Planning and Optimization

C15. M&S: Maintaining the Mapping and Some Theory

Malte Helmert and Gabriele Röger

Universität Basel

November 17, 2016

Planning and Optimization

November 17, 2016 — C15. M&S: Maintaining the Mapping and Some Theory

C15.1 Motivation

Generic Algorithm Template

Generic Abstraction Computation 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

Questions for practical implementation:

- How to represent the corresponding abstraction?
- Which abstractions to select? $\rightsquigarrow$ merging strategy
- How to shrink an abstraction? $\rightsquigarrow$ shrinking strategy
- How to choose N? $\rightsquigarrow$ usually: as high as memory allows

C15.2 Maintaining the Abstraction

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 $\mathrm{dom}(v)$ 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)

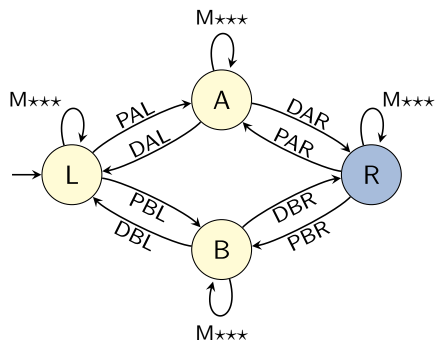
Idea: the computation of the abstraction mapping follows the sequence of product computations

- 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.

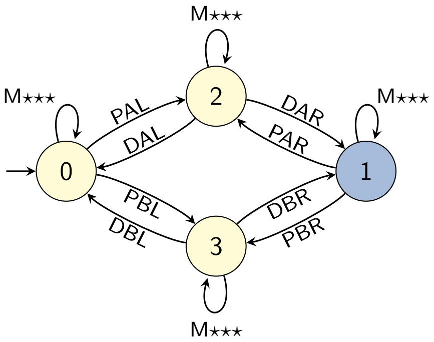
Abstraction Example: Atomic Abstractions

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

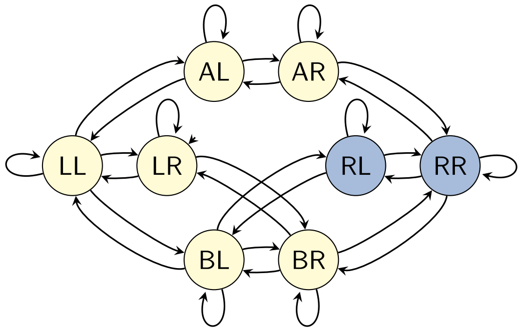
Computing abstractions for the transition systems of atomic abstractions is simple. Just number the states (domain values) consecutively and generate a table of references to the states:



L	R	A	B
0	1	2	3

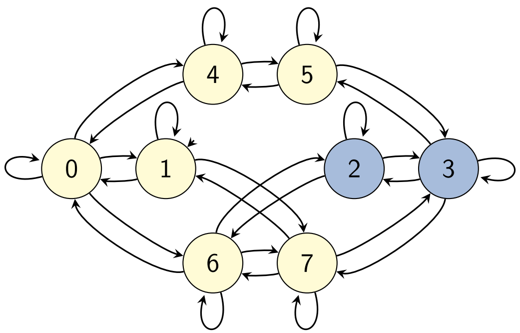
Abstraction Example: Merge Step

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

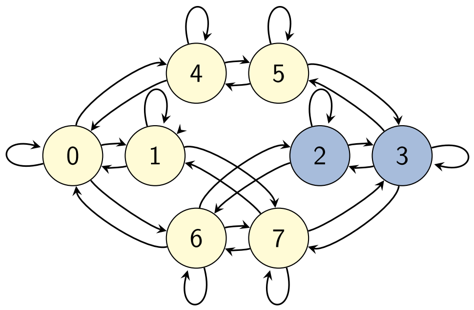
- The hard part in representing the abstraction is to keep it consistent when shrinking.
- In theory, this is easy to do:
  - When combining states i and j, arbitrarily use one of them (say i) as the number of the new state.
  - Find all table entries in the table for this abstraction which map to the 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

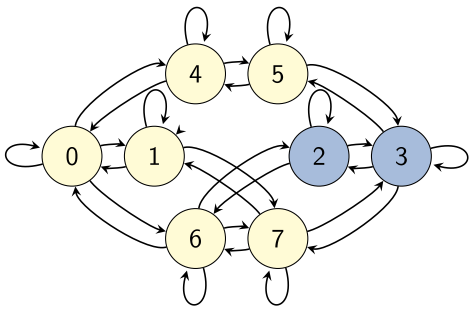
Representation before shrinking:



	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	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.

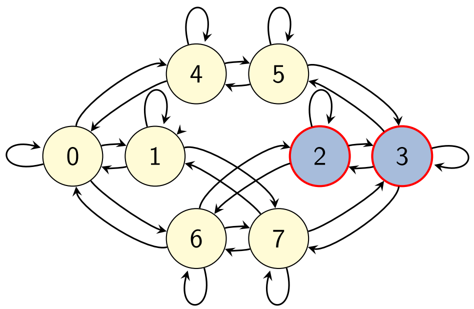


$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$
$list_2 = \{(1,0)\}$
$list_3 = \{(1,1)\}$
$list_4 = \{(2,0)\}$
$list_5 = \{(2,1)\}$
$list_6 = \{(3,0)\}$
$list_7 = \{(3,1)\}$

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

Abstraction Example: Shrink Step

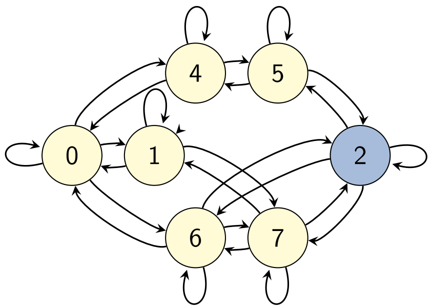
2. When combining i and j, splice $list_j$ into $list_i$.



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

Abstraction Example: Shrink Step

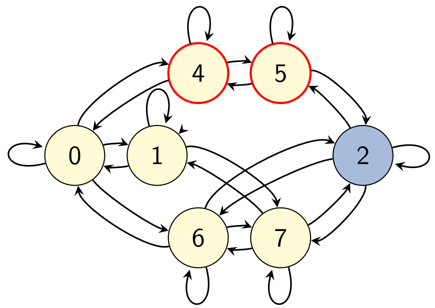
2. When combining i and j, splice $list_j$ into $list_i$.



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

Abstraction Example: Shrink Step

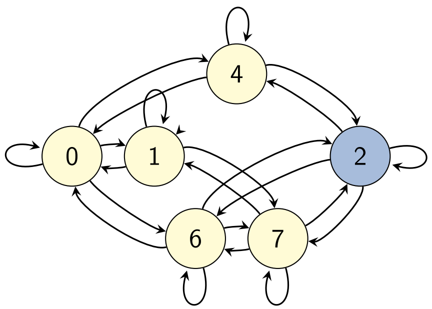
2. When combining i and j, splice $list_j$ into $list_i$.



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

Abstraction Example: Shrink Step

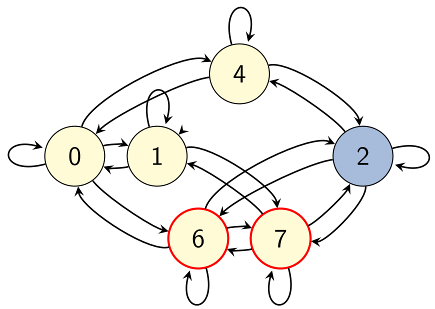
2. When combining i and j, splice $list_j$ into $list_i$.



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

Abstraction Example: Shrink Step

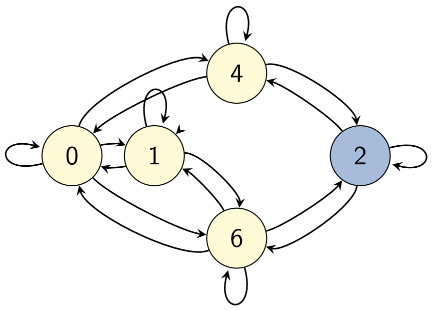
2. When combining i and j, splice $list_j$ into $list_i$.



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

Abstraction Example: Shrink Step

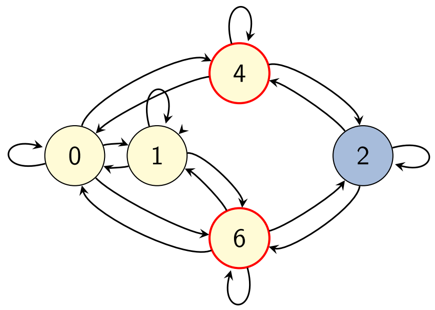
2. When combining i and j, splice $list_j$ into $list_i$.



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

Abstraction Example: Shrink Step

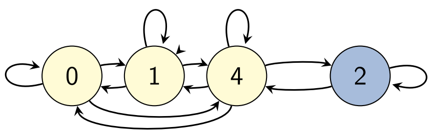
2. When combining i and j, splice $list_j$ into $list_i$.



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

Abstraction Example: Shrink Step

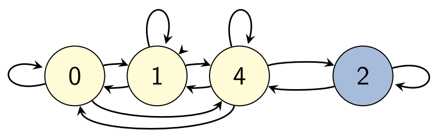
2. When combining i and j, splice $list_j$ into $list_i$.



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

Abstraction Example: Shrink Step

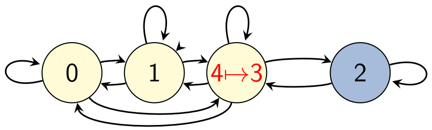
2. When combining i and j, splice $list_j$ into $list_i$.



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

Abstraction Example: Shrink Step

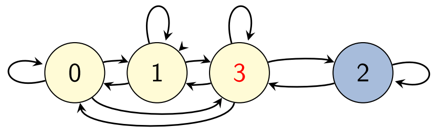
3. Renumber abstract states consecutively.



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

Abstraction Example: Shrink Step

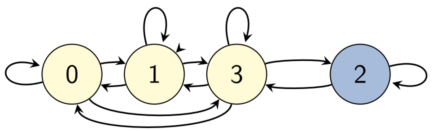
3. Renumber abstract states consecutively.



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

Abstraction Example: Shrink Step

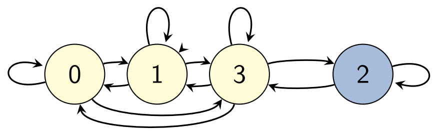
4. Regenerate the mapping table from the linked lists.



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

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	1	0

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

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

Generic Algorithm Template

Generic abstraction 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

We first need a bit more theory...

C15.3 Safe and Exact Transformations

Collections of Transition Systems

Definition (Collection of Transition Systems)

A set X of transition systems is a collection of transition systems if all $\mathcal{T} \in X$ have the same set of labels and the same cost function.
The combined system is $\mathcal{T}_X := \bigotimes_{\mathcal{T} \in X} \mathcal{T}$.

Safe Transformations

Definition (Safe Transformation)

Let X and X' be collections of transition systems with label sets L and L' and cost functions c and c', respectively.

The transformation from X to X' is safe if there exist functions σ and τ mapping the states and labels of $\mathcal{T}_X$ to the states and labels of $\mathcal{T}_{X'}$ such that

- $L' = \{\tau(\ell) \mid \ell \in L\}$,
- $c'(\tau(\ell)) \leq c(\ell)$ for all $\ell \in L$,
- if $\langle s, \ell, t\rangle$ is a transition of $\mathcal{T}_X$ then $\langle \sigma(s), \tau(\ell), \sigma(t)\rangle$ is a transition of $\mathcal{T}_{X'}$, and
- if s is a goal state of $\mathcal{T}_X$ then $\sigma(s)$ is a goal state of $\mathcal{T}_{X'}$.

Examples

X: Collection of transition systems

Replacement with Synchronized Product is Safe

Let $\mathcal{T}_1, \mathcal{T}_2 \in X$ with $\mathcal{T}_1 \neq \mathcal{T}_2$. The transformation from X to $X' := (X \setminus \{\mathcal{T}_1, \mathcal{T}_2\}) \cup \{\mathcal{T}_1 \otimes \mathcal{T}_2\}$ is safe with $\sigma = \mathrm{id}$ and $\tau = \mathrm{id}$.

Abstraction is Safe

Let α be an abstraction for $\mathcal{T}_i \in X$. The transformation from X to $X' := (X \setminus \{\mathcal{T}_i\}) \cup \{\mathcal{T}_i^\alpha\}$ is safe with $\tau = \mathrm{id}$ and $\sigma(\langle s_1, \dots, s_n\rangle) = \langle s_1, \dots, s_{i-1}, \alpha(s_i), s_{i+1}, \dots, s_n\rangle$.

(Proofs omitted.)

Heuristic Properties (1)

Theorem

Let X and X' be collections of transition systems. If the transformation from X to X' is safe with functions σ and τ then $h(s) = h^*_{\mathcal{T}_{X'}}(\sigma(s))$ is a safe, goal-aware, admissible, and consistent heuristic for $\mathcal{T}_X$.

Proof.

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

Goal-awareness: For all goal states $s_\star$ of $\mathcal{T}_X$, state $\sigma(s_\star)$ is a goal state of $\mathcal{T}_{X'}$ and therefore $h(s_\star) = h^*_{\mathcal{T}_{X'}}(\sigma(s_\star)) = 0$. ...

Heuristic Properties (2)

Proof (continued).

Consistency: Let c and c' be the label cost functions of X and X', respectively. Consider state s of $\mathcal{T}_X$ and transition $\langle s, \ell, t\rangle$. As $\mathcal{T}_{X'}$ has a transition $\langle \sigma(s), \tau(\ell), \sigma(t)\rangle$, it holds that

$$\begin{aligned} h(s) &= h^*_{\mathcal{T}_{X'}}(\sigma(s)) \\ &\leq c'(\tau(\ell)) + h^*_{\mathcal{T}_{X'}}(\sigma(t)) \\ &= c'(\tau(\ell)) + h(t) \\ &\leq c(\ell) + h(t) \end{aligned}$$

The second inequality holds due to the requirement on the label costs. □

Exact Transformations

Definition (Exact Transformation)

Let X and X' be collections of transition systems with label sets L and L' and cost functions c and c', respectively.

The transformation from X to X' is exact if there exist functions σ and τ mapping the states and labels of $\mathcal{T}_X$ to the states and labels of $\mathcal{T}_{X'}$ such that

1. σ and τ satisfy the requirements of safe transformations,
2. if $\langle s', \ell', t'\rangle$ is a transition of $\mathcal{T}_{X'}$ then $\langle s, \ell, t\rangle$ is a transition of $\mathcal{T}_X$ for all $s \in \sigma^{-1}(s'), t \in \sigma^{-1}(t')$ and some $\ell \in \tau^{-1}(\ell')$,
3. if s' is a goal state of $\mathcal{T}_{X'}$ then all states $s \in \sigma^{-1}(s')$ are goal states of $\mathcal{T}_X$, and
4. $c(\ell) = c'(\tau(\ell))$ for all $\ell \in L$.

$\rightsquigarrow$ no "new" transitions and goal states, no cheaper labels

Examples

Replacement with Synchronized Product is Exact

Let $\mathcal{T}_1, \mathcal{T}_2 \in X$ with $\mathcal{T}_1 \neq \mathcal{T}_2$. The transformation from X to $X' := (X \setminus \{\mathcal{T}_1, \mathcal{T}_2\}) \cup \{\mathcal{T}_1 \otimes \mathcal{T}_2\}$ is exact with $\sigma = \mathrm{id}$ and $\tau = \mathrm{id}$.

(Proof omitted.)

More examples will follow.

Heuristic Properties with Exact Transformations (1)

Theorem

Let X and X' be collections of transition systems. If the transformation from X to X' is exact with functions σ and τ then $h^*_{\mathcal{T}_X}(s) = h^*_{\mathcal{T}_{X'}}(\sigma(s))$.

Proof.

As the transformation is safe, $h^*_{\mathcal{T}_{X'}}(\sigma(s))$ is admissible for $\mathcal{T}_X$ and therefore $h^*_{\mathcal{T}_X}(s) \geq h^*_{\mathcal{T}_{X'}}(\sigma(s))$.
For the other direction, we show that for every state s' of $\mathcal{T}_{X'}$ and goal path π' for s', there is for each $s \in \sigma^{-1}(s')$ a goal path in $\mathcal{T}_X$ that has the same cost. ...

Heuristic Properties with Exact Transformations (2)

Proof (continued).

Proof via induction over the length of π'.

$|\pi'| = 0$: If s' is a goal state of $\mathcal{T}_{X'}$ then each $s \in \sigma^{-1}(s')$ is a goal state of $\mathcal{T}_X$ and the empty path is a goal path for s in $\mathcal{T}_X$.

$|\pi'| = i + 1$: Let $\pi' = \langle s', \ell', t'\rangle \pi'_{t'}$, where $\pi'_{t'}$ is a goal path of length i from t'. Then there is for each $t \in \sigma^{-1}(t')$ a goal path π_t of the same cost in $\mathcal{T}_X$. Furthermore, for all $s \in \sigma^{-1}(s')$ there is a label $\ell \in \tau^{-1}(\ell')$ such that $\mathcal{T}_X$ has a transition $\langle s, \ell, t\rangle$ with $t \in \sigma^{-1}(t')$. The path $\pi = \langle s, \ell, t\rangle \pi_t$ is a solution for s in $\mathcal{T}$. As ℓ and ℓ' must have the same cost and π_t and $\pi'_{t'}$ have the same cost, π has the same cost as π'. $\square$

Sequences of Transformations

Theorem (Sequences of Transformations)

*Let $X_1, \dots, X_n$ be collections of transition systems.
If for $i \in \{1, \dots, n-1\}$ the transformation from X_i to X_{i+1} is safe (exact) then the transformation from X_1 to X_n is safe (exact).*

Proof idea: The composition of the σ and τ functions of each step satisfy the required conditions.

Consequences

Generic Abstraction Computation Algorithm

abs $:= \{\mathcal{T}^{\pi_{\{v\}}} \mid v \in V\} =: X_0$
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*

- Initially $\mathcal{T}_{abs}$ is the concrete transition system.
- Each iteration performs a safe transformation of *abs*.
  $\rightarrow$ the corresponding abstraction heuristic is safe, goal-aware, consistent, and admissible.
- If shrinking is exact, the corresponding heuristic is perfect.

C15.4 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.
- As we only use safe transformations, the resulting heuristic is safe, goal-aware, admissible, and consistent.
- If we use only exact transformations, the resulting heuristic is perfect.