State identification using characterizing sets or separating sequences

Kati Overbeeke

January 21, 2026

Overview

- Motivation for testing
- Prerequisites
- Difference in the papers
- Minimal Separating Sequences for All Pairs of States
- Efficient State Identification for Finite State Machine-based Testing
- Comparison
- Conclusion

Motivation for testing

- Testing is costly
  - Manual configuration
  - Test case design

Motivation for testing

- Testing is costly
  - Manual configuration
  - Test case design
- Automation testing
  - Faster
  - Consistency and accuracy
  - Frequent testing
- Using a Finite State Machine

Motivation for testing

- Testing is costly
  - Manual configuration
  - Test case design
- Automation testing
  - Faster
  - Consistency and accuracy
  - Frequent testing
- Using a Finite State Machine
  - Does an implementation conform to its specification?

Motivation for testing

- Testing is costly
  - Manual configuration
  - Test case design
- Automation testing
  - Faster
  - Consistency and accuracy
  - Frequent testing
- Using a Finite State Machine
  - Does an implementation conform to its specification?
- Other methods:
  - Property based testing
  - Extended FSM Models
  - Formal verification

Overview prerequisites

- Finite State Machines
- Separating sequence
- Characterisation set
- Test suite

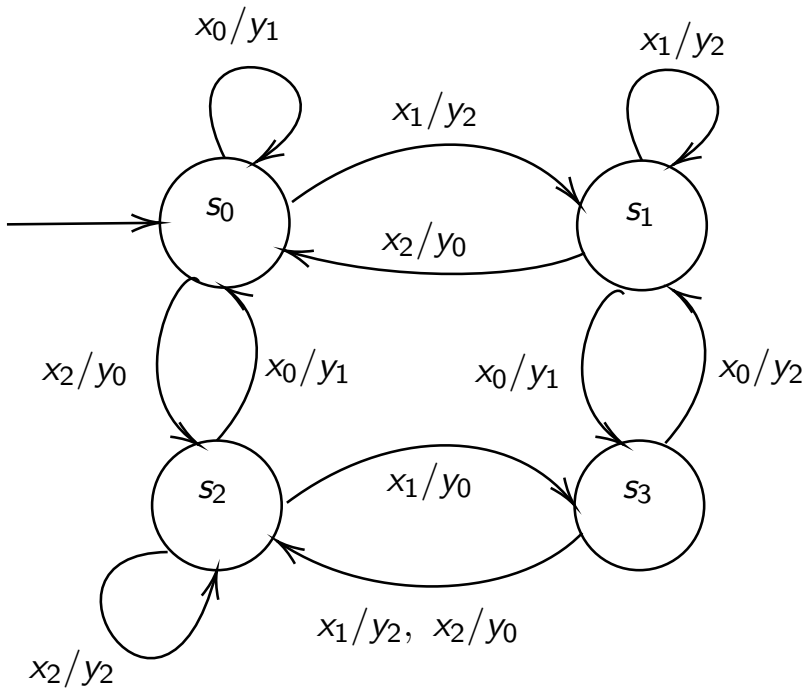
Finite State Machines

- Set of states, S
- Inputs, I
- Outputs, O
- $\lambda : S \times I \rightarrow O$
- $\delta : S \times I \rightarrow S$

Finite State Machines

- Set of states, S
- Inputs, I
- Outputs, O
- $\lambda : S \times I \rightarrow O$
- $\delta : S \times I \rightarrow S$

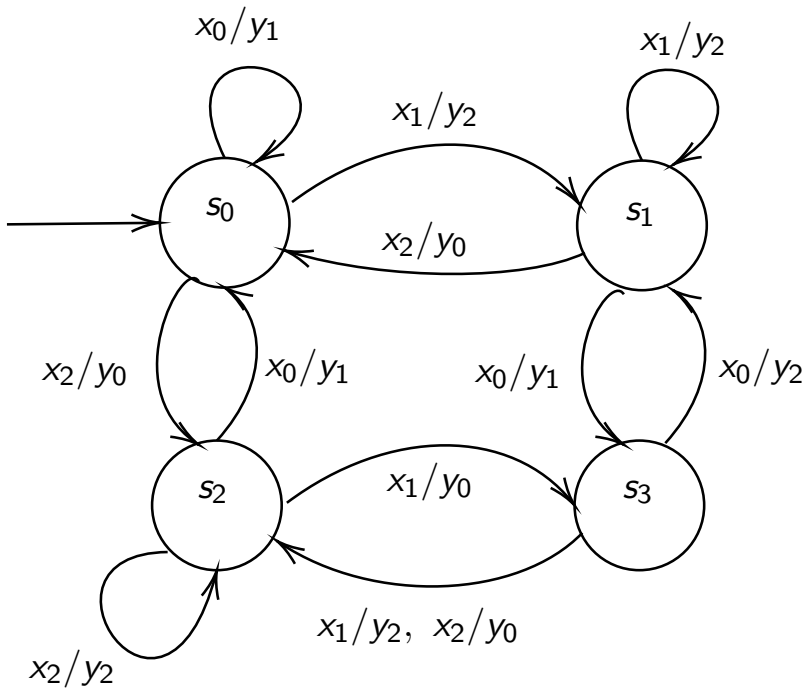
Example



Finite State Machines

- Set of states, S
- Inputs, I
- Outputs, O
- $\lambda : S \times I \rightarrow O$
- $\delta : S \times I \rightarrow S$

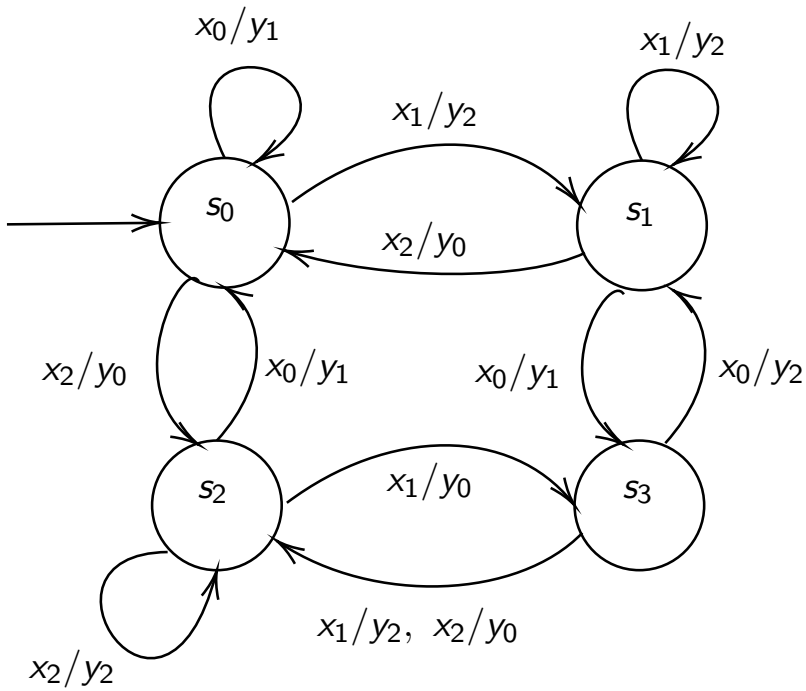
Example



Finite State Machines

- Set of states, S
- Inputs, I
- Outputs, O
- $\lambda : S \times I \rightarrow O$
- $\delta : S \times I \rightarrow S$

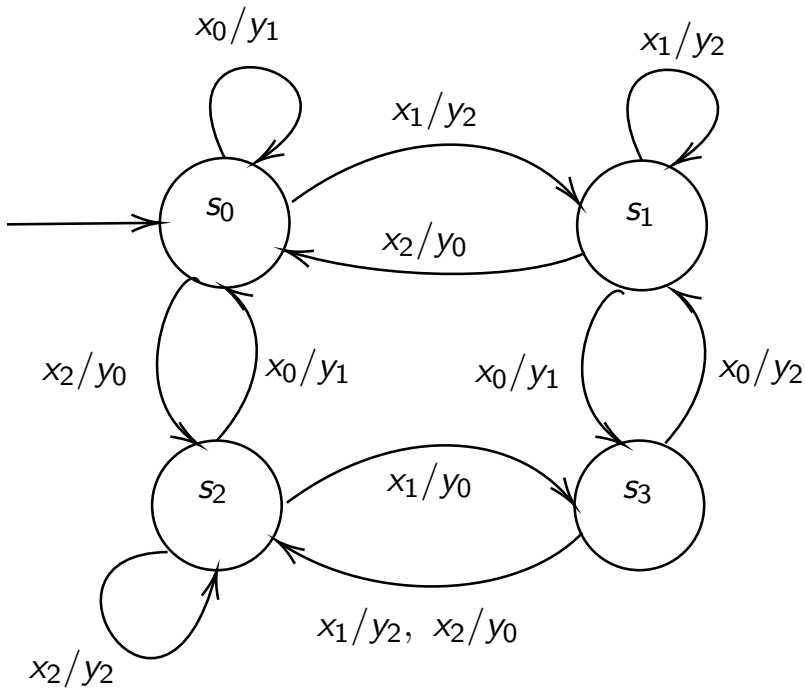
Example



Finite State Machines

- Set of states, S
- Inputs, I
- Outputs, O
- $\lambda : S \times I \rightarrow O$
- $\delta : S \times I \rightarrow S$

Example



Seperating Sequence

Definition (Separating Sequence)

A separating sequence for states s and t is a sequence $x \in I^*$ such that $\lambda^*(s, x) \neq \lambda^*(t, x)$.

Seperating Sequence

Definition (Separating Sequence)

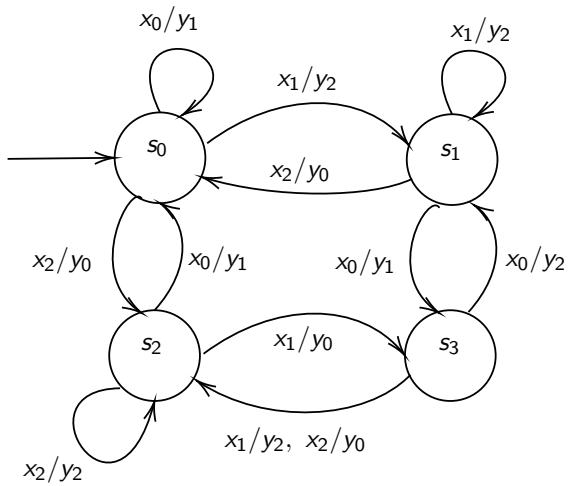
A separating sequence for states s and t is a sequence $x \in I^*$ such that $\lambda^*(s, x) \neq \lambda^*(t, x)$. We say x is minimal if $|y| \geq |x|$ for all separating sequences y for s and t.

Seperating Sequence

Definition (Separating Sequence)

A separating sequence for states s and t is a sequence $x \in I^*$ such that $\lambda^*(s, x) \neq \lambda^*(t, x)$. We say x is minimal if $|y| \geq |x|$ for all separating sequences y for s and t.

Example



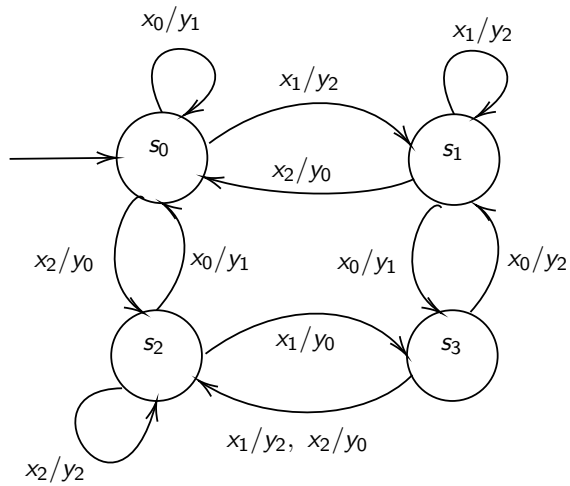
$x := x_0x_0$ is a minimal separating sequence for s_0 and s_1.

Seperating Sequence

Definition (Separating Sequence)

A separating sequence for states s and t is a sequence $x \in I^*$ such that $\lambda^*(s, x) \neq \lambda^*(t, x)$. We say x is minimal if $|y| \geq |x|$ for all separating sequences y for s and t.

Example



$x := x_0x_0$ is a minimal separating sequence for s_0 and s_1.
x_0, x_1, x_2 do not separate s_0 and s_1.

Characterization set

Definition (Characterization set)

A set $W \subseteq I^*$ is a characterisation set if for every pair of states (s, t) in FSM M there exists a w in W such that w separates s and t.

Characterization set

Definition (Characterization set)

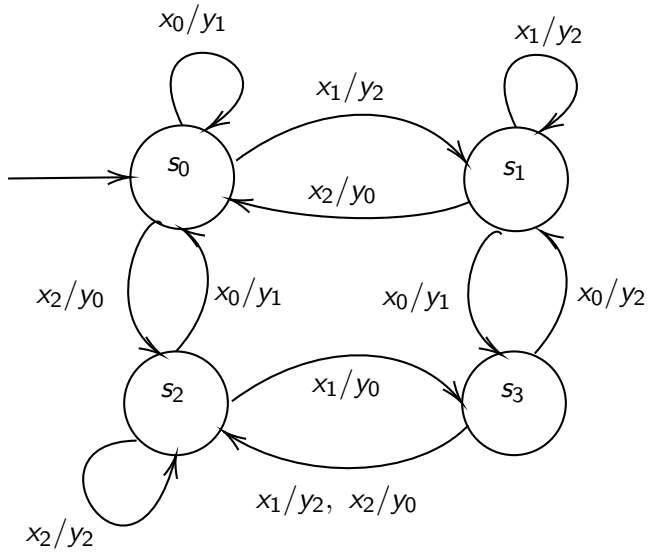
A set $W \subseteq I^*$ is a characterisation set if for every pair of states (s, t) in FSM M there exists a w in W such that w separates s and t.
$\lambda^*(s, w) \neq \lambda^*(t, w)$, where λ^* is the extended output.

Characterization set

Definition (Characterization set)

A set $W \subseteq I^*$ is a characterisation set if for every pair of states (s, t) in FSM M there exists a w in W such that w separates s and t.
$\lambda^*(s, w) \neq \lambda^*(t, w)$, where λ^* is the extended output.

Example



The set $W = \{x_0, x_2x_2\}$ is a characterizing set.

Test suite

- Set of test cases

Test suite

- Set of test cases
  - Input sequence
  - Corresponding expected output
- One of the definitions of the cost of a test suite:
  - Size of inputs and input sequences
  - Time required

Difference in the papers

Difference in the papers

- Minimal Separating Sequences for All Pairs of States
  - by Rick Smetsers, Joshua Moerman, and David N. Jansen
  - 2016

Difference in the papers

- Minimal Separating Sequences for All Pairs of States
  - by Rick Smetsers, Joshua Moerman, and David N. Jansen
  - 2016
- Efficient State Identification for Finite State Machine-based Testing
  - by Uraz Cengiz Turker, Robert M. Hierons, Mohammad Reza Mousavi, and Khaled El-Fakih
  - 2025

Difference in the papers

- Minimal Separating Sequences for All Pairs of States
  - by Rick Smetsers, Joshua Moerman, and David N. Jansen
  - 2016
- Efficient State Identification for Finite State Machine-based Testing
  - by Uraz Cengiz Turker, Robert M. Hierons, Mohammad Reza Mousavi, and Khaled El-Fakih
  - 2025
- Why do this work?
  - Finite State Machine Testing is simple
  - Still a few costly operations:
    - Reset sequences
    - Transfer sequences

Difference in the papers

- Minimal Separating Sequences for All Pairs of States
  - by Rick Smetsers, Joshua Moerman, and David N. Jansen
  - 2016
- Efficient State Identification for Finite State Machine-based Testing
  - by Uraz Cengiz Turker, Robert M. Hierons, Mohammad Reza Mousavi, and Khaled El-Fakih
  - 2025
- Why do this work?
  - Finite State Machine Testing is simple
  - Still a few costly operations:
    - Reset sequences
    - Transfer sequences
- Difference in the papers
  - Building characterization sets

Difference in the papers

- Minimal Separating Sequences for All Pairs of States
  - by Rick Smetsers, Joshua Moerman, and David N. Jansen
  - 2016
- Efficient State Identification for Finite State Machine-based Testing
  - by Uraz Cengiz Turker, Robert M. Hierons, Mohammad Reza Mousavi, and Khaled El-Fakih
  - 2025
- Why do this work?
  - Finite State Machine Testing is simple
  - Still a few costly operations:
    - Reset sequences
    - Transfer sequences
- Difference in the papers
  - Building characterization sets
  - Minimize costly operations.

Difference in the papers

- Minimal Separating Sequences for All Pairs of States
  - by Rick Smetsers, Joshua Moerman, and David N. Jansen
  - 2016
- Efficient State Identification for Finite State Machine-based Testing
  - by Uraz Cengiz Turker, Robert M. Hierons, Mohammad Reza Mousavi, and Khaled El-Fakih
  - 2025
- Why do this work?
  - Finite State Machine Testing is simple
  - Still a few costly operations:
    - Reset sequences
    - Transfer sequences
- Difference in the papers
  - Building characterization sets
  - Minimize costly operations.
  - Minimal FSMs

Minimal Separating Sequences for All Pairs of States

Minimal Separating Sequences for All Pairs of States

- Splitting Tree
- Splitting Tree to characterising set
- How to get a minimal splitting tree

Splitting tree

Concepts by Moore (1956), first explicit splitting tree by Yannakakis (1994):

Splitting tree

Concepts by Moore (1956), first explicit splitting tree by Yannakakis (1994):

Definition (Splitting tree)

For a Finite State Machine M, we define the splitting tree T with a finite set of nodes such that:

- a node u in T is labelled by a subset of S, denoted $l(u)$
- the root is labelled by S
- For each inner node u, $l(u)$ is partitioned by the labels of its children
- inner node u is associated with a sequence $\sigma(u)$ separating states in the children of u.

Splitting tree

Original algorithm

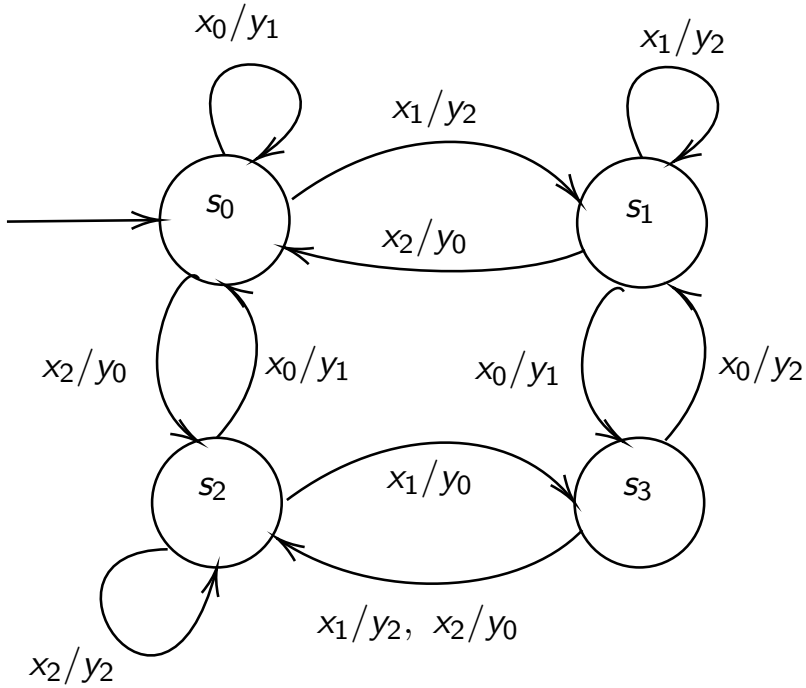
- Two ways to split:
  - split-output
    - Node is split with $a \in I$, if $\lambda(s, a) \neq \lambda(t, a)$
  - split-state
    - Node is split with $a\sigma(u)$ if we can split s and v with node after a, and that has label $\sigma(u)$

Splitting tree

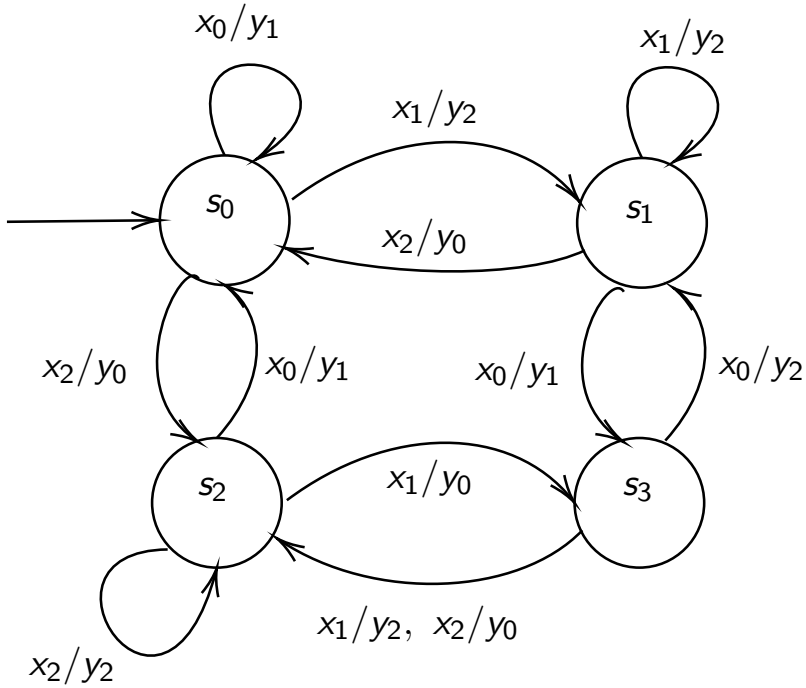
Original algorithm

- Two ways to split:
  - split-output
    - Node is split with $a \in I$, if $\lambda(s, a) \neq \lambda(t, a)$
  - split-state
    - Node is split with $a\sigma(u)$ if we can split s and v with node after a, and that has label $\sigma(u)$

Example



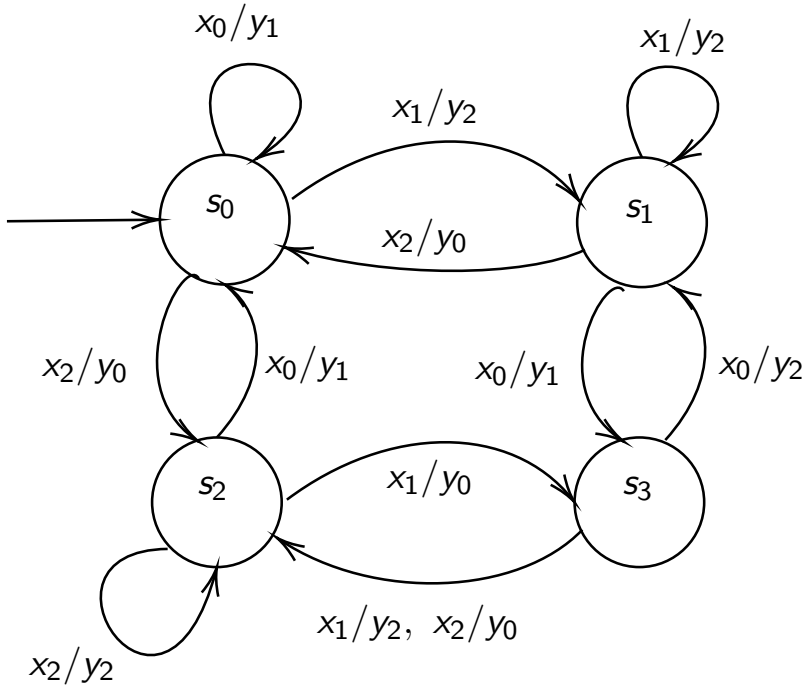
Splitting tree to characterising set



Example

$W = \{x_0, x_1, x_0x_0\}$

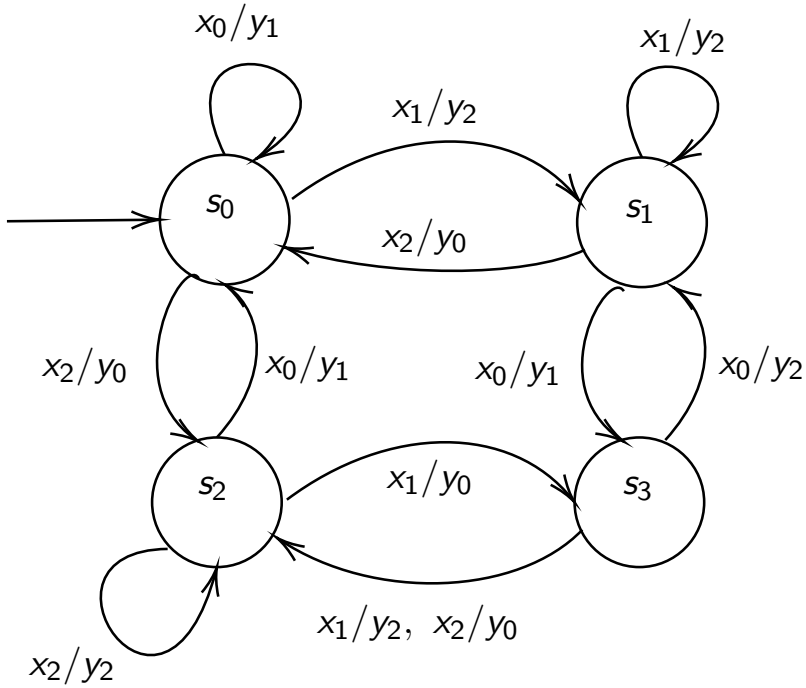
Verifying characterising set



Example

Characterising set: $W = \{x_0, x_1, x_0x_0\}$

Verifying characterising set

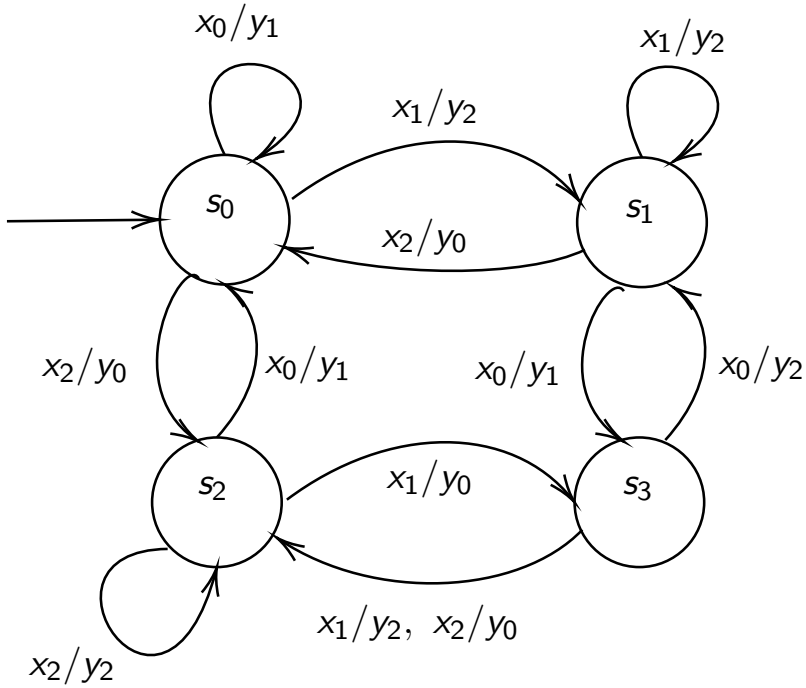


Example

Characterising set: $W = \{x_0, x_1, x_0x_0\}$

We do the checks:

Verifying characterising set



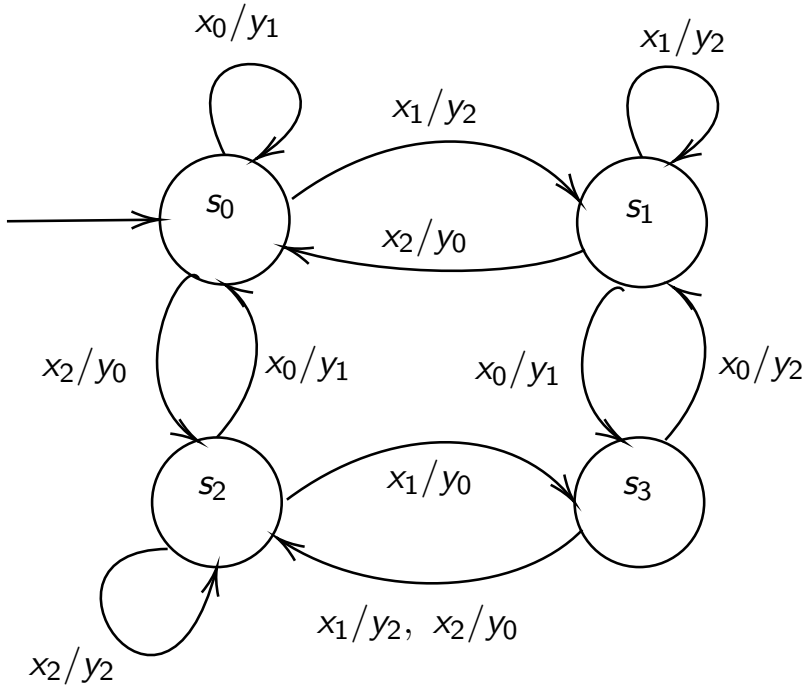
Example

Characterising set: $W = \{x_0, x_1, x_0x_0\}$

We do the checks:

$\{s_0, s_1\} : x_0x_0,$

Verifying characterising set



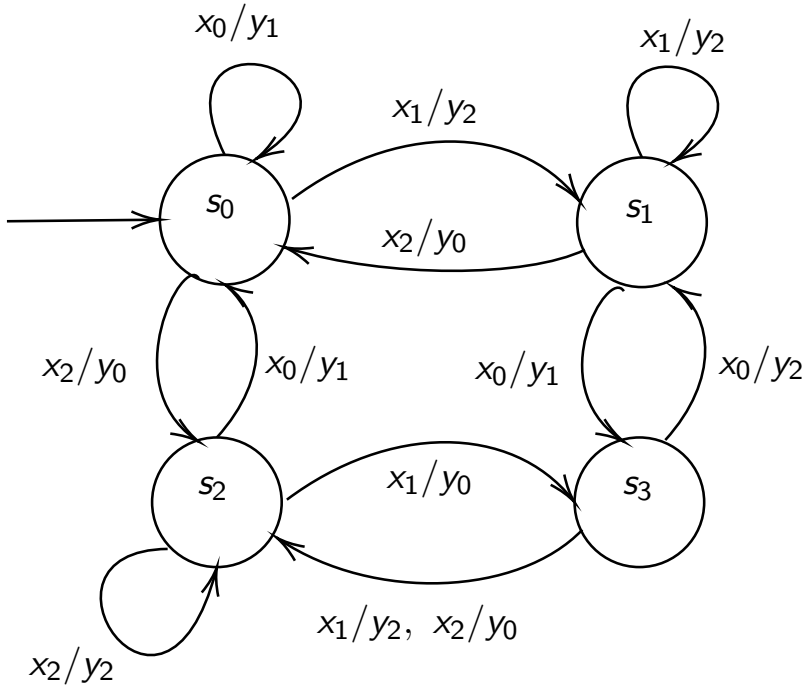
Example

Characterising set: $W = \{x_0, x_1, x_0x_0\}$

We do the checks:

$\{s_0, s_1\} : x_0x_0, \{s_0, s_2\} : x_1,$

Verifying characterising set



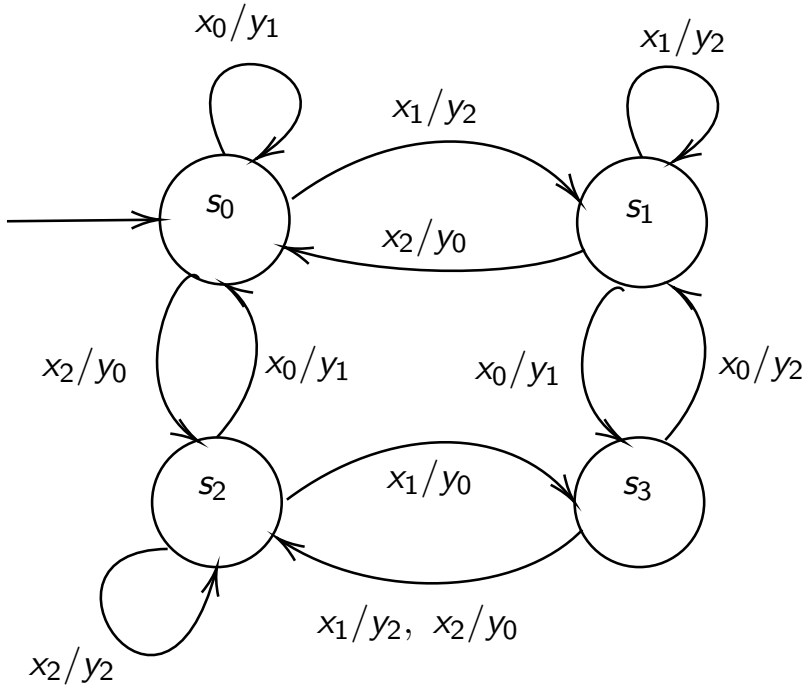
Example

Characterising set: $W = \{x_0, x_1, x_0x_0\}$

We do the checks:

$\{s_0, s_1\} : x_0x_0, \{s_0, s_2\} : x_1, \{s_0, s_3\} : x_0,$

Verifying characterising set



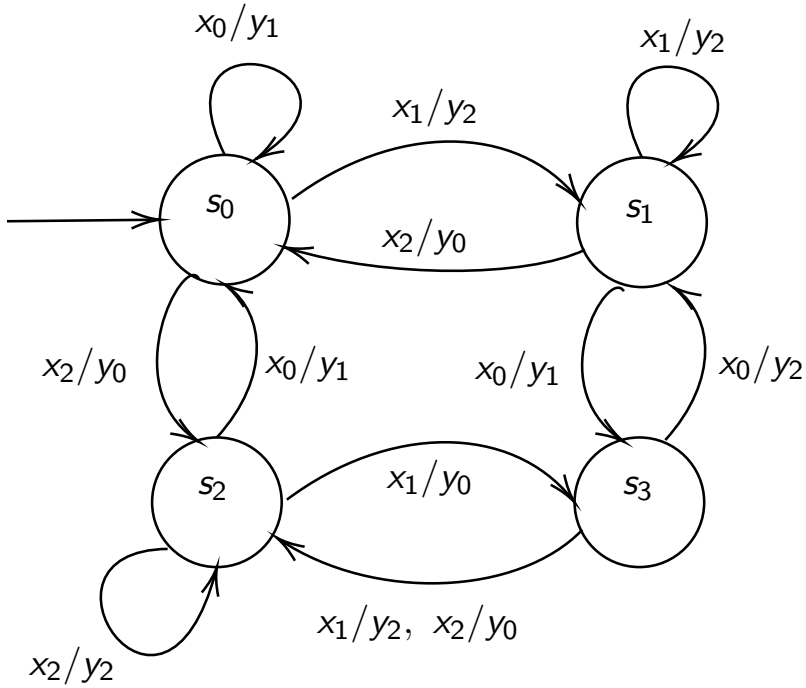
Example

Characterising set: $W = \{x_0, x_1, x_0x_0\}$

We do the checks:

$\{s_0, s_1\} : x_0x_0, \{s_0, s_2\} : x_1,\ \{s_0, s_3\} : x_0,\ \{s_1, s_2\} : x_1,$

Verifying characterising set



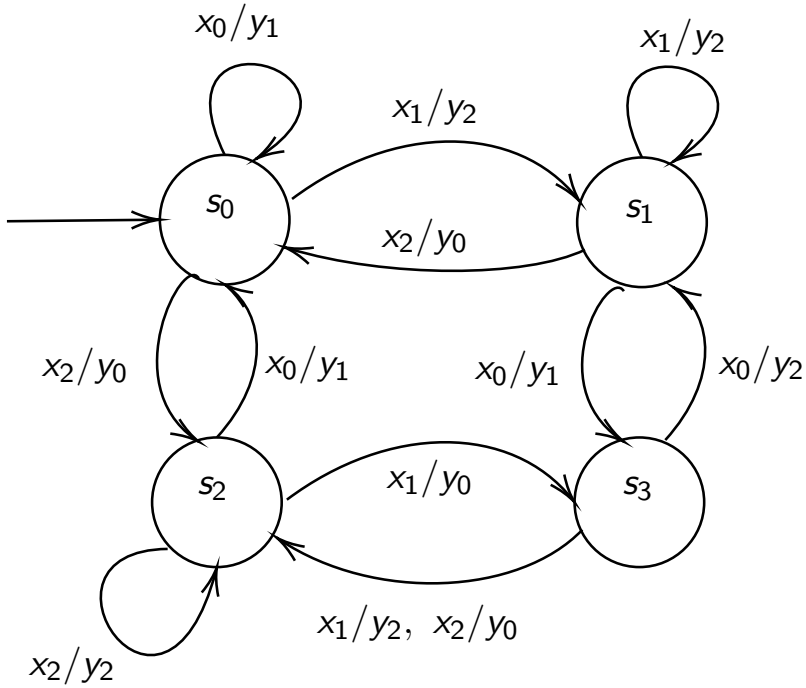
Example

Characterising set: $W = \{x_0, x_1, x_0x_0\}$

We do the checks:

$\{s_0, s_1\} : x_0x_0, \{s_0, s_2\} : x_1,\ \{s_0, s_3\} : x_0,\ \{s_1, s_2\} : x_1,\ \{s_1, s_3\} : x_0,$

Verifying characterising set



Example

Characterising set: $W = \{x_0, x_1, x_0x_0\}$

We do the checks:

$\{s_0, s_1\} : x_0x_0$, $\{s_0, s_2\} : x_1$, $\{s_0, s_3\} : x_0$, $\{s_1, s_2\} : x_1$, $\{s_1, s_3\} : x_0$, $\{s_2, s_3\} : x_0$

Lowest Common Ancestor

Definition (Lowest Common Ancestor)

The Lowest Common Ancestor (LCA) for a set of states $S' \subseteq S$ is the node u such that $S' \subseteq l(u)$, and for all children v of u: $S' \not\subseteq l(v)$.

Lowest Common Ancestor

Definition (Lowest Common Ancestor)

The Lowest Common Ancestor (LCA) for a set of states $S' \subseteq S$ is the node u such that $S' \subseteq I(u)$, and for all children v of u: $S' \not\subseteq I(v)$.

Example

For $S' := \{s_0, s_2\}$

Lowest Common Ancestor

Definition (Lowest Common Ancestor)

The Lowest Common Ancestor (LCA) for a set of states $S' \subseteq S$ is the node u such that $S' \subseteq l(u)$, and for all children v of u: $S' \not\subseteq l(v)$.

Example

For $S' := \{s_0, s_2\}$ this is $\{s_0, s_1, s_2\}$.

Lowest Common Ancestor

Definition (Lowest Common Ancestor)

The Lowest Common Ancestor (LCA) for a set of states $S' \subseteq S$ is the node u such that $S' \subseteq l(u)$, and for all children v of u: $S' \not\subseteq l(v)$.

Example

For $S' := \{s_0, s_2\}$ this is $\{s_0, s_1, s_2\}$. And for $S'' := \{s_2, s_3\}$

Lowest Common Ancestor

Definition (Lowest Common Ancestor)

The Lowest Common Ancestor (LCA) for a set of states $S' \subseteq S$ is the node u such that $S' \subseteq l(u)$, and for all children v of u: $S' \not\subseteq l(v)$.

Example

For $S' := \{s_0, s_2\}$ this is $\{s_0, s_1, s_2\}$. And for $S'' := \{s_2, s_3\}$ this is $\{s_0, s_1, s_2, s_3\}$.

Acceptable splitting tree

Definition (acceptable partition)

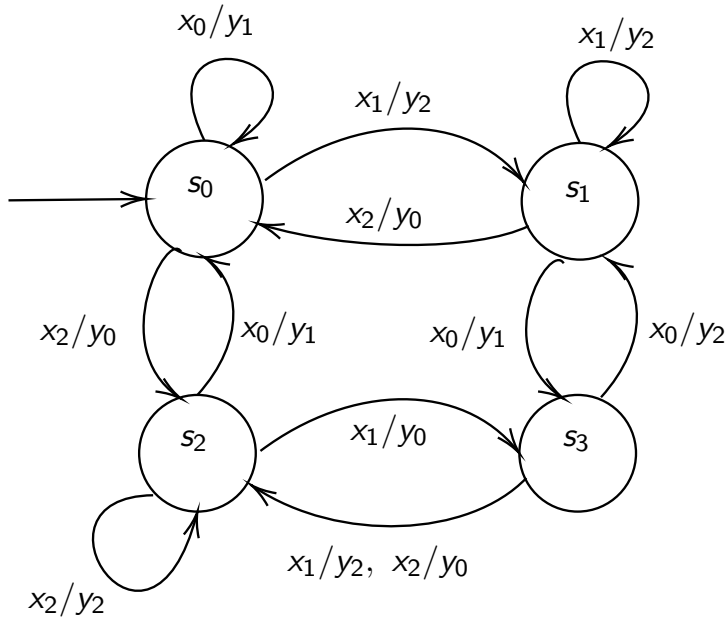
A partition is acceptable if for all s, t in the same block and for all $a \in I$, $\lambda(s, a) = \lambda(t, a)$

Acceptable splitting tree

Definition (acceptable partition)

A partition is acceptable if for all s, t in the same block and for all $a \in I$, $\lambda(s, a) = \lambda(t, a)$

Example

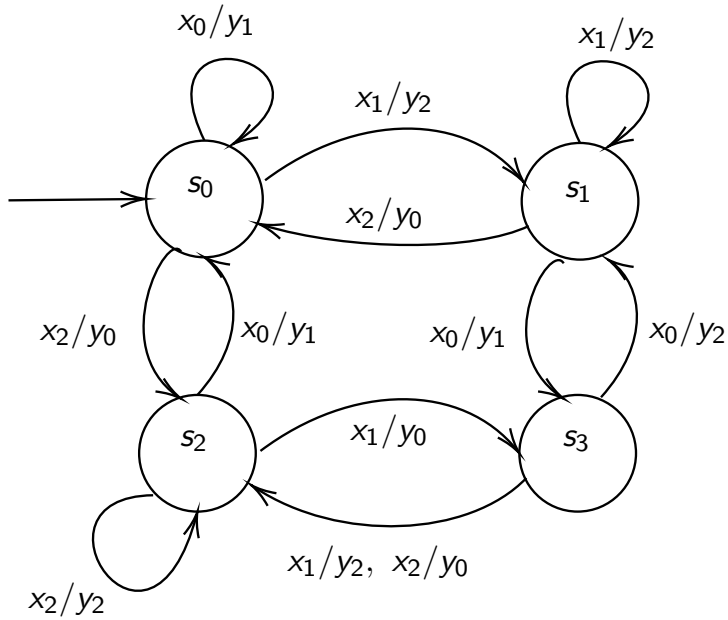


Acceptable splitting tree

Definition (acceptable partition)

A partition is acceptable if for all s, t in the same block and for all $a \in I$, $\lambda(s, a) = \lambda(t, a)$

Example



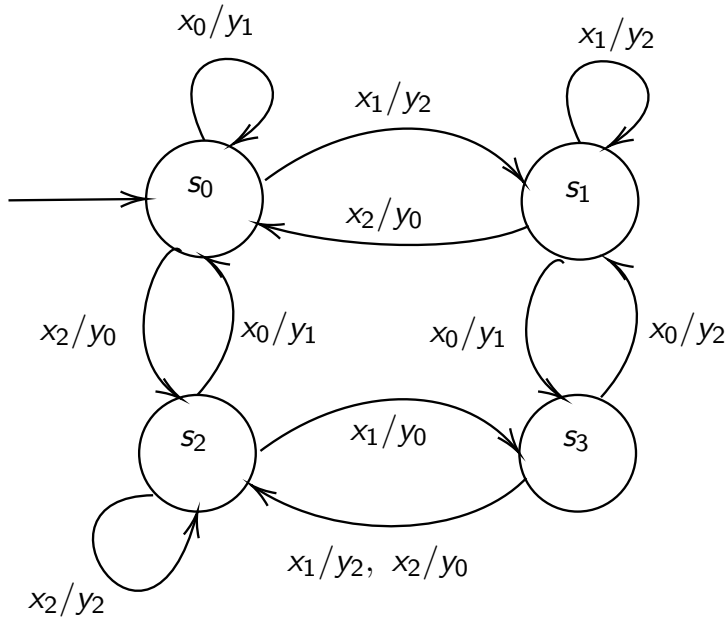
$[\{s_0, s_1, s_2\}, \{s_3\}]$ is not acceptable.

Acceptable splitting tree

Definition (acceptable partition)

A partition is acceptable if for all s, t in the same block and for all $a \in I$, $\lambda(s, a) = \lambda(t, a)$

Example



$[\{s_0, s_1, s_2\}, \{s_3\}]$ is not acceptable.
$[\{s_0, s_1\}, \{s_2\}, \{s_3\}]$ is acceptable.

Stable splitting tree

Definition (Stable partition)

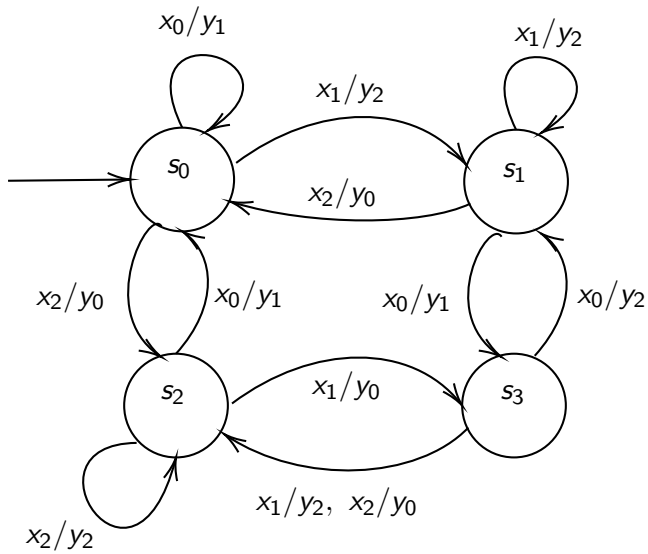
A partition is stable if it is acceptable, and for all s, t in the same block, and all $a \in I$, we have that $\delta(s, a)$ and $\delta(t, a)$ are in the same block.

Stable splitting tree

Definition (Stable partition)

A partition is stable if it is acceptable, and for all s, t in the same block, and all $a \in I$, we have that $\delta(s, a)$ and $\delta(t, a)$ are in the same block.

Example



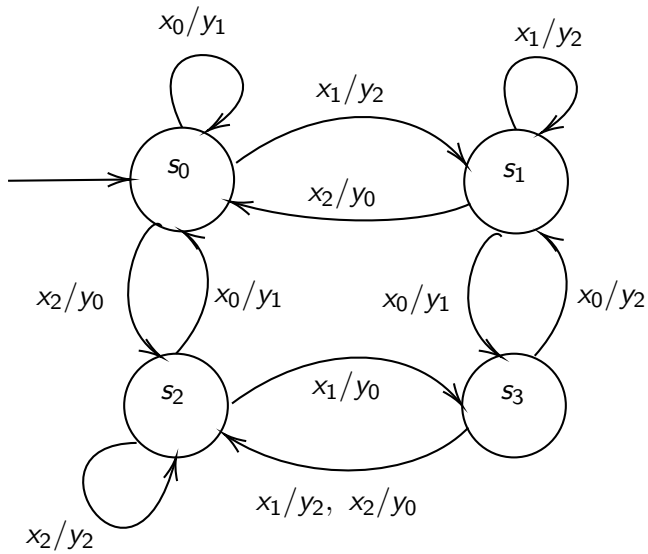
$[\{s_0, s_1\}, \{s_2\}, \{s_3\}]$ is not stable.

Stable splitting tree

Definition (Stable partition)

A partition is stable if it is acceptable, and for all s, t in the same block, and all $a \in I$, we have that $\delta(s, a)$ and $\delta(t, a)$ are in the same block.

Example



$[\{s_0, s_1\}, \{s_2\}, \{s_3\}]$ is not stable.
$[\{s_0\}, \{s_1\}, \{s_2\}, \{s_3\}]$ is stable.

k-Stable splitting tree

Definition (k-stable splitting tree)

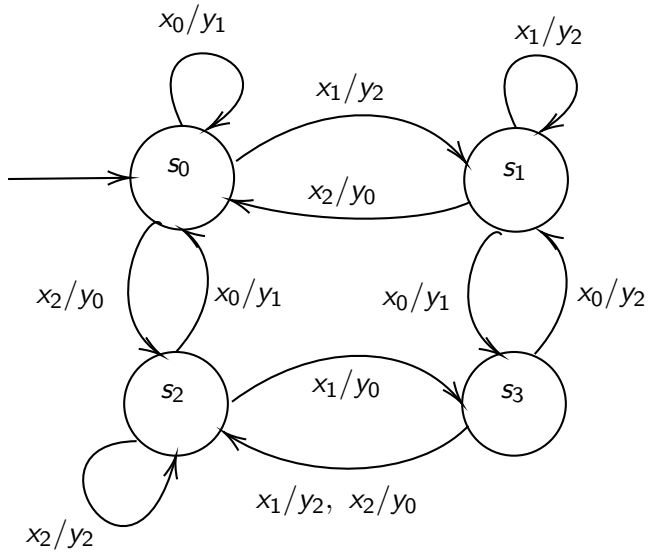
A splitting tree is k-stable if for all states s, t in the same block $\lambda(s, x) = \lambda(t, x)$ for all $x \in I^{\leq k}$

k-Stable splitting tree

Definition (k-stable splitting tree)

A splitting tree is k-stable if for all states s, t in the same block $\lambda(s, x) = \lambda(t, x)$ for all $x \in I^{\leq k}$

Example

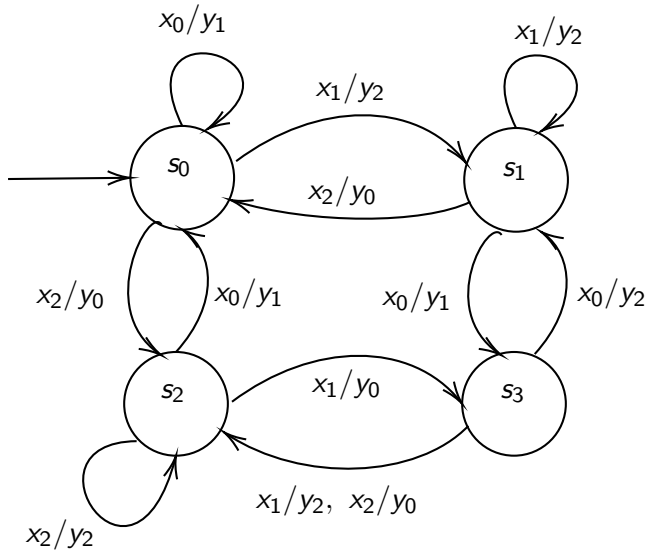


k-Stable splitting tree

Definition (k-stable splitting tree)

A splitting tree is k-stable if for all states s, t in the same block $\lambda(s, x) = \lambda(t, x)$ for all $x \in I^{\leq k}$

Example



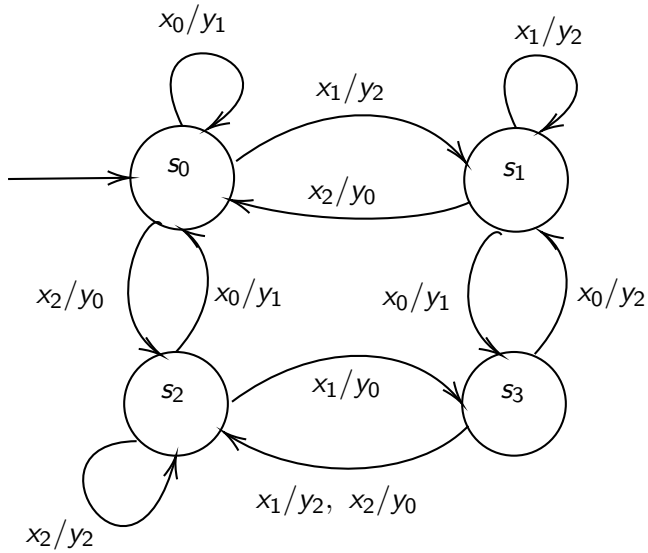
$[\{s_0, s_1\}, \{s_2\}, \{s_3\}]$ is 1-stable.

k-Stable splitting tree

Definition (k-stable splitting tree)

A splitting tree is k-stable if for all states s, t in the same block $\lambda(s, x) = \lambda(t, x)$ for all $x \in I^{\leq k}$

Example



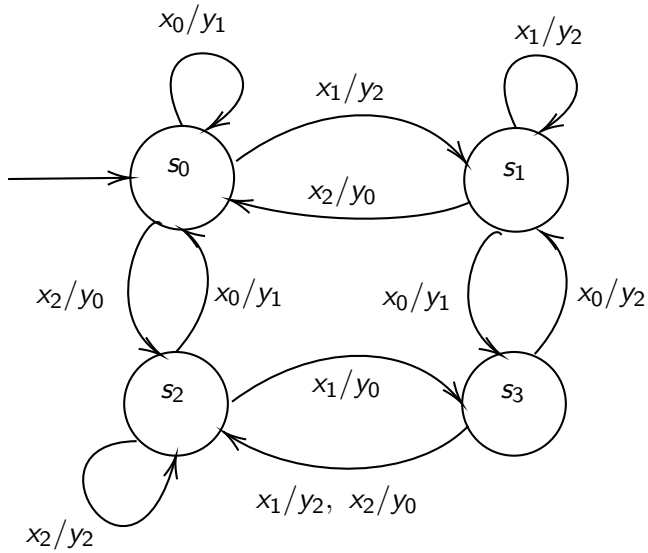
$[\{s_0, s_1\}, \{s_2\}, \{s_3\}]$ is 1-stable.
$[\{s_0, s_1\}, \{s_2\}, \{s_3\}]$ is not 2-stable.

k-Stable splitting tree

Definition (k-stable splitting tree)

A splitting tree is k-stable if for all states s, t in the same block $\lambda(s, x) = \lambda(t, x)$ for all $x \in I^{\leq k}$

Example



$[\{s_0, s_1\}, \{s_2\}, \{s_3\}]$ is 1-stable.
$[\{s_0, s_1\}, \{s_2\}, \{s_3\}]$ is not 2-stable.
$[\{s_0\}, \{s_1\}, \{s_2\}, \{s_3\}]$ is 2-stable.

Minimal splitting tree

Definition (minimal splitting tree)

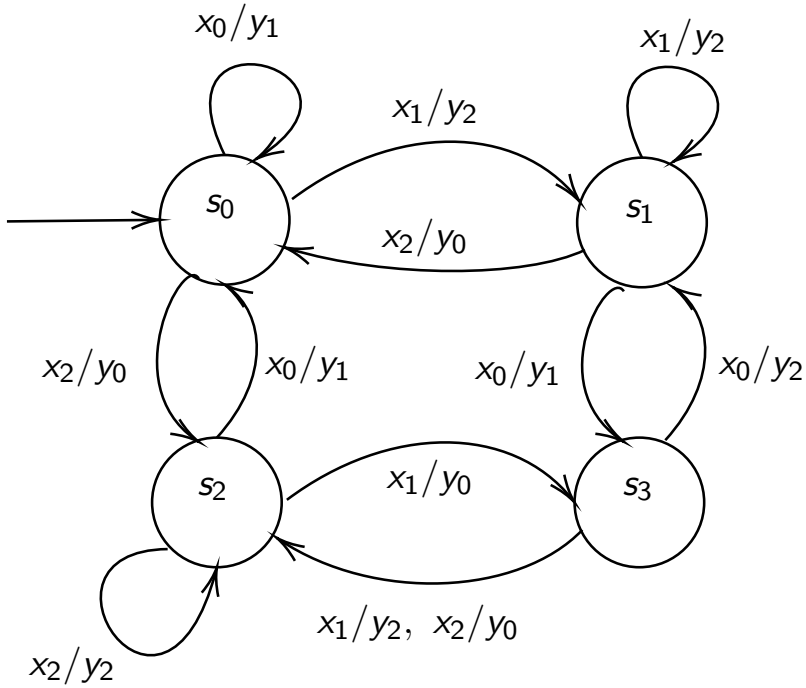
T is minimal if for all states s, t in different leaves $\lambda(s, x) \neq \lambda(t, x)$ implies $|x| \geq |\sigma(lca(s, t))|$ for all $x \in I^*$.

Minimal splitting tree

Definition (minimal splitting tree)

T is minimal if for all states s, t in different leaves $\lambda(s, x) \neq \lambda(t, x)$ implies $|x| \geq |\sigma(lca(s, t))|$ for all $x \in I^*$.

Example



How to get a Minimal splitting tree

- Get an acceptable splitting tree

How to get a Minimal splitting tree

- Get an acceptable splitting tree
- Go from a k-stable to $k+1$-stable tree.

How to get a Minimal splitting tree

- Get an acceptable splitting tree
- Go from a k-stable to $k+1$-stable tree.
  - $u \in T$, $a \in I$, $lca(\delta(\{u\}, a))$
  - $\sigma(u) = a\sigma(v)$, $v := \delta(\{u\}, a)$

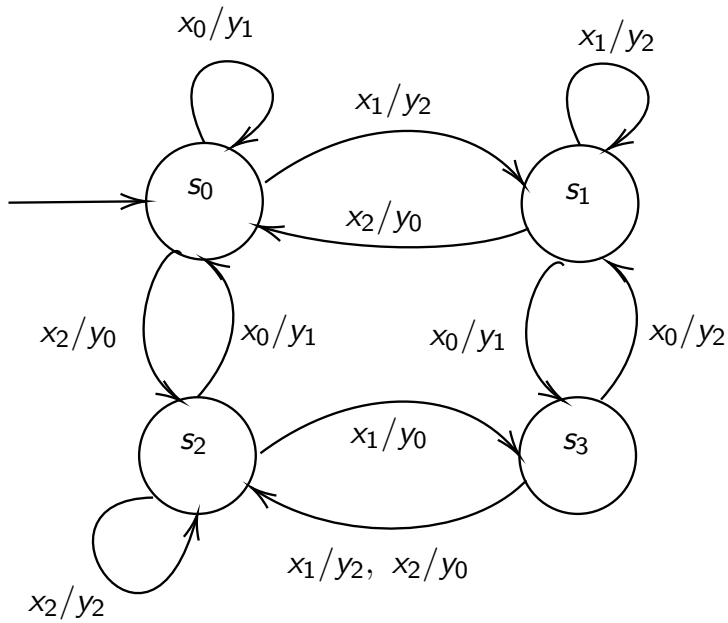
How to get a Minimal splitting tree

- Get an acceptable splitting tree
- Go from a k-stable to $k+1$-stable tree.
  - $u \in T$, $a \in I$, $lca(\delta(\{u\}, a))$
  - $\sigma(u) = a\sigma(v)$, $v := \delta(\{u\}, a)$
- Note: an acceptable splitting tree is a 1-stable tree.

How to get a Minimal splitting tree

- Get an acceptable splitting tree
- Go from a k-stable to $k+1$-stable tree.
  - $u \in T$, $a \in I$, $lca(\delta(\{u\}, a))$
  - $\sigma(u) = a\sigma(v)$, $v := \delta(\{u\}, a)$
- Note: an acceptable splitting tree is a 1-stable tree.

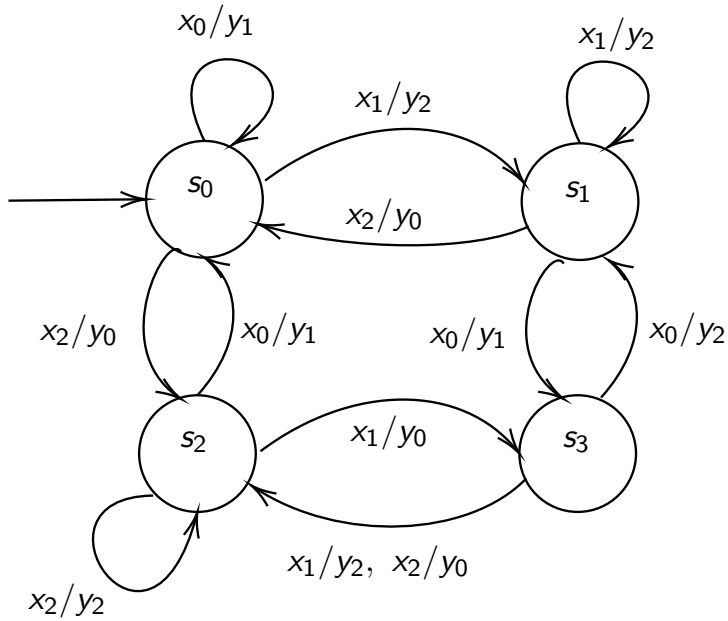
Example



How to get a Minimal splitting tree

- Get an acceptable splitting tree
- Go from a k-stable to $k+1$-stable tree.
  - $u \in T$, $a \in I$, $lca(\delta(\{u\}, a))$
  - $\sigma(u) = a\sigma(v)$, $v := \delta(\{u\}, a)$
- Note: an acceptable splitting tree is a 1-stable tree.

Example



Note that this example is small, so it yields no difference.

Complexity of the algorithm

Speed up improvements:

Complexity of the algorithm

Speed up improvements:

- Building the split-state parts bottom up

Complexity of the algorithm

Speed up improvements:

- Building the split-state parts bottom up
- Adding a data structure

Complexity of the algorithm

Speed up improvements:

- Building the split-state parts bottom up
- Adding a data structure

Complexity of the splitting tree algorithm with speed up improvements: $\mathcal{O}(m \log n)$.

Complexity of the algorithm

Speed up improvements:

- Building the split-state parts bottom up
- Adding a data structure

Complexity of the splitting tree algorithm with speed up improvements: $\mathcal{O}(m \log n)$.
It was $\mathcal{O}(m\, n)$ in Moore's version.

Efficient state identification for finite state machine-based testing

Efficient state identification for finite state machine-based testing

- Redundant Characterizing set
- Identification path
- (B)O-Wset
- Complexity of the (B)O-Wset problem

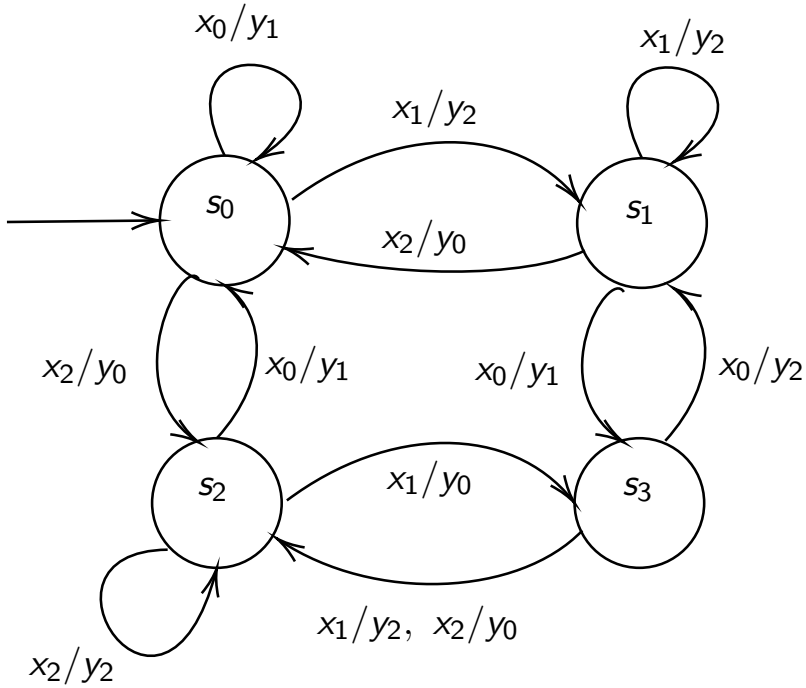
Redundant Characterizing set

Definition (Redundant Characterizing set)

A characterizing set W is redundant if either:

- $W \setminus \{w\}$ is a characterizing set for some $w \in W$.
- $W \cup \{w'\} \setminus \{w\}$ is a characterizing set for some $w \in W$ and a proper prefix w' of w.

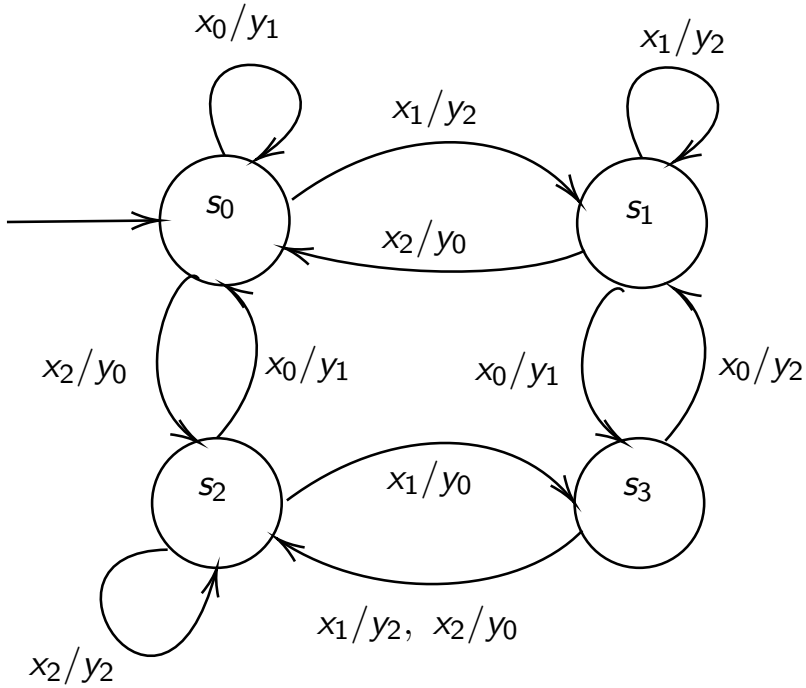
Redundant Characterizing set Example



Example

$W_1 = \{x_0x_0x_0, x_2\}$ is redundant: x_0x_0 separates s_0, s_2 and s_1 and s_3.

Redundant Characterizing set Example



Example

$W_1 = \{x_0x_0x_0, x_2\}$ is redundant: x_0x_0 separates s_0, s_2 and s_1 and s_3.

$W_2 = \{x_2, x_0x_0\}$ is non-redundant.

Identification path

- Transfer-free identification path
- State identification path with transfers

Identification path

- Transfer-free identification path
- State identification path with transfers
- Why transfer-free?

Identification path

- Transfer-free identification path
- State identification path with transfers
- Why transfer-free?
  - Transfers are often manual
  - Transfers are costly

Identification path

- Transfer-free identification path
- State identification path with transfers
- Why transfer-free?
  - Transfers are often manual
  - Transfers are costly

Example

An example of a transfer is when we have a sequence $x_1 b x_2$, and $b \notin W$.

Transfer-free identification path

Definition (Transfer-free identification path)

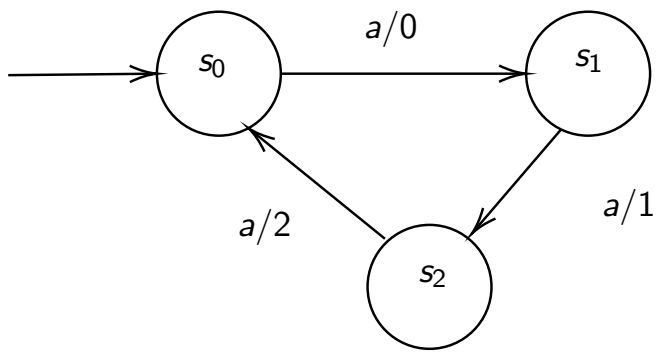
This is a path $\alpha_1\alpha_2 \ldots \alpha_k$ starting in the initial state, such that input$(\alpha_i) \in W$. Furthermore, for all $w \in W$, and for all states s, there is an α_i such that it corresponds with (s, w).

Transfer-free identification path

Definition (Transfer-free identification path)

This is a path $\alpha_1\alpha_2 \ldots \alpha_k$ starting in the initial state, such that input$(\alpha_i) \in W$. Furthermore, for all $w \in W$, and for all states s, there is an α_i such that it corresponds with (s, w).

Example



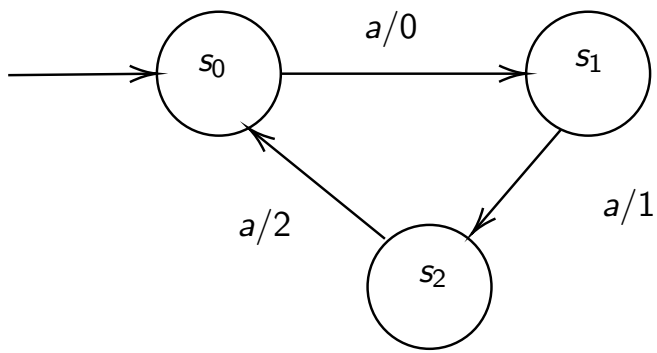
$W = \{a\}$

Transfer-free identification path

Definition (Transfer-free identification path)

This is a path $\alpha_1\alpha_2 \ldots \alpha_k$ starting in the initial state, such that input(α_i) $\in W$. Furthermore, for all $w \in W$, and for all states s, there is an α_i such that it corresponds with (s, w).

Example



$W = \{a\}$ $s_0 \rightarrow_a s_1 \rightarrow_a s_2 \rightarrow_a s_0$

State identification path with transfers

Definition (Identification path with transfers)

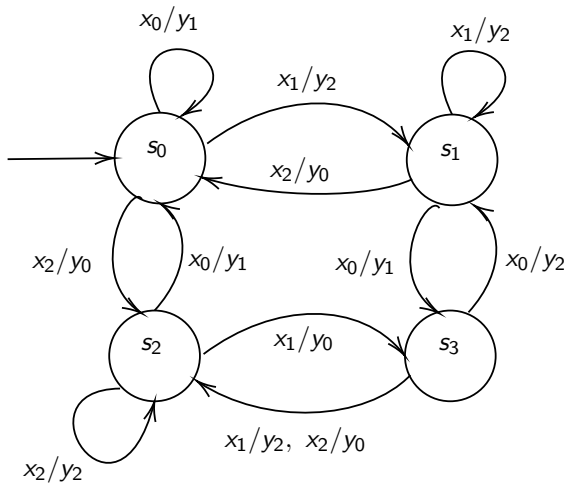
This is a path $\alpha_1\beta_1\alpha_2\beta_2 \ldots \alpha_k$ starting in the initial state, such that input(α_i) $\in W$ Furthermore, for all $w \in W$, and for all states s, there is an α_i such that it corresponds with (s, w), and β_j is a transfer sequence.

State identification path with transfers

Definition (Identification path with transfers)

This is a path $\alpha_1\beta_1\alpha_2\beta_2 \ldots \alpha_k$ starting in the initial state, such that input(α_i) $\in W$ Furthermore, for all $w \in W$, and for all states s, there is an α_i such that it corresponds with (s, w), and β_j is a transfer sequence.

Example



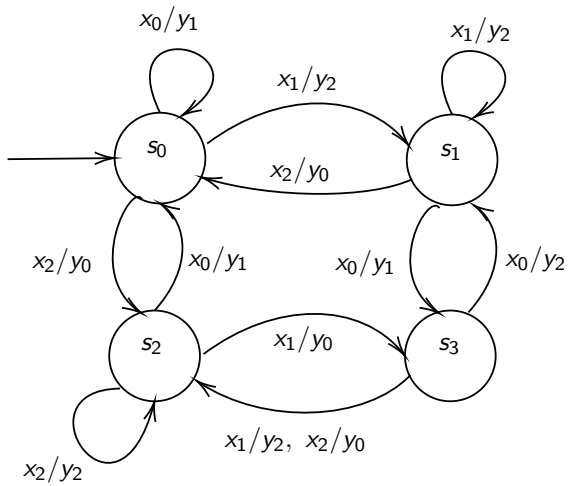
$W = \{x_1, x_2x_2\}$

State identification path with transfers

Definition (Identification path with transfers)

This is a path $\alpha_1\beta_1\alpha_2\beta_2\ldots\alpha_k$ starting in the initial state, such that input(α_i) $\in W$ Furthermore, for all $w \in W$, and for all states s, there is an α_i such that it corresponds with (s, w), and β_j is a transfer sequence.

Example



$W = \{x_1, x_2x_2\}$, $s_0 \rightarrow_{x_1} s_1 \rightarrow_{x_1} s_1 \rightarrow^*_{x_2x_2} s_2 \rightarrow_{x_1} s_3 \rightarrow_{x_1} s_2 \rightarrow^*_{x_2x_2}$
$s_2 \rightarrow_{x_0} s_0 \rightarrow^*_{x_2x_2} s_2 \rightarrow^*_{x_0x_1x_0} s_3 \rightarrow^*_{x_2x_2} s_2$

(B)O-Wset

- Minimal characterising set
- O-Wset
- BO-Wset

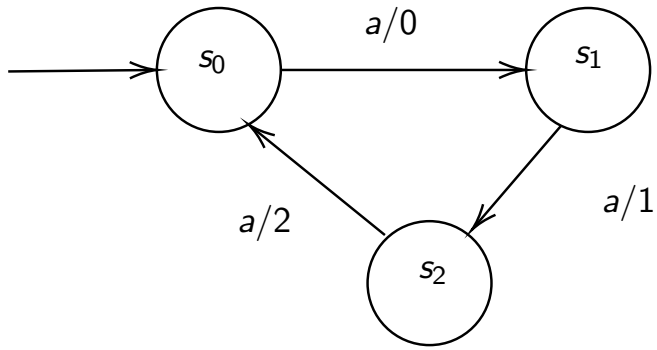
Minimal characterising set

Definition (Minimal characterising set)

A characterising set W is called minimal if for all $s, s' \in S$ where $s \neq s'$ there is a prefix w' of a sequence $w \in W$ such that:

- w' separates s and s'
- no shorter sequence separates s and s'

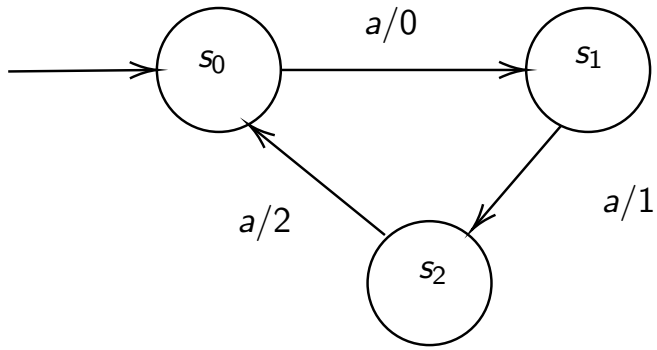
Minimal Characterizing set - Example 1



Example

- a separates s_0, s_1, s_2

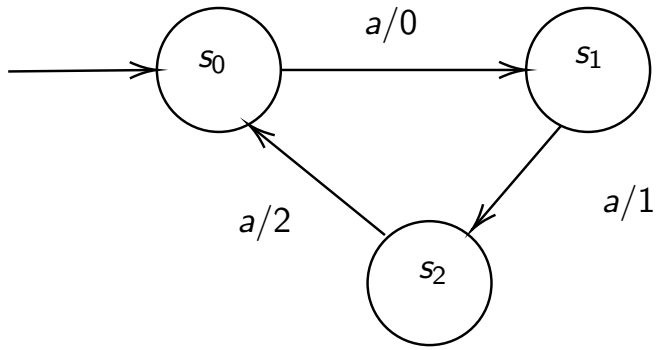
Minimal Characterizing set - Example 1



Example

- a separates s_0, s_1, s_2
- No shorter sequence separates the states

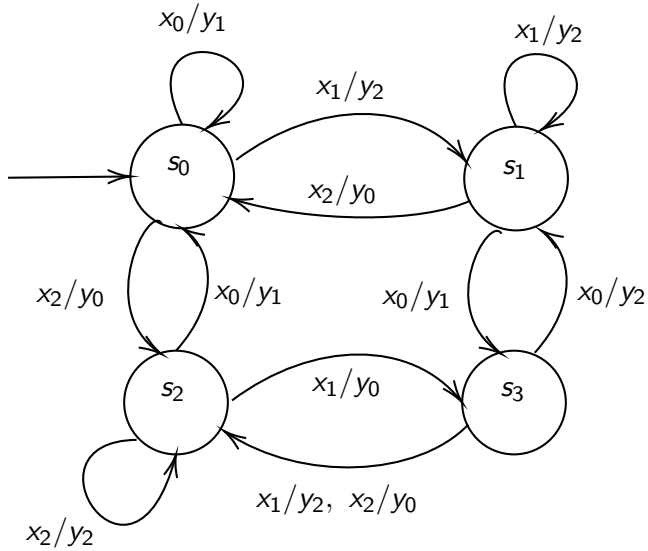
Minimal Characterizing set - Example 1



Example

- a separates s_0, s_1, s_2
- No shorter sequence separates the states
- So $W = \{a\}$ is minimal

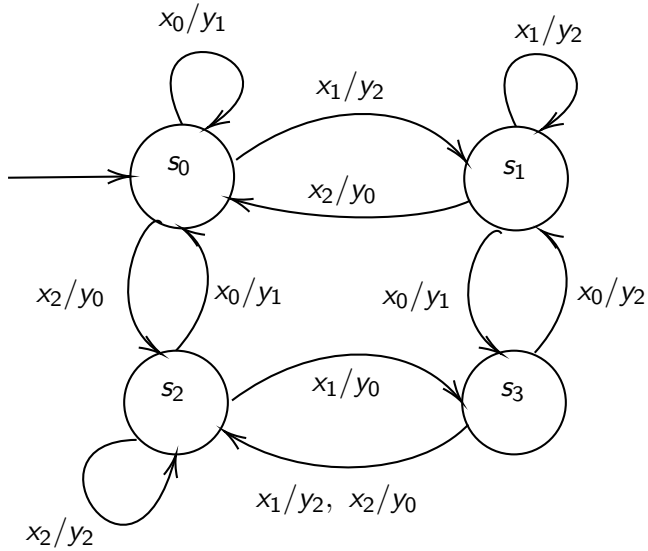
Minimal Characterizing set - Example 2



Example

$W = \{x_1, x_2x_2\}$.

Minimal Characterizing set - Example 2

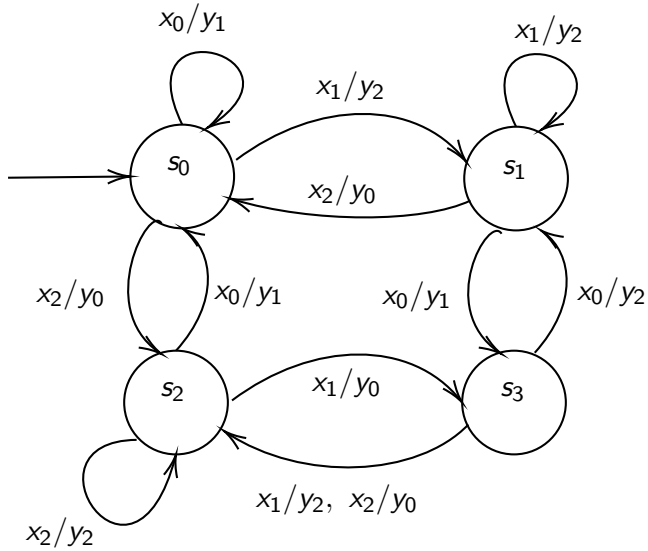


Example

$W = \{x_1, x_2x_2\}$.

- x_1 separates s_2 from s_0, s_1, s_3, and no shorter sequence does.

Minimal Characterizing set - Example 2

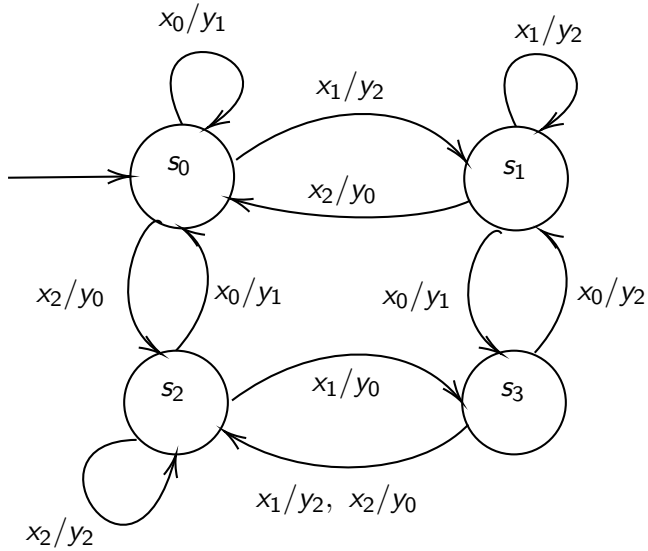


Example

$W = \{x_1, x_2x_2\}$.

- x_1 separates s_2 from s_0, s_1, s_3, and no shorter sequence does.
- x_2 separates s_3 from s_0 and s_1, and is a prefix of x_2x_2, no shorter sequence does.

Minimal Characterizing set - Example 2

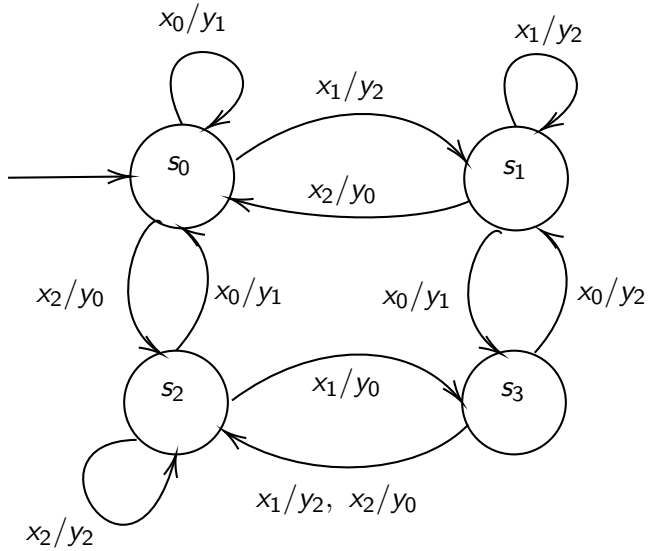


Example

$W = \{x_1, x_2x_2\}$.

- x_1 separates s_2 from s_0, s_1, s_3, and no shorter sequence does.
- x_2 separates s_3 from s_0 and s_1, and is a prefix of x_2x_2, no shorter sequence does.
- x_2x_2 separates s_0 and s_1 and no shorter sequence does.

Minimal Characterizing set - Example 2



Example

$W = \{x_1, x_2x_2\}$.

- x_1 separates s_2 from s_0, s_1, s_3, and no shorter sequence does.
- x_2 separates s_3 from s_0 and s_1, and is a prefix of x_2x_2, no shorter sequence does.
- x_2x_2 separates s_0 and s_1 and no shorter sequence does.
- So $W = \{x_1, x_2x_2\}$ is minimal.

O-Wset

Definition

W is an O-Wset if

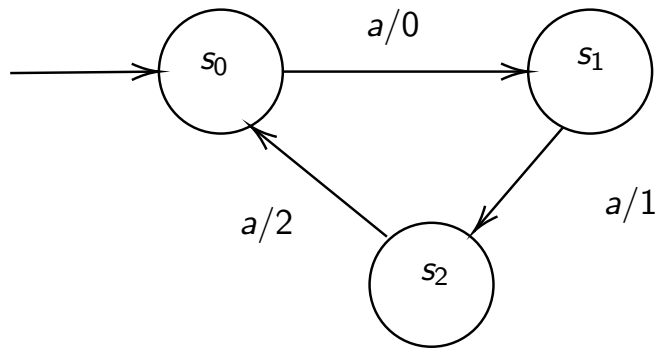
- W is a minimal characterizing set
- There exists a transfer-free state identification path for W.

O-Wset

Definition

W is an O-Wset if

- W is a minimal characterizing set
- There exists a transfer-free state identification path for W.



Example

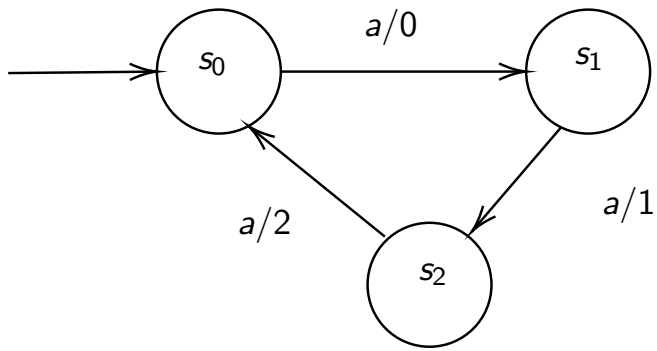
$W = \{a\}$ is an O-Wset:

O-Wset

Definition

W is an O-Wset if

- W is a minimal characterizing set
- There exists a transfer-free state identification path for W.



Example

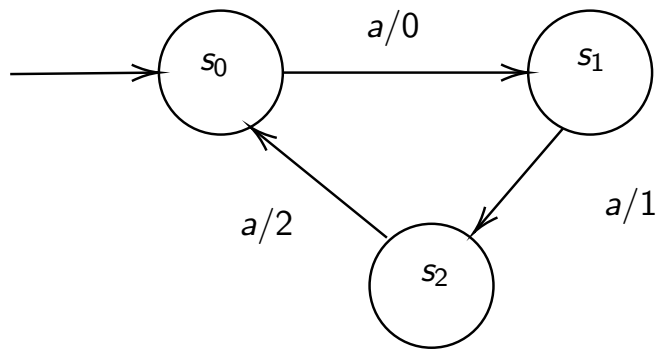
$W = \{a\}$ is an O-Wset: W is minimal

O-Wset

Definition

W is an O-Wset if

- W is a minimal characterizing set
- There exists a transfer-free state identification path for W.



Example

$W = \{a\}$ is an O-Wset: W is minimal and $s_0 \rightarrow_a s_1 \rightarrow_a s_2 \rightarrow_a s_0$ is a transfer-free path.

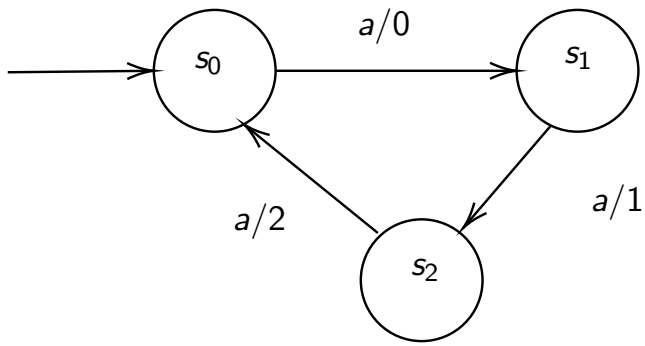
BO-Wset

Definition

W is a BO-Wset if there exists a k such that:

- W is minimal
- There exists a state identification path with at most k transfers.

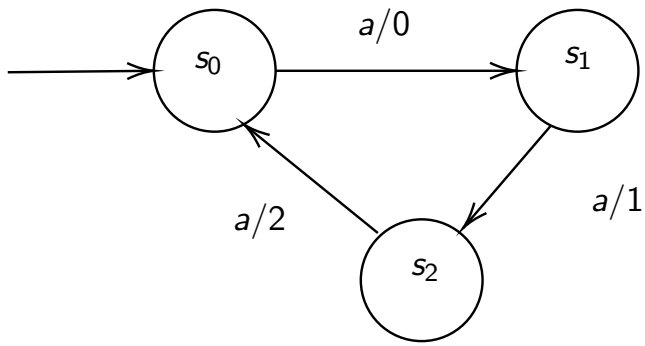
BO-Wset, $k = 0$



Example

$W = \{a\}$ is a BO-Wset:

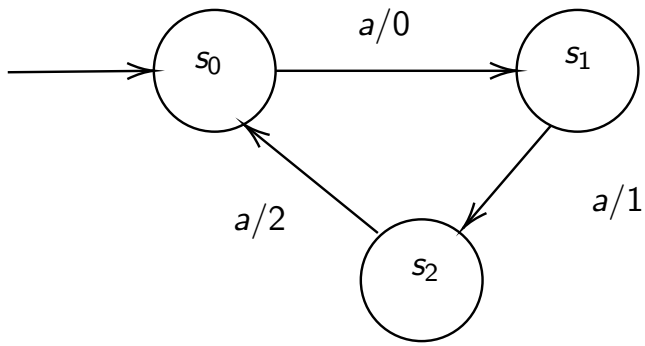
BO-Wset, $k = 0$



Example

$W = \{a\}$ is a BO-Wset: W is minimal

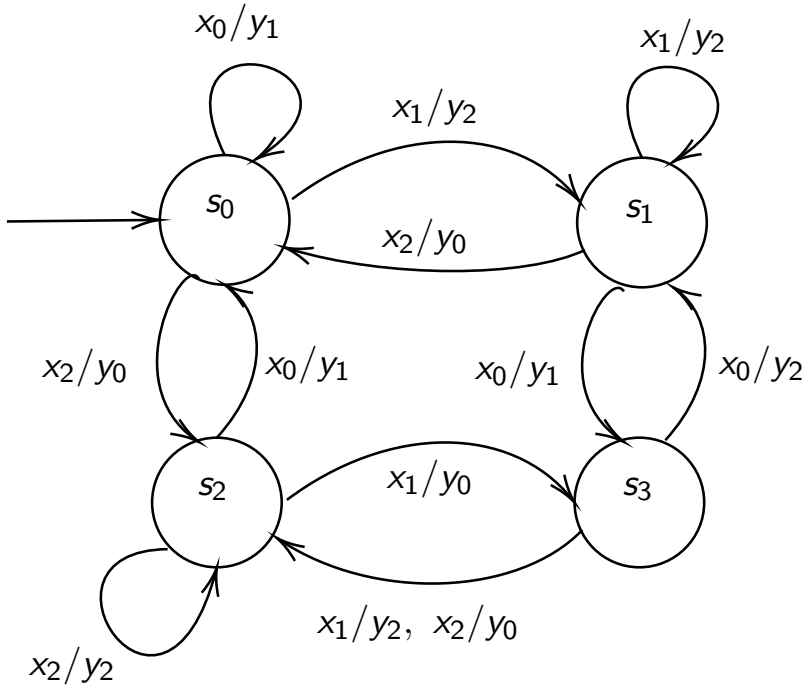
BO-Wset, $k = 0$



Example

$W = \{a\}$ is a BO-Wset: W is minimal and $s_0 \to_a s_1 \to_a s_2 \to_a s_0$ is a transfer-free path, so there are at most $k = 0$ transfers.

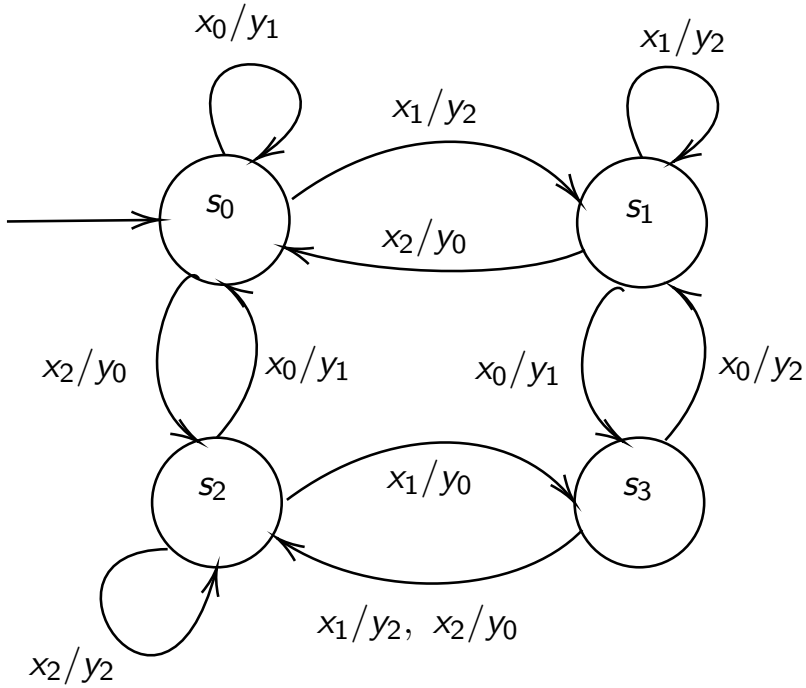
BO-Wset, $k = 4$



Example

$W = \{x_1, x_2x_2\}$ is a BO-Wset:

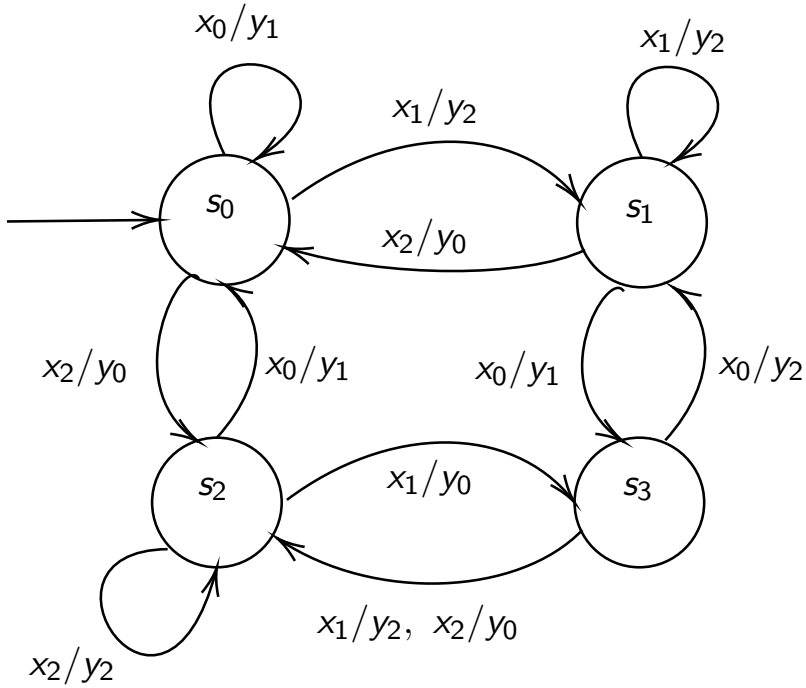
BO-Wset, $k = 4$



Example

$W = \{x_1, x_2x_2\}$ is a BO-Wset: W is minimal

BO-Wset, $k = 4$



Example

$W = \{x_1, x_2x_2\}$ is a BO-Wset: W is minimal and

$s_0 \to_{x_1} s_1 \to_{x_1} s_1 \to^*_{x_2x_2} s_2 \to_{x_1} s_3 \to_{x_1} s_2 \to^*_{x_2x_2} s_2 \to_{x_0} s_0 \to^*_{x_2x_2}$

$s_2 \to^*_{x_0x_1x_0} s_3 \to^*_{x_2x_2} s_2$ has at most $k = 4$ transfers.

Complexity of finding a (B)O-WSet

- NP-complete

Complexity of finding a (B)O-WSet

- NP-complete
- X3C

Complexity of finding a (B)O-WSet

- NP-complete
- X3C
- Mapping X3C to O-WSet

X3C

- Set of elements U

X3C

- Set of elements U
- Finite set of subsets of the elements (of size 3): C

X3C

- Set of elements U
- Finite set of subsets of the elements (of size 3): C
- Find $\mathbb{C} \subseteq C$

X3C

- Set of elements U
- Finite set of subsets of the elements (of size 3): C
- Find $\mathbb{C} \subseteq C$
  - $\bigcup_{e_i \in \mathbb{C}} e_i = U$

X3C

- Set of elements U
- Finite set of subsets of the elements (of size 3): C
- Find $\mathbb{C} \subseteq C$
  - $\bigcup_{e_i \in \mathbb{C}} e_i = U$
  - for $e_i, e_j \in \mathbb{C}$ with $i \neq j$: $e_i \cap e_j = \emptyset$

X3C

- Set of elements U
- Finite set of subsets of the elements (of size 3): C
- Find $\mathbb{C} \subseteq C$
  - $\bigcup_{e_i \in \mathbb{C}} e_i = U$
  - for $e_i, e_j \in \mathbb{C}$ with $i \neq j$: $e_i \cap e_j = \emptyset$
- Complexity: NP-complete

X3C

- Set of elements U
- Finite set of subsets of the elements (of size 3): C
- Find $\mathbb{C} \subseteq C$
  - $\bigcup_{e_i \in \mathbb{C}} e_i = U$
  - for $e_i, e_j \in \mathbb{C}$ with $i \neq j$: $e_i \cap e_j = \emptyset$
- Complexity: NP-complete

Example

$U := \{1, 2, 3, 4, 5, 6\}$, $C := \{\{1, 2, 3\}, \{2, 3, 4\}, \{4, 5, 6\}\}$.
Is there a satisfying assignment?

X3C

- Set of elements U
- Finite set of subsets of the elements (of size 3): C
- Find $\mathbb{C} \subseteq C$
  - $\bigcup_{e_i \in \mathbb{C}} e_i = U$
  - for $e_i, e_j \in \mathbb{C}$ with $i \neq j$: $e_i \cap e_j = \emptyset$
- Complexity: NP-complete

Example

$U := \{1, 2, 3, 4, 5, 6\}$, $C := \{\{1, 2, 3\}, \{2, 3, 4\}, \{4, 5, 6\}\}$.
Is there a satisfying assignment?
Yes $\mathbb{C} = \{\{1, 2, 3\}, \{4, 5, 6\}\}$

X3C

- Set of elements U
- Finite set of subsets of the elements (of size 3): C
- Find $\mathbb{C} \subseteq C$
  - $\bigcup_{e_i \in \mathbb{C}} e_i = U$
  - for $e_i, e_j \in \mathbb{C}$ with $i \neq j$: $e_i \cap e_j = \emptyset$
- Complexity: NP-complete

Example

$U := \{1, 2, 3, 4, 5, 6\}$, $C := \{\{1, 2, 3\}, \{2, 3, 4\}, \{4, 5, 6\}\}$.
Is there a satisfying assignment?
Yes $\mathbb{C} = \{\{1, 2, 3\}, \{4, 5, 6\}\}$

Example

$U := \{1, 2, 3, 4, 5, 6\}$, $C := \{\{1, 3, 5\}, \{2, 3, 4\}, \{4, 5, 6\}\}$.
Is there a satisfying assignment?

X3C

- Set of elements U
- Finite set of subsets of the elements (of size 3): C
- Find $\mathbb{C} \subseteq C$
  - $\bigcup_{e_i \in \mathbb{C}} e_i = U$
  - for $e_i, e_j \in \mathbb{C}$ with $i \neq j$: $e_i \cap e_j = \emptyset$
- Complexity: NP-complete

Example

$U := \{1, 2, 3, 4, 5, 6\}$, $C := \{\{1, 2, 3\}, \{2, 3, 4\}, \{4, 5, 6\}\}$.
Is there a satisfying assignment?
Yes $\mathbb{C} = \{\{1, 2, 3\}, \{4, 5, 6\}\}$

Example

$U := \{1, 2, 3, 4, 5, 6\}$, $C := \{\{1, 3, 5\}, \{2, 3, 4\}, \{4, 5, 6\}\}$.
Is there a satisfying assignment?
No.

X3C to O-Wset

Theorem

There is a satisfying assignment $\mathbb{C}$ for a given X3C problem iff there is a transfer-free path in the FSM M.

X3C to O-Wset

Theorem

There is a satisfying assignment $\mathbb{C}$ *for a given X3C problem iff there is a transfer-free path in the FSM M. M is defined by the mapping* $S := \{s_i | u_i \in U\} \cup \{s_{|U|+1}\}$, $I := \{x_i | c_i \in C\}$,

$O := \{y_i | u_i \in U\} \cup \{0\}$,

X3C to O-Wset

Theorem

There is a satisfying assignment $\mathbb{C}$ *for a given X3C problem iff there is a transfer-free path in the FSM M. M is defined by the mapping* $S := \{s_i | u_i \in U\} \cup \{s_{|U|+1}\}$, $I := \{x_i | c_i \in C\}$, $O := \{y_i | u_i \in U\} \cup \{0\}$, *and* $\lambda(s_i, x_j) = \begin{cases} y_i \text{ if } u_i \in e_j \\ 0 \text{ otherwise} \end{cases}$,

X3C to O-Wset

Theorem

There is a satisfying assignment $\mathbb{C}$ *for a given X3C problem iff there is a transfer-free path in the FSM M. M is defined by the mapping* $S := \{s_i | u_i \in U\} \cup \{s_{|U|+1}\}$, $I := \{x_i | c_i \in C\}$, $O := \{y_i | u_i \in U\} \cup \{0\}$, *and* $\lambda(s_i, x_j) = \begin{cases} y_i \text{ if } u_i \in e_j \\ 0 \text{ otherwise} \end{cases}$, *and*

$$\delta(s_i, x_j) = \begin{cases} s_{i+1} \text{ if } u_i \in e_j \\ s_i \text{ otherwise} \end{cases}.$$

X3C to O-Wset

Theorem

There is a satisfying assignment $\mathbb{C}$ *for a given X3C problem iff there is a transfer-free path in the FSM M. M is defined by the mapping* $S := \{s_i | u_i \in U\} \cup \{s_{|U|+1}\}$, $I := \{x_i | c_i \in C\}$,

$O := \{y_i | u_i \in U\} \cup \{0\}$, *and* $\lambda(s_i, x_j) = \begin{cases} y_i \text{ if } u_i \in e_j \\ 0 \text{ otherwise} \end{cases}$, *and*

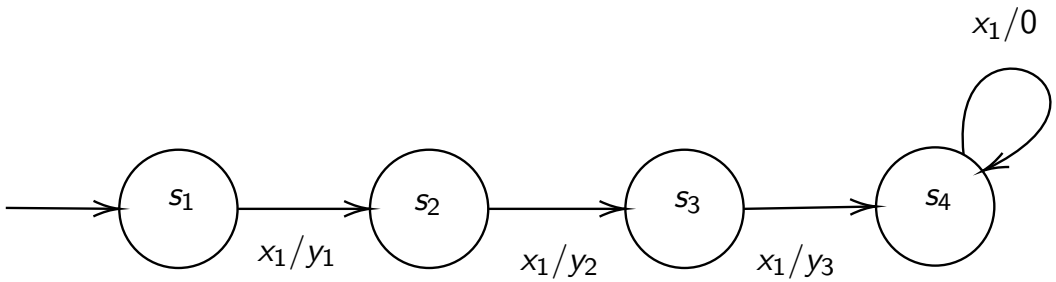
$\delta(s_i, x_j) = \begin{cases} s_{i+1} \text{ if } u_i \in e_j \\ s_i \text{ otherwise} \end{cases}$.

Example

$U = \{u_1, u_2, u_3\}$, $C = \{e_1\} = \{\{u_1, u_2, u_3\}\}$.

Mapping X3C to O-Wset

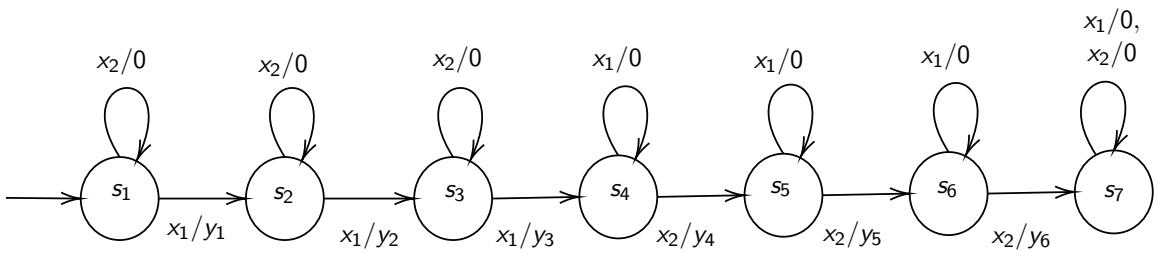
$U = \{u_1, u_2, u_3\}$, $S = \{s_1, s_2, s_3, s_4\}$, $C = \{e_1\} = \{\{u_1, u_2, u_3\}\}$,
$I = \{x_1\}$,
$O = \{y_1, y_2, y_3, 0\}$.



Transition for state s_1: For x_1, u_1 in e_1 => goes to s_2, with output y_1 as it is state 1.
Transition for state s_2: For x_1, u_1 in e_1 => goes to s_3,with output y_2 as it is state 2.
Transition for state s_3: For x_1, u_1 in e_1 => goes to s_4,with output y_3 as it is state 3.
Transition for state s_4: For x_1, u_1 not in e_1 => goes to s_4, with output 0.

X3C to O-Wset - path

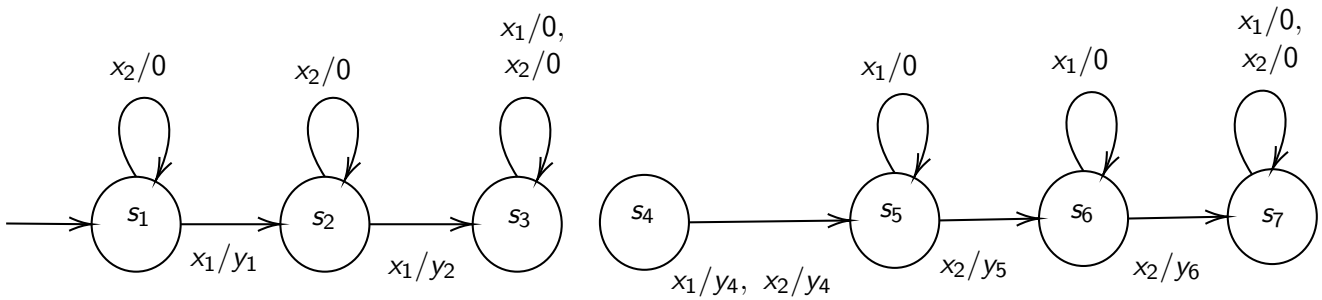
$U = \{u_1, u_2, u_3, u_4, u_5, u_6\}$,
$C = \{e_1, e_2\} = \{\{u_1, u_2, u_3\}, \{u_4, u_5, u_6\}\}$,
$S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$,
$I = \{x_1, x_2\}$, $O = \{y_1, y_2, y_3, y_4, y_5, y_6, 0\}$



$W := \{x_1, x_2\}$.
Path is : $s_1 \rightarrow_{x_2} s_1 \rightarrow_{x_1} s_2 \rightarrow_{x_2} s_2 \rightarrow_{x_1} s_3 \rightarrow_{x_2} s_3 \rightarrow_{x_1} s_4 \rightarrow_{x_1}$
$s_4 \rightarrow_{x_2} s_5 \rightarrow_{x_1} s_5 \rightarrow_{x_2} s_6 \rightarrow_{x_1} s_6 \rightarrow_{x_2} s_7 \rightarrow_{x_1} s_7 \rightarrow_{x_2} s_7$

X3C to O-Wset - path

$U = \{u_1, u_2, u_3, u_4, u_5, u_6\}$,
$C = \{e_1, e_2\} = \{\{u_1, u_2, u_4\}, \{u_4, u_5, u_6\}\}$,
$S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$,
$I = \{x_1, x_2\}$, $O = \{y_1, y_2, y_3, y_4, y_5, y_6, 0\}$



No path.

NP completeness for (B)O-Wset

- Mapping from X3C to O-Wset

NP completeness for (B)O-Wset

- Mapping from X3C to O-Wset
- X3C to BO-Wset

Comparison of the papers

Recap: Minimal splitting tree or (B)O-Wset?

Comparison of the papers

Recap: Minimal splitting tree or (B)O-Wset?
When should we use a splitting tree and when is a (B)O-Wset better?

Comparison of the papers

Recap: Minimal splitting tree or (B)O-Wset?
When should we use a splitting tree and when is a (B)O-Wset better?
Resets needed for splitting tree.

Comparison of the papers

Recap: Minimal splitting tree or (B)O-Wset?
When should we use a splitting tree and when is a (B)O-Wset better?
Resets needed for splitting tree.
Resets can be costly.

Comparison of the papers

Recap: Minimal splitting tree or (B)O-Wset?
When should we use a splitting tree and when is a (B)O-Wset better?
Resets needed for splitting tree.
Resets can be costly.
(B)O-Wset yield a long path for all states, splitting tree small ones for every pair of states.

Comparison of the papers

Recap: Minimal splitting tree or (B)O-Wset?
When should we use a splitting tree and when is a (B)O-Wset better?
Resets needed for splitting tree.
Resets can be costly.
(B)O-Wset yield a long path for all states, splitting tree small ones for every pair of states.
Is a combination possible?

Conclusion

- Splitting Tree
- Splitting Tree to characterising set
- (B)O-Wset
- Comparison of methods