June 26,

2017

Synchronizing and Homing Experiments for Input/Output Automata

Natalia Kushik
(SAMOVAR, CNRS, Télécom SudParis / Université Paris-Saclay)

Nina Yevtushenko
(Tomsk State University,
Institute for System Programming of the Russian Academy of Sciences)

Igor Burdonov
(Institute for System Programming of the Russian Academy of Sciences)

Alexandr Kossatchev
(Institute for System Programming of the Russian Academy of Sciences)

Outline

- Motivation
- Introducing the class of Input/Output automata we are dealing with
- Defining homing and synchronizing sequences for Input/Output automata
- Deriving synchronizing sequences
- Deriving homing sequences
- Conclusions & future work

Motivation

Just now mostly theoretical interest ☺

o Finite State Machines and synchronizing/homing experiments with them are well studied and are widely used for test derivation

However…

o Not each reactive system produces an output immediately after applying an input

o Synchronizing/homing experiments with Input/Output Automata should be also considered

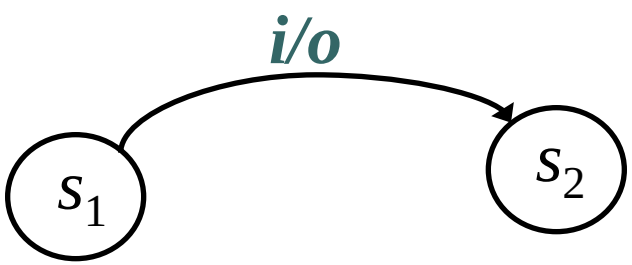
⇓

We define the synchronizing and homing sequences for Input/Output Automata and propose how to derive them

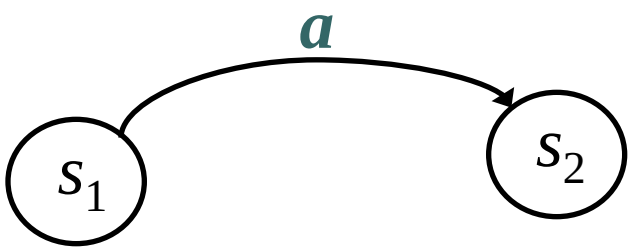
Finite State Machines and Automata

- *Finite state machines* (*FSMs*) and *Automata* describe the behavior of discrete event systems
- FSMs work in request/response mode

- FSM transitions are labeled with input/output pairs



FSM transition



Automata transition

- Both models are widely used for test generation and verification of software and hardware systems

Finite State Machine

$\boldsymbol{S} = (S, I, O, T_S)$ is a FSM

- S is a finite nonempty set of states
- I and O are finite input and output alphabets
- $T_S \subseteq S \times (I \times O) \times S$ is transition relation

Input/Output Automaton

$S = (S, I, O, T_S)$ is an I/O automaton

- S is a finite nonempty set of states
- I and O are finite input and output alphabets, $I \cap O = \varnothing$
- $T_S \subseteq S \times I \times S \cup S \times O \times S$ is transition relation

- $S = S_1 \cup S_2$, $S_1 \cap S_2 = \varnothing$ and $T_S \subseteq S_1 \times I \times S \cup S_2 \times O \times S$
- NO loops labeled with outputs
- $\exists\, \delta \notin O\ \forall s \in S_1$ $(s, \delta, s) \in T_S$

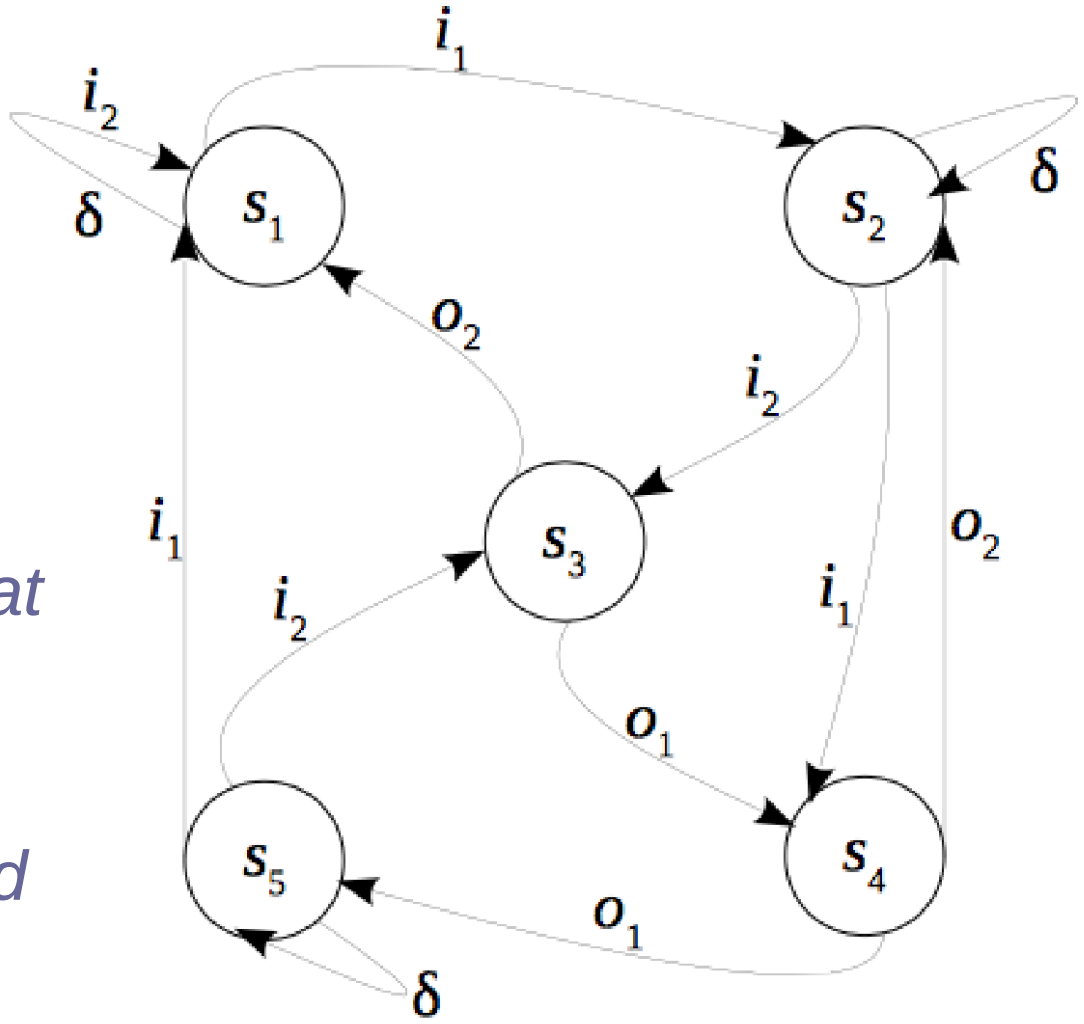
Some example...

- $S = \{s_1, \ldots, s_5\}$
- $S_1 = \{s_1, s_2, s_5\}$
- $S_2 = \{s_3, s_4\}$

$\Downarrow$

Inputs are accepted at s_1, s_2, s_5

Outputs are produced at s_3, s_4



When testing against FSMs…

1) Reaching each FSM state *s*

2) Traversing a transition to check the output and final state *s′*

3) Distinguishing the final state *s′* of the transition from any other FSM state

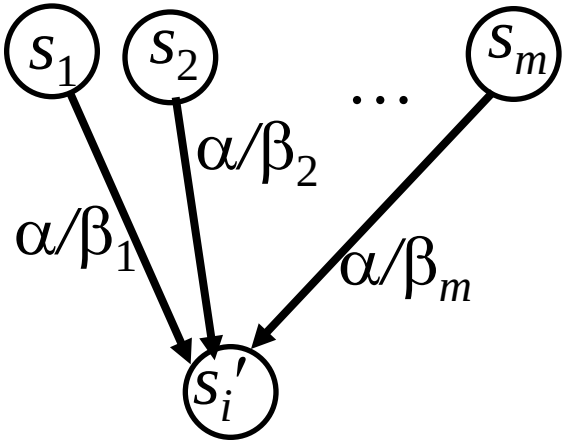
The reachability issue can be solved via an application of the homing or synchronizing sequence

FSM

Synchronizing sequence (SS)

- The sequence $\alpha = i_1 i_2 \ldots i_k$ allows to recognize the current state of the machine under experiment without observing the output response
- After applying α at any state s_i the final state s_i' becomes known

Synchronizing sequence α



Independently of $\beta_i = o_1 o_2 \ldots o_k$

FSM

Synchronizing sequence (SS)

γ-successor of the state $s \in S$ is the set of states that can be reached from state s through the trace γ

while the γ-successor of S is the set of states that can be reached from some state of S through the trace γ

FSM

Synchronizing sequence (SS)

$\alpha = i_1 i_2 \ldots i_k$ is ***synchronizing***

if

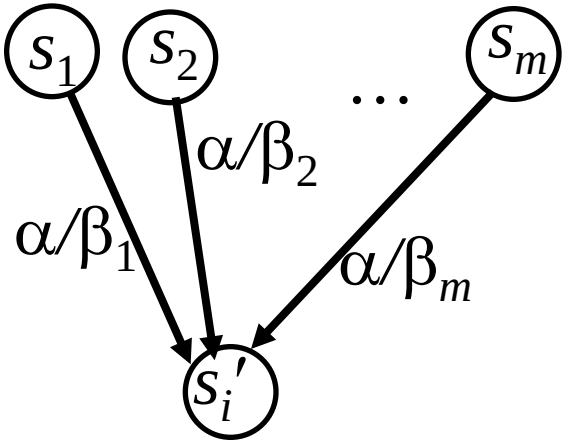
α **is defined at all states of** S

and $\exists\ s \in S$ **such that**

$\forall\ \gamma = i_1 o_1 i_2 o_2 \ldots i_k o_k$, **where**
$o_j \in O, j = 0, \ldots, k$

it holds that γ-state-after-S
is either empty
or is equal to $\{s\}$

Synchronizing sequence α



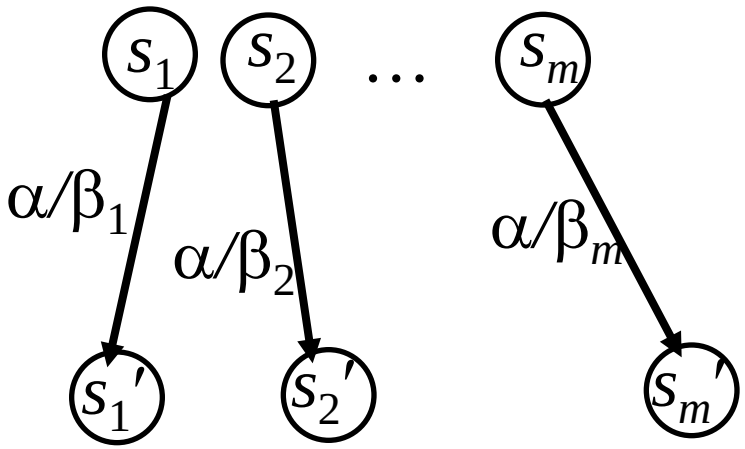
Independently of $\beta_i = o_1 o_2 \ldots o_k$

FSM

Homing sequence (HS)

- The sequence $\alpha = i_1 i_2 \ldots i_k$ allows to recognize the current state of the machine under experiment after observing the output response
- After applying α at any state s_i and observing an output response β_i the final state s_i' becomes known

Homing sequence α



$$\beta_1 \neq \beta_2 \neq \ldots \neq \beta_m$$

FSM

Homing sequence (HS)

$\alpha = i_1 i_2 \ldots i_k$ is ***homing***

if

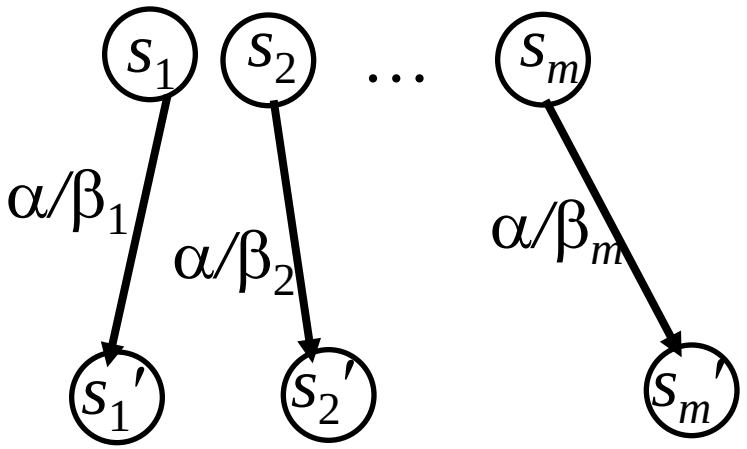
α **is defined at all states of** S

and $\exists\ s \in S$ **such that**

$\forall\ \gamma = i_1 o_1 i_2 o_2 \ldots i_k o_k$, **where**
$o_j \in O,\ j = 0, \ldots, k$

it holds that γ-state-after-S
is either empty
or is a singleton $\{ s_i' \}$

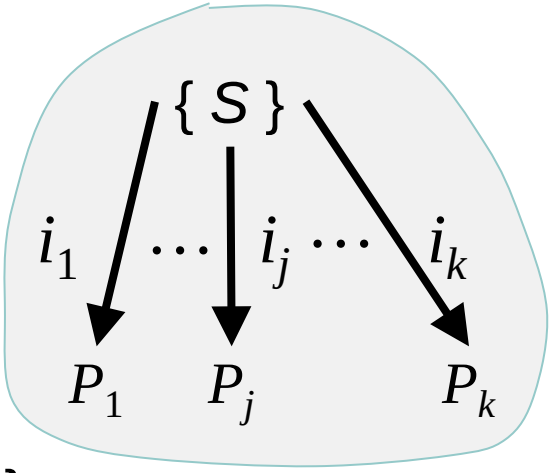
Homing sequence α



$\beta_1 \neq \beta_2 \neq \ldots \neq \beta_m$

FSM

‘Classical’ approach for SS/HS derivation for FSMs



Derive a successor tree (ST)

$s \in S$: *io*-successor $s =_{def} \{ s' \mid (s, i, o, s') \in T_S \}$

$R \subseteq S$: *io*-successor $R =_{def} \cup \{$ *io*-successor $s \mid s \in R \}$

$R \subseteq S$: *i*-successor $R =_{def} \{$ *io*-successor $R \neq \varnothing \mid o \in O \}$

$P \subseteq 2^S$: *i*-successor $P =_{def} \cup \{$ *i*-successor $R \mid R \in P \} =$

$= \{$ *io*-successor $R \neq \varnothing \mid R \in P \ \&\ o \in O \}$

$P \subseteq 2^S$, $P \xrightarrow{i}$ *i*-successor P **iff** $\forall R \in P$ *i*-successor $p \neq \varnothing$

FSM

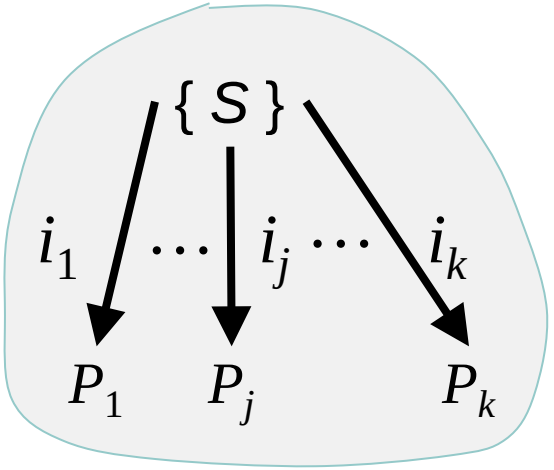
‘Classical’ approach for SS derivation for FSMs

Derive a truncated successor tree (ST)

$s \in S$: *io*-successor $s =_{\text{def}} \{ s' \mid (s, i, o, s') \in T_s \}$

$R \subseteq S$: *io*-successor $R =_{\text{def}} \cup \{$ *io*-successor $s \mid s \in R \}$

$P \subseteq 2^S$: *i*-successor $P =_{\text{def}} \cup \{$ *io*-successor $R \neq \varnothing \mid R \in P \ \& \ o \in O \}$



$P \subseteq 2^S$, $P \text{—}i\text{→}$ *i*-successor P **iff** $\forall R \in P$ *i*-successor $p \neq \varnothing$

Truncating rules:

Rule 1 P has only one singleton $\exists s \in S \quad P = \{ \{ s \} \}$

Rule 2 Set P coincides with the set P' that is met ‘upper’ in the tree

α is a synchronizing sequence iff it labels the path truncated by Rule 1

FSM

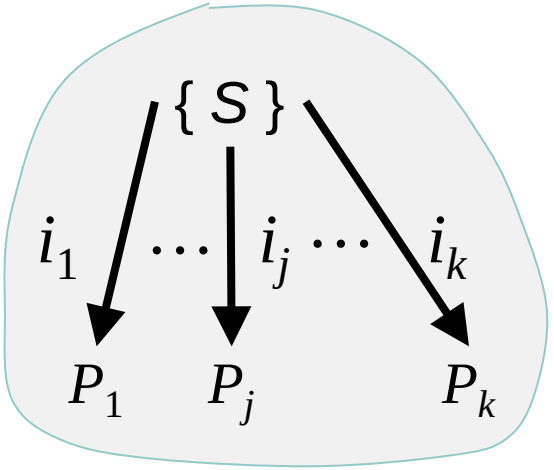
‘Classical’ approach for HS derivation for FSMs

Derive a truncated successor tree (ST)

$s \in S$: *io*-successor $s =_{\text{def}} \{ s' \mid (s, i, o, s') \in T_S \}$

$R \subseteq S$: *io*-successor $R =_{\text{def}} \cup \{$ *io*-successor $s \mid s \in R \}$

$P \subseteq 2^S$: *i*-successor $P =_{\text{def}} \cup \{$ *io*-successor $R \neq \varnothing \mid R \in P \ \&\ o \in O \}$



$P \subseteq 2^S$, $P \text{—}i\rightarrow$ *i*-successor P **iff** $\forall R \in P$ *i*-successor $p \neq \varnothing$

Truncating rules:

Rule 1 P has only singletons $\forall p \in P \quad \exists s \in S \quad p = \{ s \}$

Rule 2 Set P coincides with the set P' that is met ‘upper’ in the tree

α is a homing sequence iff it labels the path truncated by Rule 1

Adapting FSM based techniques to I/O Automata

The problem statement

o How to define HS and SS for Input/Output Automata?

o How to use the FSM ‘classics’ for their derivation?

⇓

Hypothesis : before applying any input, a tester waits for a given maximal output timeout t

If no output is produced by the system in *t* time units

Then the tester applies the next input (if any) and resets the timer

FSM

Synchronizing sequence (SS)

$\alpha = i_1 i_2 \ldots i_k$ is ***synchronizing***

if $\exists\ s \in S$ **such that**

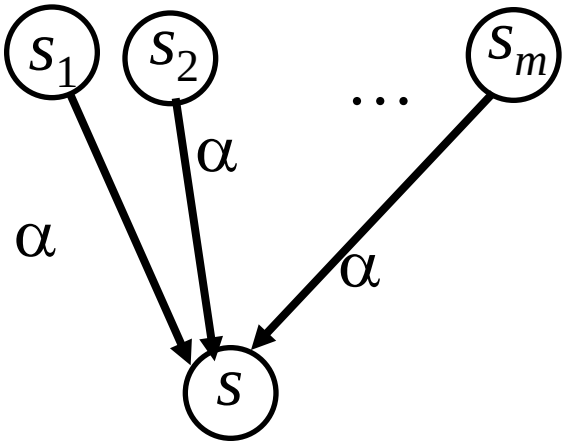
$\forall\ \gamma = i_1 o_1 i_2 o_2 \ldots i_k o_k$, **where**

$o_j \in O$, $j = 0, \ldots, k$

it holds that

γ-state-after-S is equal to $\{s\}$

Synchronizing sequence α



Synchronizing sequence (SS)

I/O Automaton

$\alpha = i_1 i_2 \ldots i_k$ is ***synchronizing***

if $\exists\ s \in S$ **such that**

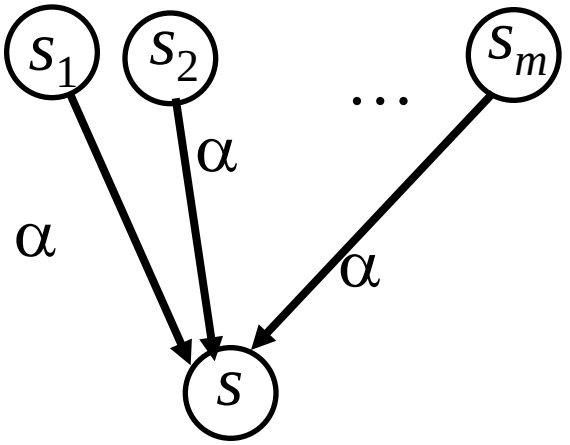
$\forall\ \gamma = \beta_0 i_1 \beta_1 i_2 \beta_2 \ldots i_k \beta_k$, **where**

$\beta_j \in (O \cup \{\delta\})^p$, $j = 0, \ldots, k$

it holds that

γ-state-after-S is equal to $\{s\}$

Synchronizing sequence α



p is the length of a longest sequence of consecutive outputs

FSM

Homing sequence (HS)

$\alpha = i_1 i_2 \ldots i_k$ is ***homing***

if $\exists\, s \in S$ **such that**

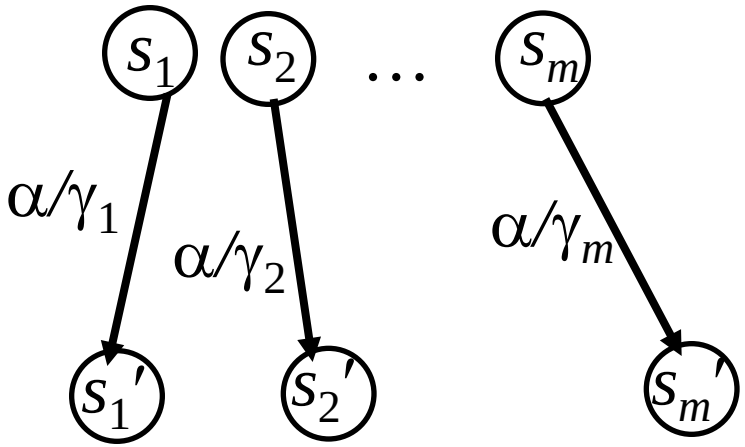
$\forall\, \gamma = i_1 o_1 i_2 o_2 \ldots i_k o_k$, **where**

$o_j \in O$, $j = 0, \ldots, k$

it holds that

γ-state-after-S is a singleton $\{ s_i' \}$

Homing sequence α



Homing sequence (HS)

I/O Automaton

$\alpha = i_1 i_2 \ldots i_k$ is ***homing***

if $\exists\ s \in S$ **such that**

$\forall\ \gamma = \beta_0 i_1 \beta_1 i_2 \beta_2 \ldots i_k \beta_k$, **where**

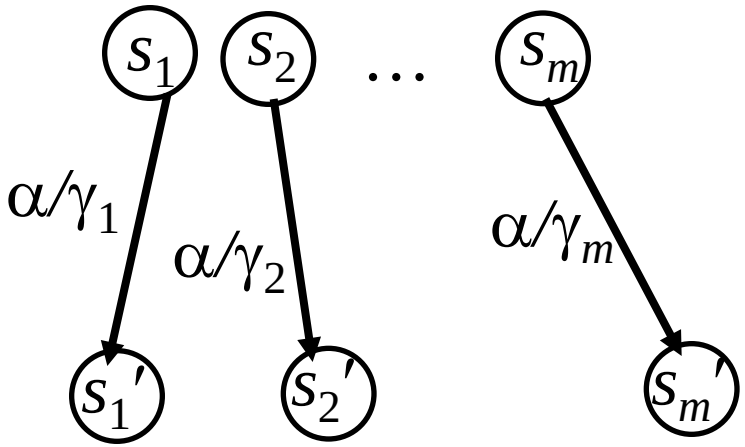
$\beta_j \in (O \cup \{\delta\})^p$, $j = 0, \ldots, k$

it holds that

γ-state-after-S is a singleton

$\{ s_i' \}$

Homing sequence α



p is the length of a longest sequence of consecutive outputs

Converting an input-output automaton to an FSM

Input: Input/Output automaton $\boldsymbol{S} = (S, I, O, T_{\boldsymbol{S}})$, $S = S_1 \cup S_2$

Output: FSM $\boldsymbol{M} = (S_1, I, O \cup O^2 \cup \ldots \cup O^p \cup \{\delta\}, T_{\boldsymbol{M}})$

p - length of a longest output trace of $\boldsymbol{S}$

Step 1 $T_{\boldsymbol{M}} := \varnothing$.

Step 2

For each state $s \in S_1$, such that $(s, i, s') \in T_{\boldsymbol{S}}$, $s' \in S_1$,
add to the $T_{\boldsymbol{M}}$ the transition (s, i, δ, s')

For each state $s \in S_1$, such that $(s, i, s'') \in T_{\boldsymbol{S}}$, $s'' \in S_2$,
add to the $T_{\boldsymbol{M}}$ the transition $(s, i, o_1\, o_2 \ldots o_k, s''')$, where
$k \leq p$, $s''' \in S_1$ is the $o_1\, o_2 \ldots o_k$-successor of state s''

Deriving a synchronizing sequence for an I/O automaton ***S***

Input: Input/Output automaton ***S*** = (S, I, O, T_S)

Output: SS α or a message “The automaton ***S*** is not synchronizing”

Step 1 Converting an input-output automaton ***S*** to an FSM ***M***

Step 2

Check the existence and derive a synchronizing sequence α for ***M***:

If the sequence α is derived then **Return** α

Else Return the message “The automaton ***S*** is not synchronizing”

Deriving a homing sequence for an I/O automaton ***S***

Input: Input/Output automaton ***S*** = (S, I, O, T_S)

Output: HS α or a message “The automaton ***S*** is not homing”

Step 1 Converting an input-output automaton ***S*** to an FSM ***M***

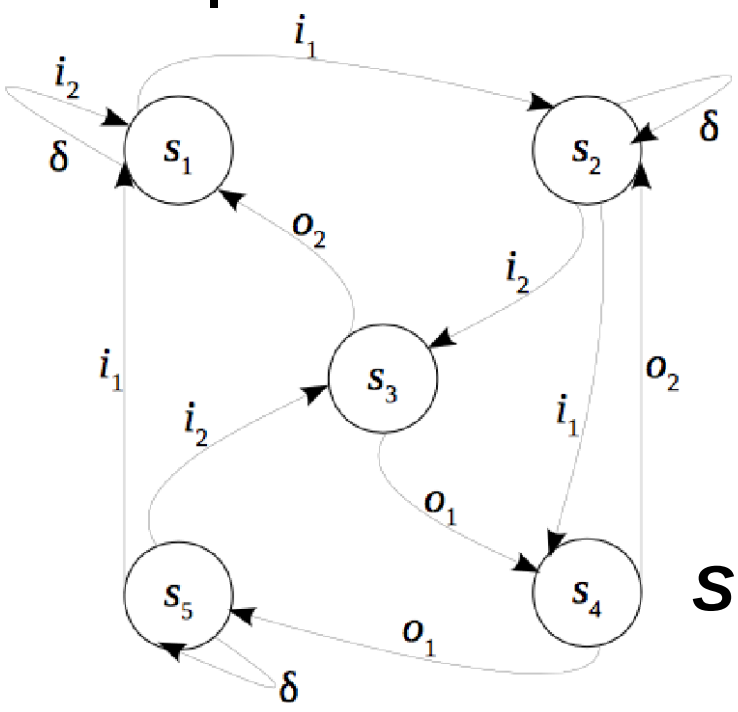
Step 2

Check the existence and derive an HS α for ***M***:

If the sequence α is derived then **Return** α

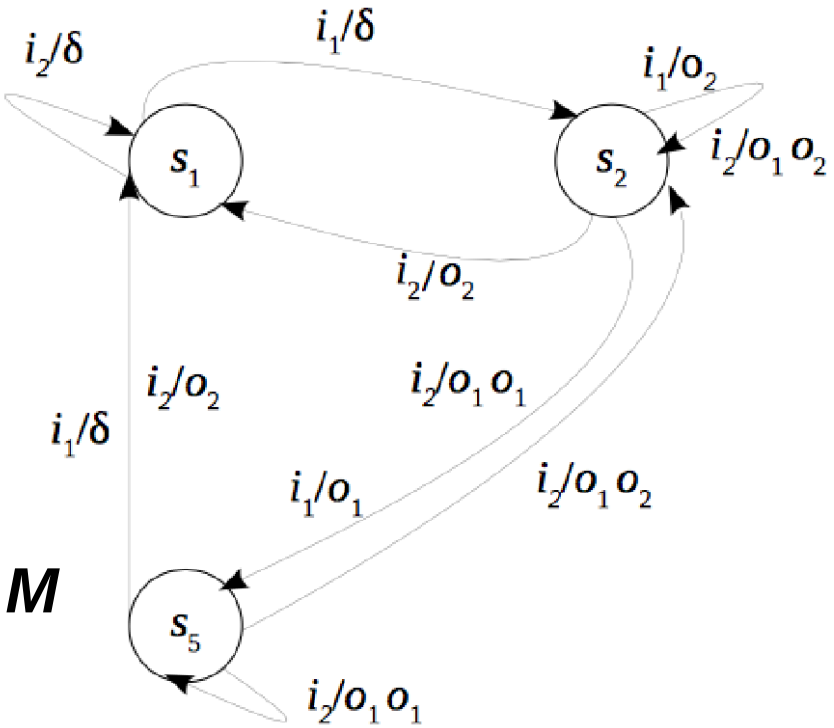
Else Return the message “The automaton ***S*** is not homing”

Back to the example...



S

M



An automaton ***S***
Withoit SS
HS $\alpha = i_1 i_1$

An FSM ***M***
Not synchronizing
HS $\alpha = i_1 i_1$

Conclusions and future work

- New problems are raised!
- Synchronizing and Homing Sequences for Input/Output Automata are introduced!

It is of course interesting…

- To extend the class of the machines, *i.e. delete so many restrictions*
- To define distinguishing sequences/experiments
- To consider adaptive experiments instead of preset
- …

Thank you!