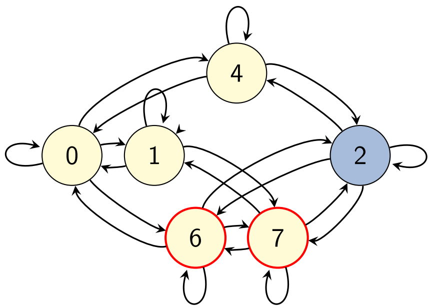
2. When combining i and j, splice $list_j$ into $list_i$.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \emptyset$
$list_4 = \{(2,0),(2,1)\}$
$list_5 = \emptyset$
$list_6 = \{(3,0)\}$
$list_7 = \{(3,1)\}$

Abstraction Example: Shrink Step

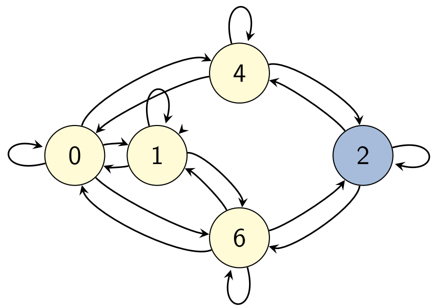
2. When combining i and j, splice $list_j$ into $list_i$.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \emptyset$
$list_4 = \{(2,0),(2,1)\}$
$list_5 = \emptyset$
$list_6 = \{(3,0)\}$
$list_7 = \{(3,1)\}$

Abstraction Example: Shrink Step

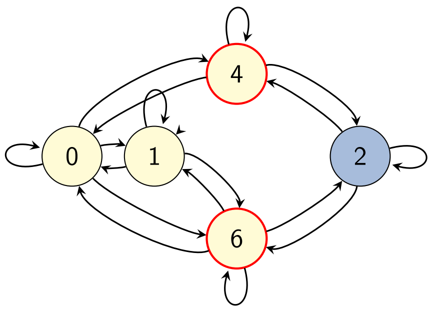
2. When combining i and j, splice $list_j$ into $list_i$.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \emptyset$
$list_4 = \{(2,0),(2,1)\}$
$list_5 = \emptyset$
$list_6 = \{(3,0),(3,1)\}$
$list_7 = \emptyset$

Abstraction Example: Shrink Step

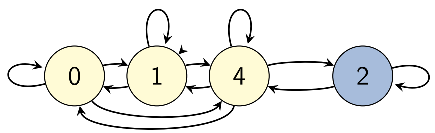
2. When combining i and j, splice $list_j$ into $list_i$.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \emptyset$
$list_4 = \{(2,0),(2,1)\}$
$list_5 = \emptyset$
$list_6 = \{(3,0),(3,1)\}$
$list_7 = \emptyset$

Abstraction Example: Shrink Step

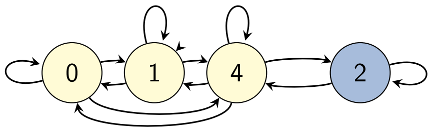
2. When combining i and j, splice $list_j$ into $list_i$.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \emptyset$
$list_4 = \{(2,0),(2,1),$
$(3,0),(3,1)\}$
$list_5 = \emptyset$
$list_6 = \emptyset$
$list_7 = \emptyset$

Abstraction Example: Shrink Step

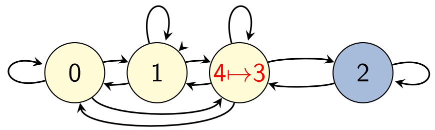
2. When combining i and j, splice $list_j$ into $list_i$.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \emptyset$
$list_4 = \{(2,0),(2,1),$
$(3,0),(3,1)\}$
$list_5 = \emptyset$
$list_6 = \emptyset$
$list_7 = \emptyset$

Abstraction Example: Shrink Step

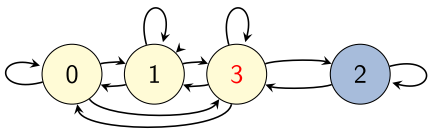
3. Renumber abstract states consecutively.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \emptyset$
$list_4 = \{(2,0),(2,1),$
$(3,0),(3,1)\}$
$list_5 = \emptyset$
$list_6 = \emptyset$
$list_7 = \emptyset$

Abstraction Example: Shrink Step

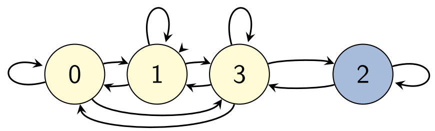
3. Renumber abstract states consecutively.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \{(2,0),(2,1),$
$(3,0),(3,1)\}$
$list_4 = \emptyset$
$list_5 = \emptyset$
$list_6 = \emptyset$
$list_7 = \emptyset$

Abstraction Example: Shrink Step

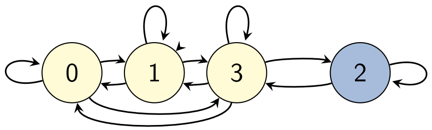
4. Regenerate the mapping table from the linked lists.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \{(2,0),(2,1),$
$(3,0),(3,1)\}$
$list_4 = \emptyset$
$list_5 = \emptyset$
$list_6 = \emptyset$
$list_7 = \emptyset$

Abstraction Example: Shrink Step

4. Regenerate the mapping table from the linked lists.



$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0),(1,1)\}$
$list_3 = \{(2,0),(2,1),$
$(3,0),(3,1)\}$
$list_4 = \emptyset$
$list_5 = \emptyset$
$list_6 = \emptyset$
$list_7 = \emptyset$

	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	0	1
$s_1 = 1$	2	2
$s_1 = 2$	3	3
$s_1 = 3$	3	3

The Final Heuristic Representation

At the end, our heuristic is represented by six tables:

- three one-dimensional tables for the atomic abstractions:

T_{package}	L	R	A	B
	0	1	2	3

$T_{\text{truck A}}$	L	R
	0	1

$T_{\text{truck B}}$	L	R
	0	1

- two tables for the two merge and subsequent shrink steps:

$T^1_{\text{m\&s}}$	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	0	1
$s_1 = 1$	2	2
$s_1 = 2$	3	3
$s_1 = 3$	3	3

$T^2_{\text{m\&s}}$	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	1	1
$s_1 = 1$	1	0
$s_1 = 2$	2	2
$s_1 = 3$	3	3

- one table with goal distances for the final transition system:

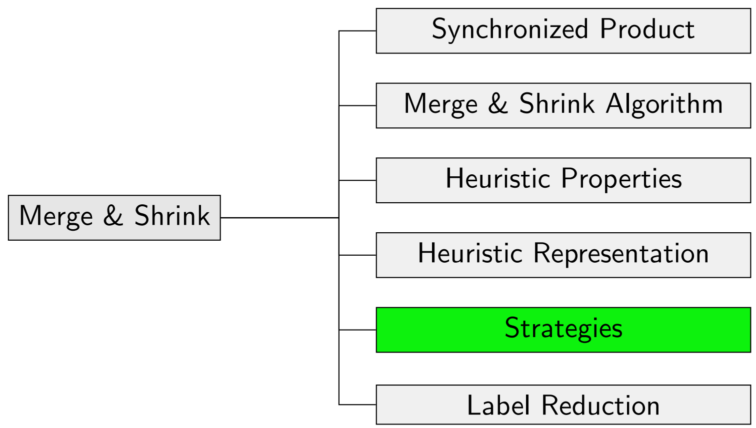
T_h	$s = 0$	$s = 1$	$s = 2$	$s = 3$
$h(s)$	3	2	0	1

Given a state $s = \{\text{package} \mapsto L, \text{truck A} \mapsto L, \text{truck B} \mapsto R\}$, its heuristic value is then looked up as:

- $h(s) = T_h[T^2_{\text{m\&s}}[T^1_{\text{m\&s}}[T_{\text{package}}[L], T_{\text{truck A}}[L]], T_{\text{truck B}}[R]]]$

D9.2 Shrinking Strategies

Shrinking Strategies



Generic Algorithm Template

Generic M&S computation algorithm

$abs := \{\mathcal{T}^{\pi_{\{v\}}} \mid v \in V\}$
while abs contains more than one abstraction:
    select $\mathcal{A}_1$, $\mathcal{A}_2$ from abs
    shrink $\mathcal{A}_1$ and/or $\mathcal{A}_2$ until $size(\mathcal{A}_1) \cdot size(\mathcal{A}_2) \leq N$
    $abs := abs \setminus \{\mathcal{A}_1, \mathcal{A}_2\} \cup \{\mathcal{A}_1 \otimes \mathcal{A}_2\}$
return the remaining abstraction in abs

N: parameter bounding number of abstract states

Remaining Questions:

- Which abstractions to select? $\rightsquigarrow$ merging strategy
- How to shrink an abstraction? $\rightsquigarrow$ shrinking strategy

Shrinking Strategies

How to shrink an abstraction?

We cover two common approaches:

- f-preserving shrinking
- bisimulation-based shrinking

f-preserving Shrinking Strategy

f-preserving Shrinking Strategy

Repeatedly combine abstract states with
identical abstract goal distances (h values) and
identical abstract initial state distances (g values).

Rationale: preserves heuristic value and overall graph shape

Tie-breaking Criterion

Prefer combining states where $g + h$ is high.
In case of ties, combine states where h is high.

Rationale: states with high $g + h$ values are less likely to be explored by A*, so inaccuracies there matter less

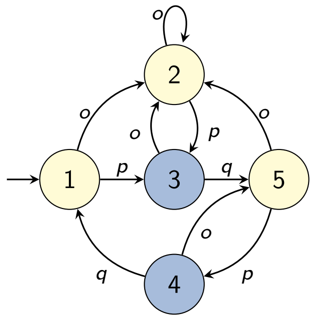
Bisimulation

Definition (Bisimulation)

Let $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$ be a transition system. An equivalence relation $\sim$ on S is a bisimulation for $\mathcal{T}$ if for every $\langle s, \ell, s' \rangle \in T$ and every $t \sim s$ there is a transition $\langle t, \ell, t' \rangle \in T$ with $t' \sim s'$.

A bisimulation $\sim$ is goal-respecting if $s \sim t$ implies that either $s, t \in S_\star$ or $s, t \notin S_\star$.

Bisimulation: Example



$\sim$ with equivalence classes $\{\{1,2,5\},\{3,4\}\}$ is a goal-respecting bisimulation.

Bisimulations as Abstractions

Theorem (Bisimulations as Abstractions)

Let $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$ be a transition system and $\sim$ be a bisimulation for $\mathcal{T}$. Then $\alpha_\sim : S \to \{[s]_\sim \mid s \in S\}$ with $\alpha_\sim(s) = [s]_\sim$ is an abstraction of $\mathcal{T}$.

Note: $[s]_\sim$ denotes the equivalence class of s.

Note: Surjectivity follows from the definition of the codomain as the image of $\alpha_\sim$.

Abstractions as Bisimulations

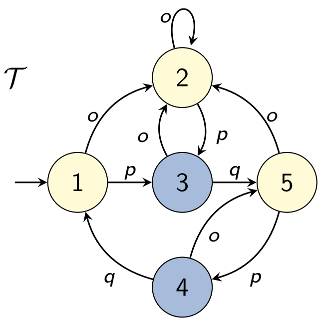
Definition (Abstraction as Bisimulation)

Let $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$ be a transition system and $\alpha : S \to S'$ be an abstraction of $\mathcal{T}$. The abstraction induces the equivalence relation $\sim_\alpha$ as $s \sim_\alpha t$ iff $\alpha(s) = \alpha(t)$.
We say that α is a (goal-respecting) bisimulation for $\mathcal{T}$ if $\sim_\alpha$ is a (goal-respecting) bisimulation for $\mathcal{T}$.

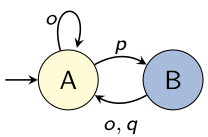
Abstraction as Bisimulations: Example

Abstraction α with
$\alpha(1) = \alpha(2) = \alpha(5) = A$ and $\alpha(3) = \alpha(4) = B$
is a goal-respecting bisimulation for $\mathcal{T}$.

$\mathcal{T}$



$\mathcal{T}^\alpha$



Goal-respecting Bisimulations are Exact (1)

Theorem

Let X be a collection of transition systems. Let α be an abstraction for $\mathcal{T}_i \in X$. If α is a goal-respecting bisimulation then the transformation from X to $X' := (X \setminus \{\mathcal{T}_i\}) \cup \{\mathcal{T}_i^\alpha\}$ is exact.

Proof.

Let $\mathcal{T}_X = \mathcal{T}_1 \otimes \cdots \otimes \mathcal{T}_n = \langle S, L, c, T, s_0, S_\star\rangle$ and w.l.o.g. $\mathcal{T}_{X'} = \mathcal{T}_1 \otimes \cdots \otimes \mathcal{T}_{i-1} \otimes \mathcal{T}_i^\alpha \otimes \mathcal{T}_{i+1} \otimes \cdots \otimes \mathcal{T}_n = \langle S', L', c', T', s_0', S_\star'\rangle$.
Consider $\sigma(\langle s_1, \dots, s_n\rangle) = \langle s_1, \dots, s_{i-1}, \alpha(s_i), s_{i+1}, \dots, s_n\rangle$ for the mapping of states and $\lambda = \mathrm{id}$ for the mapping of labels.

1. Mappings σ and λ satisfy the requirements of safe transformations because α is an abstraction and we have chosen the mapping functions as before.

...

Goal-respecting Bisimulations are Exact (2)

Proof (continued).

2. If $\langle s', \ell, t'\rangle \in T'$ with $s' = \langle s_1', \dots, s_n'\rangle$ and $t' = \langle t_1', \dots, t_n'\rangle$, then for $j \neq i$ transition system $\mathcal{T}_j$ has transition $\langle s_j', \ell, t_j'\rangle$ (*) and $\mathcal{T}_i^\alpha$ has transition $\langle s_i', \ell, t_i'\rangle$. This implies that $\mathcal{T}_i$ has a transition $\langle s_i'', \ell, t_i''\rangle$ for some $s_i'' \in \alpha^{-1}(s_i')$ and $t_i'' \in \alpha^{-1}(t_i')$. As α is a bisimulation, there must be such a transition for *all* such s_i'' and t_i'' (**).
Each $s \in \sigma^{-1}(s')$ has the form $s = \langle s_1, \dots, s_n\rangle$ with $s_j = s_j'$ for $j \neq i$ and $s_i \in \alpha^{-1}(s_i')$. Analogously for each $t = \langle t_1, \dots, t_n\rangle \in \sigma^{-1}(t')$. From (*) and (**) follows that $\mathcal{T}_j$ has a transition $\langle s_j, \ell, t_j\rangle$ for all $j \in \{1, \dots, n\}$, so for each such s and t, T contains the transition $\langle s, \ell, t\rangle$.

...

Goal-respecting Bisimulations are Exact (3)

Proof (continued).

3. For $s_\star' = \langle s_1', \dots, s_n'\rangle \in S_\star'$, each s_j' with $j \neq i$ must be a goal state of $\mathcal{T}_j$ (*) and s_i' must be a goal state of $\mathcal{T}_i^\alpha$. The latter implies that at least on $s_i'' \in \alpha^{-1}(s_i')$ is a goal state of $\mathcal{T}_i$. As α is goal-respecting, all states from $\alpha^{-1}(s_i')$ are goal states of $\mathcal{T}_i$ (**).
Consider $s_\star = \langle s_1, \dots, s_n\rangle \in \sigma^{-1}(s_\star')$. By the definition of σ, $s_j = s_j'$ for $j \neq i$ and $s_i \in \alpha^{-1}(s_i')$. From (*) and (**), each s_j ($j \in \{1, \dots, n\}$) is a goal state of $\mathcal{T}_j$ and, hence, $s_\star$ a goal state of $\mathcal{T}_X$.
4. As $\lambda = \mathrm{id}$ and the transformation does not change the label cost function, $c(\ell) = c'(\lambda(\ell))$ for all $\ell \in L$.

□

Bisimulations: Discussion

- As all bisimulations preserve all relevant information, we are interested in the coarsest such abstraction (to shrink as much as possible).
- There is always a unique coarsest bisimulation for $\mathcal{T}$ and it can be computed efficiently (from the explicit representation).
- In some cases, computing the bisimulation is still too expensive or it cannot sufficiently shrink a transition system.

Greedy Bisimulations

Definition (Greedy Bisimulation)

Let $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$ be a transition system. An equivalence relation $\sim$ on S is a greedy bisimulation for $\mathcal{T}$ if it is a bisimulation for the system $\langle S, L, c, T^G, s_0, S_\star \rangle$, where $T^G = \{\langle s, \ell, t \rangle \mid \langle s, \ell, t \rangle \in T, h^*(s) = h^*(t) + c(\ell)\}$.

Greedy bisimulation only considers transitions that are used in an optimal solution of some state of $\mathcal{T}$.

Greedy Bisimulation is h-preserving

Theorem

Let $\mathcal{T}$ be a transition system and let α be an abstraction of $\mathcal{T}$. If $\sim_\alpha$ is a goal-respecting greedy bisimulation for $\mathcal{T}$ then $h^*_{\mathcal{T}^\alpha} = h^*_{\mathcal{T}}$.

(Proof omitted.)

Note: This does not mean that replacing $\mathcal{T}$ with $\mathcal{T}^\alpha$ in a collection of transition systems is a safe transformation! Abstraction α preserves solution costs "locally" but not "globally".

D9.3 Summary

Summary

- Merge-and-shrink abstractions are represented by a set of reference tables, one for each atomic abstraction and one for each merge-and-shrink step.
- The heuristic representation uses an additional table for the goal distances in the final abstract transition system.
- Bisimulation is an exact shrinking method.