TEST SCENARIOS GENERATION FOR LOGICAL CONTROLLERS USED IN EDF NUCLEAR POWER PLANTS

Aziz SFAR
Thesis 2022-2025

Supervisors:

- **Dina IROFTI (EDF)**
- **Madalina CROITORU (LIRMM)**

Logic Controllers

Logic Controllers

Logic Controllers

Logic Controllers

inputs

Logic Control System

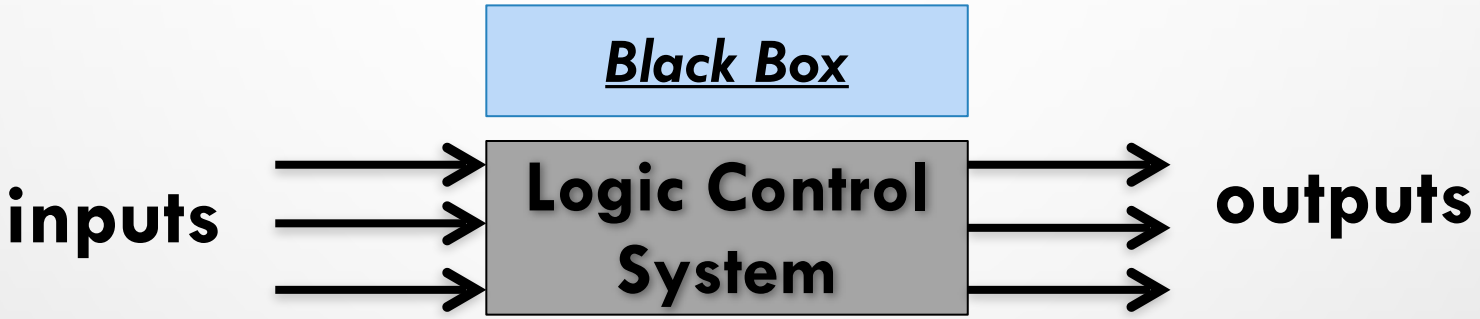
outputs

sensors

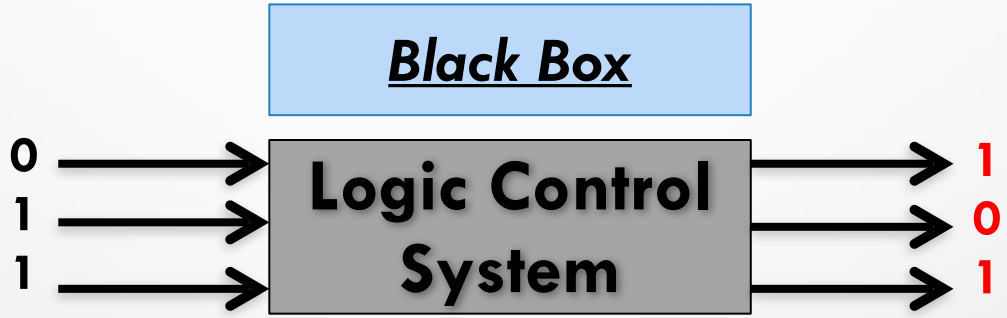
actuators

Validation Tests

Validation Tests

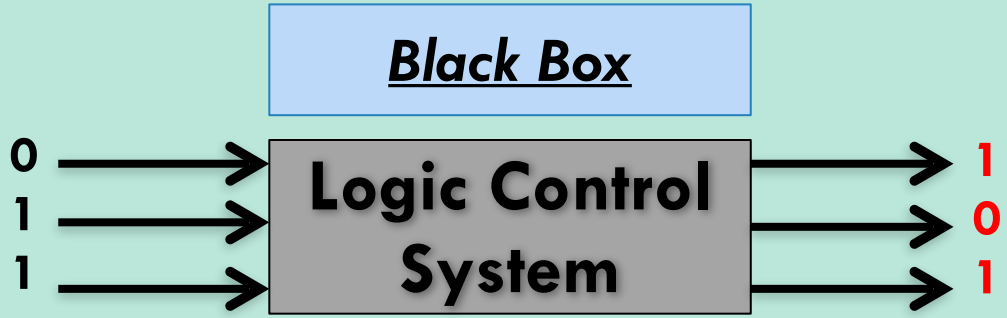


Validation Tests



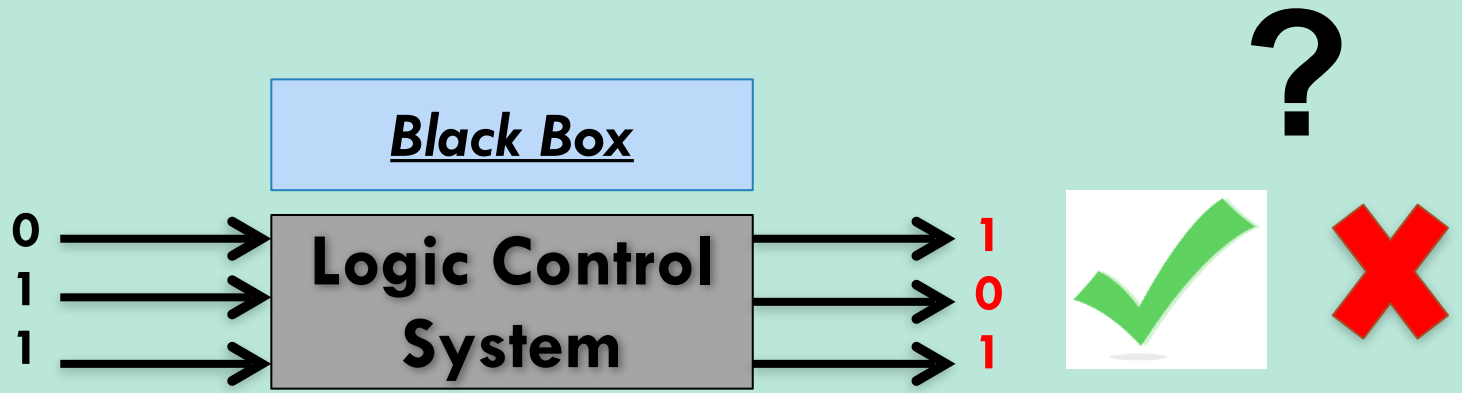
Validation Tests

THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



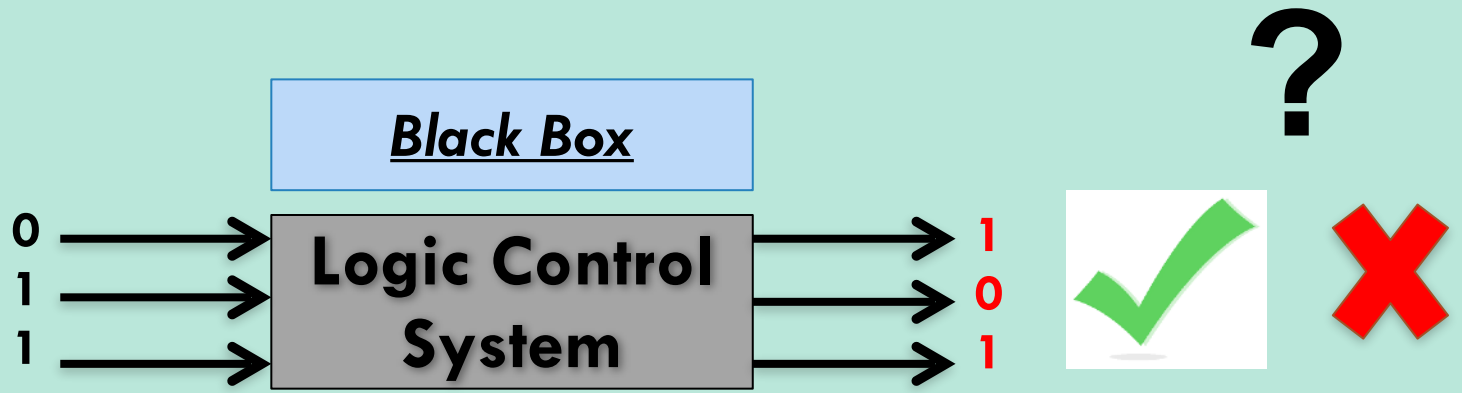
Validation Tests

THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



Validation Tests

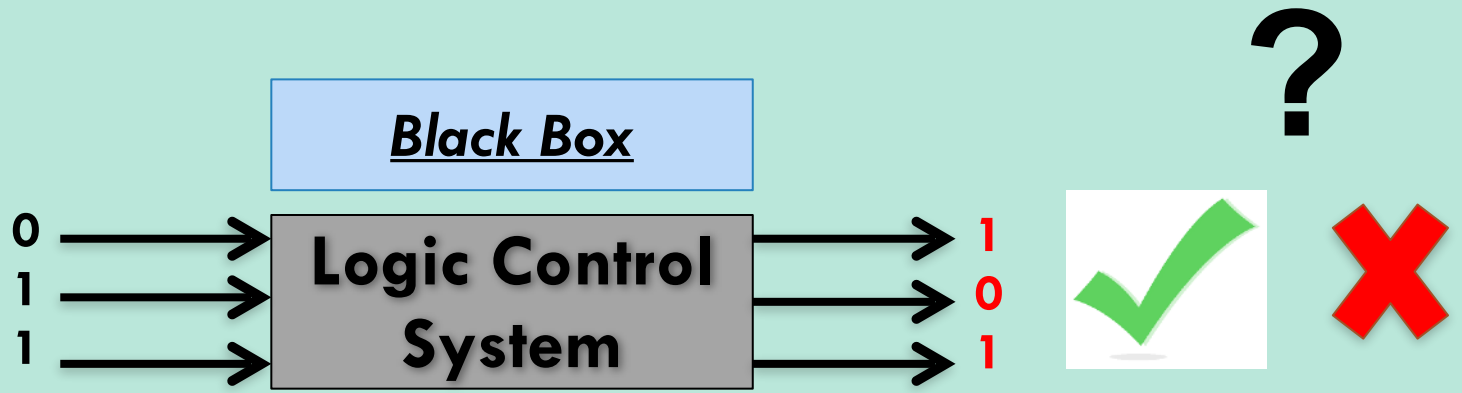
THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



Funtional Specification (System Design)

Validation Tests

THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



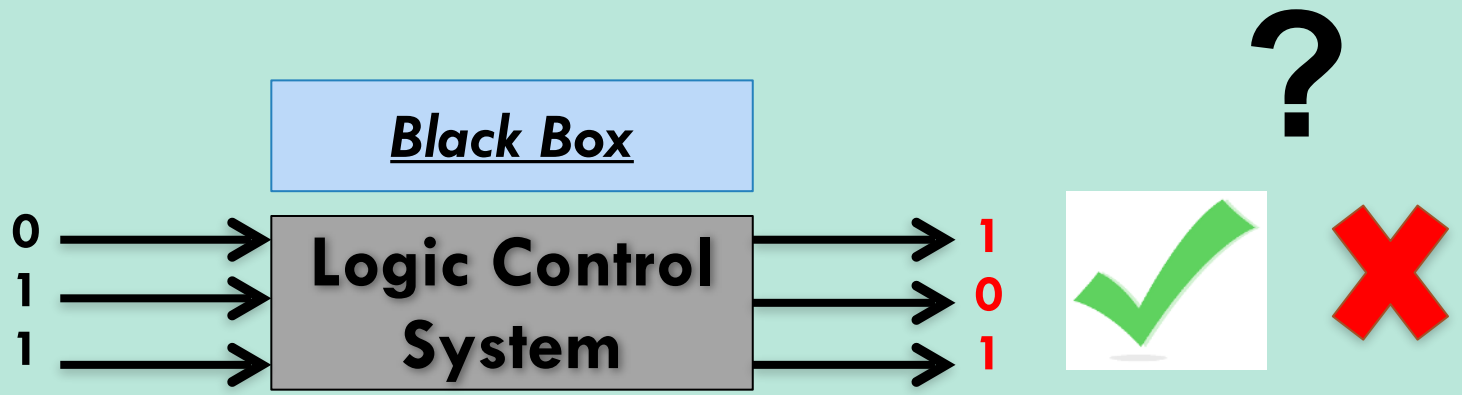
A model that describes how the control system should operate.

Funtional Specification (System Design)

THE EXPECTED BEHAVIOR OF THE SYSTEM

Validation Tests

THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



A model that describes how the control system should operate.

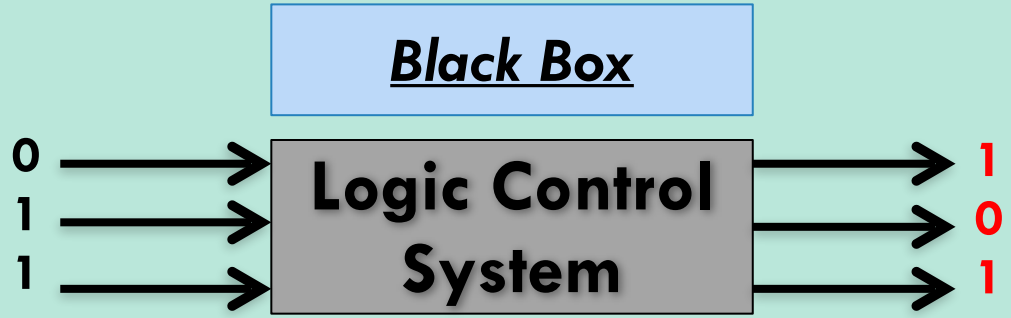
Funtional Specification (System Design)

Program

THE EXPECTED BEHAVIOR OF THE SYSTEM

Validation Tests

THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



A model that describes how the control system should operate.

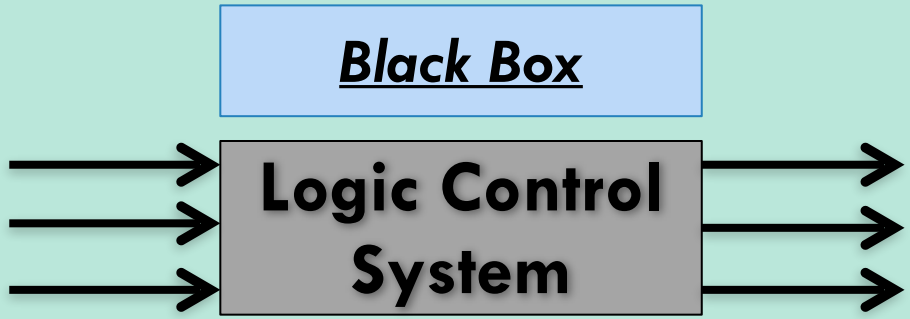
Funtional Specification (System Design)

Program

THE EXPECTED BEHAVIOR OF THE SYSTEM

Validation Tests

THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



Funtional Specification (System Design)

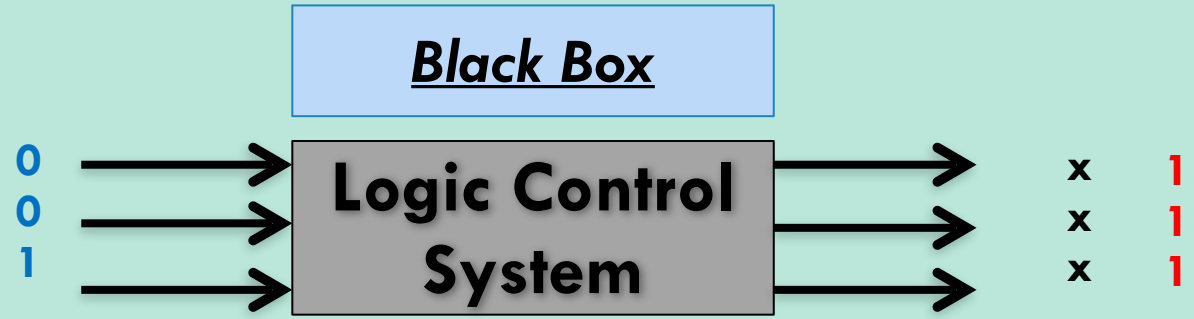
Test Generation

Input sequence:	0	0	1 / 0	0	0 / 1	0	0
Output sequence:	1	1	1 / 0	1	0 / 0	0	0

THE EXPECTED BEHAVIOR OF THE SYSTEM

Validation Tests

THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



Funtional Specification (System Design)

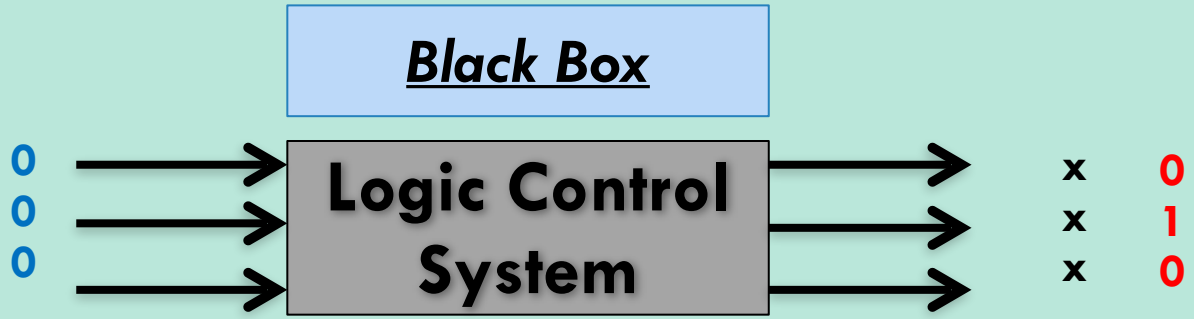
Test Generation

Input sequence:	0 0 1 / 0 0 0 / 1 0 0
Output sequence:	1 1 1 / 0 1 0 / 0 0 0

THE EXPECTED BEHAVIOR OF THE SYSTEM

Validation Tests

THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



Funtional Specification (System Design)

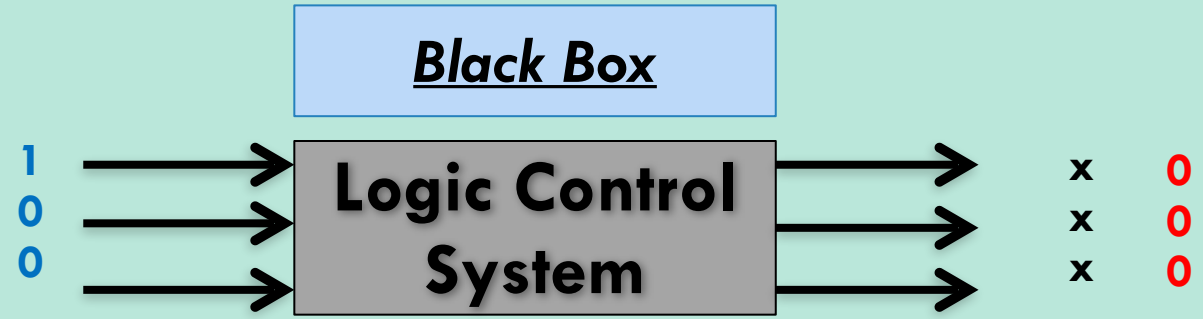
Test Generation

Input sequence:	0	0	1	/	0	0	0	/	1	0	0
Output sequence:	1	1	1	/	0	1	0	/	0	0	0

THE EXPECTED BEHAVIOR OF THE SYSTEM

Validation Tests

THE ACTUAL OBSERVED BEHAVIOR OF THE SYSTEM



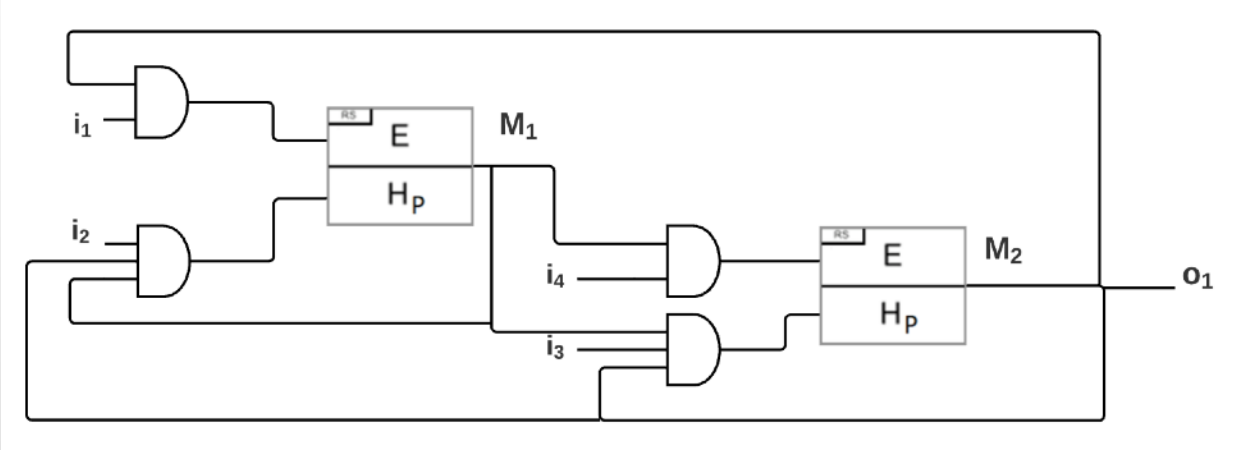
Funtional Specification (System Design)

Test Generation

Input sequence:	0 0 1 / 0 0 0 / 1 0 0
Output sequence:	1 1 1 / 0 1 0 / 0 0 0

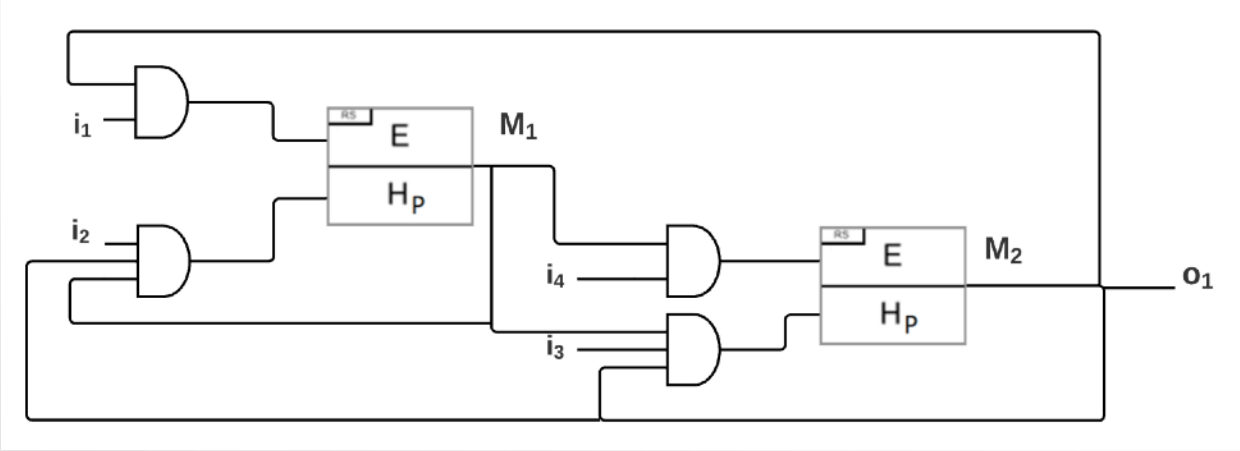
THE EXPECTED BEHAVIOR OF THE SYSTEM

System specification

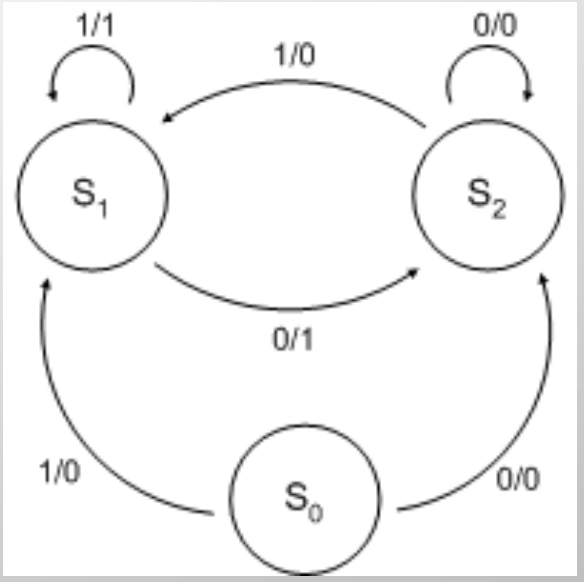


Logical Diagram

System specification

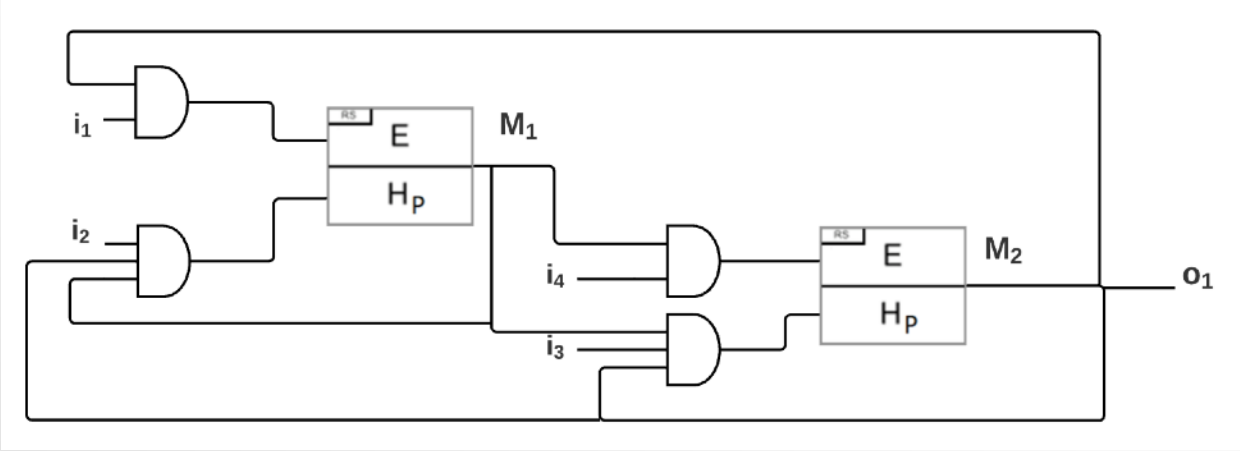


Logical Diagram

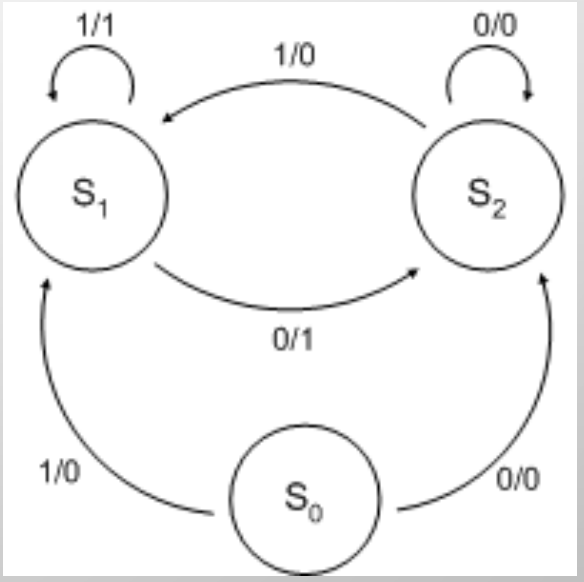


Mealy Machine

System specification



Logical Diagram

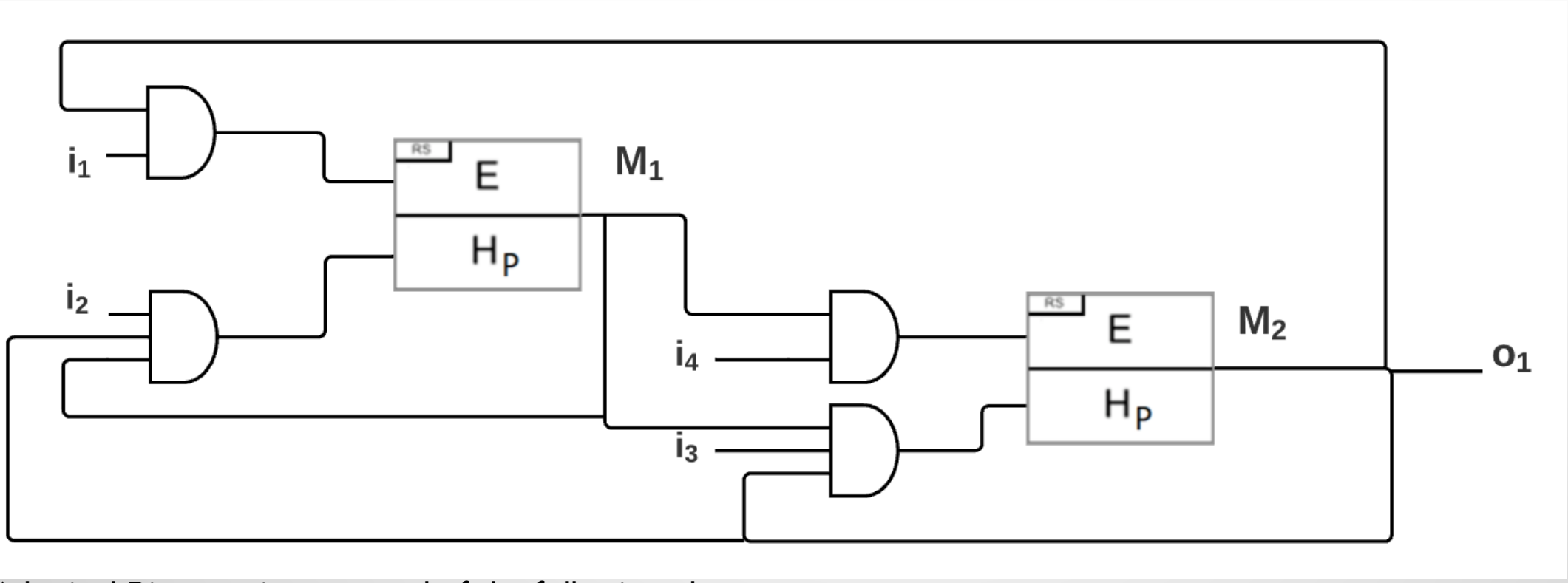


Mealy Machine

- How is the transformation done ?
- KB representation of the problem ?

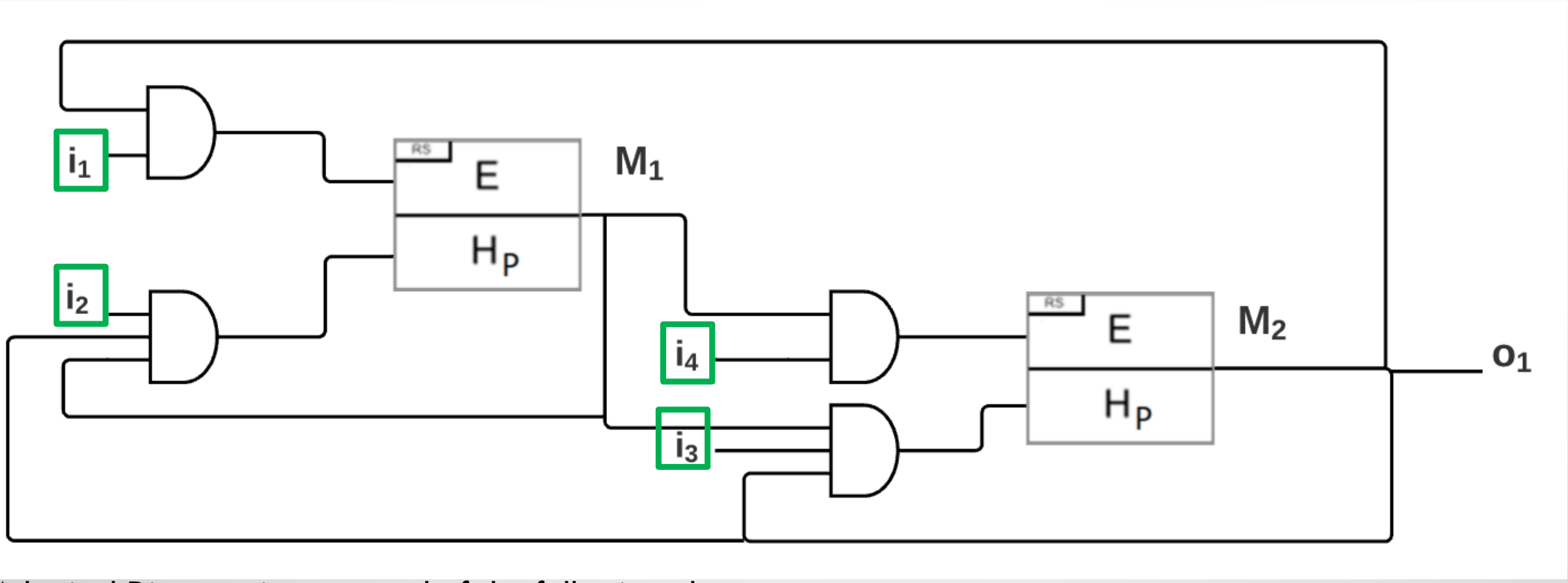
Specifications : Logical Diagrams

Specifications : Logical Diagrams



A logical Diagram is composed of the following elements

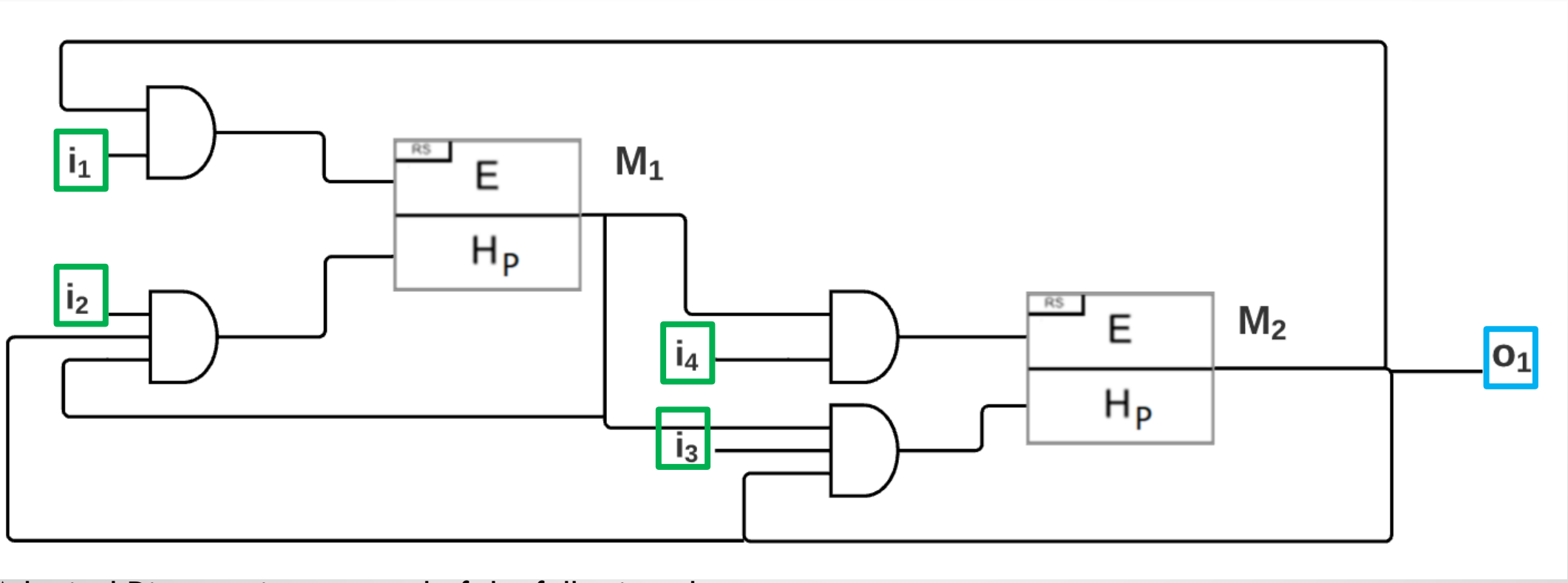
Specifications : Logical Diagrams



A logical Diagram is composed of the following elements

- I: inputs

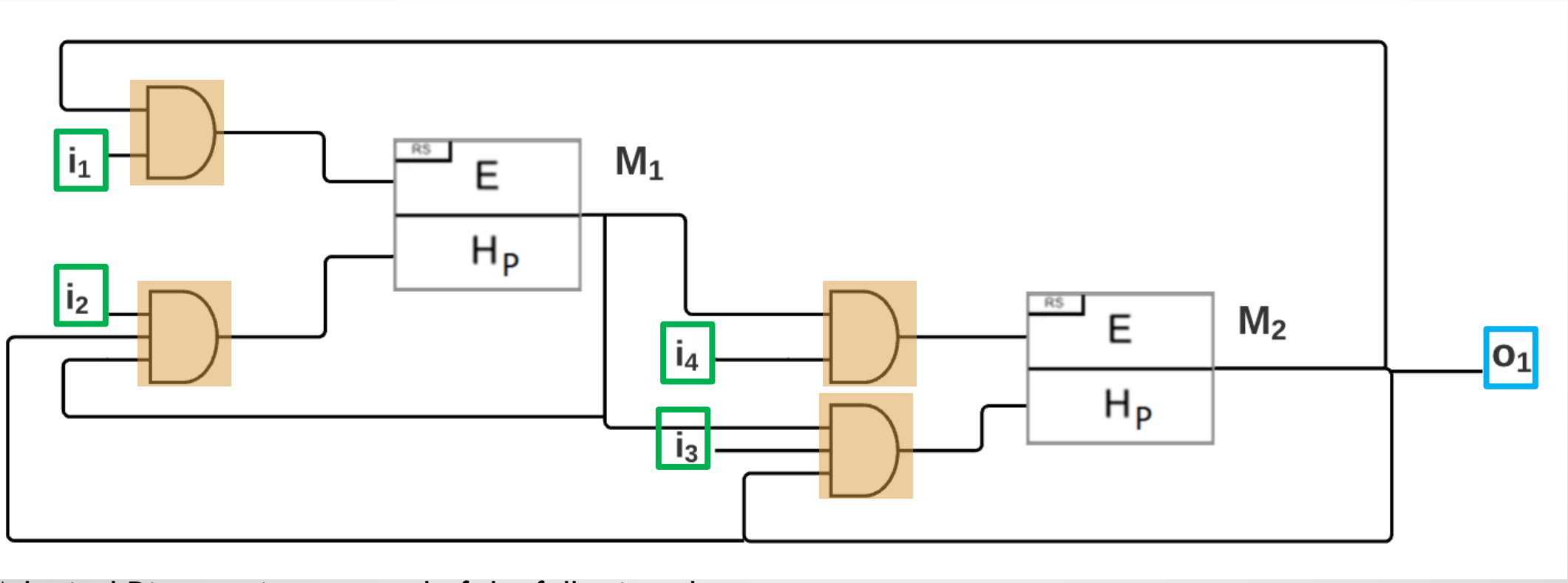
Specifications : Logical Diagrams



A logical Diagram is composed of the following elements

- I: inputs
- O: outputs

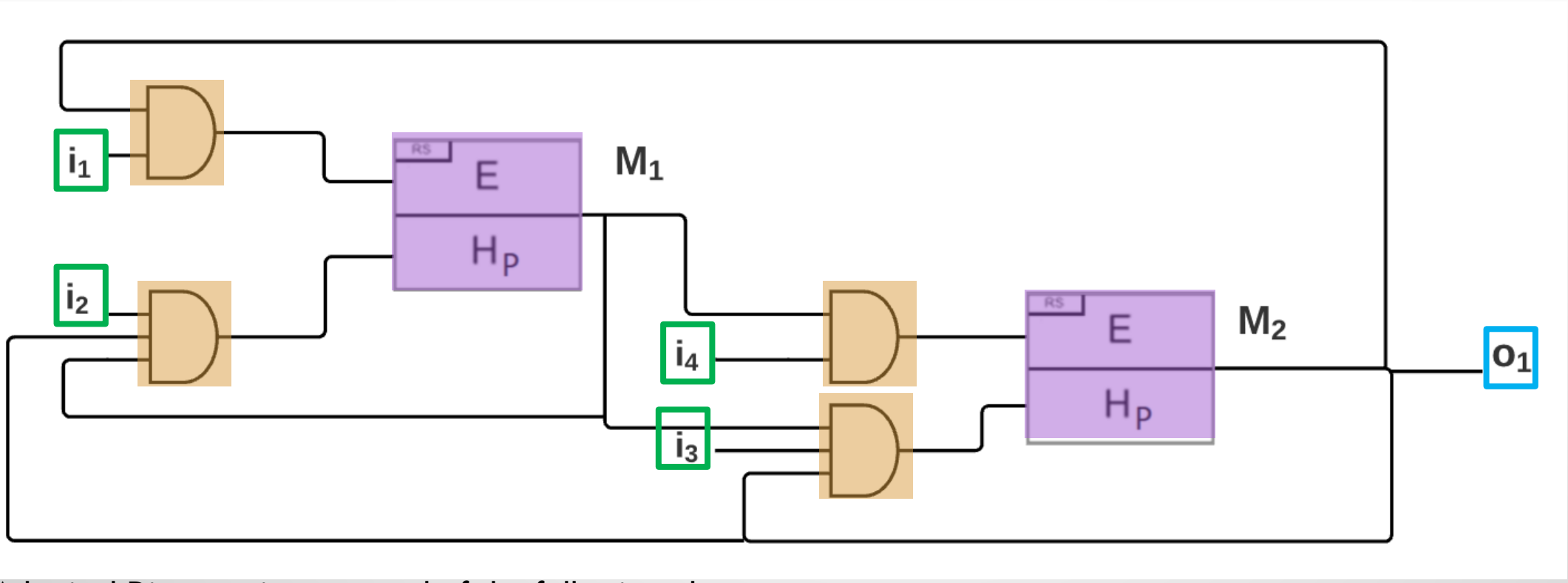
Specifications : Logical Diagrams



A logical Diagram is composed of the following elements

- I: inputs
- O: outputs
- LG: logic gates

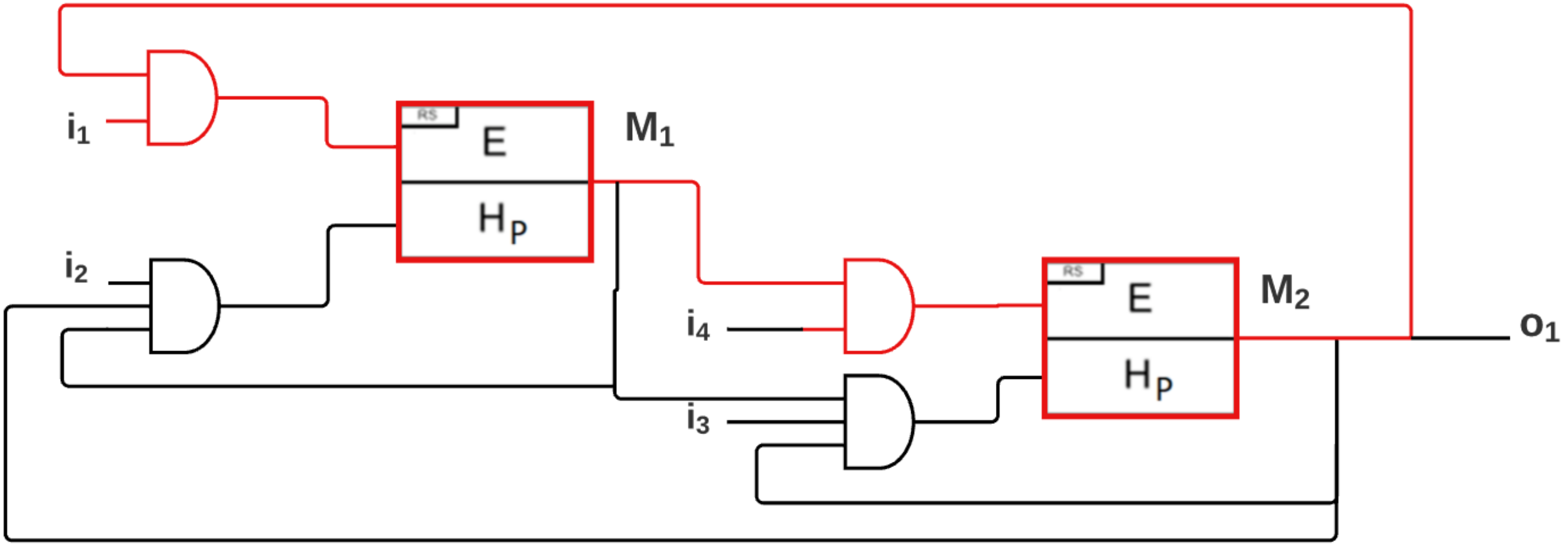
Specifications : Logical Diagrams



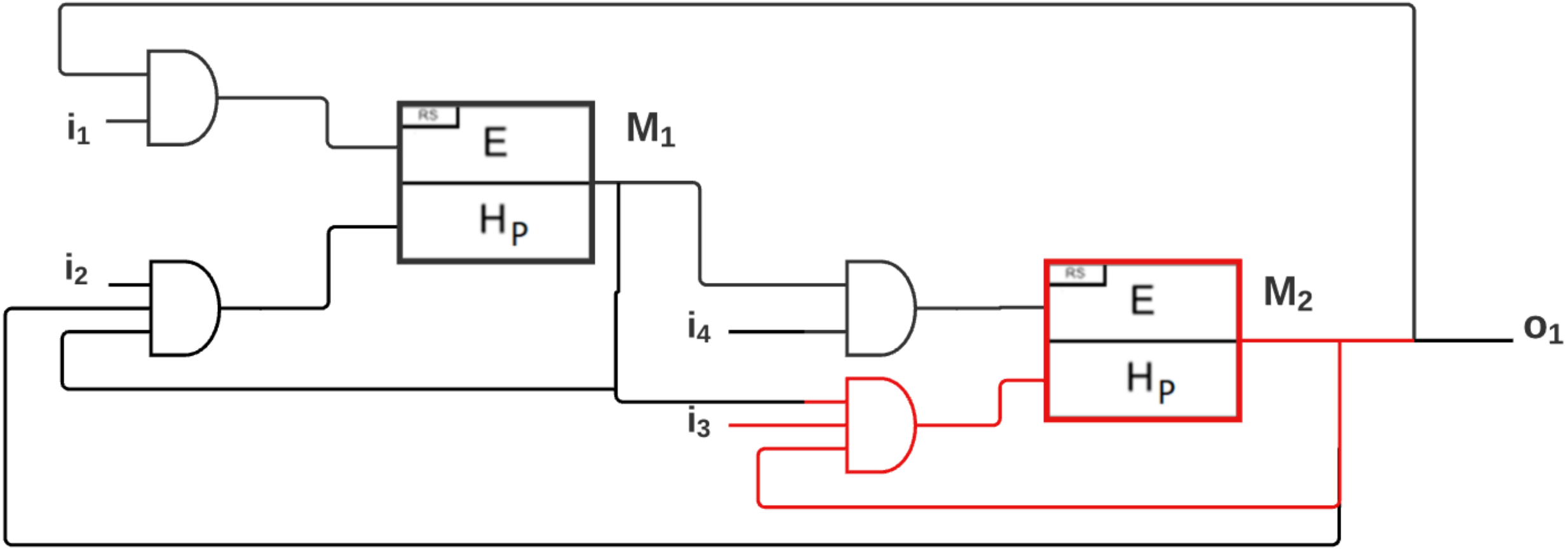
A logical Diagram is composed of the following elements

- I: inputs
- O: outputs
- LG: logic gates
- M: memory blocks

Specifications : Logical Diagrams

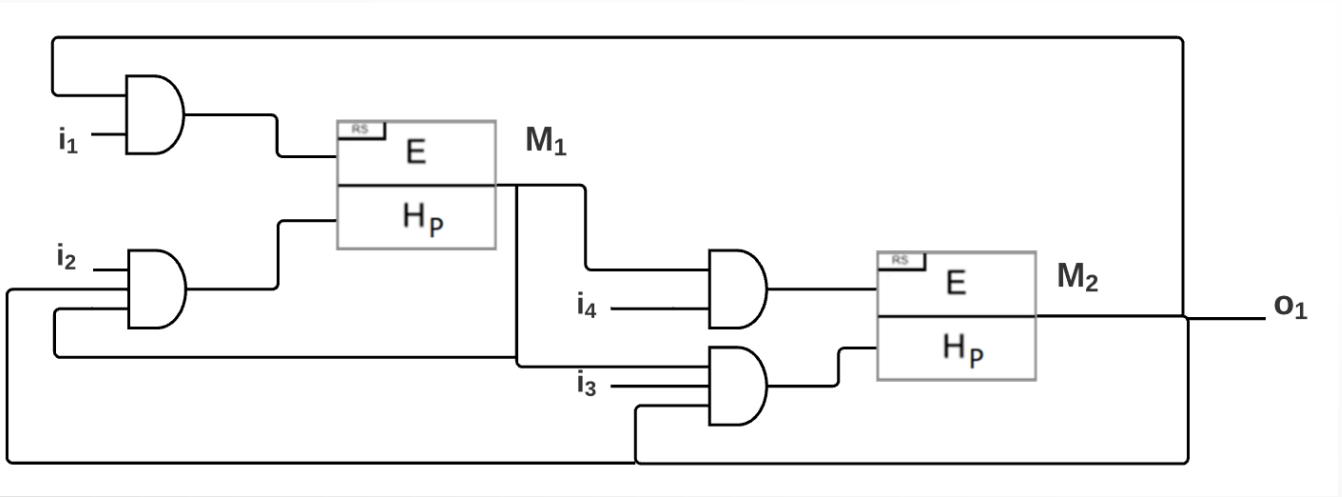


Specifications : Logical Diagrams

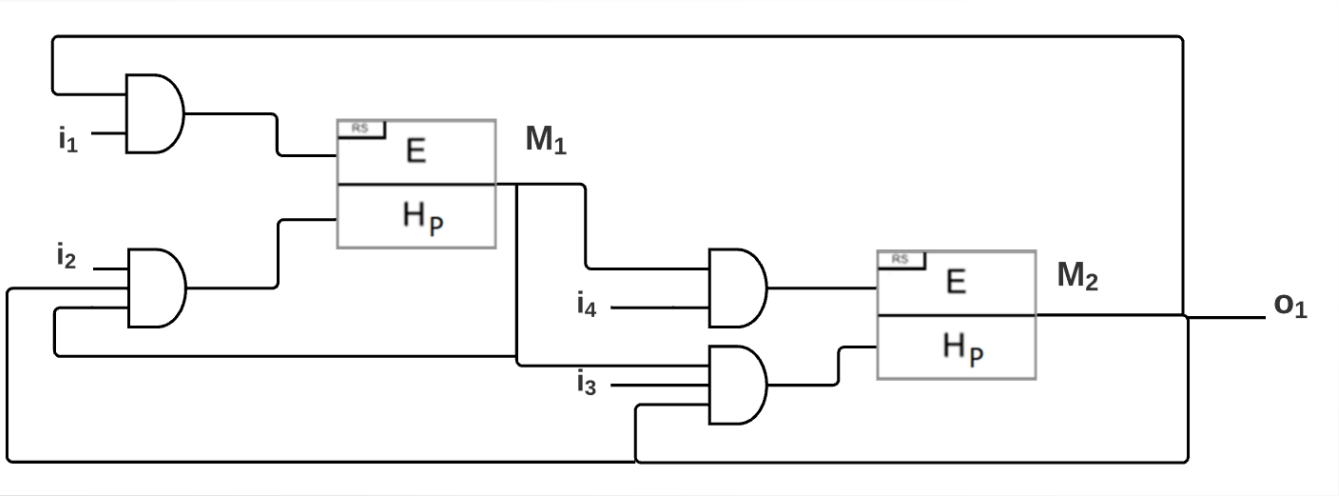


Evaluation of Logical Diagrams

Evaluation of Logical Diagrams



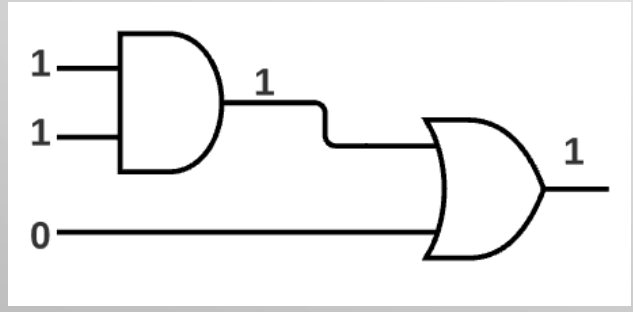
Evaluation of Logical Diagrams



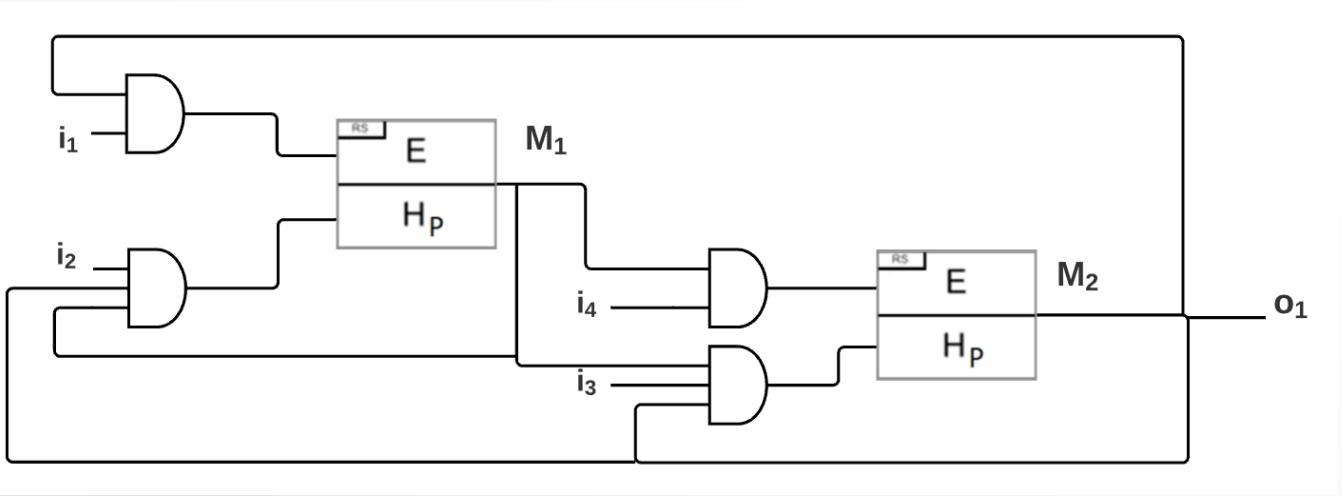
Evaluation of logic gates :

Logic gates: AND, OR , NOT

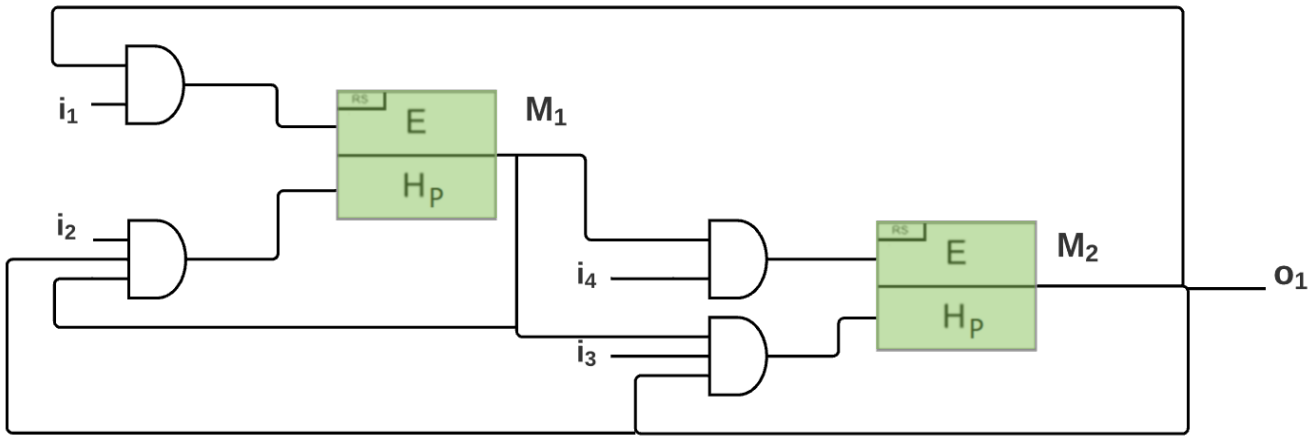
Evaluated from left to right



Evaluation of Logical Diagrams

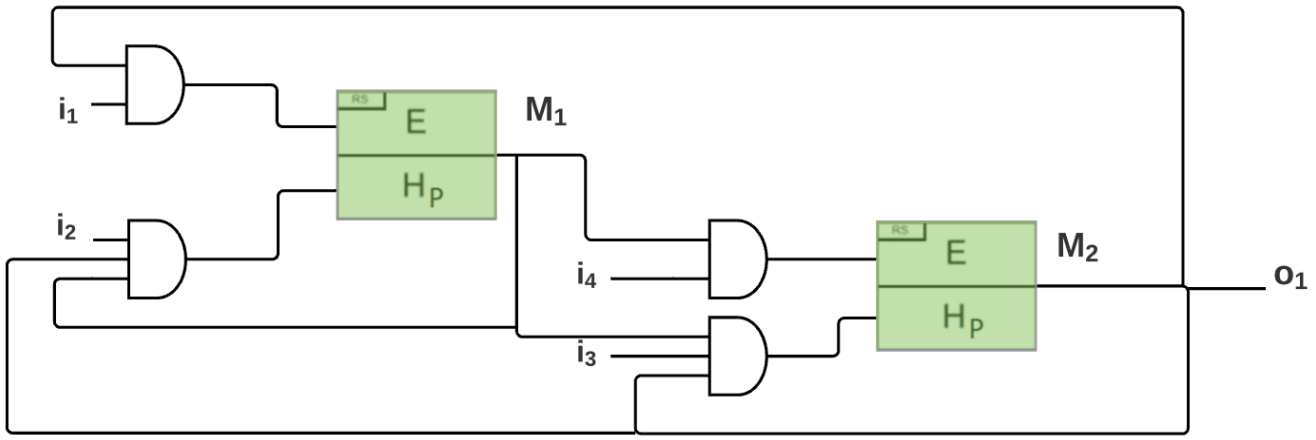


Evaluation of Logical Diagrams

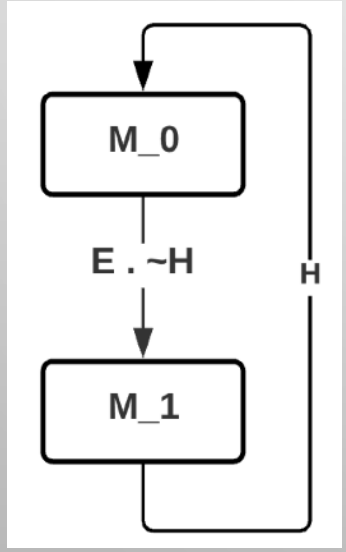


Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.

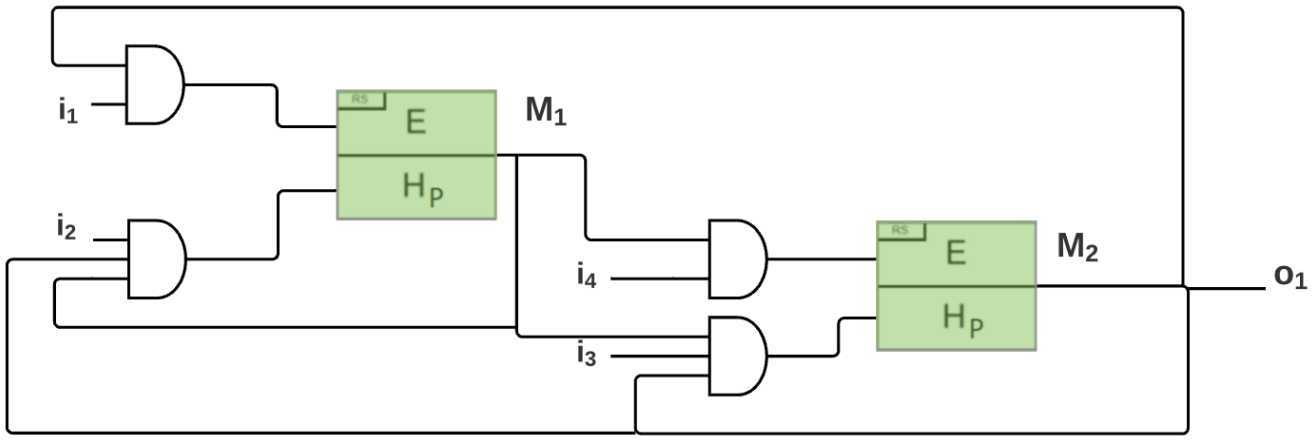
Evaluation of Logical Diagrams



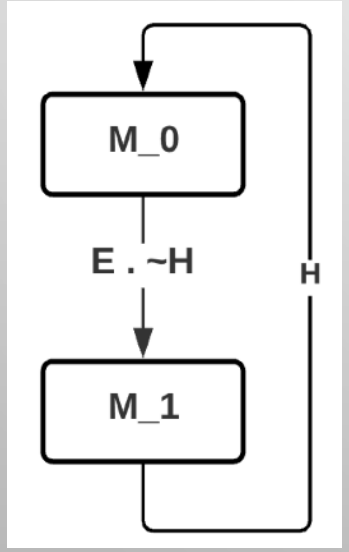
Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.



Evaluation of Logical Diagrams

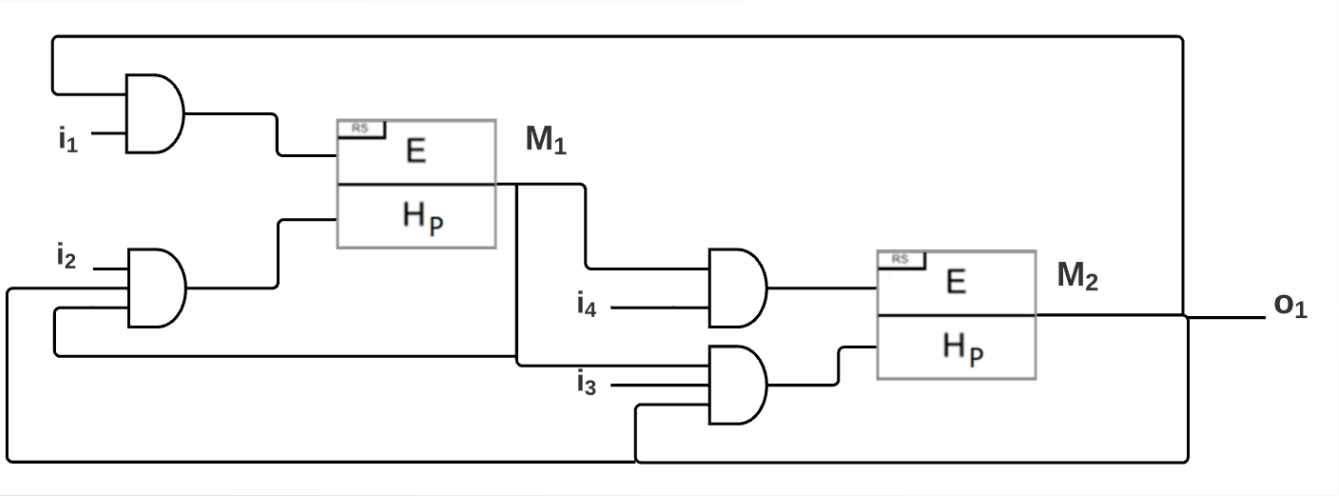


Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.

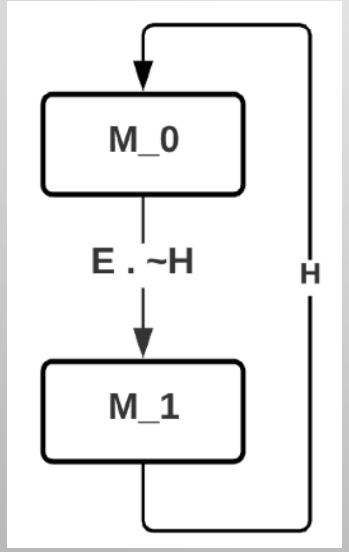


Sequential evaluation of memory blocks in accordance to an order ω

Evaluation of Logical Diagrams



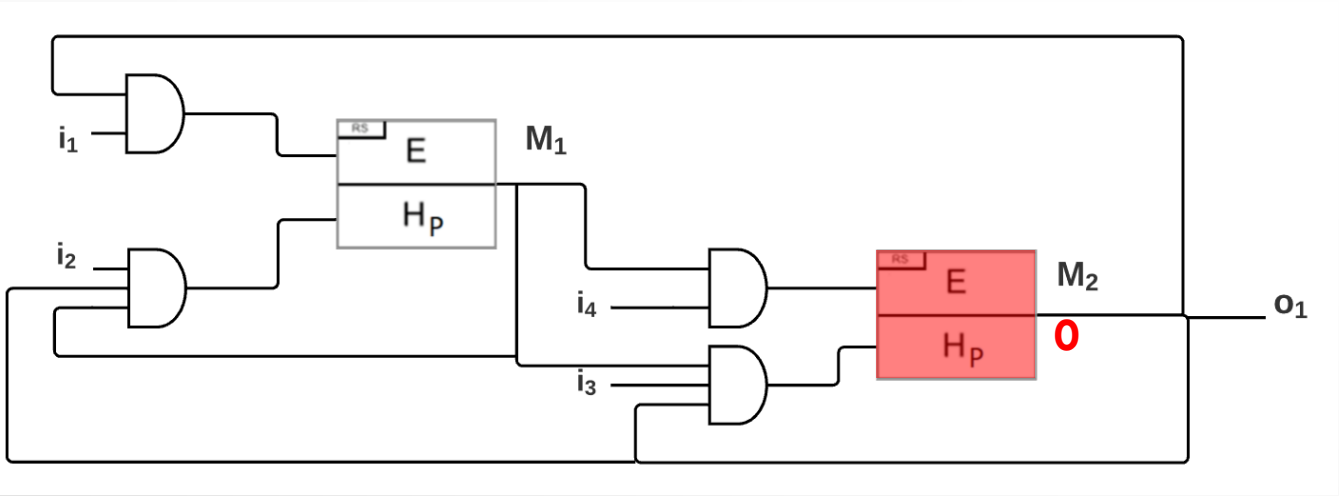
Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.



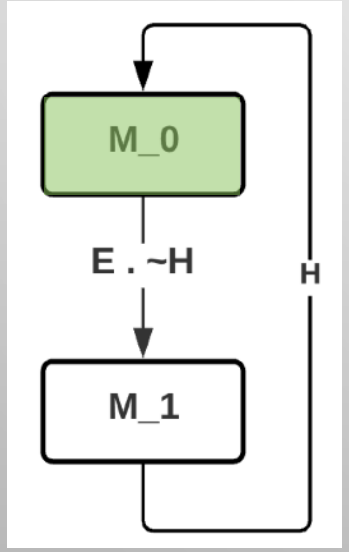
Sequential evaluation of memory blocks in accordance to an order ω

Example ω = (M2, M1)

Evaluation of Logical Diagrams



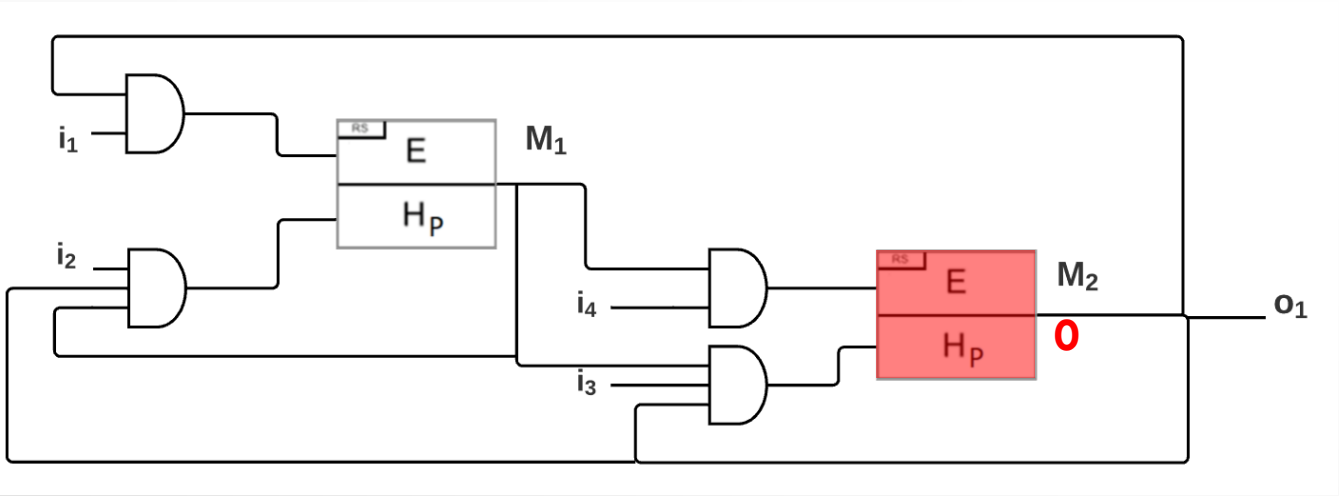
Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.



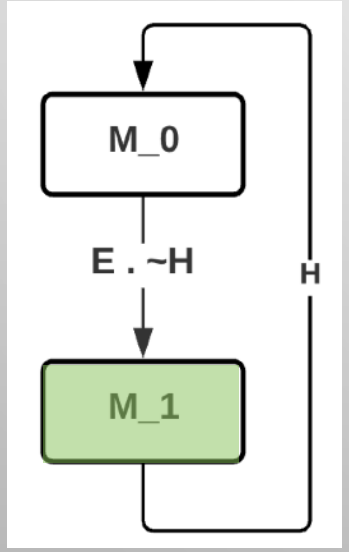
Sequential evaluation of memory blocks in accordance to an order ω

Example ω = (M2, M1)

Evaluation of Logical Diagrams



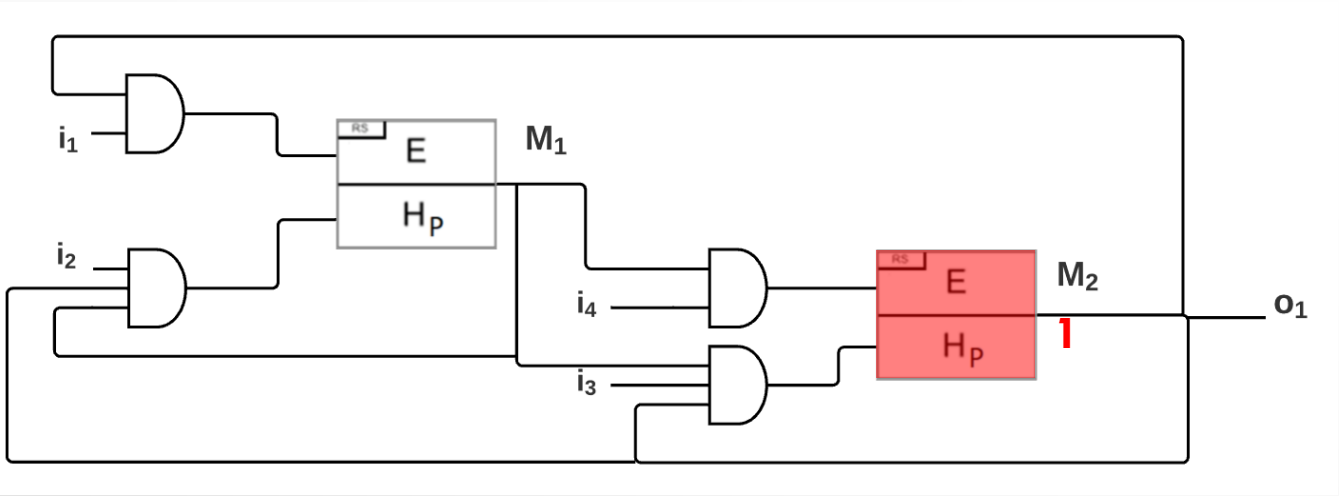
Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.



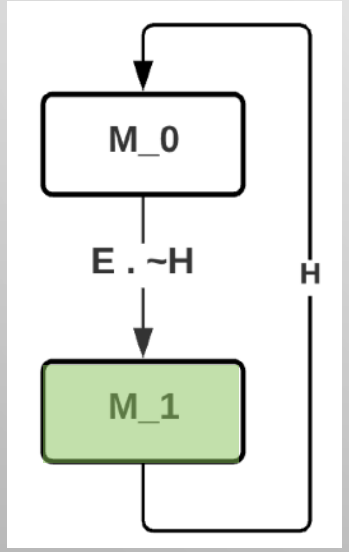
Sequential evaluation of memory blocks in accordance to an order ω

Example ω = (M2, M1)

Evaluation of Logical Diagrams



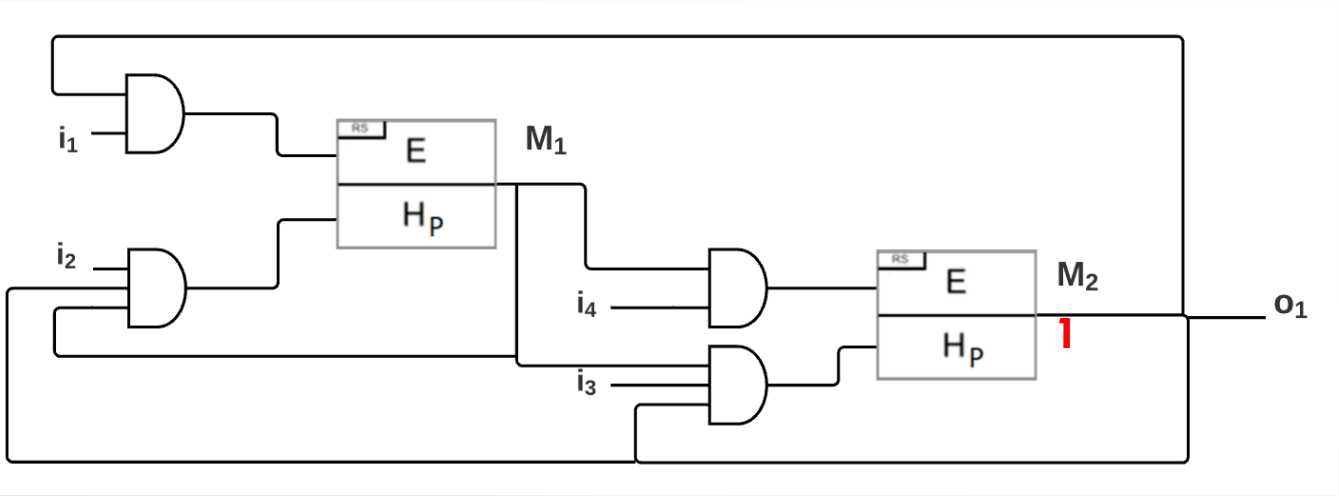
Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.



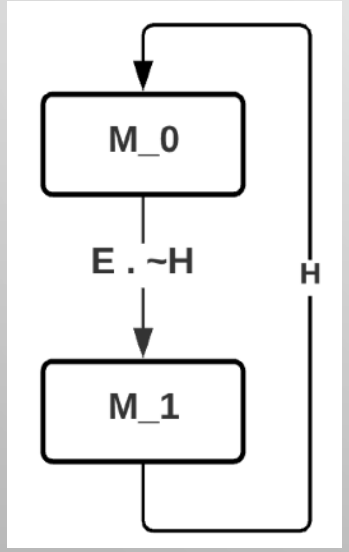
Sequential evaluation of memory blocks in accordance to an order ω

Example ω = (M2, M1)

Evaluation of Logical Diagrams



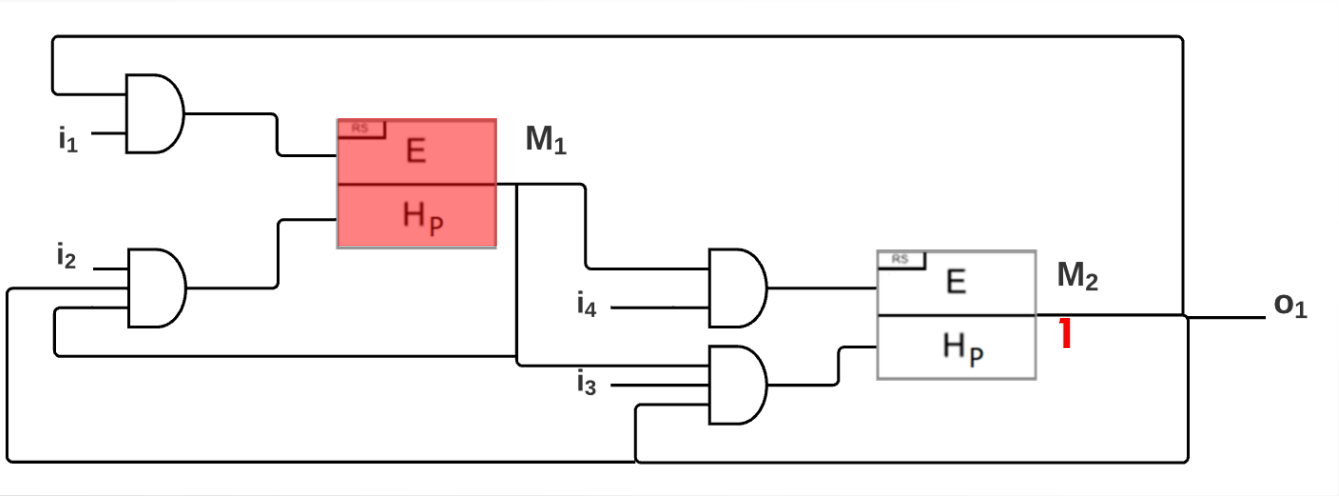
Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.



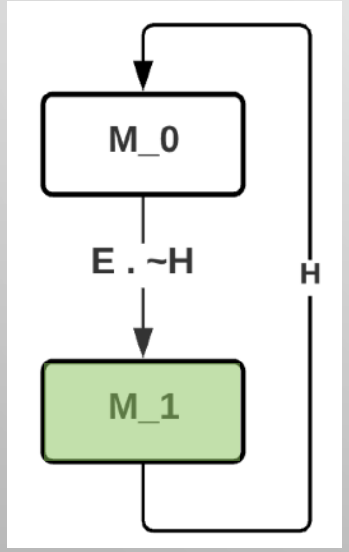
Sequential evaluation of memory blocks in accordance to an order ω

Example ω = (M2, M1)

Evaluation of Logical Diagrams



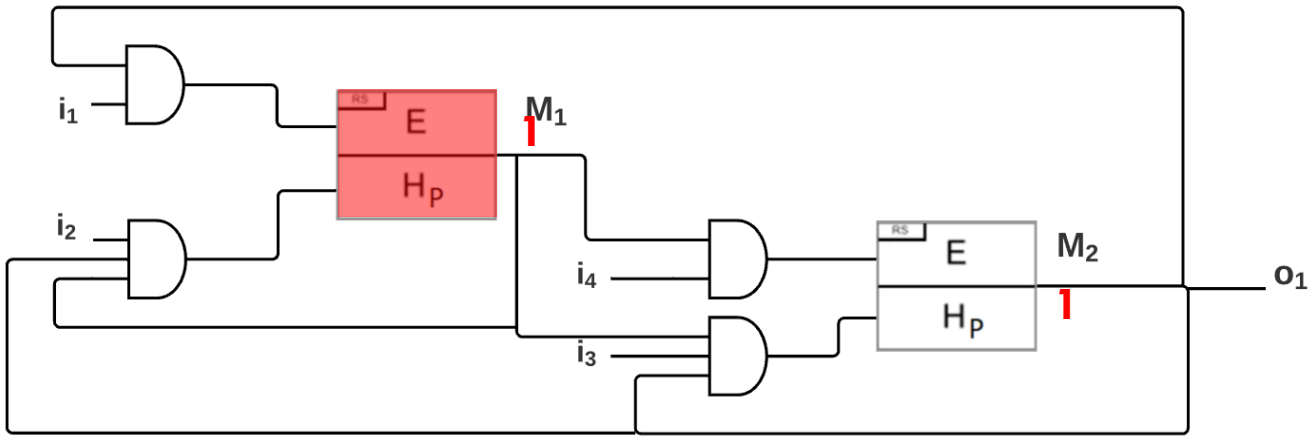
Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.



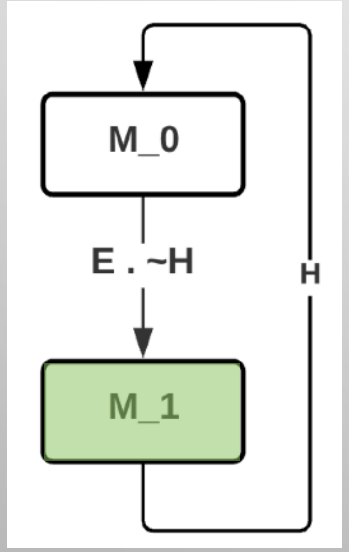
Sequential evaluation of memory blocks in accordance to an order ω

Example ω = (M2, M1)

Evaluation of Logical Diagrams



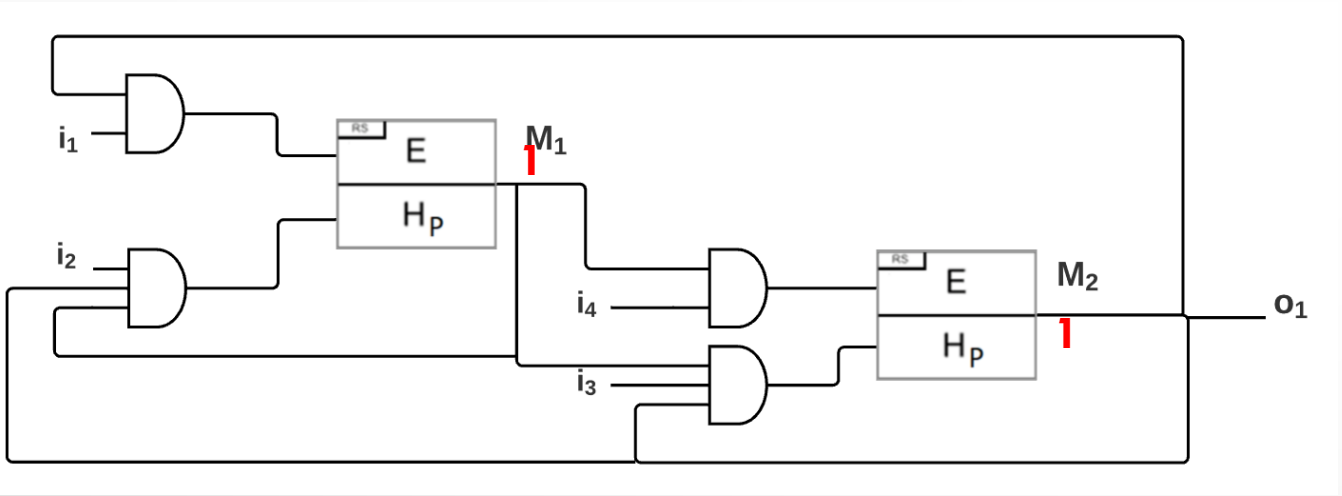
Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.



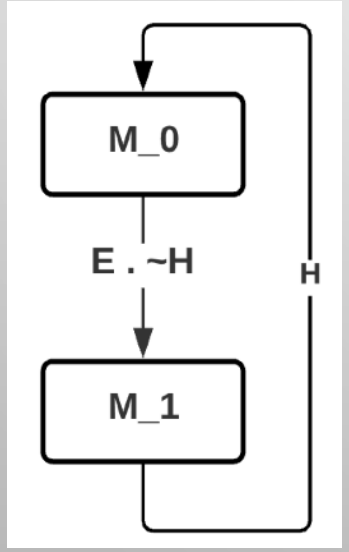
Sequential evaluation of memory blocks in accordance to an order ω

Example ω = (M2, M1)

Evaluation of Logical Diagrams



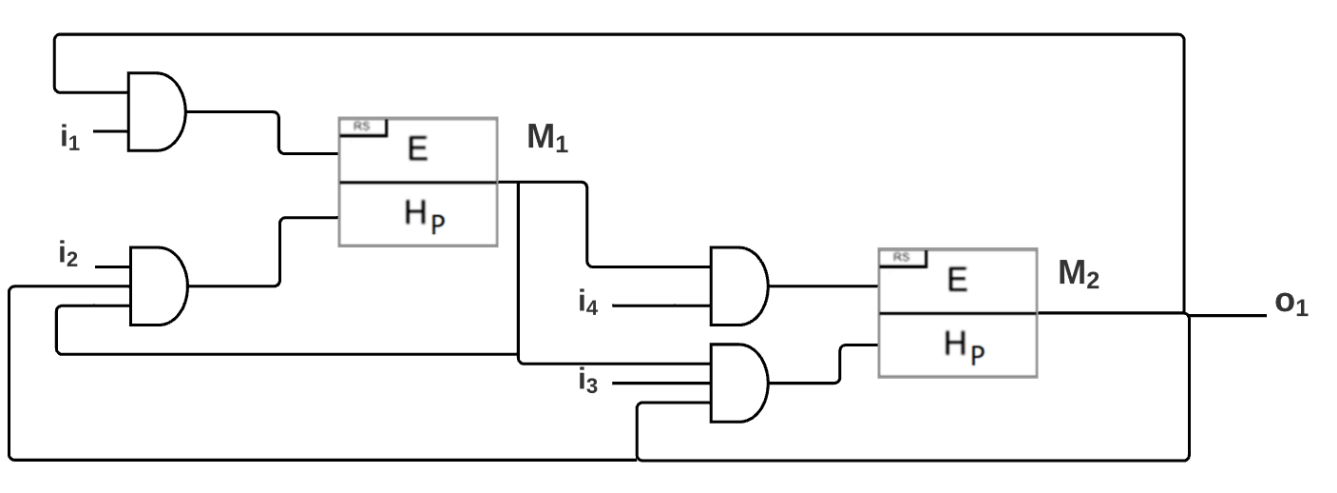
Evaluation of memory blocks: We call memory blocks as status blocks : the output value of the block depends on the values of its inputs and its current output value.



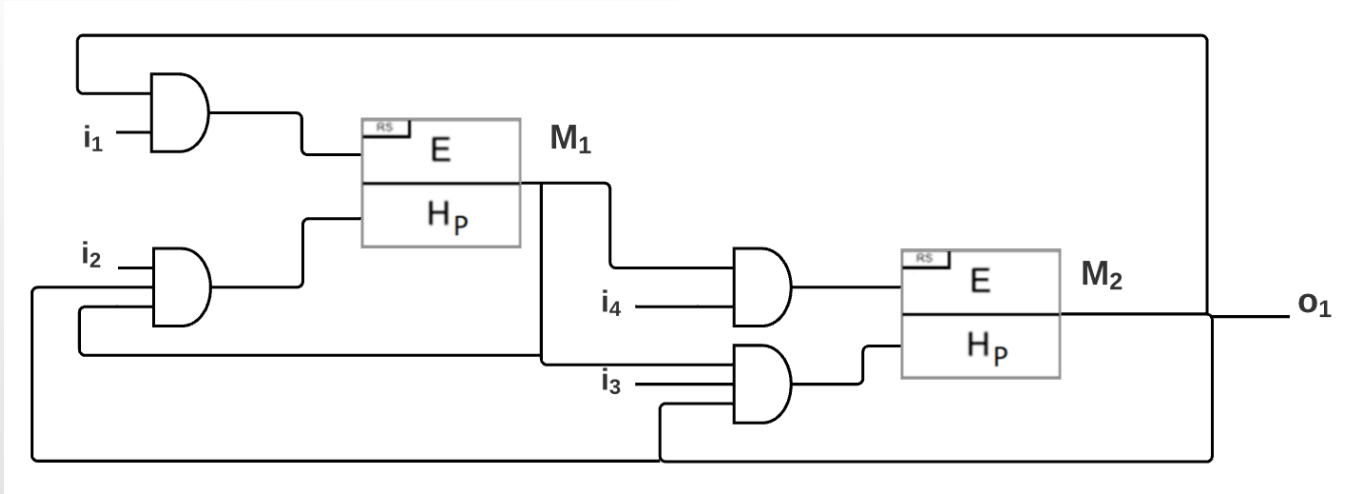
Sequential evaluation of memory blocks in accordance to an order ω

Example ω = (M2, M1)

First Problem: Test Generation based on Logical Diagrams

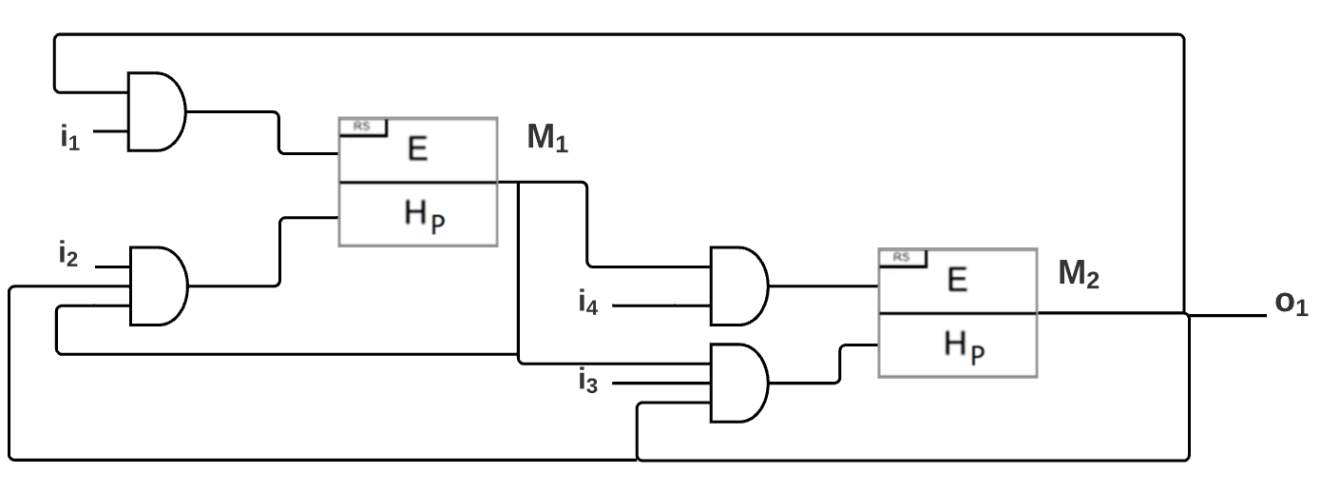


First Problem: Test Generation based on Logical Diagrams



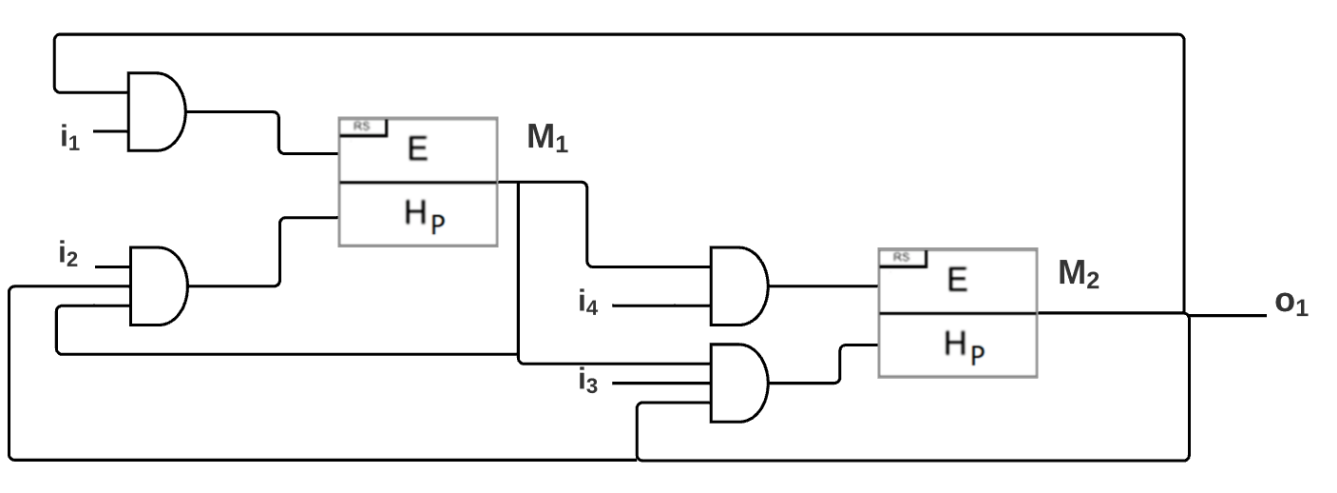
- Logical diagrams do not explicitly represent how the output values evolve in response to changes of input values.

First Problem: Test Generation based on Logical Diagrams



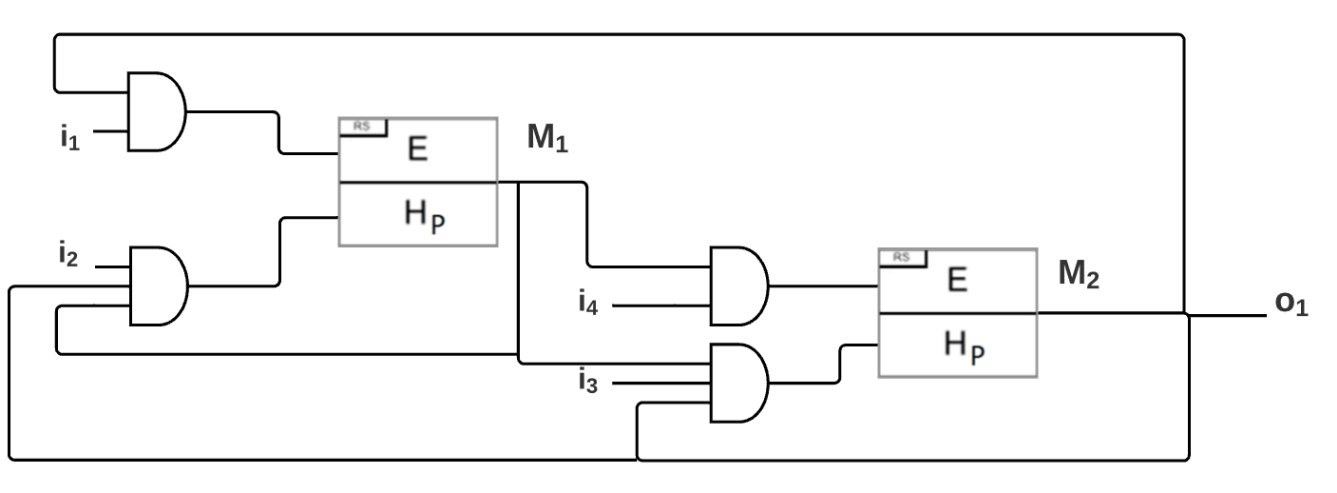
- Logical diagrams do not explicitly represent how the output values evolve in response to changes of input values.
- The whole behavior is described by the evaluation of the diagram for all the $2^I . 2^M$ possibilities.

First Problem: Test Generation based on Logical Diagrams



- Logical diagrams do not explicitly represent how the output values evolve in response to changes of input values.
- The whole behavior is described by the evaluation of the diagram for all the $2^I . 2^M$ possibilities.
- Each evaluation of the outputs for a given set of input values my have to go through many simulations of the Logical Diagram.

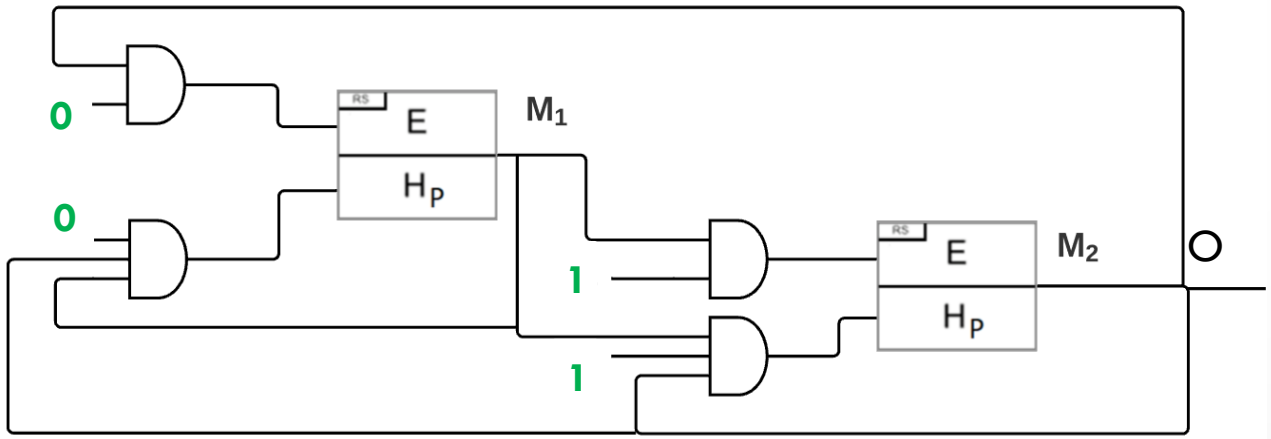
First Problem: Test Generation based on Logical Diagrams



- Logical diagrams do not explicitly represent how the output values evolve in response to changes of input values.
- The whole behavior is described by the evaluation of the diagram for all the $2^I . 2^M$ possibilities.
- Each evaluation of the outputs for a given set of input values my have to go through many simulations of the Logical Diagram.

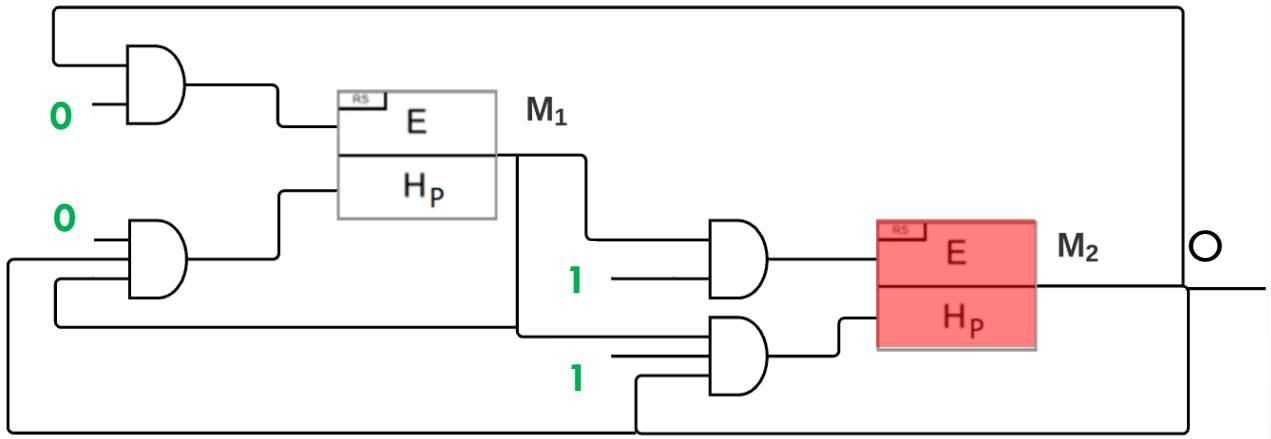
➢ Generation and selection of test sequences is not obvious.

Second problem : Output values have to converge



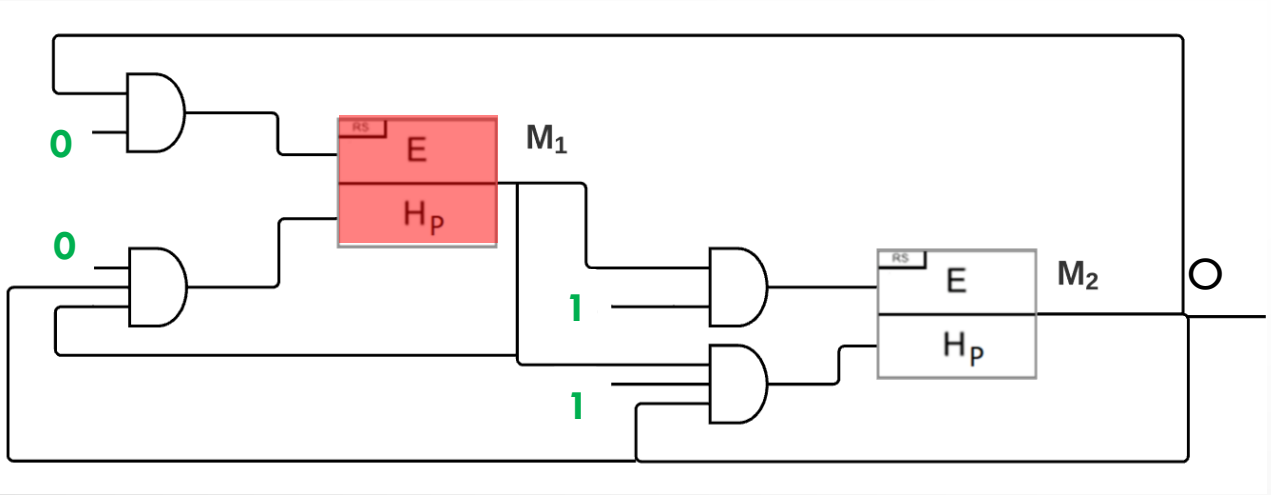
$\omega = (M2, M1)$

Second problem : Output values have to converge



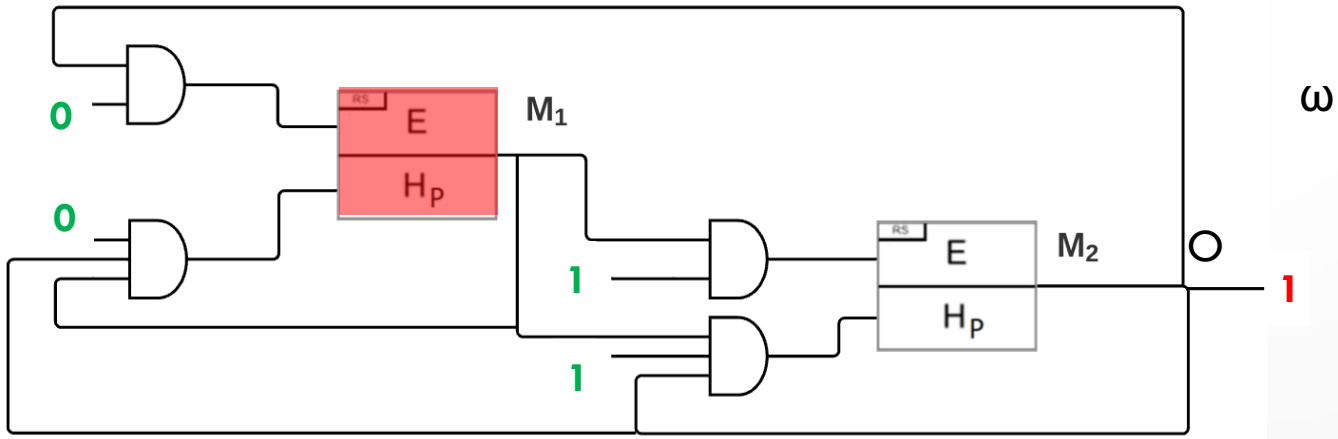
$\omega = (M2, M1)$

Second problem : Output values have to converge



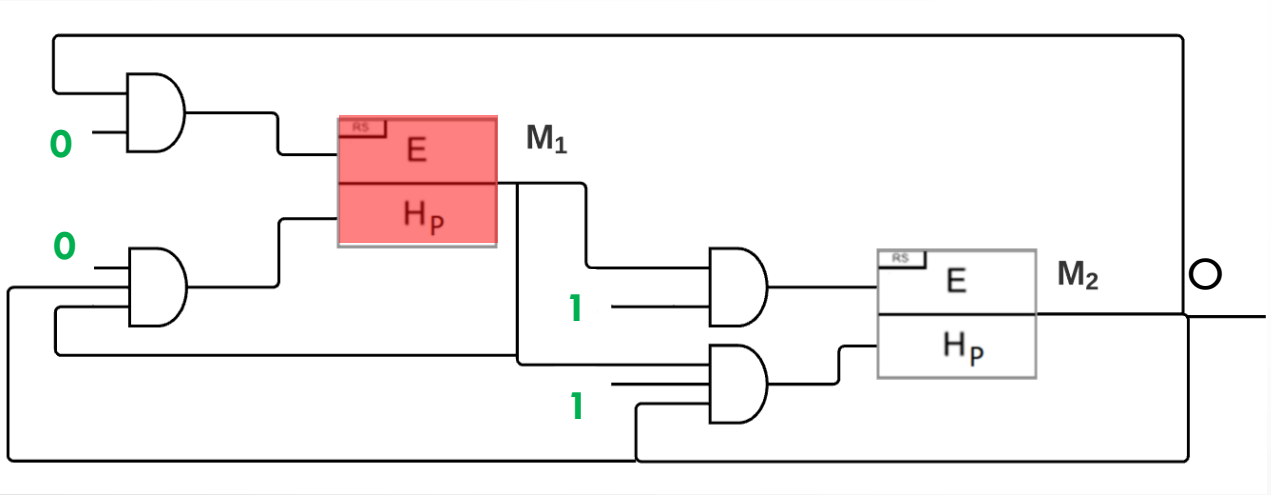
$\omega = (M2, M1)$

Second problem : Output values have to converge



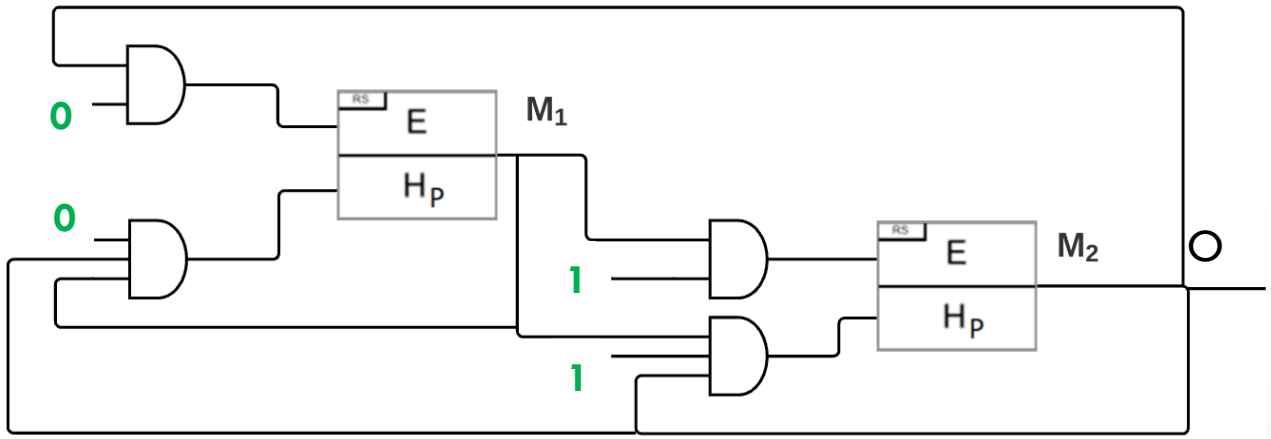
$\omega = (M2, M1)$

Second problem : Output values have to converge



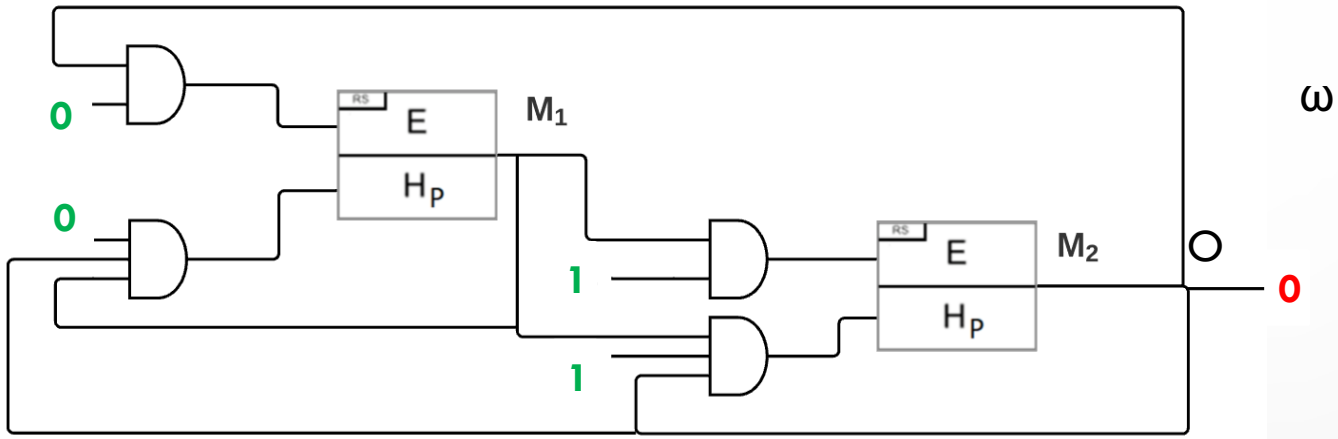
$\omega = (M2, M1)$

Second problem : Output values have to converge



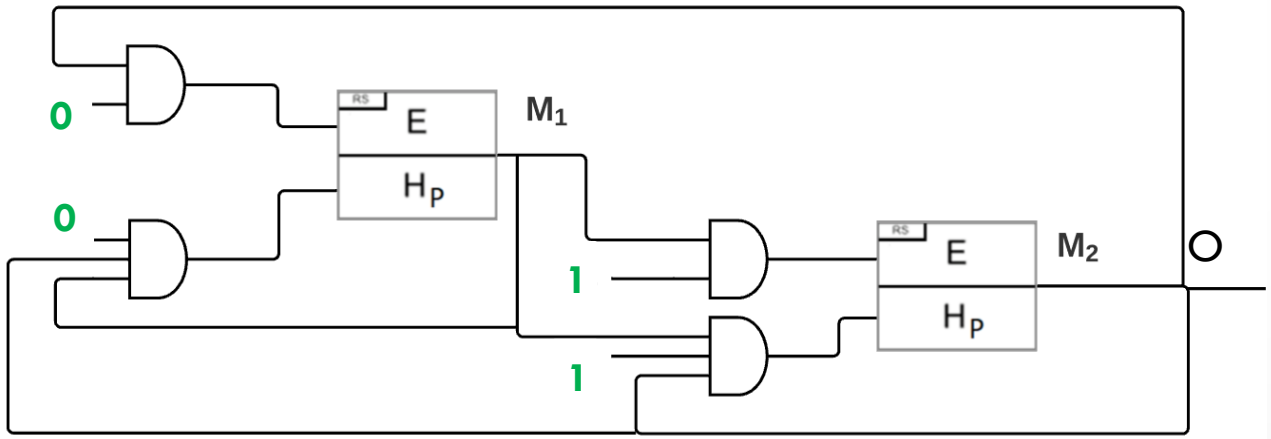
$\omega = (M2, M1)$

Second problem : Output values have to converge



$\omega = (M2, M1)$

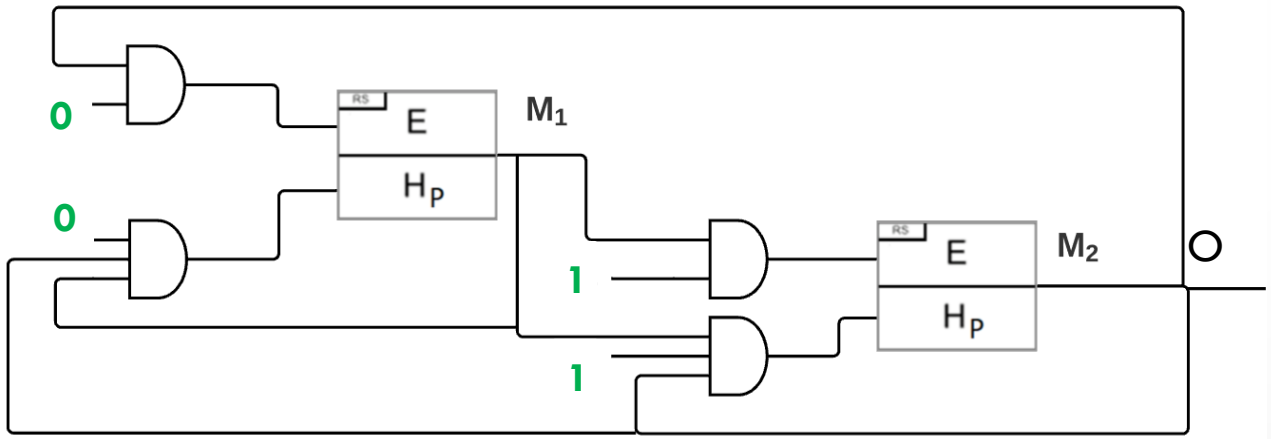
Second problem : Output values have to converge



$\omega = (M2, M1)$

0

Second problem : Output values have to converge



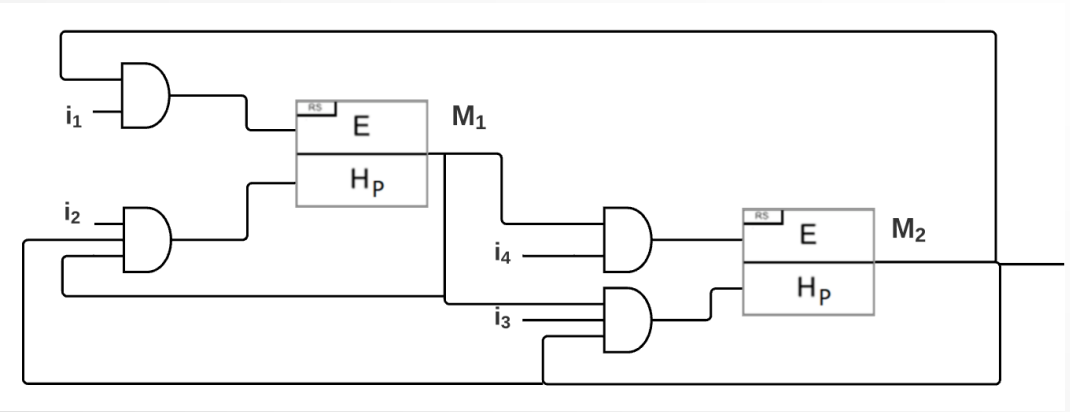
ω = (M2, M1)

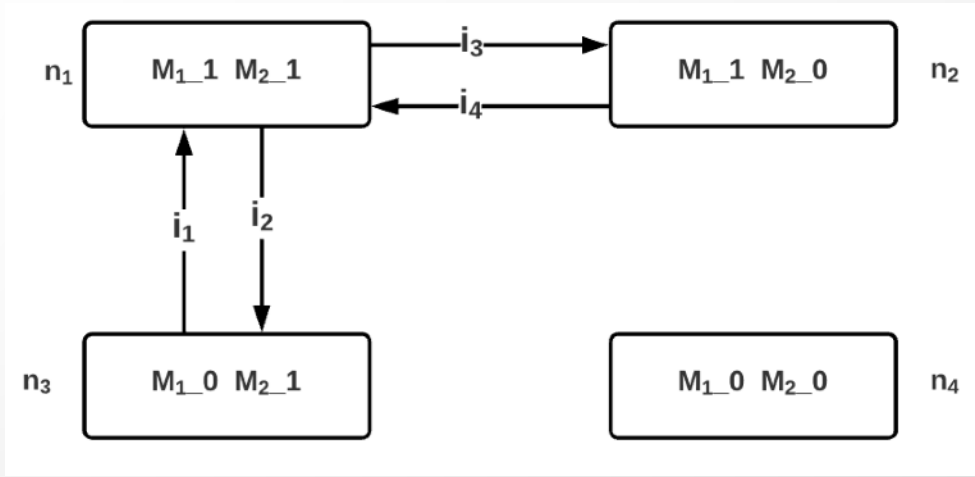
0

I = 0 0 1 1 → O = 1 or 0 ?

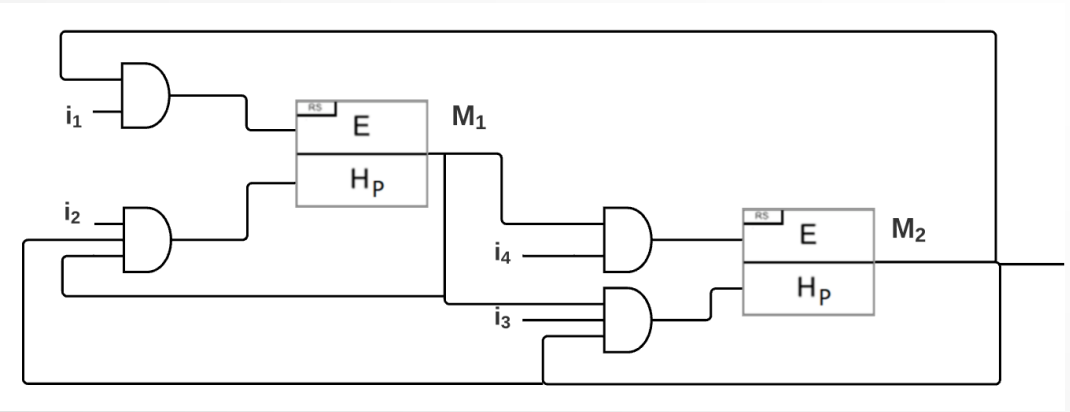
State graph representation of logical diagrams

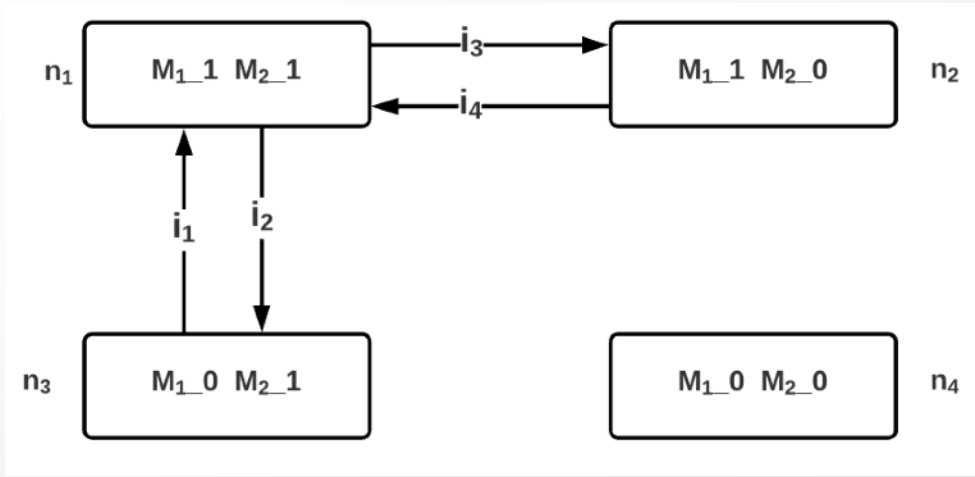
State graph representation of logical diagrams





State graph representation of logical diagrams

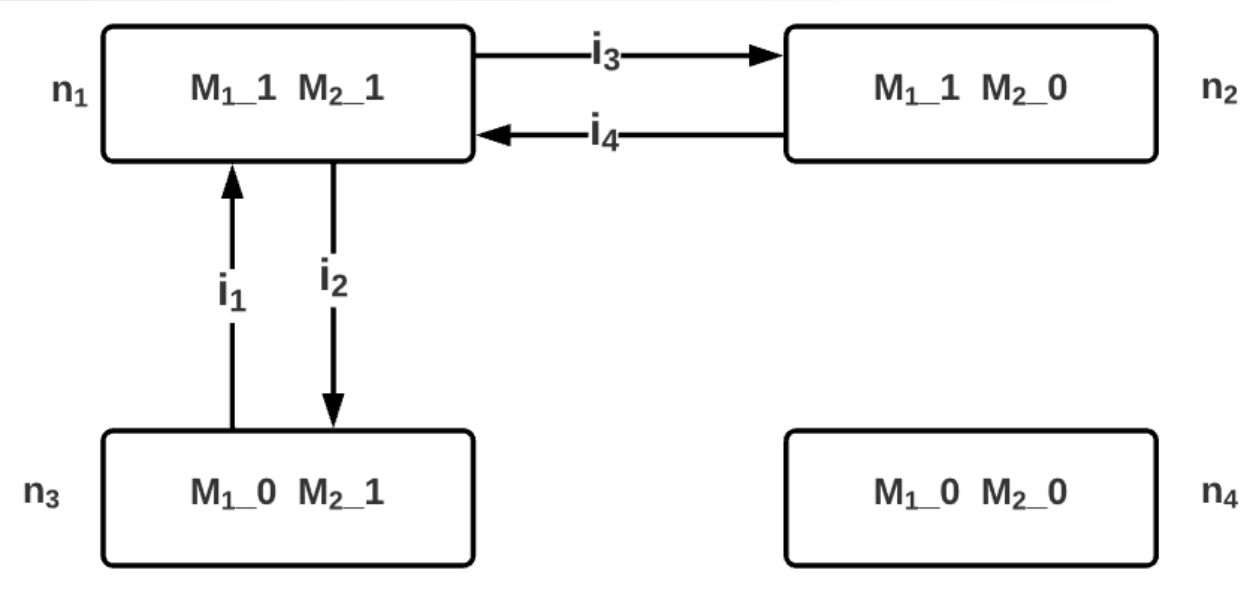




We propose a formal representation of logical diagrams for:

- Convergence property checking (and possible other properties).
- Generation of test sequences.

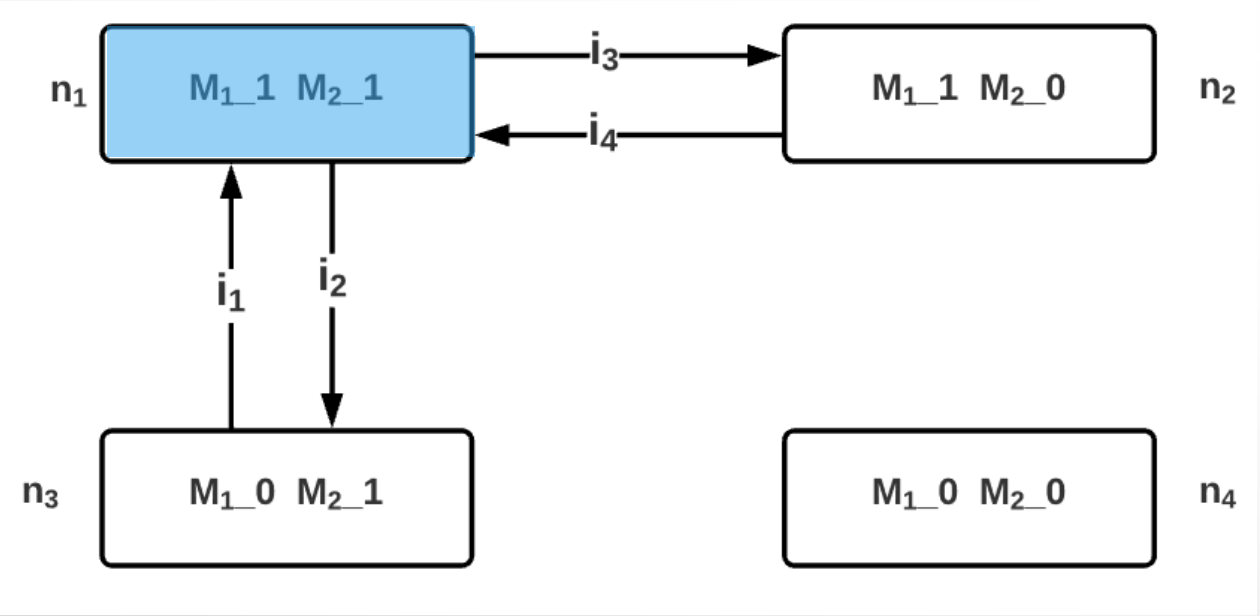
The Sequential Graph of State Transition (SGST)



SGST: (N, E)

- N: set of nodes. They represent the possible states of the logical diagram.
- E: set of edges. They represent all the theoratical evolution possibilities. Each edge corresponds to one evolution of one status block M.

Effective trails in the SGST and Convergence checking

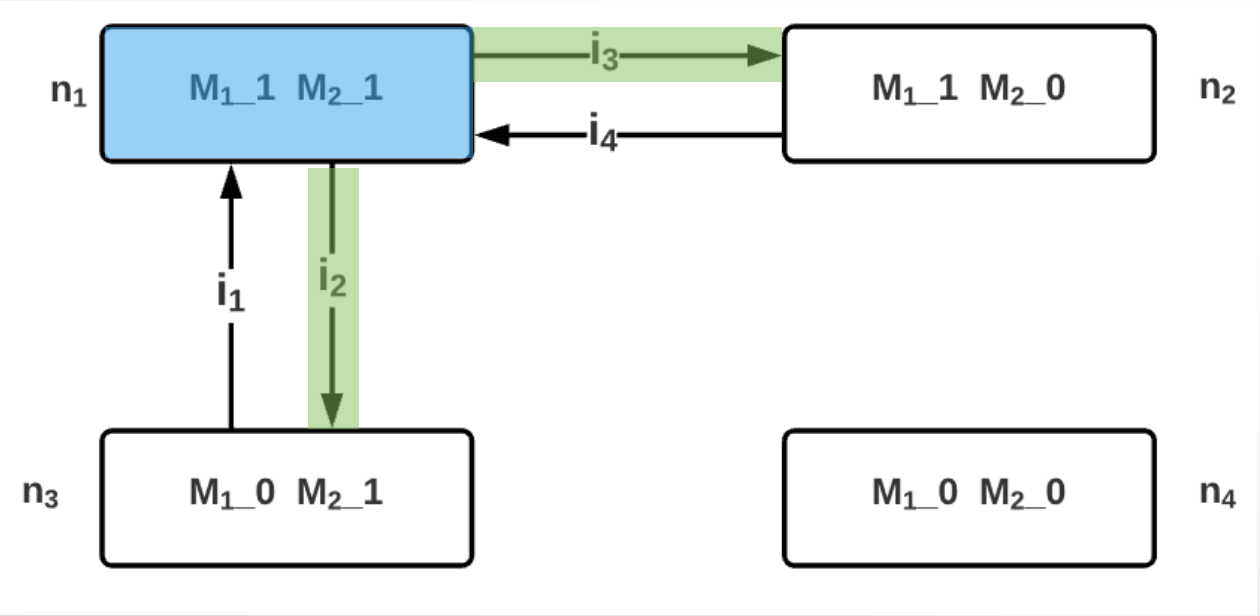


$\omega = \{ M2 , M1 \}$

Trail: n1

Condition:

Effective trails in the SGST and Convergence checking

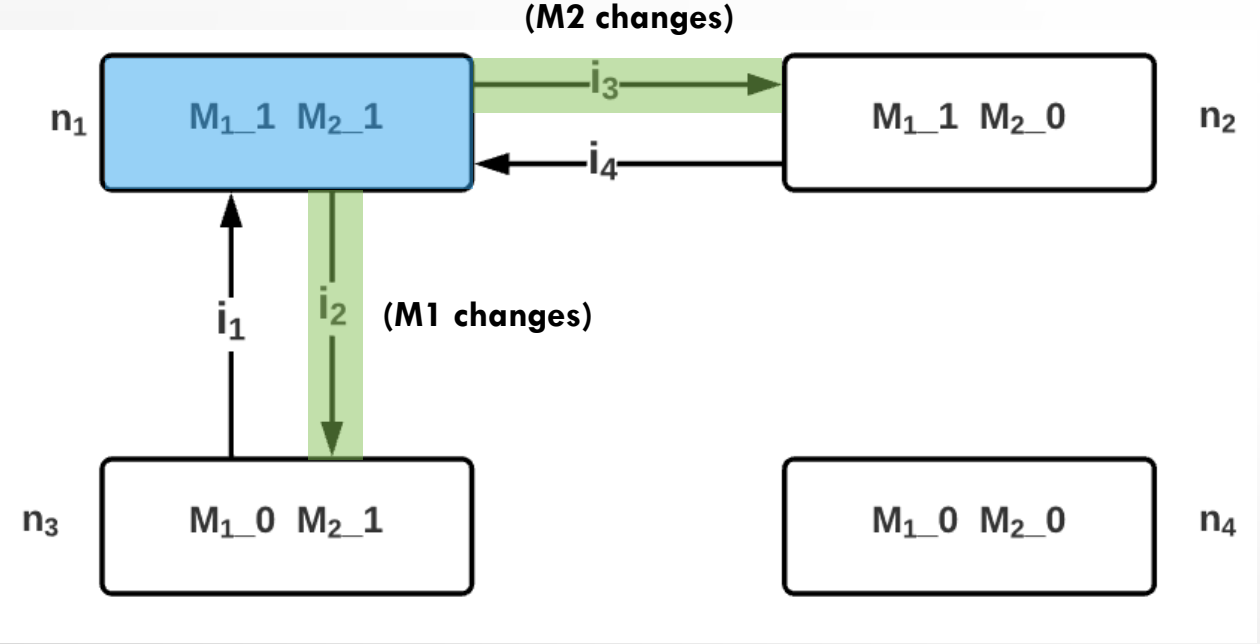


$\omega = \{ M2 , M1 \}$

Trail: n1

Condition:

Effective trails in the SGST and Convergence checking

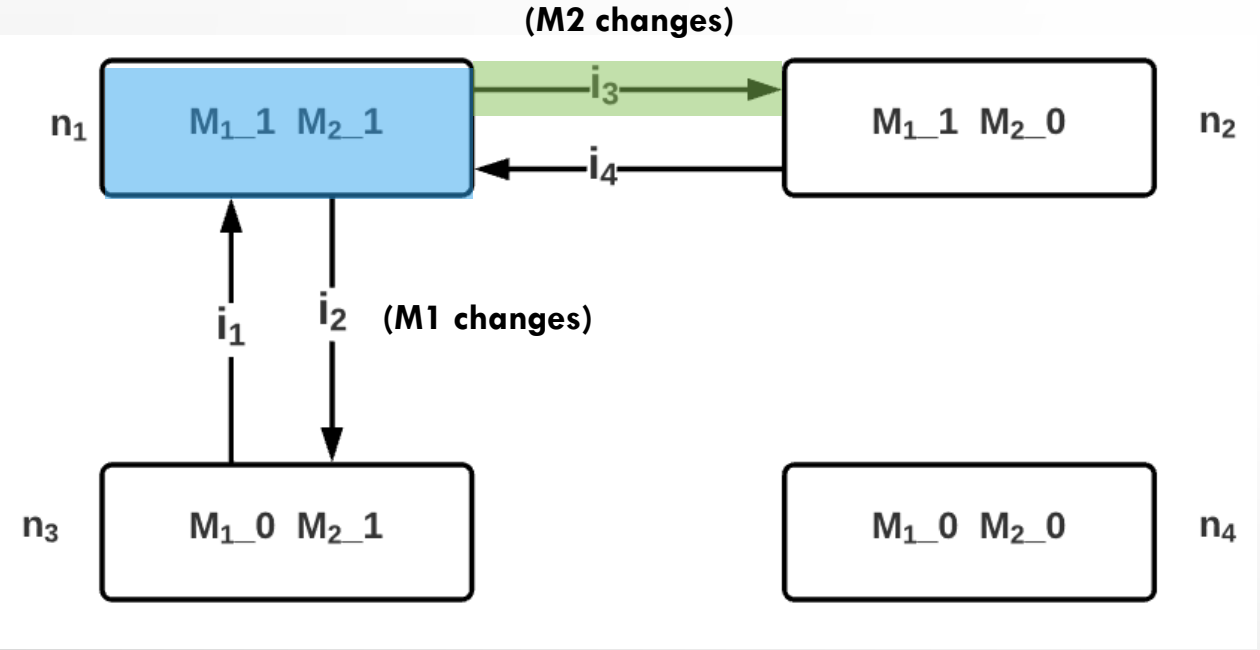


$\omega = \{ M2 , M1 \}$

Trail: n1

Condition:

Effective trails in the SGST and Convergence checking

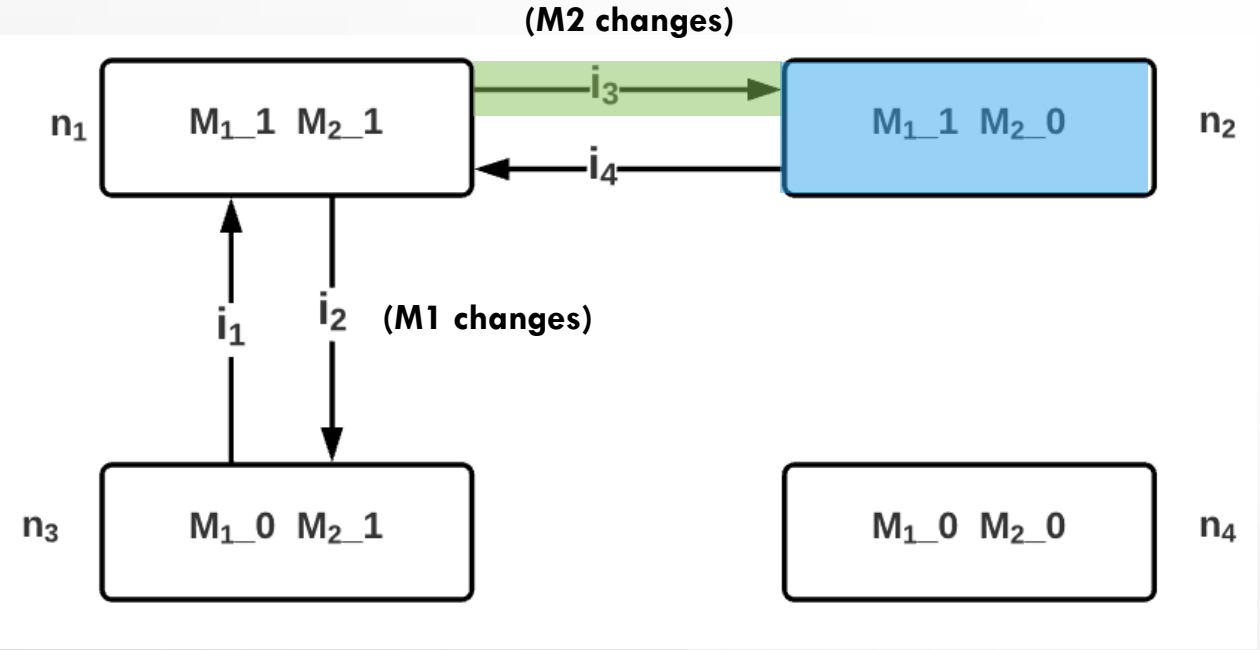


$\omega = \{ M2 , M1 \}$

Trail: n1- n2

Condition: i3

Effective trails in the SGST and Convergence checking

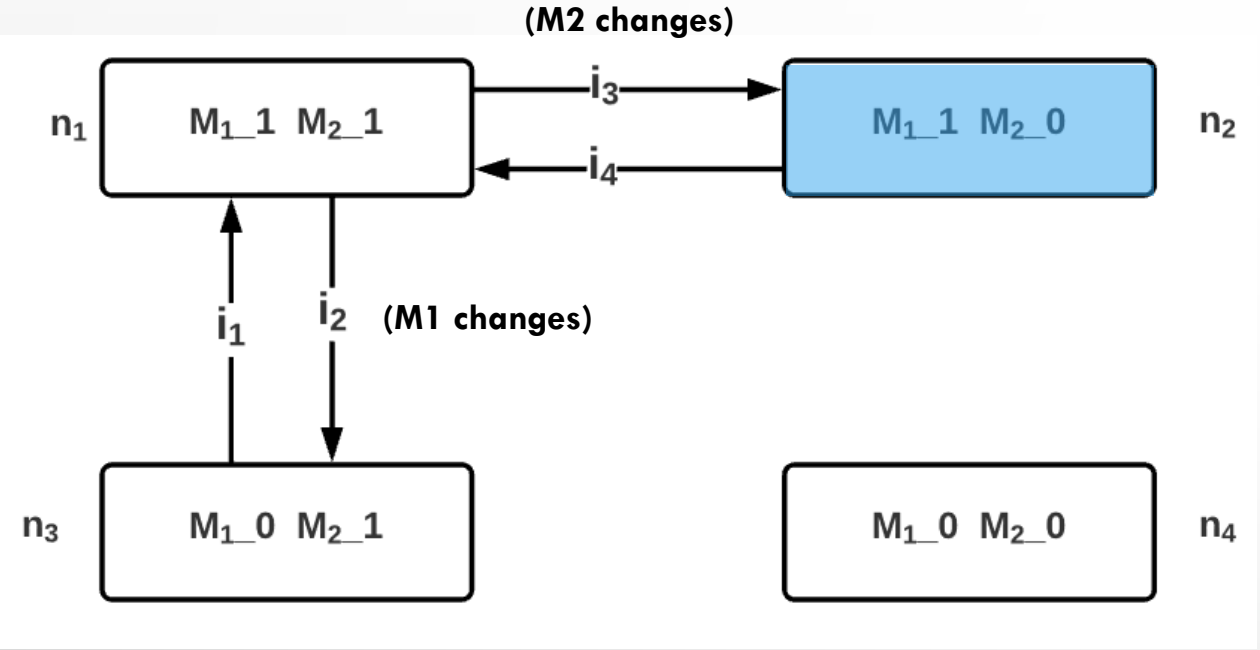


$\omega = \{ M2 , M1 \}$

Trail: n1- n2

Condition: i3

Effective trails in the SGST and Convergence checking

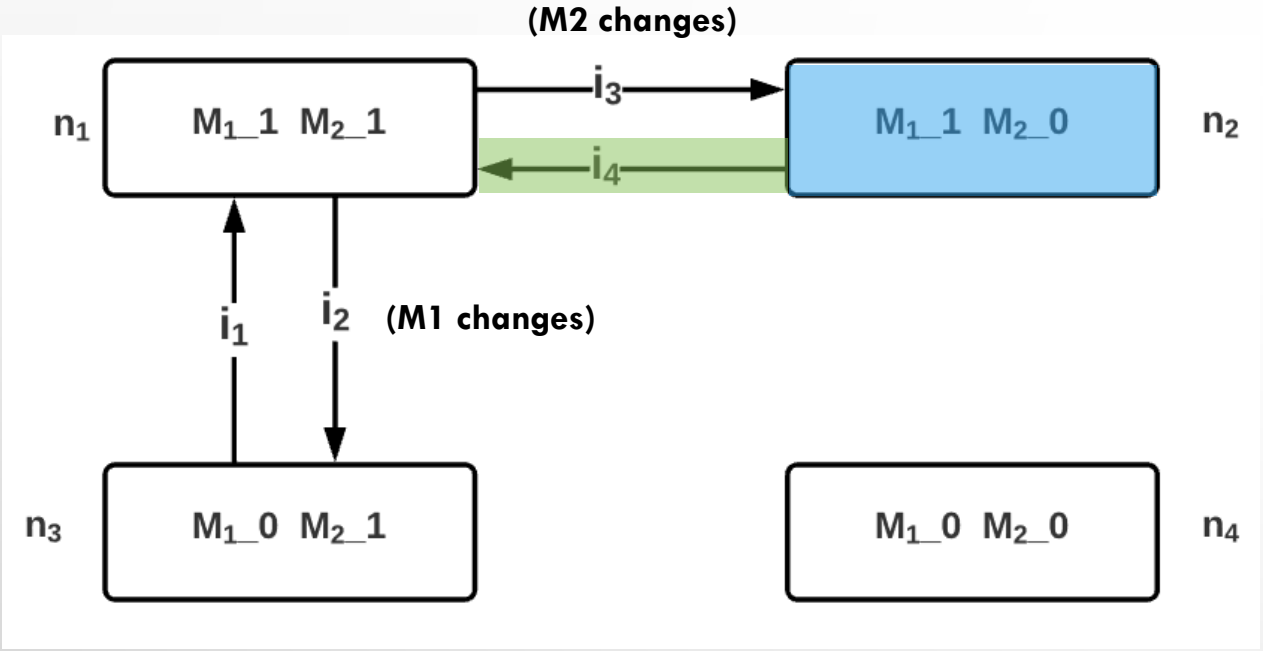


$\omega = \{ M2 , M1 \}$

Trail: n1- n2

Condition: i3

Effective trails in the SGST and Convergence checking

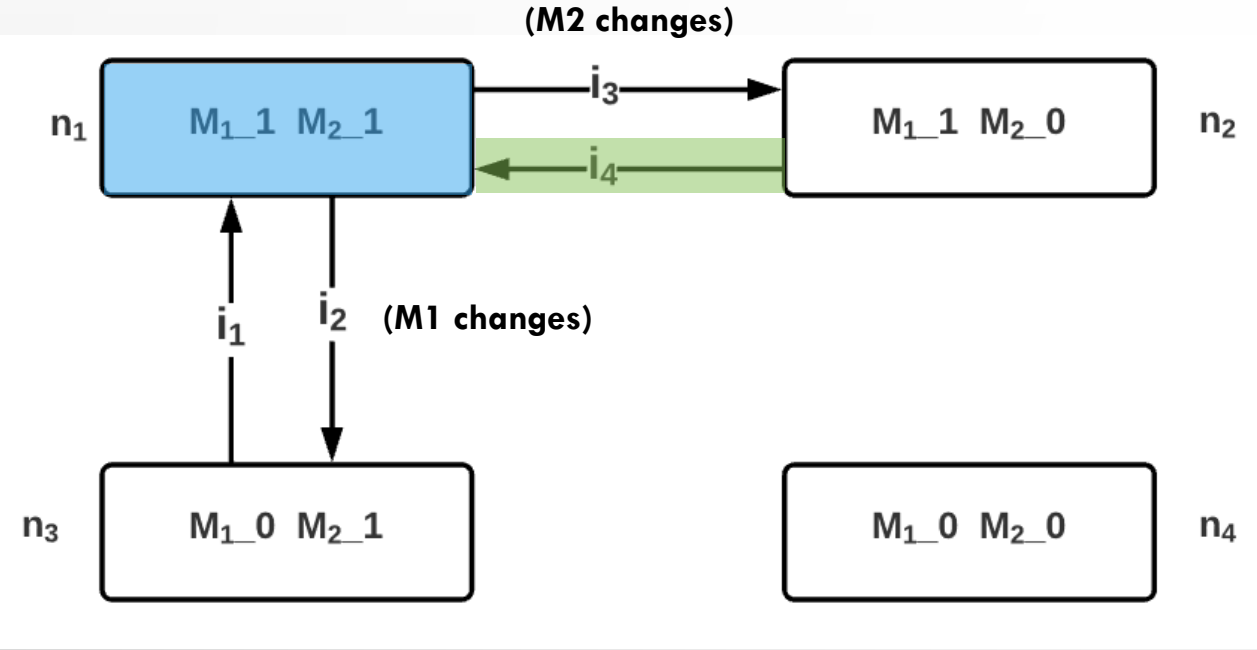


$\omega = \{ M2 , M1 \}$

Trail: n1- n2 – n1

Condition: i3 . i4

Effective trails in the SGST and Convergence checking

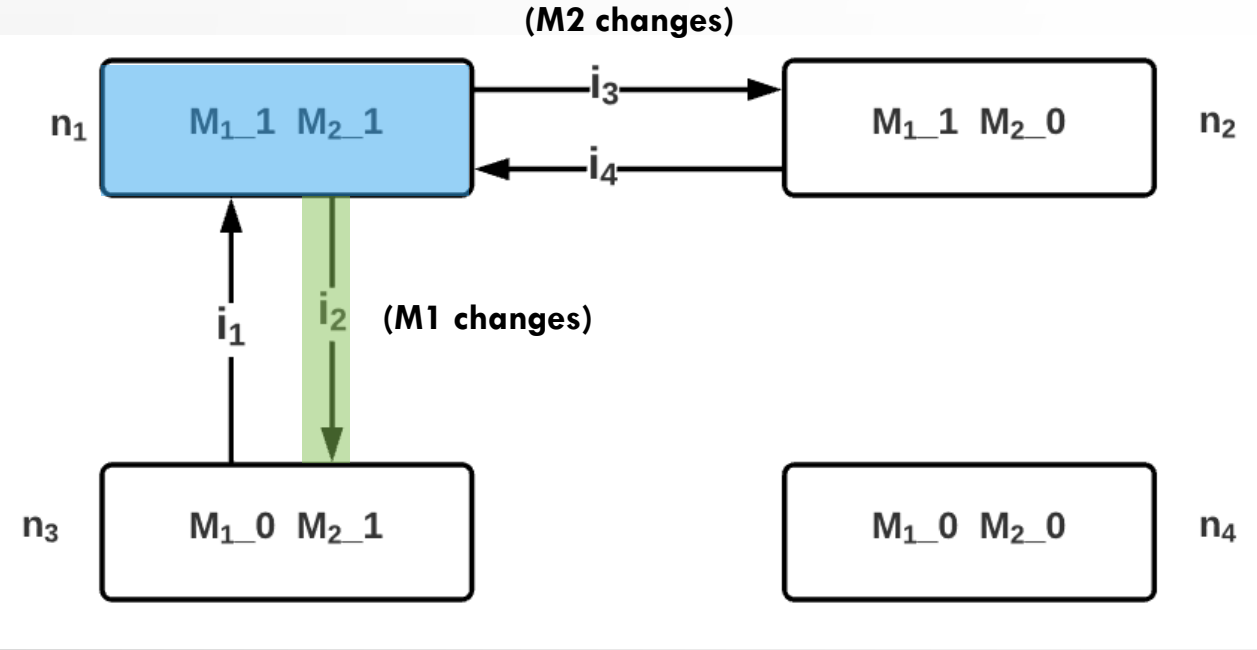


$\omega = \{ M2 , M1 \}$

Trail: n1- n2 – n1

Condition: i3 . i4

Effective trails in the SGST and Convergence checking

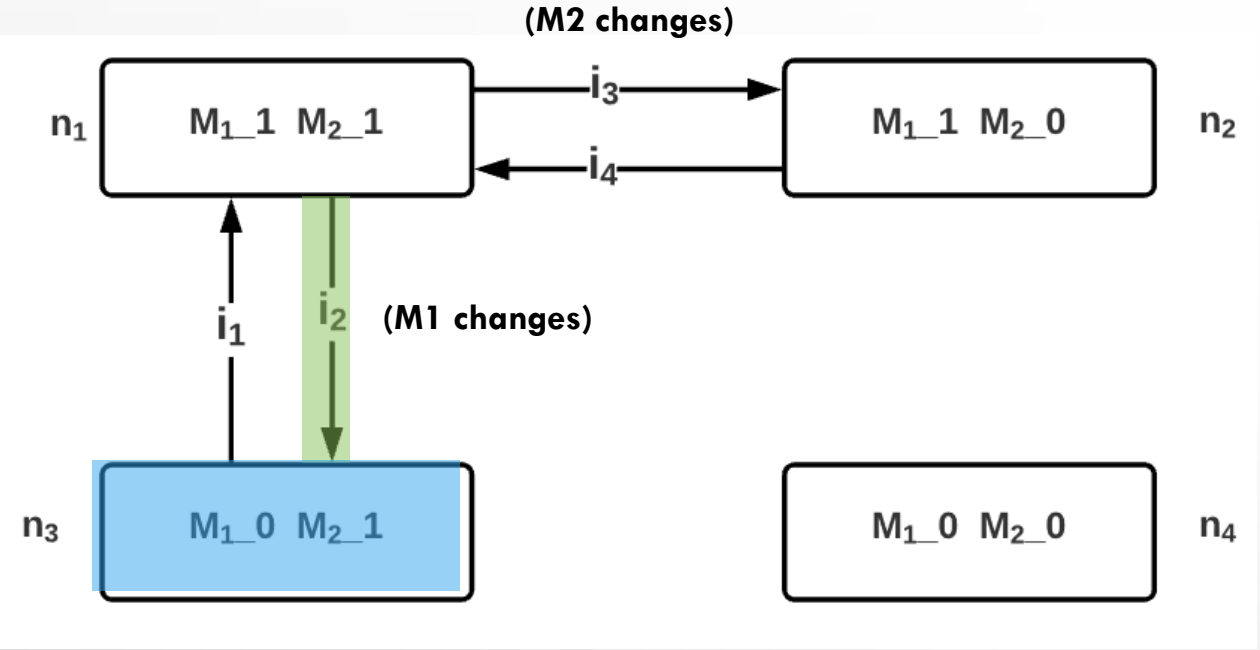


$\omega = \{ M2 , M1 \}$

Trail: n1- n2 - n1 – n3

Condition: i3 . i4 . i2

Effective trails in the SGST and Convergence checking

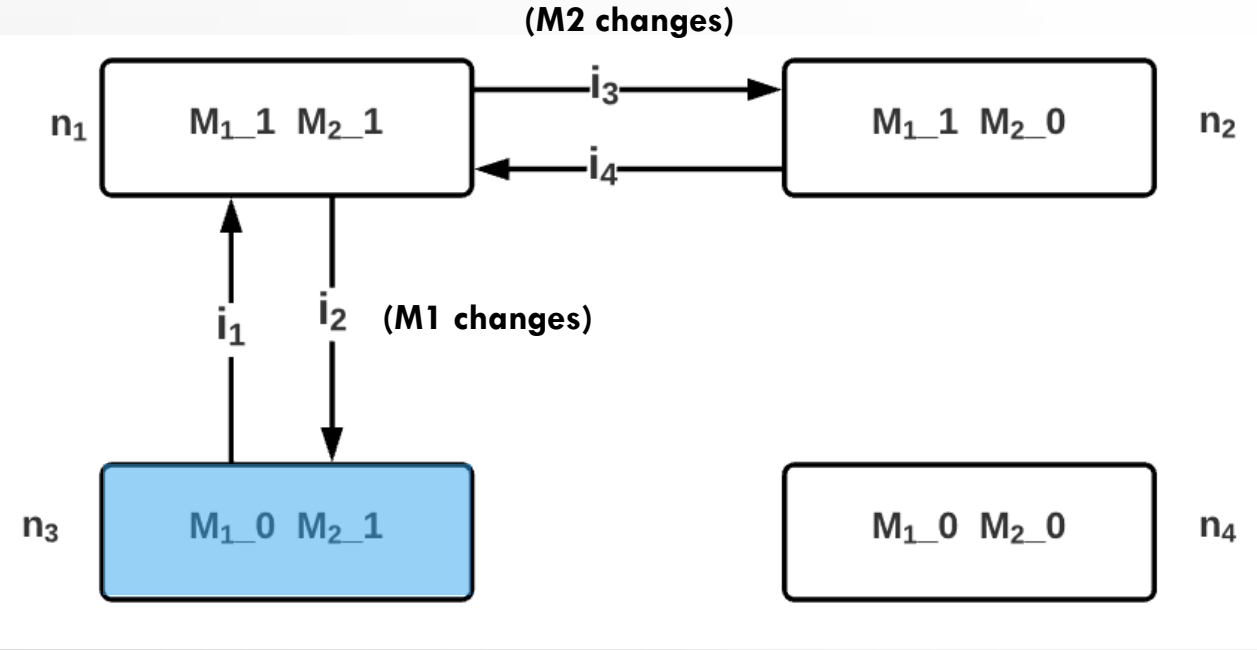


$\omega = \{ M2 , M1 \}$

Trail: n1- n2 - n1 – n3

Condition: i3 . i4 . i2

Effective trails in the SGST and Convergence checking

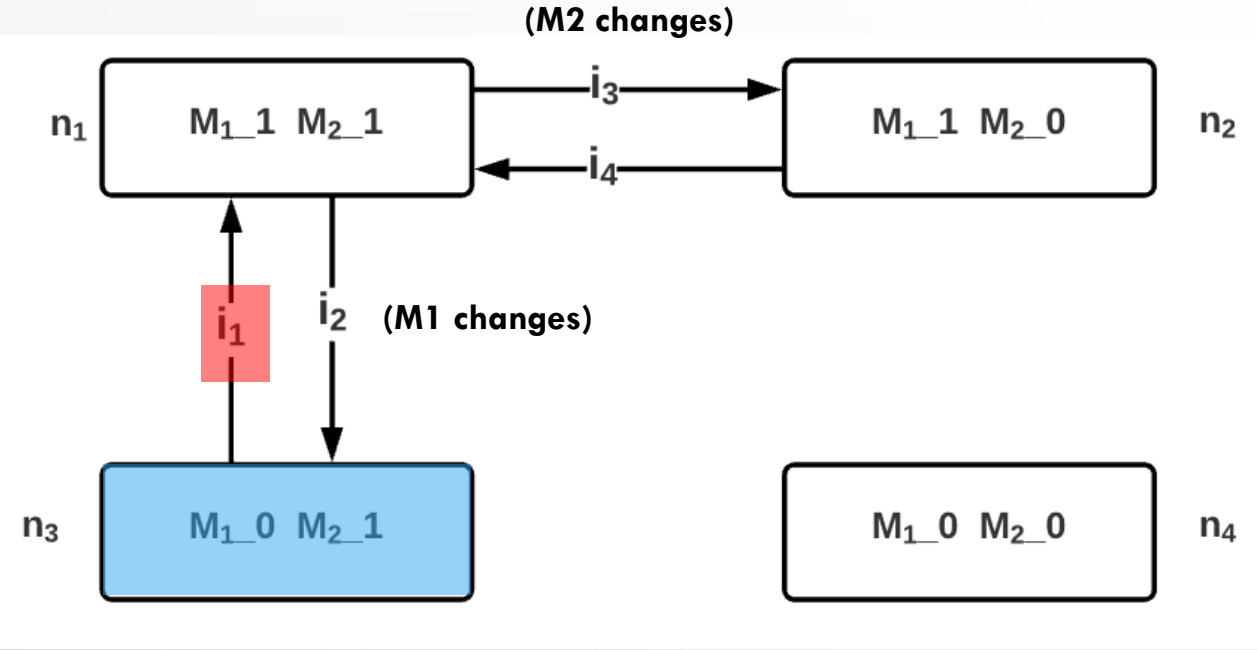


$\omega = \{ M2 , M1 \}$

Trail: n1- n2 - n1 – n3

Condition: i3 . i4 . i2

Effective trails in the SGST and Convergence checking

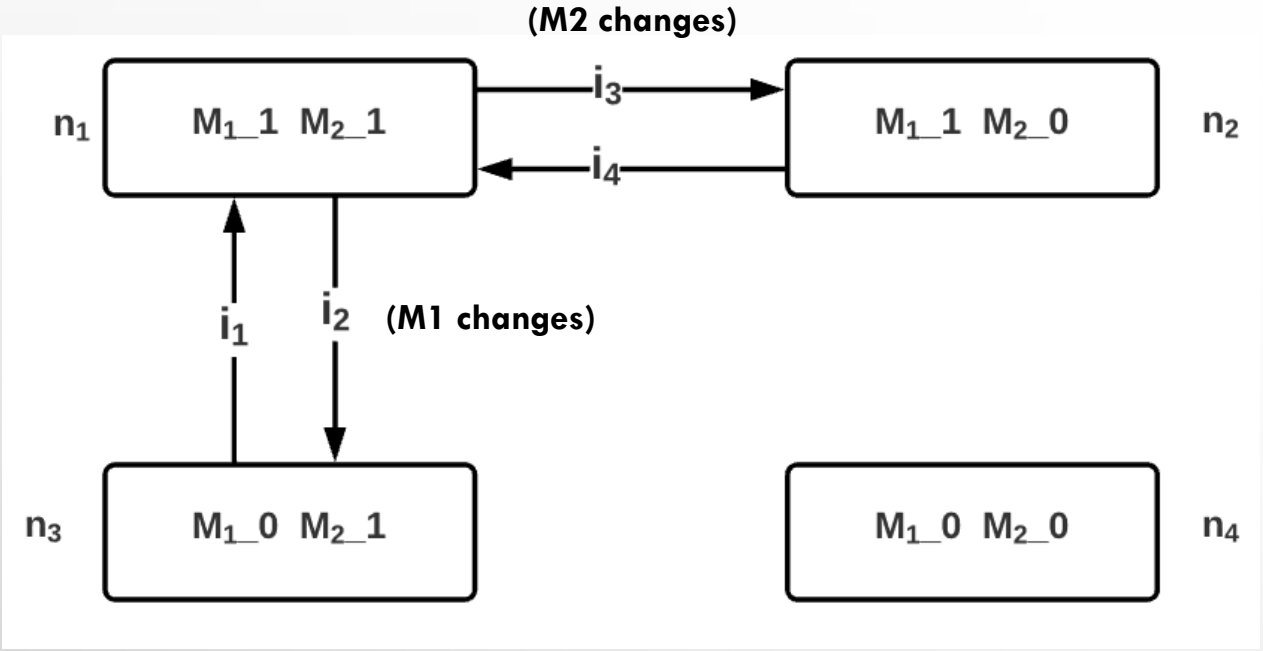


$\omega = \{ M2 , M1 \}$

Trail: n1- n2 - n1 – n3

Condition: i3 . i4 . i2 . ~i1

Effective trails in the SGST and Convergence checking

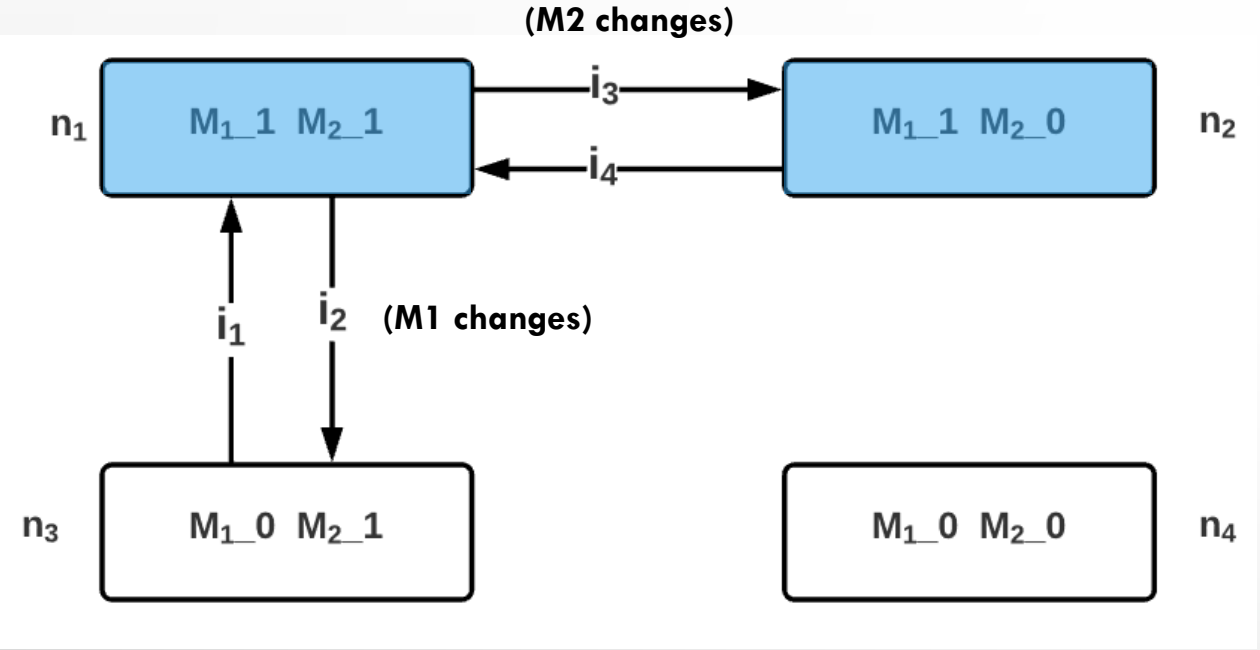


$\omega = \{ M2 , M1 \}$

Trail: n1- n2 - n1 – n3

Condition: i3 . i4 . i2 . ~i1

Effective trails in the SGST and Convergence checking



$\omega = \{ M2 , M1 \}$

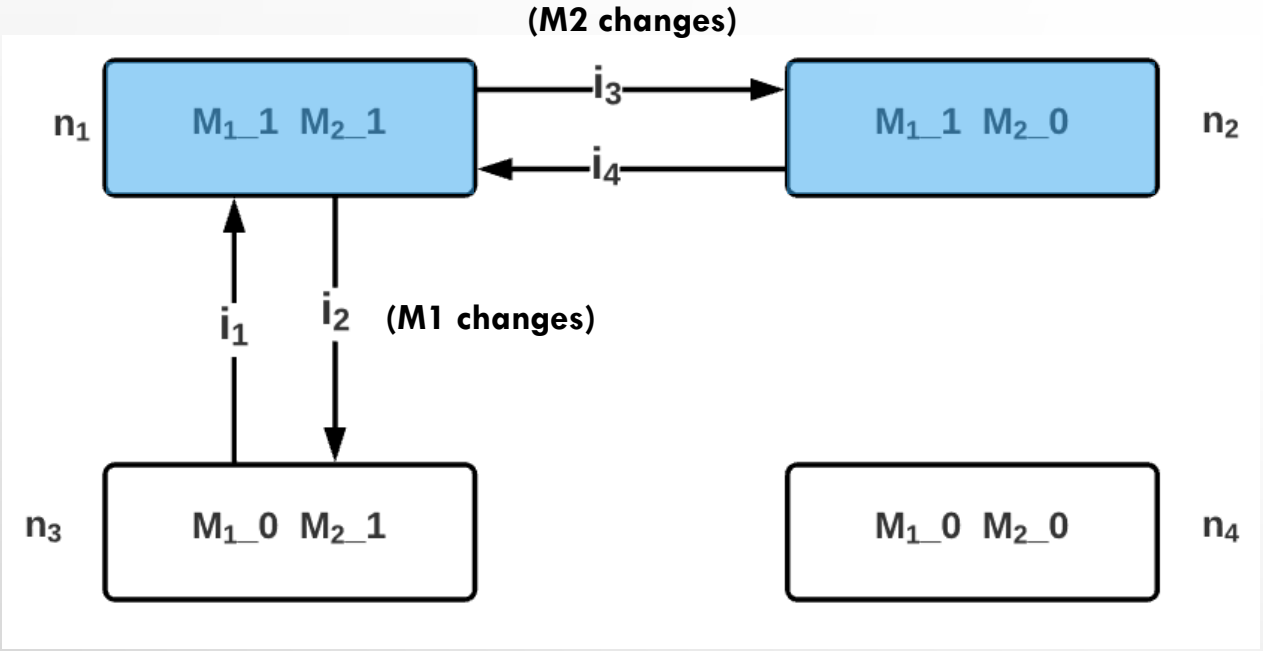
Trail: n1- n2 - n1 – n3

Condition: i3 . i4 . i2 . ~i1

Trail: n1- n2 - n1 – n2

Condition: i3 . i4 . ~i2

Effective trails in the SGST and Convergence checking



$\omega = \{ M2 , M1 \}$

Trail: n1- n2 - n1 – n3

Condition: i3 . i4 . i2 . ~i1

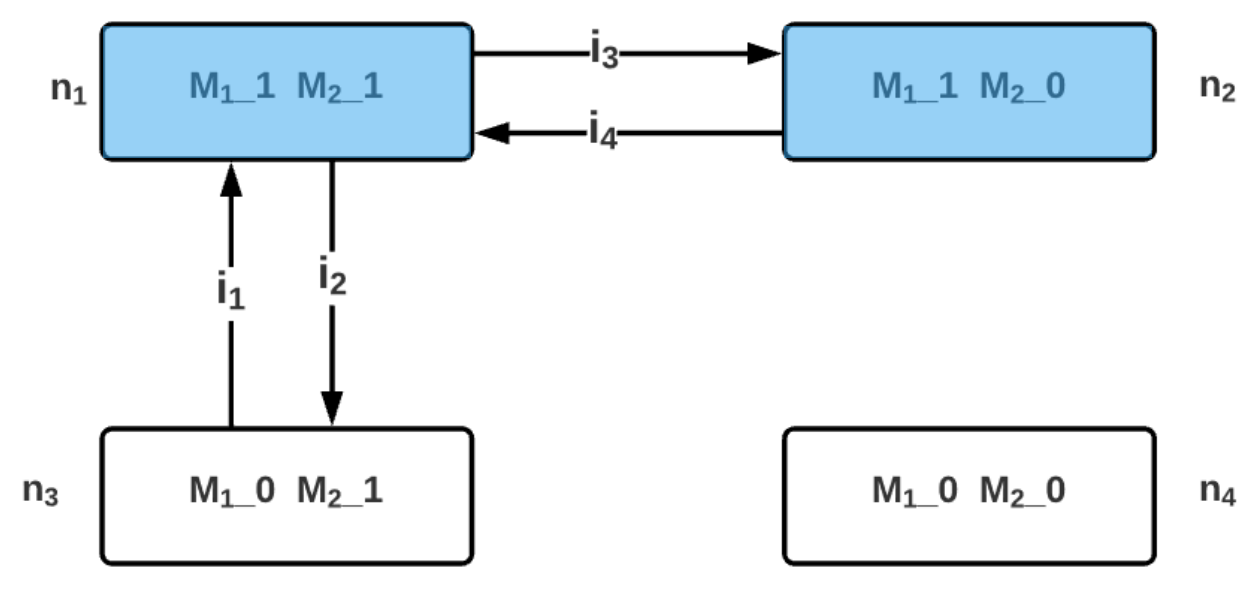
⇒ The infinite traversal of circuit n1-n2 is possible :

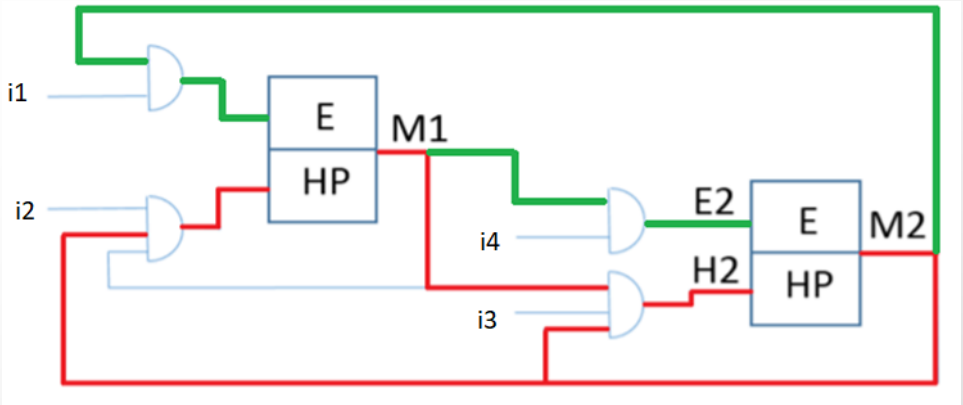
Inputs 0 0 1 1 and 1 0 1 1 cause a non-stable signal at the output of M2

Trail: n1- n2 - n1 – n2

Condition: i3 . i4 . ~i2

Effective trails in the SGST and Convergence checking

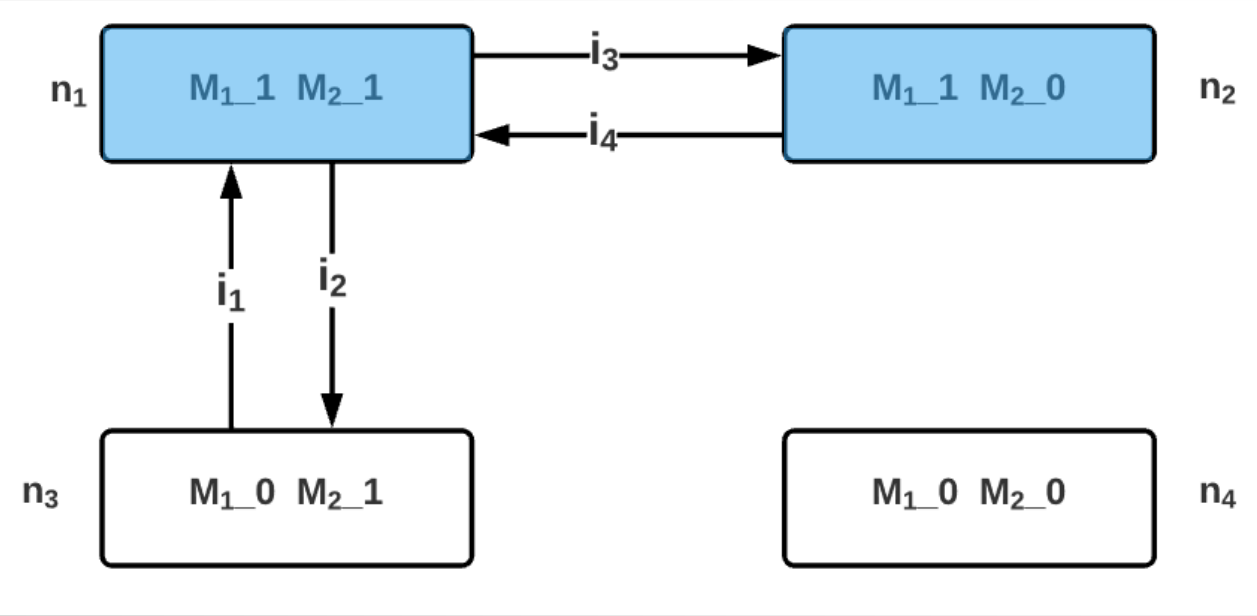


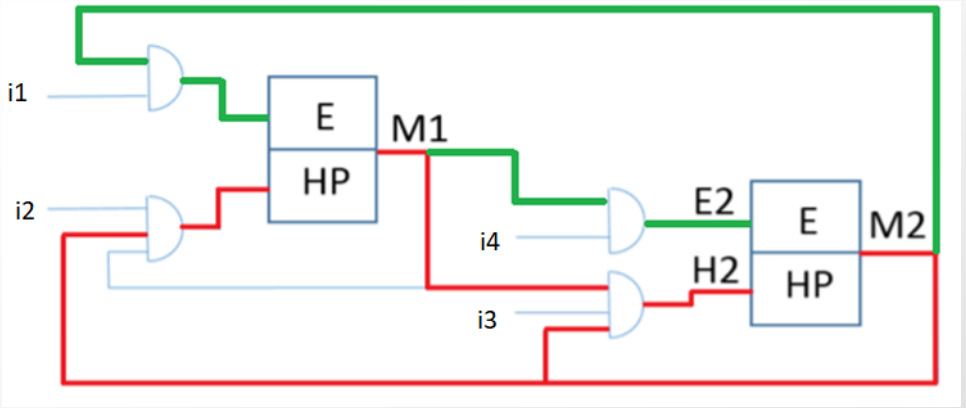


⇒ The infinite traversal of circuit n1-n2 is possible :

Inputs 0 0 1 1 and 1 0 1 1 cause a non-stable signal at the output of M2

Effective trails in the SGST and Convergence checking



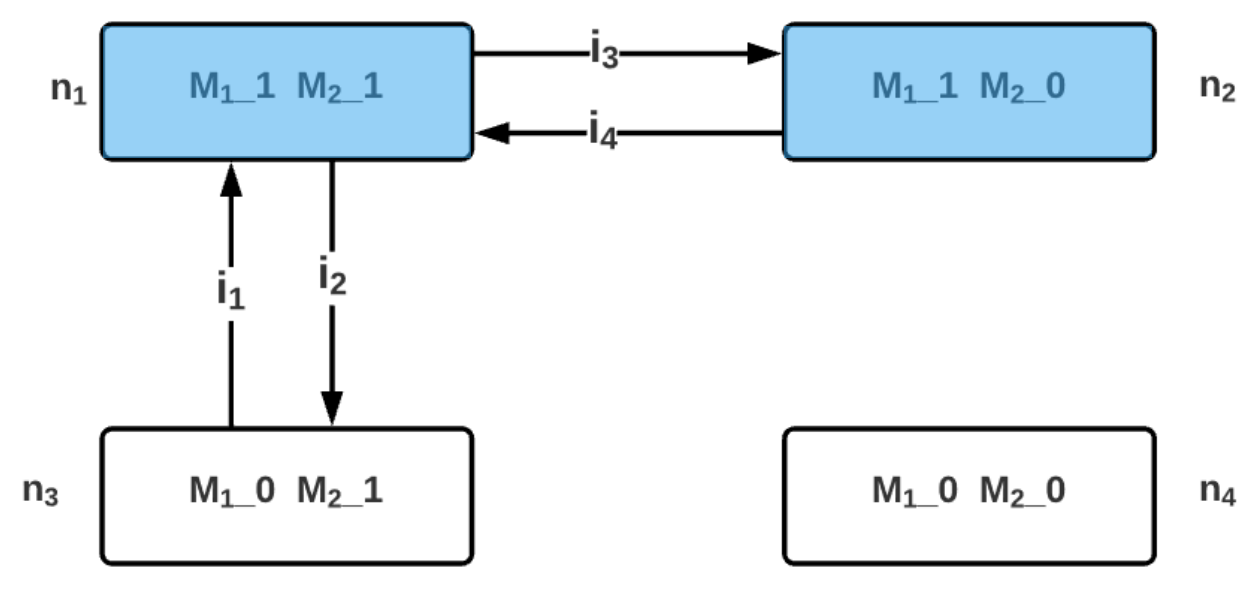


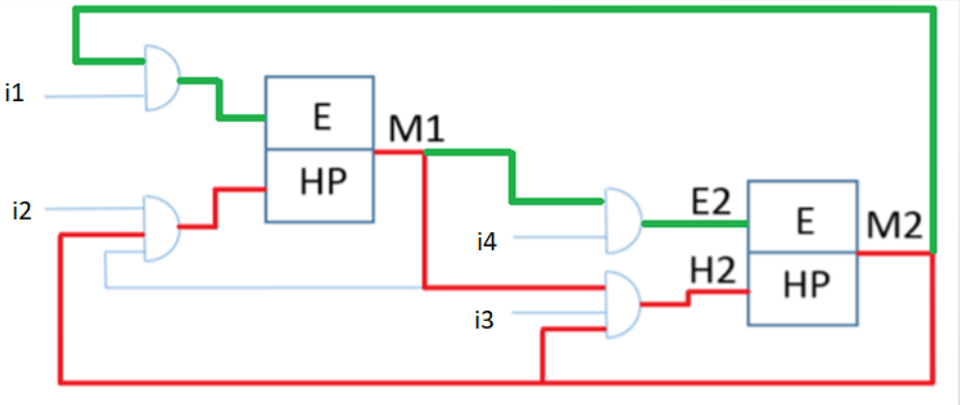
Loop structures cause this behavior.
The presence of loops doesn't necessarily mean that a non-convergent scenario exists.

⇒ The infinite traversal of circuit n1-n2 is possible :

Inputs 0 0 1 1 and 1 0 1 1 cause a non-stable signal at the output of M2

Effective trails in the SGST and Convergence checking





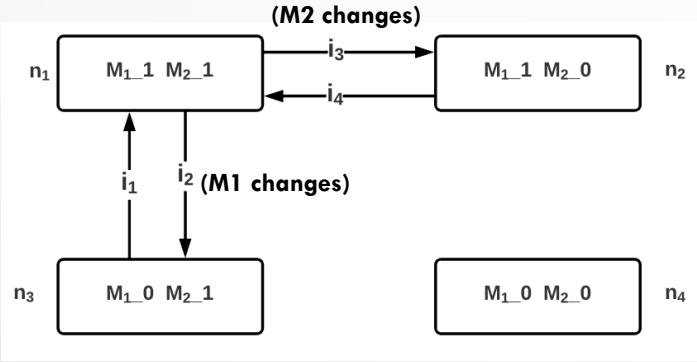
Loop structures cause this behavior.
The presence of loops doesn't necessarily mean that a non-convergent scenario exists.

⇒ The infinite traversal of circuit n1-n2 is possible :

Inputs 0 0 1 1 and 1 0 1 1 cause a non-stable signal at the output of M2

This is verifiable on the graph by finding trails that contain circuits.

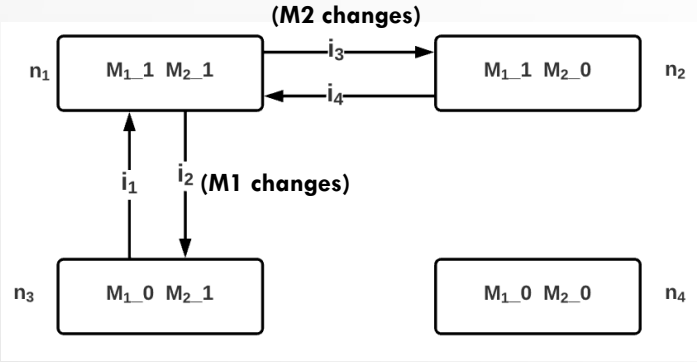
Permanent state automaton (PSA)



SGST

$\omega = \{ M2 , M1 \}$

Permanent state automaton (PSA)

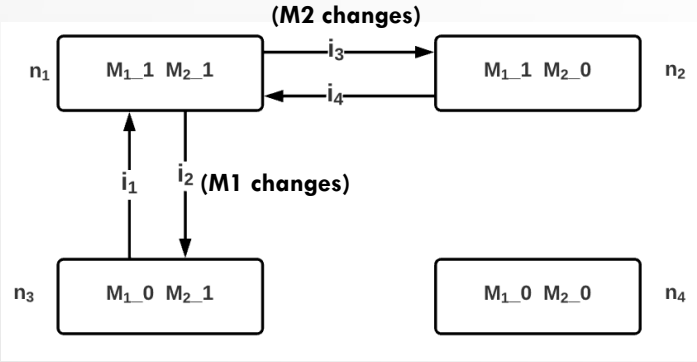


SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation

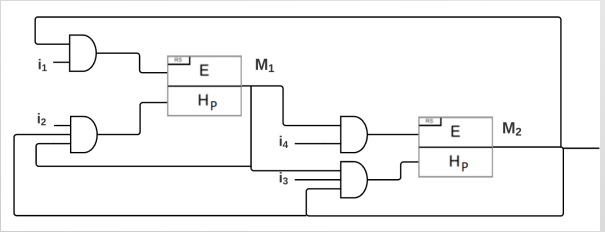
Permanent state automaton (PSA)



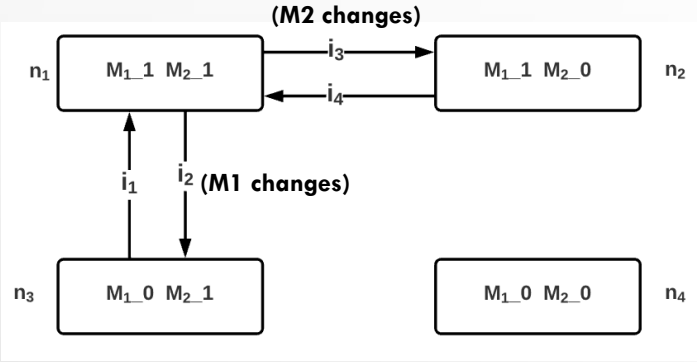
SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



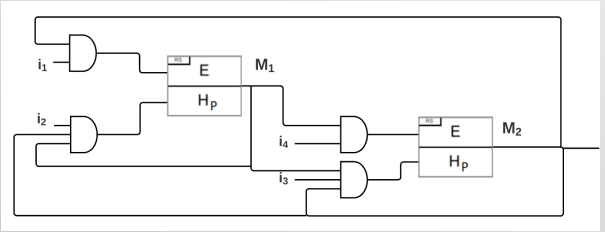
Permanent state automaton (PSA)



SGST

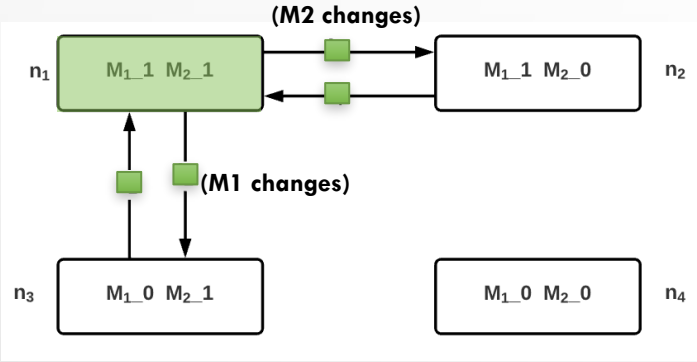
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

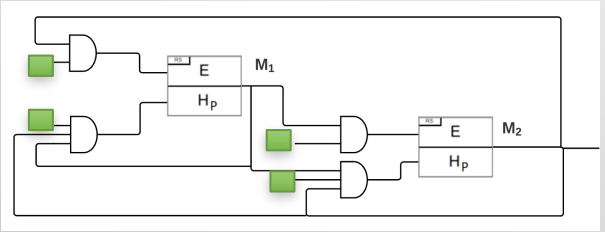
Permanent state automaton (PSA)



SGST

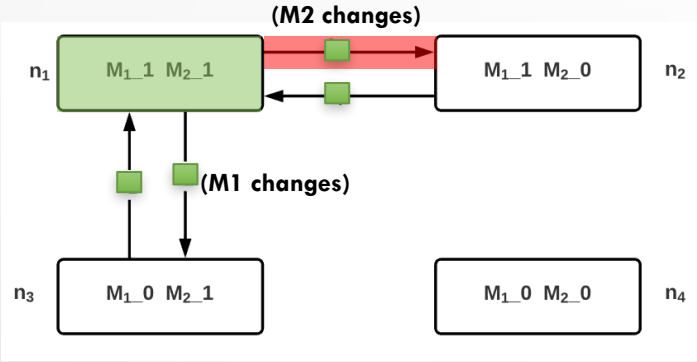
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

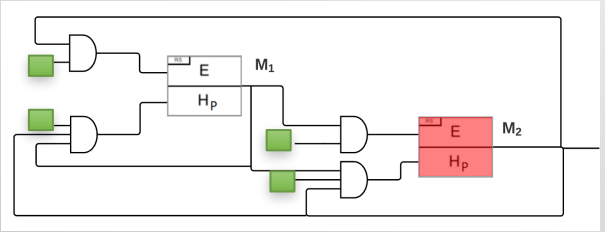
Permanent state automaton (PSA)



SGST

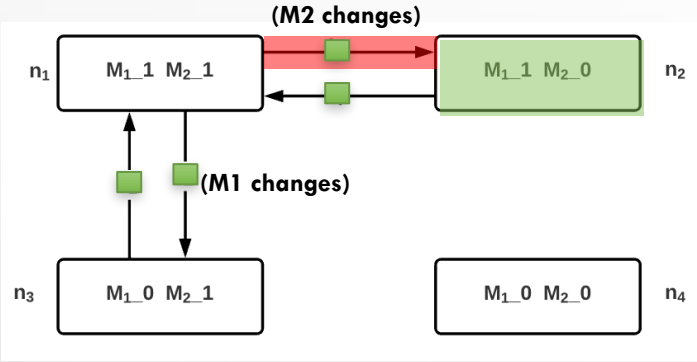
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

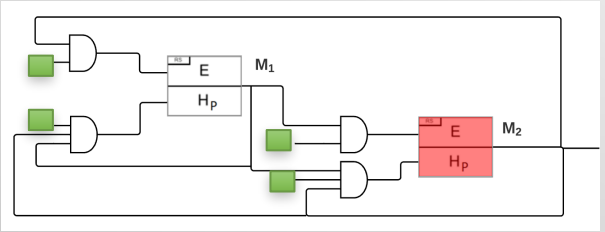
Permanent state automaton (PSA)



SGST

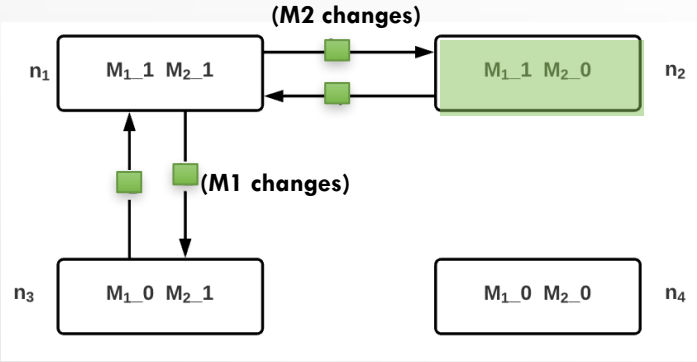
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

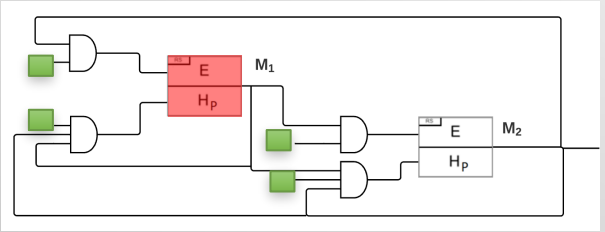
Permanent state automaton (PSA)



SGST

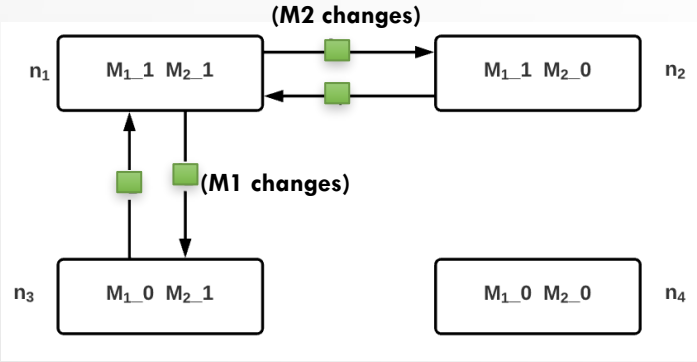
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

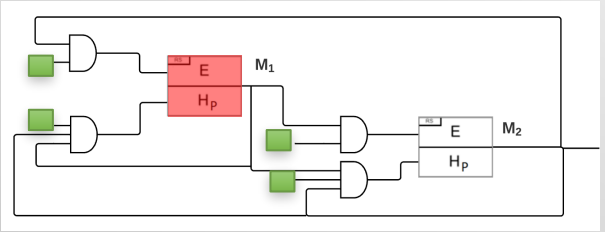
Permanent state automaton (PSA)



SGST

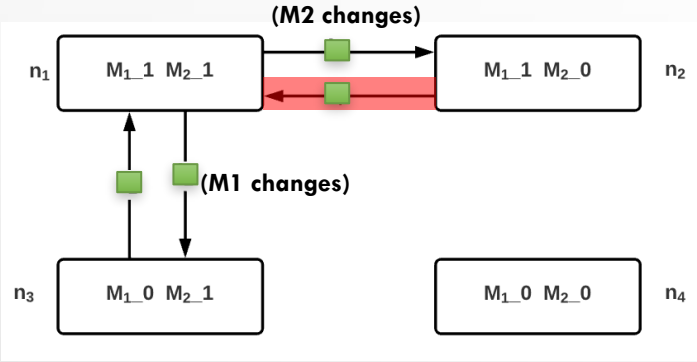
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

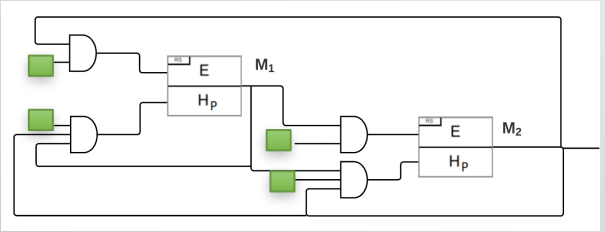
Permanent state automaton (PSA)



SGST

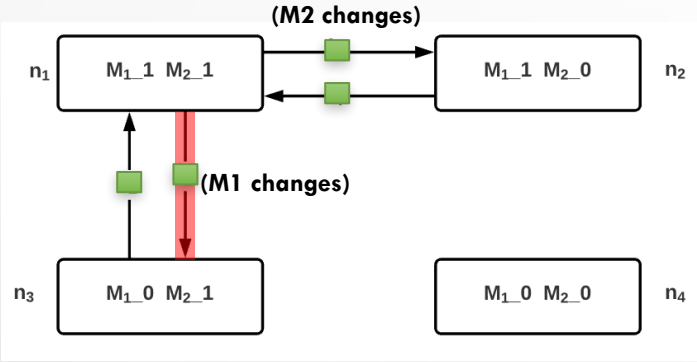
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

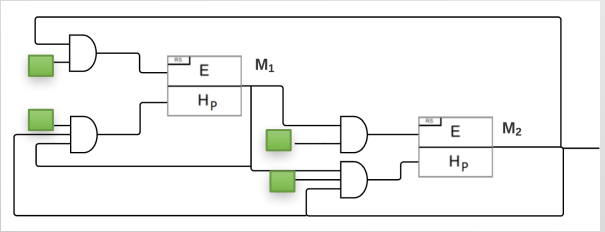
Permanent state automaton (PSA)



SGST

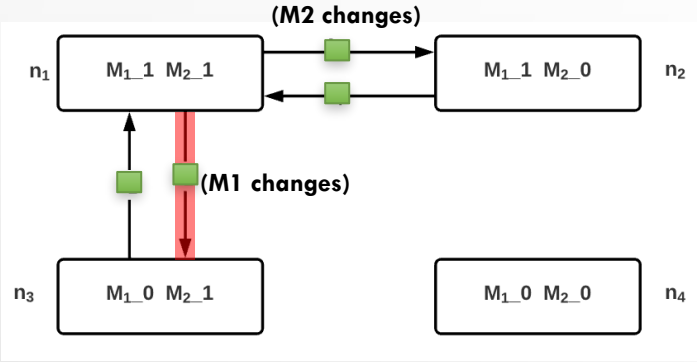
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

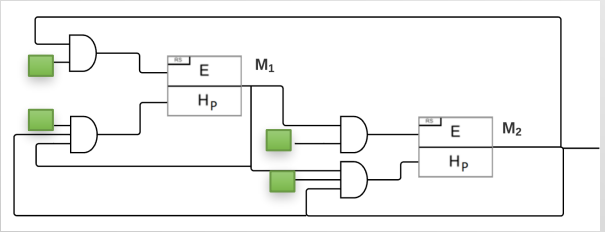
Permanent state automaton (PSA)



SGST

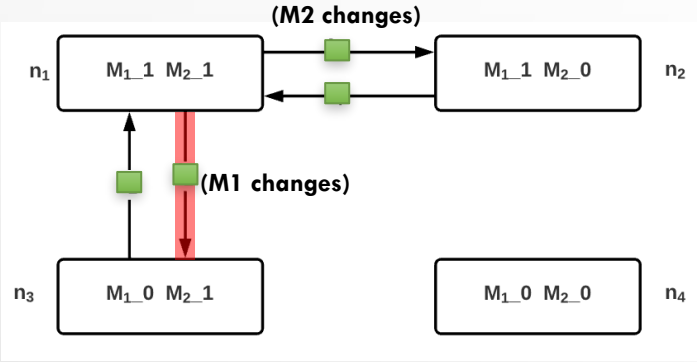
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

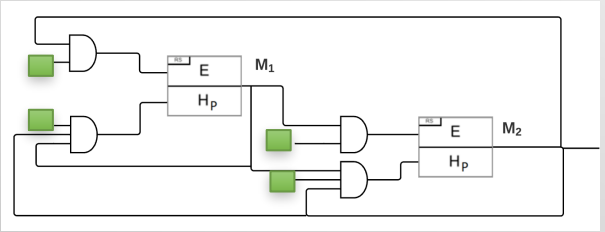
Permanent state automaton (PSA)



SGST

$\omega = \{ M2 , M1 \}$

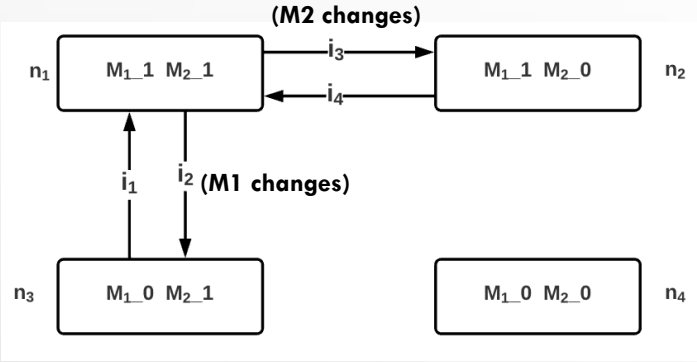
An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

Condition1 : i3.i4.i2.~i1

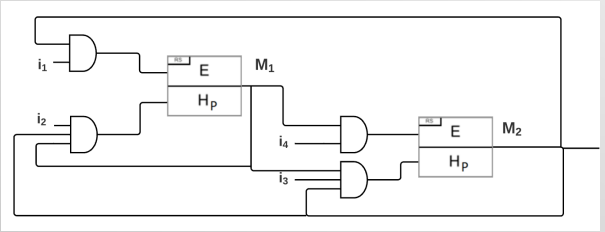
Permanent state automaton (PSA)



SGST

$\omega = \{ M2 , M1 \}$

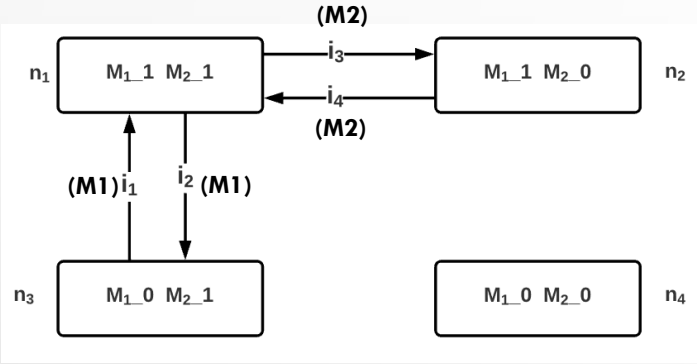
An arc represents an elementary evaluation



Trail1 : n1 – n2 – n1 – n3

Condition1 : i3.i4.i2.~i1

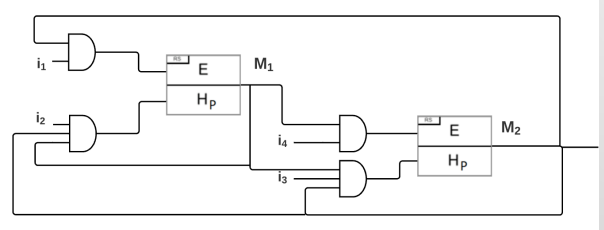
Permanent state automaton (PSA)



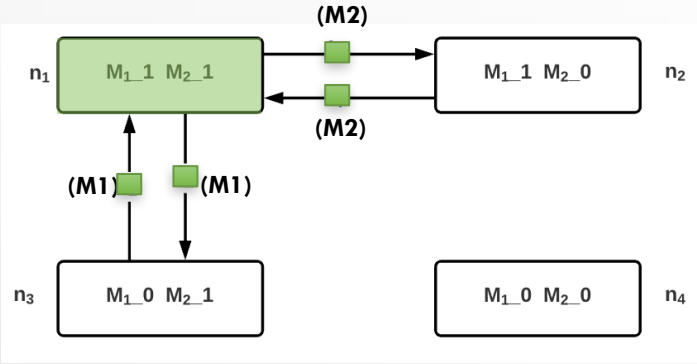
SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



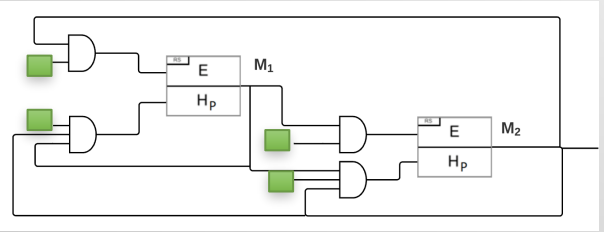
Permanent state automaton (PSA)



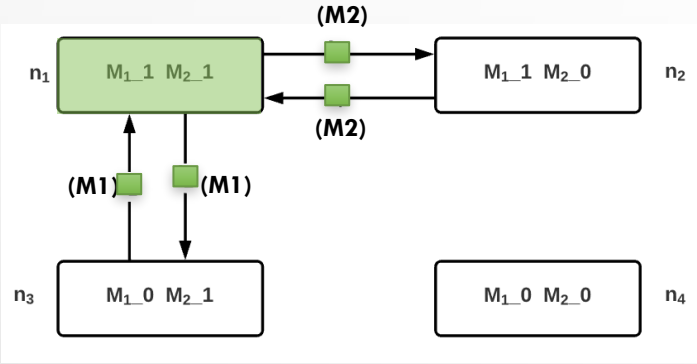
SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



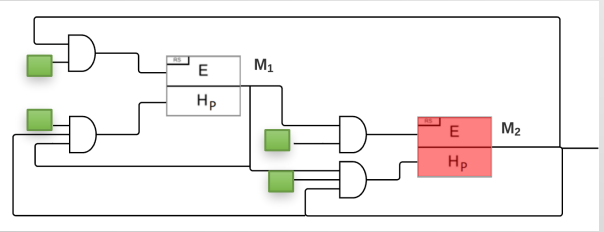
Permanent state automaton (PSA)



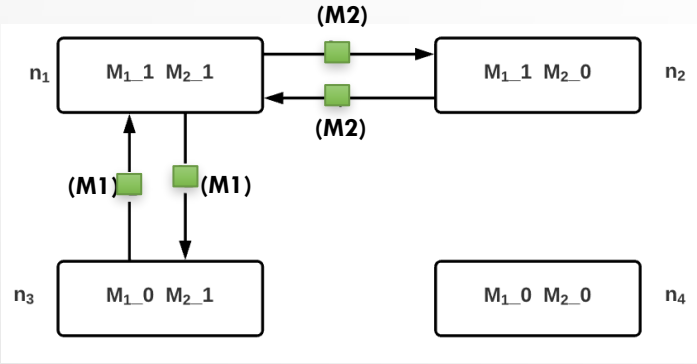
SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



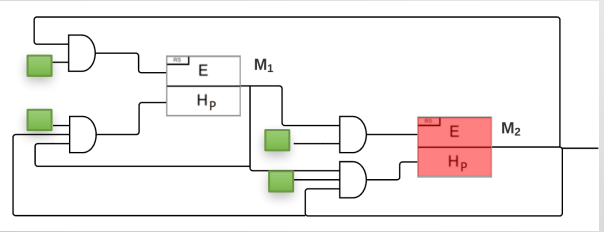
Permanent state automaton (PSA)



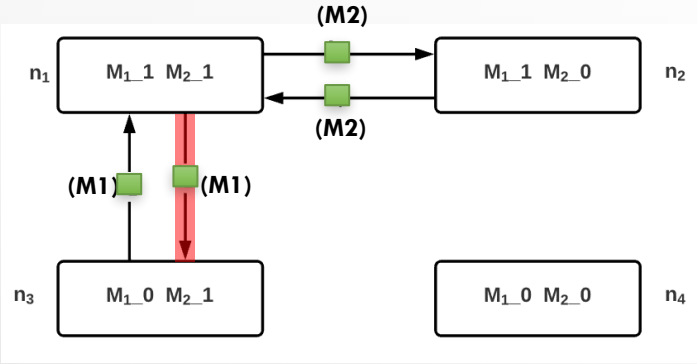
SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



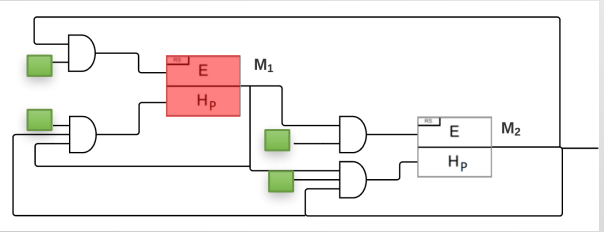
Permanent state automaton (PSA)



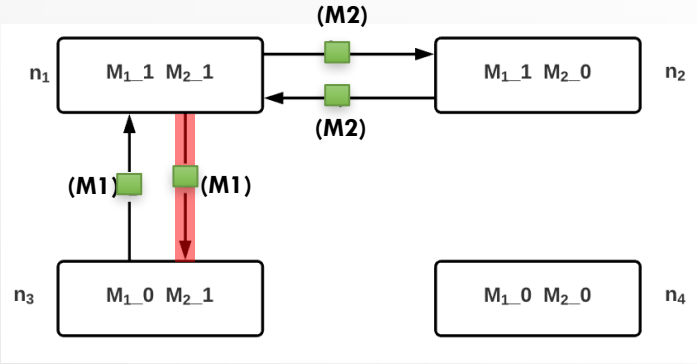
SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



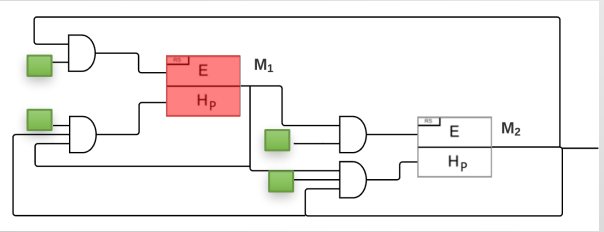
Permanent state automaton (PSA)



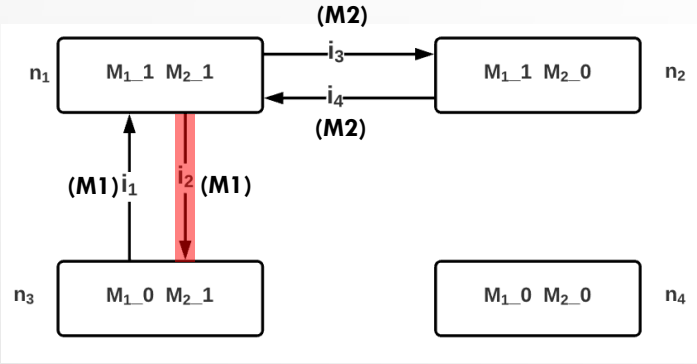
SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



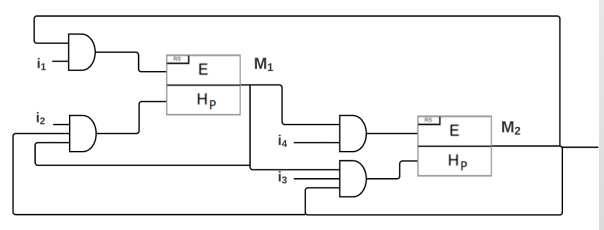
Permanent state automaton (PSA)



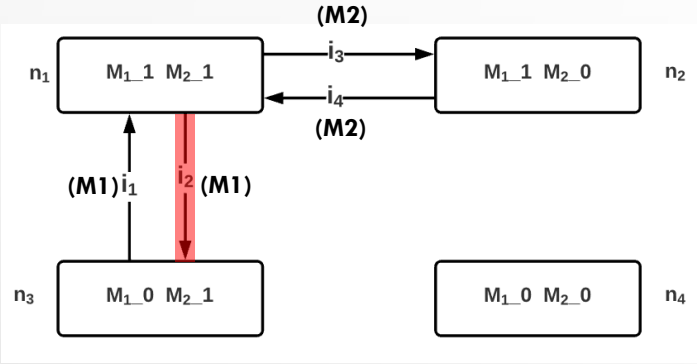
SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



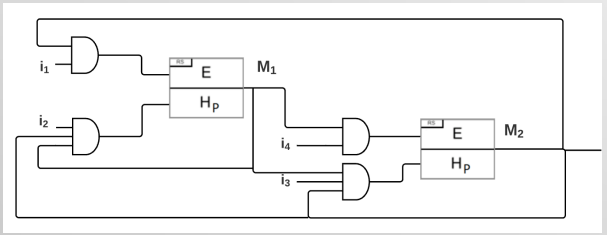
Permanent state automaton (PSA)



SGST

$\omega = \{ M2 , M1 \}$

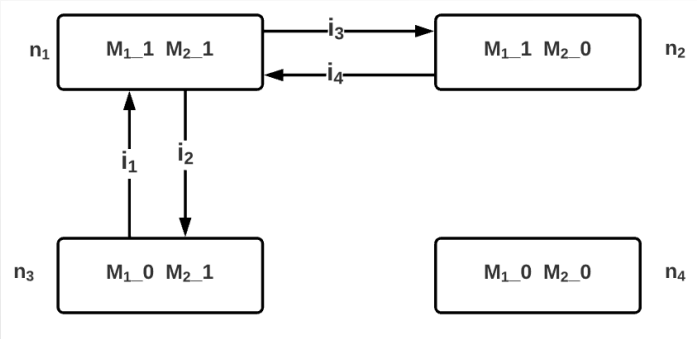
An arc represents an elementary evaluation



Trail2 : n1 – n3

Condition2 : i2.~i3.~i1

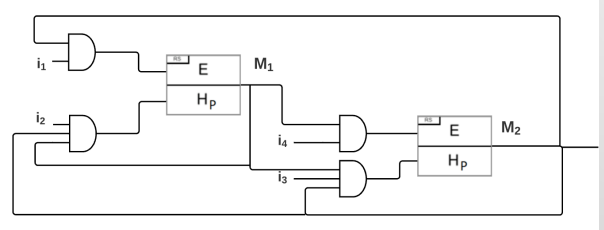
Permanent state automaton (PSA)



SGST

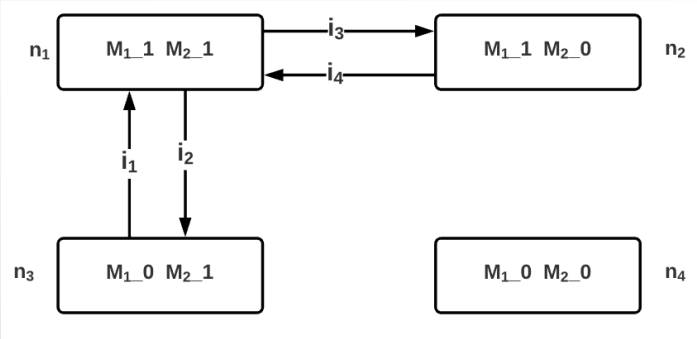
$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Two possible trails from n1 to n3:

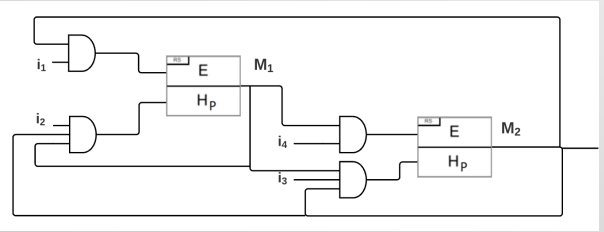
Permanent state automaton (PSA)



SGST

$\omega = \{ M2 , M1 \}$

An arc represents an elementary evaluation



Two possible trails from n1 to n3:

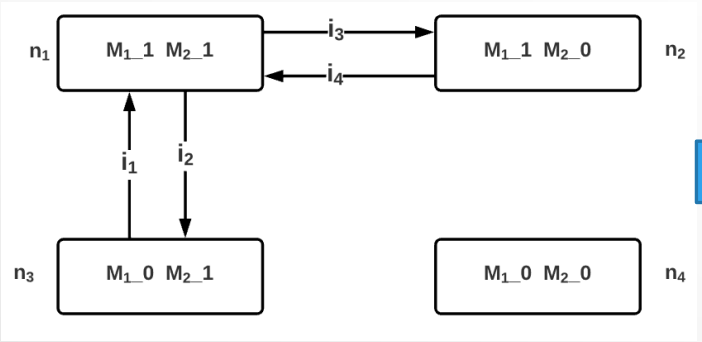
Trail1 : n1 – n2 – n1 – n3

Condition1 : i3.i4.i2.~i1

Trail2 : n1 – n3

Condition2 : i2.~i3.~i1

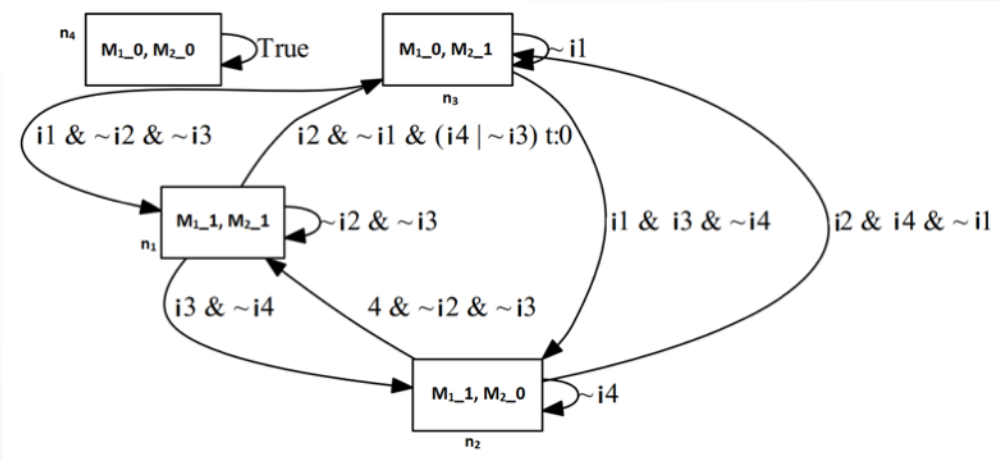
Permanent state automaton (PSA)



SGST

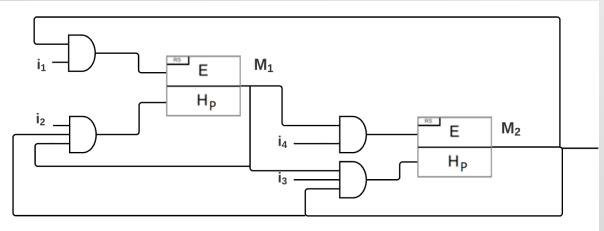
Trails calculation

$\omega = \{ M2 , M1 \}$



PSA

An arc represents an elementary evaluation



Two possible trails from n1 to n3:

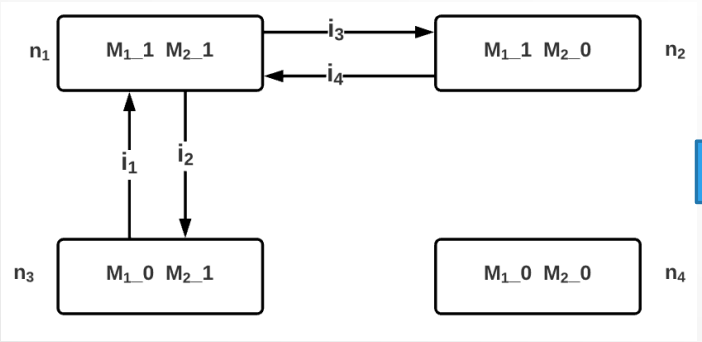
Trail1 : n1 – n2 – n1 – n3

Condition1 : i3.i4.i2.~i1

Trail2 : n1 – n3

Condition2 : i2.~i3.~i1

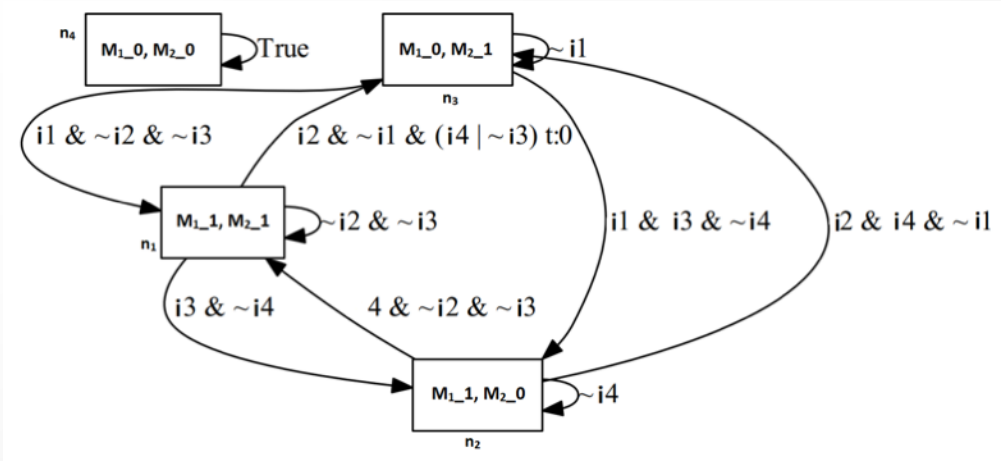
Permanent state automaton (PSA)



SGST

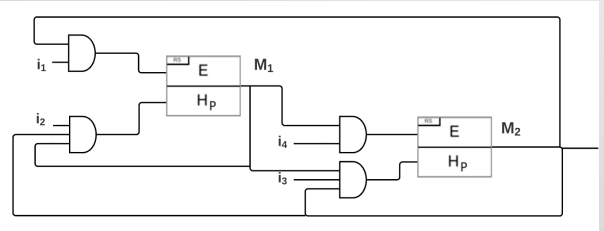
Trails calculation

$\omega = \{ M2 , M1 \}$



PSA

An arc represents an elementary evaluation



Two possible trails from n1 to n3:

Trail1 : n1 – n2 – n1 – n3

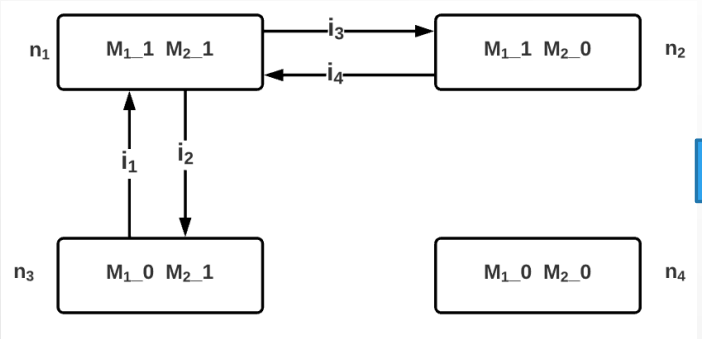
Condition1 : i3.i4.i2.~i1

Trail2 : n1 – n3

Condition2 : i2.~i3.~i1

An arc (nj, nk) represents all the full simulations of the diagram from nj to n_k

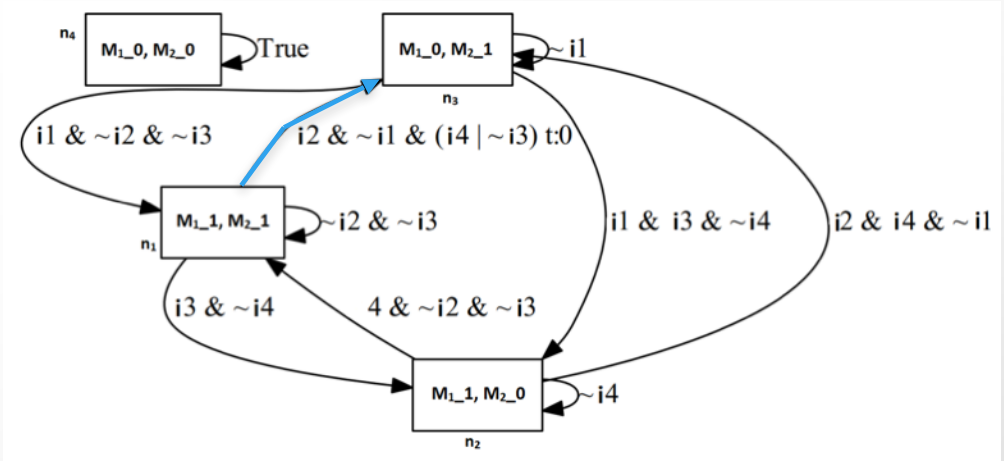
Permanent state automaton (PSA)



SGST

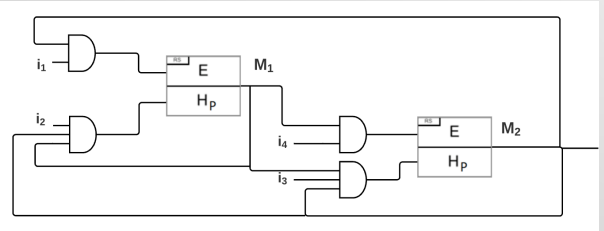
Trails calculation

$\omega = \{ M2 , M1 \}$



PSA

An arc represents an elementary evaluation



Two possible trails from n1 to n3:

Trail1 : n1 – n2 – n1 – n3

Condition1 : i3.i4.i2.~i1

Trail2 : n1 – n3

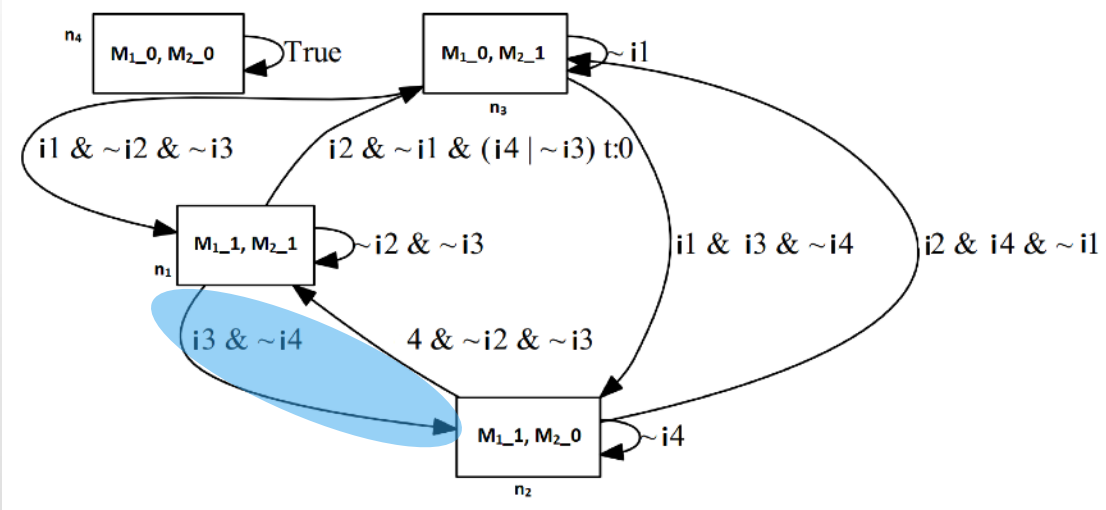
Condition2 : i2.~i3.~i1

An arc (nj, nk) represents all the full simulations of the diagram from nj to n_k

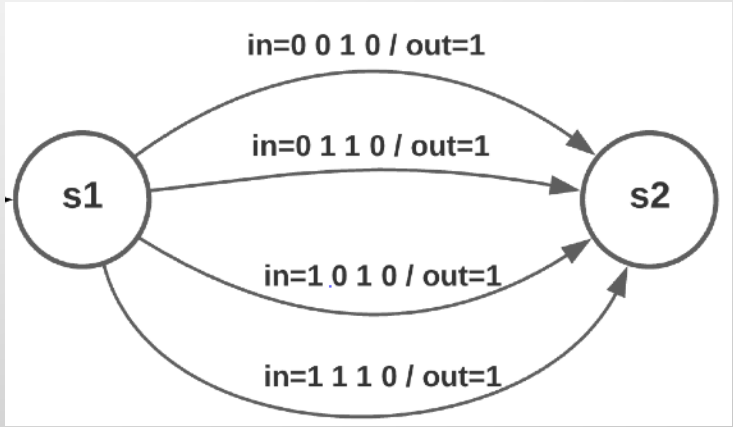
Transition : n1 – n3

Condition: Condition1 + Condition 2
=i2 & ~i1 & (i4 | ~i3)

Equivalent Mealy Machine



$in = i_1\ i_2\ i_3\ i_4$



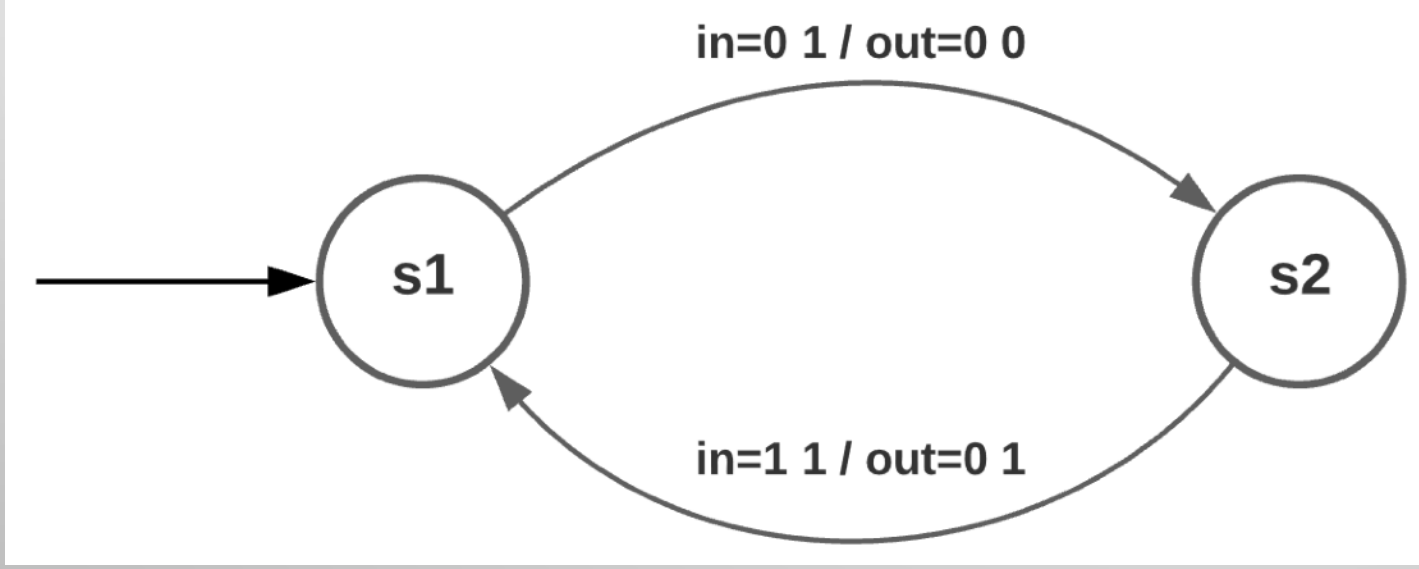
KB Representation

KB Representation

➔ **GOAL : AUTOMATICALLY GENERATE TEST SCENARIOS THE WAY EXPERTS ARE DOING MANUALLY**

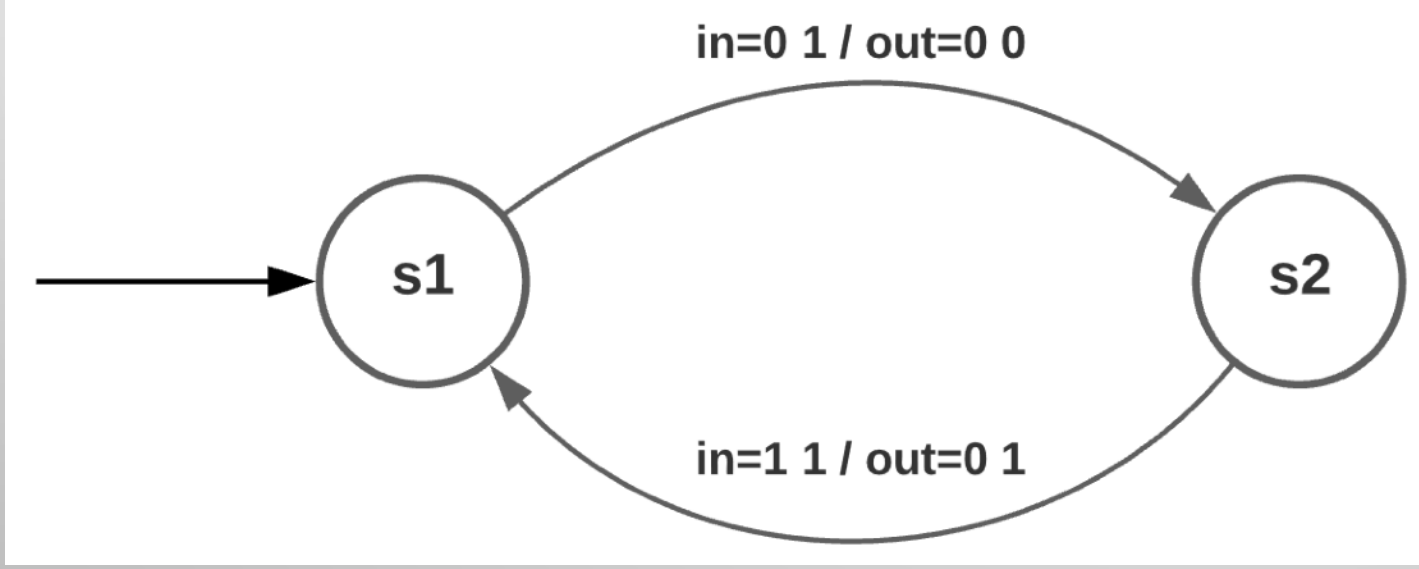
KB Representation

➔ GOAL : AUTOMATICALLY GENERATE TEST SCENARIOS THE WAY EXPERTS ARE DOING MANUALLY



KB Representation

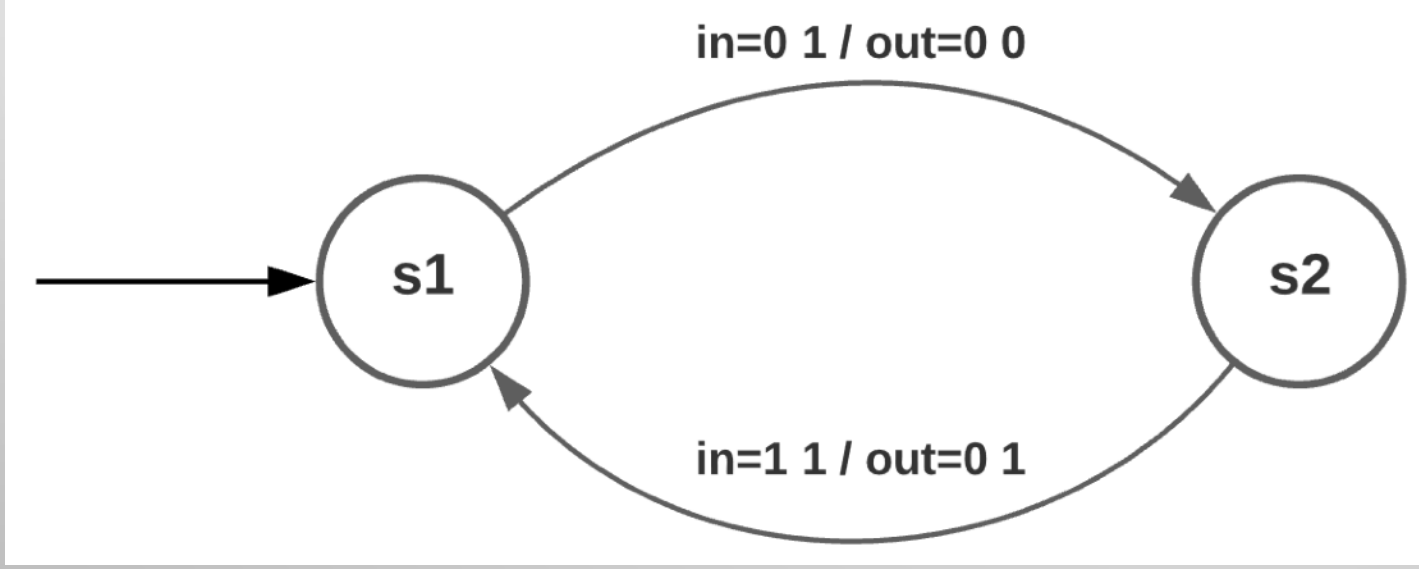
➔ GOAL : AUTOMATICALLY GENERATE TEST SCENARIOS THE WAY EXPERTS ARE DOING MANUALLY



- States: s1 and s2
- Inputs: i1 and i2; in=i1 i2
- Outputs: o1 and o2; out=o1 o2

KB Representation

➔ **GOAL : AUTOMATICALLY GENERATE TEST SCENARIOS THE WAY EXPERTS ARE DOING MANUALLY**

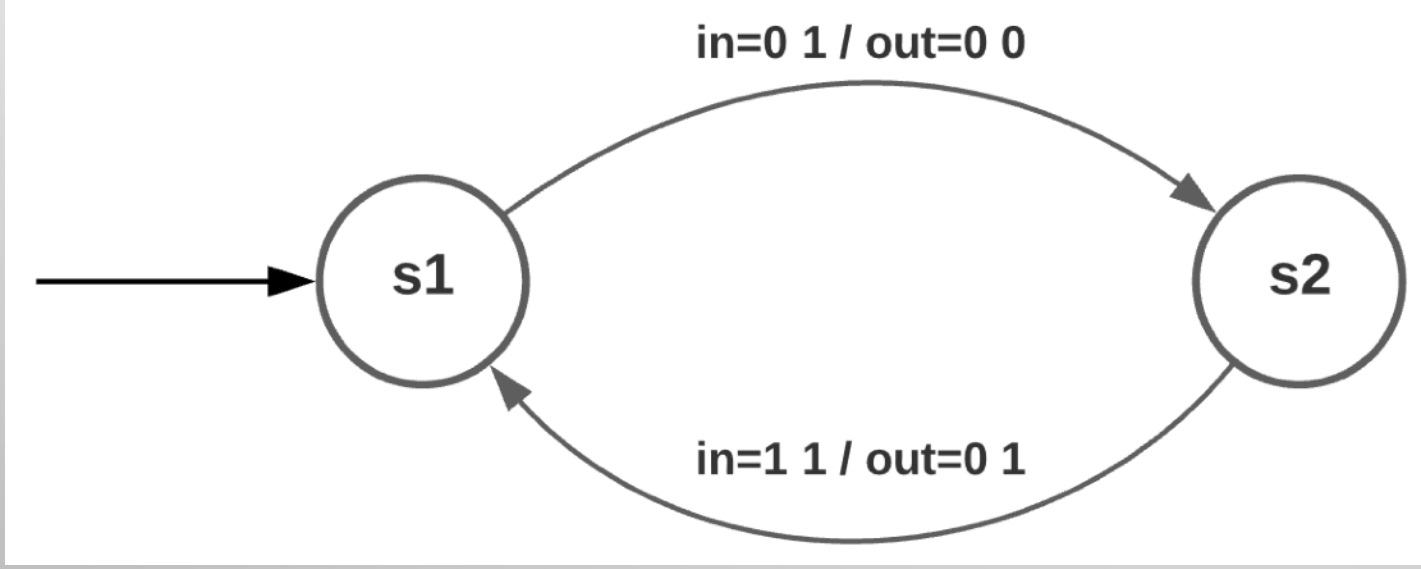


A Scenario is a sequence of input/output values. That is a succession of transitions in the Mealy Machine.

For example : <01/00> <11/01>< 01/ 00>

KB Representation

➔ **GOAL : AUTOMATICALLY GENERATE TEST SCENARIOS THE WAY EXPERTS ARE DOING MANUALLY**



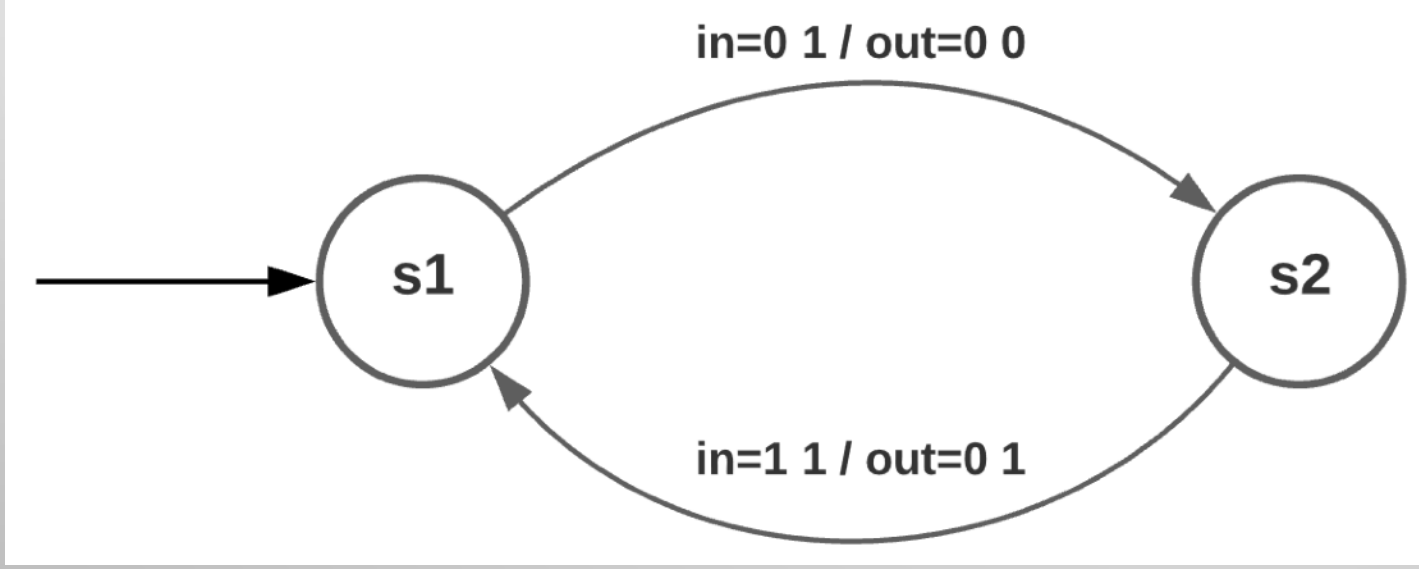
Let's say that :

- **i1 is an emergency stop hold button.**
- **o1 is a water pump**

We want to find a test sequence that verifies that the pump is always deactivated (o1=0) when the emergency button is on (i1=1)

KB Representation

➔ **GOAL : AUTOMATICALLY GENERATE TEST SCENARIOS THE WAY EXPERTS ARE DOING MANUALLY**



Let's say that :

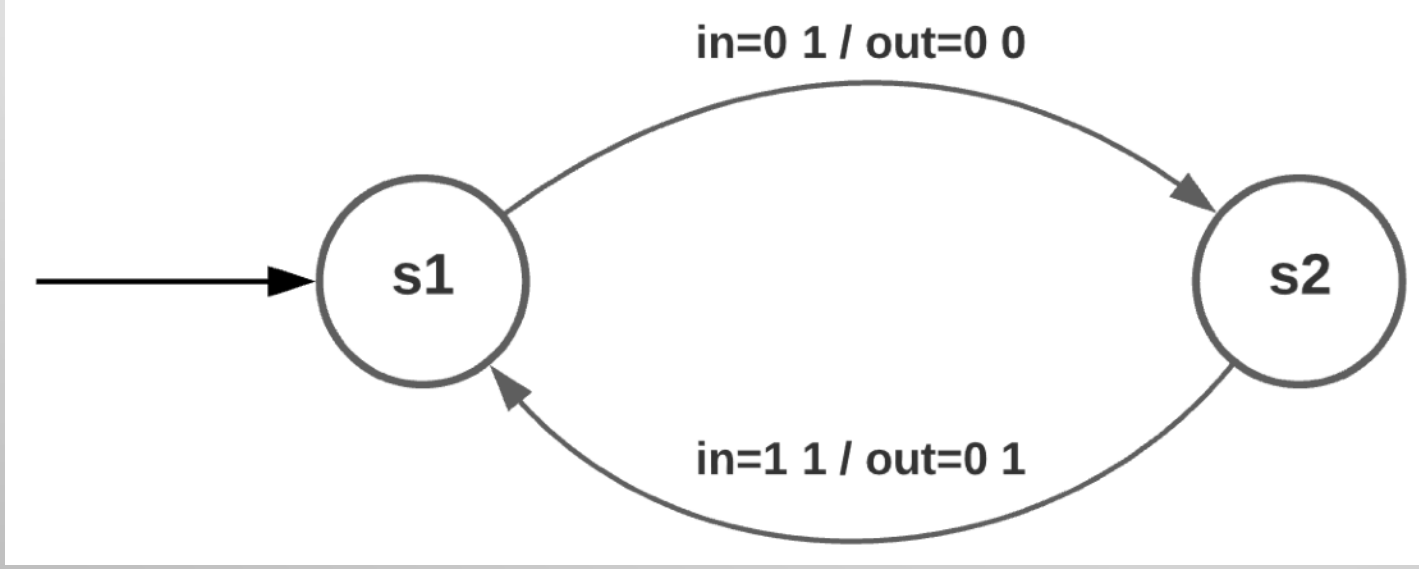
- **i1 is an emergency stop hold button.**
- **o1 is a water pump**

We want to find a test sequence that verifies that the pump is always deactivated (o1=0) when the emergency button is on (i1=1)

➔ <01/00> <11/01>

KB Representation

➔ **GOAL : AUTOMATICALLY GENERATE TEST SCENARIOS THE WAY EXPERTS ARE DOING MANUALLY**



Let's say that :

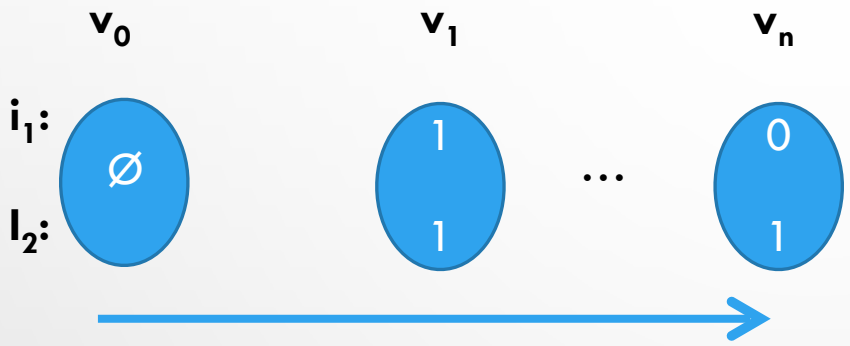
- **i1 is an emergency stop hold button.**
- **o1 is a water pump**

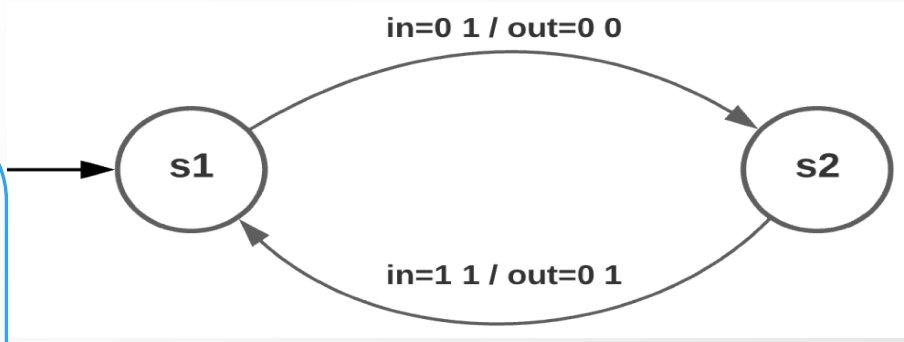
We want to find a test sequence that verifies that the pump is always deactivated (o1=0) when the emergency button is on (i1=1)

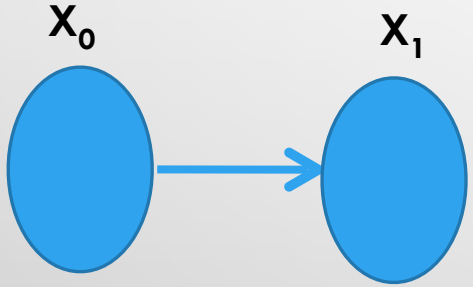
➔ <01/00> <11/01>

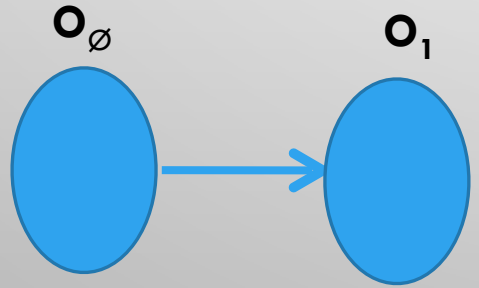
KB Representation

Fact Base:





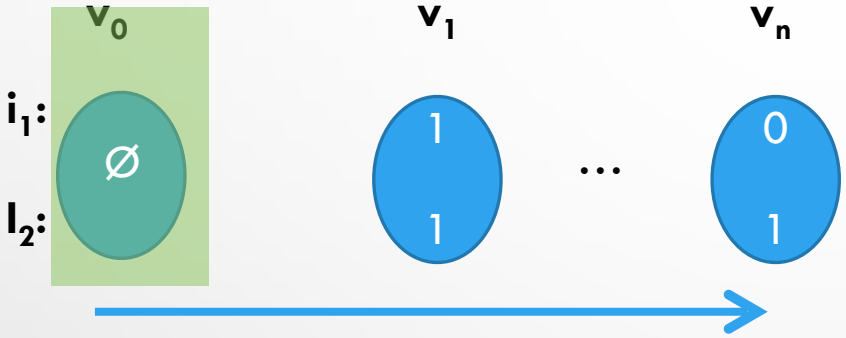




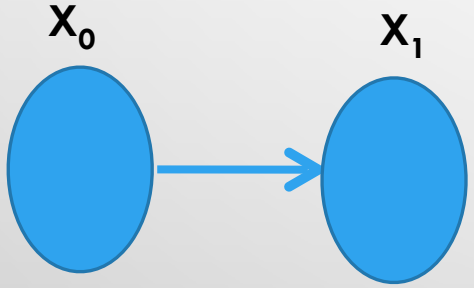
KB Representation

Fact Base:

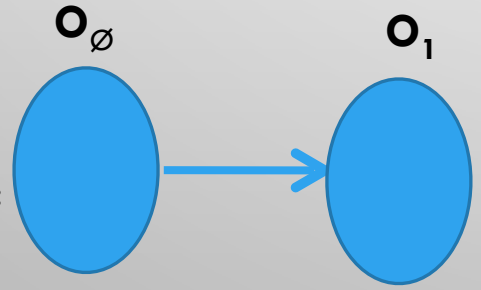
Input sequence:

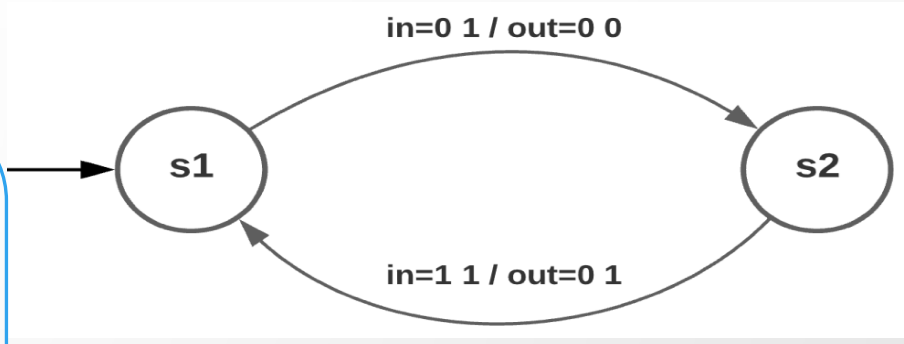


Visited states:



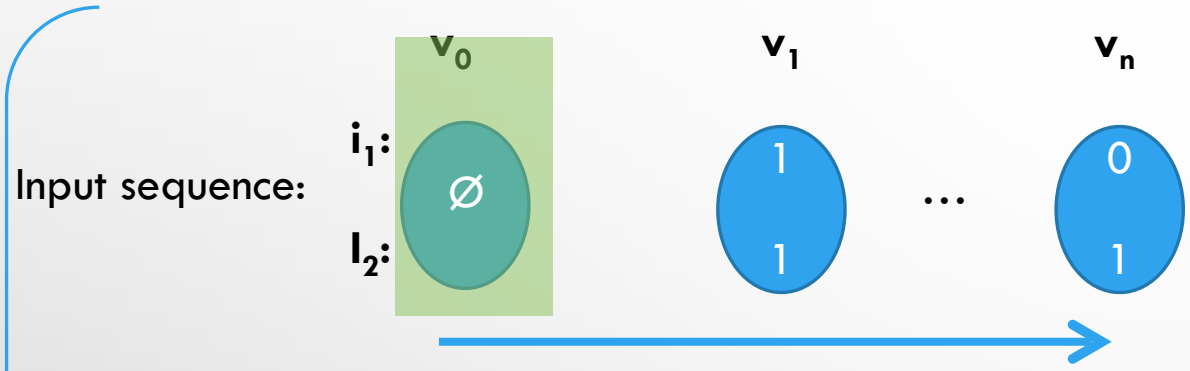
Generated output sequence:

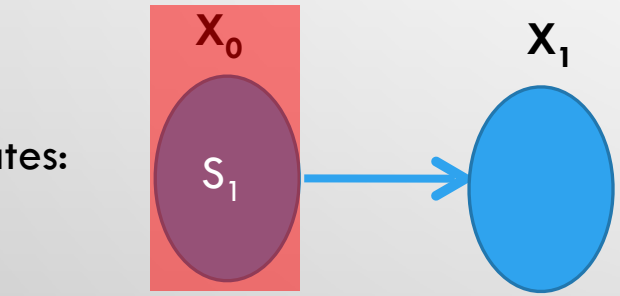


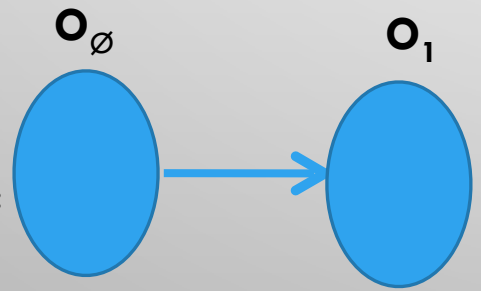


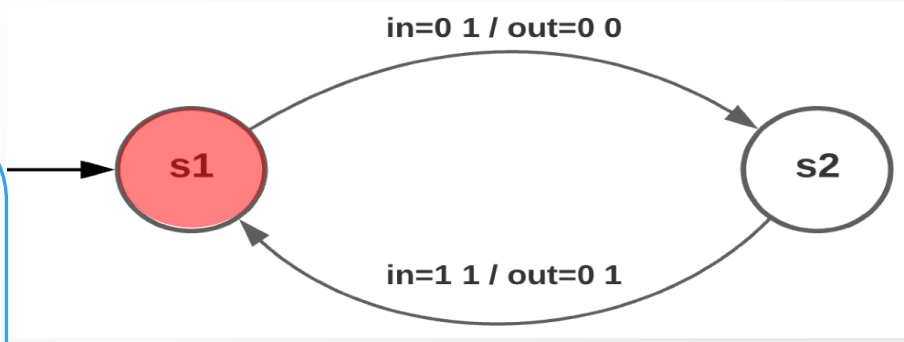
KB Representation

Fact Base:





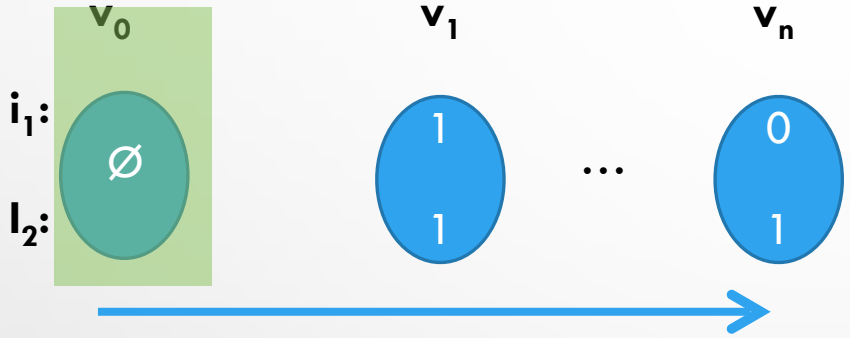




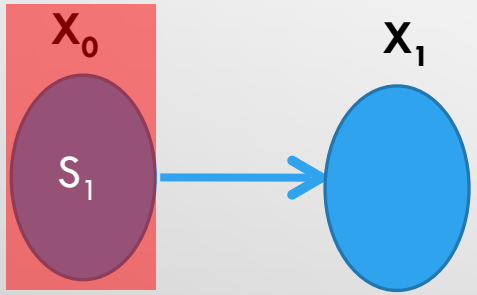
KB Representation

Fact Base:

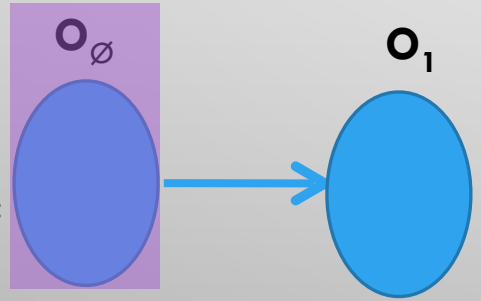
Input sequence:

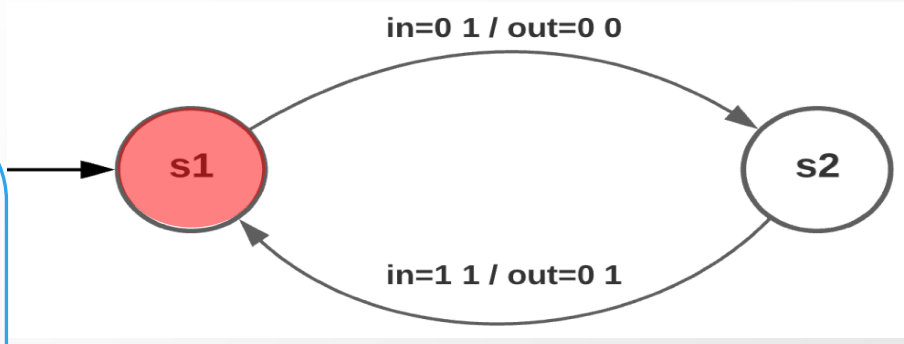


Visited states:



Generated output sequence:

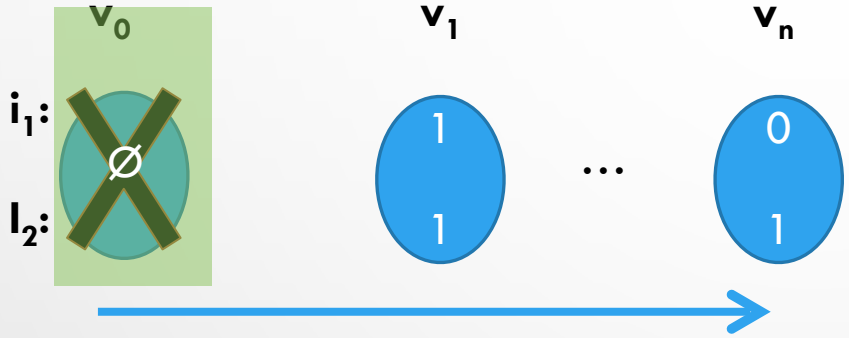




KB Representation

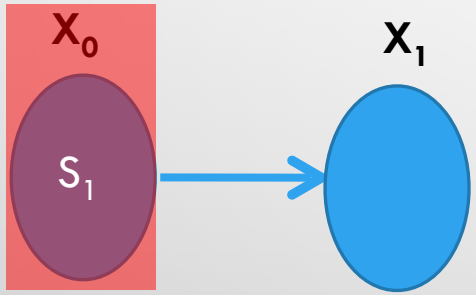
Fact Base:

Input sequence:

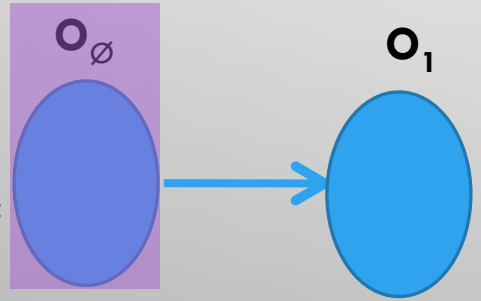


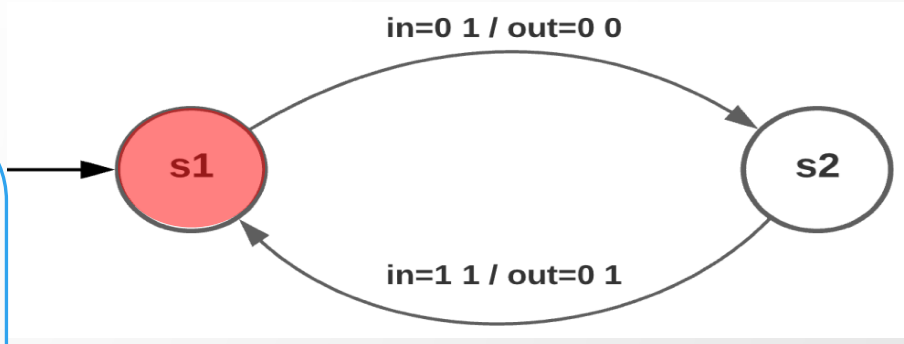
Timeline

Visited states:



Generated output sequence:

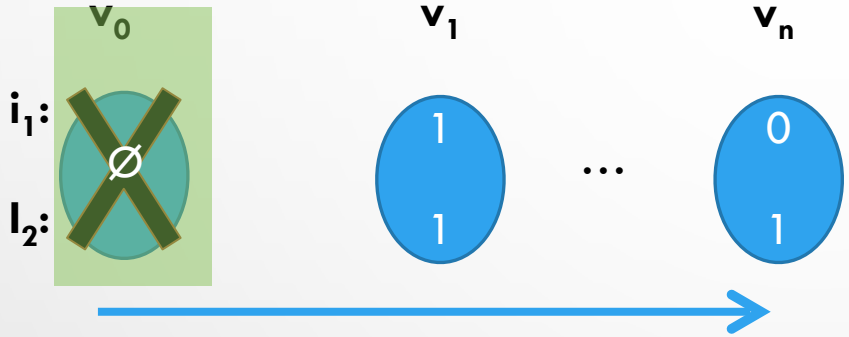




KB Representation

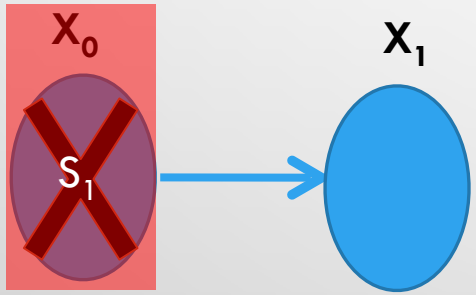
Fact Base:

Input sequence:

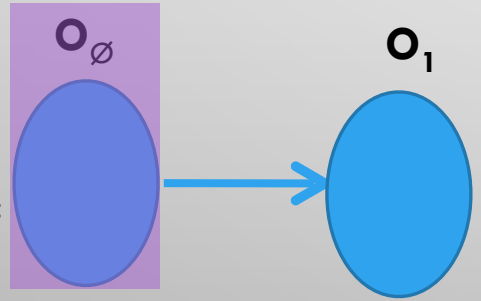


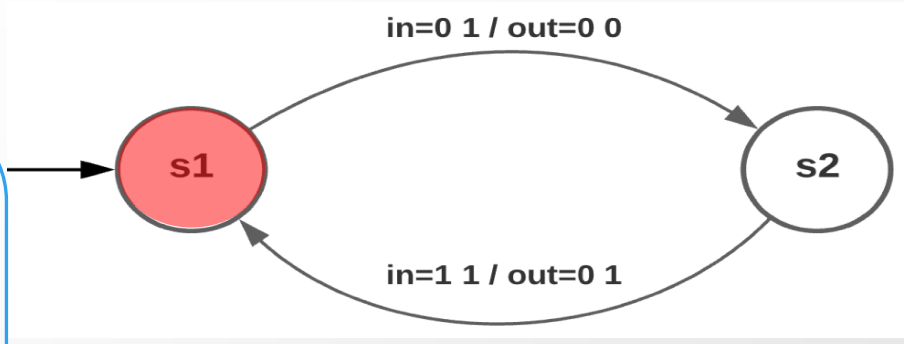
Timeline

Visited states:



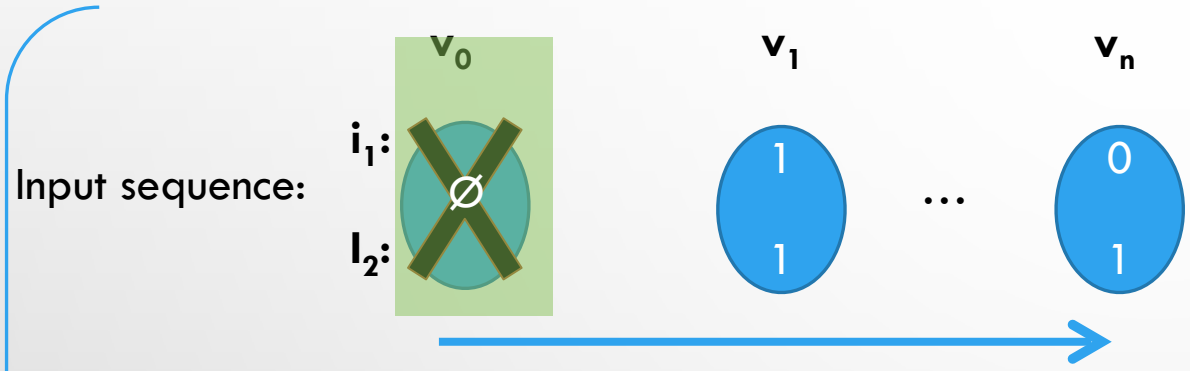
Generated output sequence:





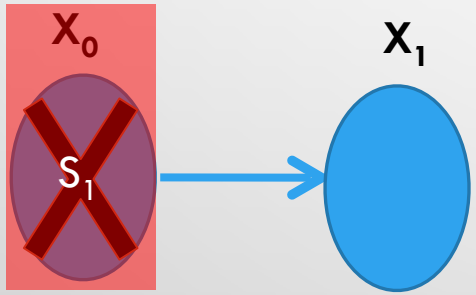
KB Representation

Fact Base:

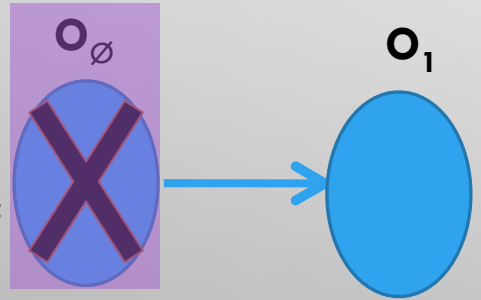


Input sequence:

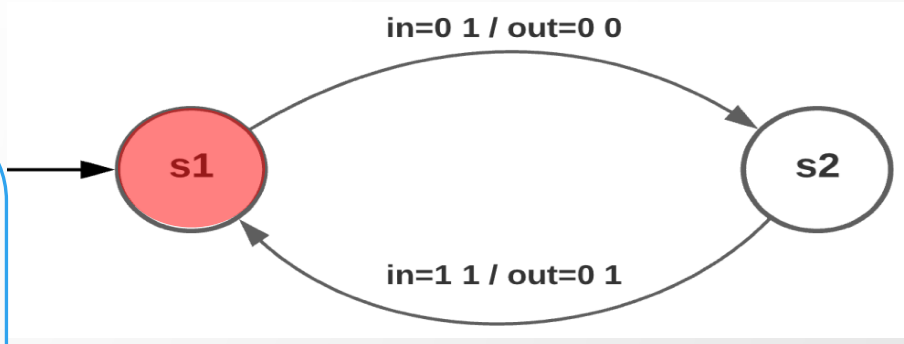
Timeline



Visited states:

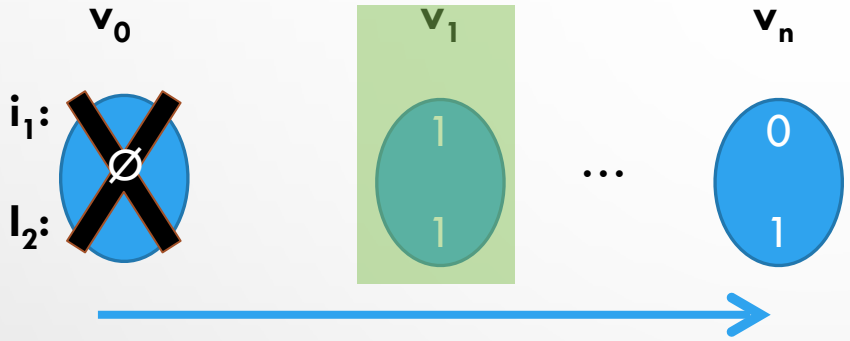


Generated output sequence:



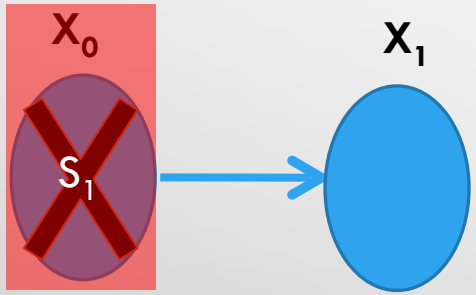
KB Representation

Fact Base:

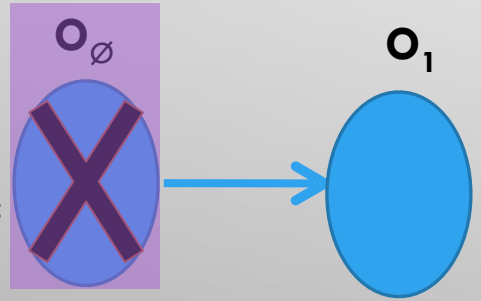


Input sequence:

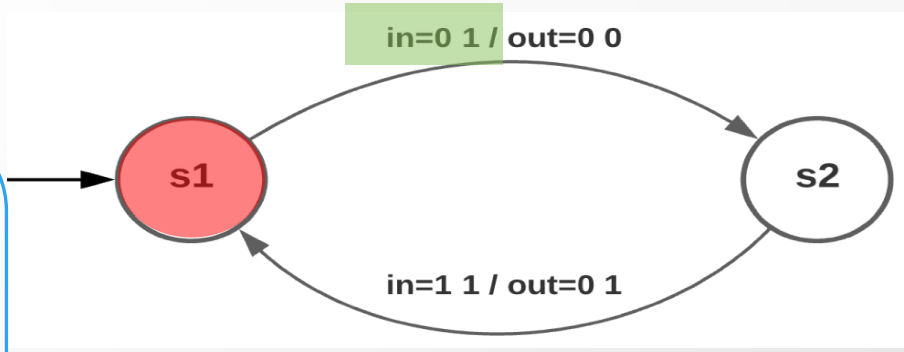
Timeline



Visited states:

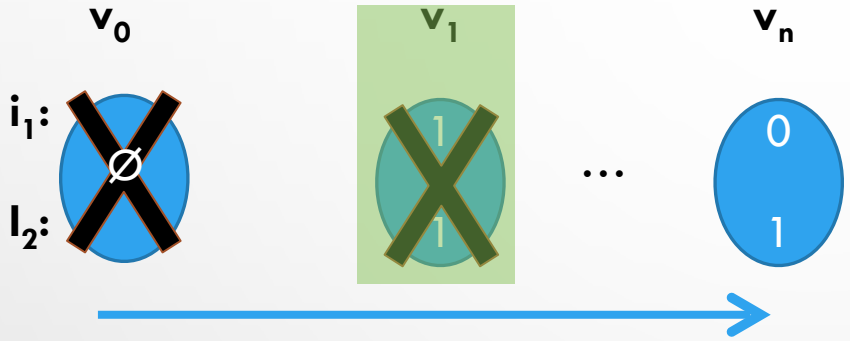


Generated output sequence:



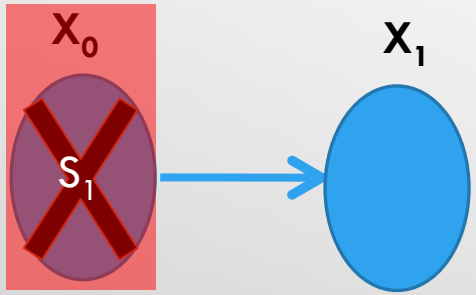
KB Representation

Fact Base:

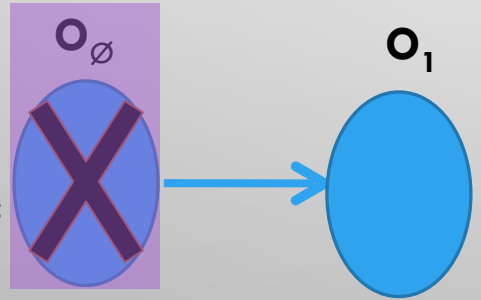


Input sequence:

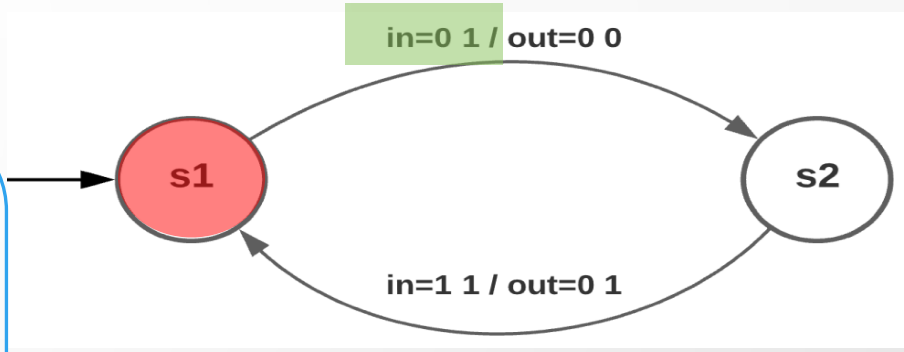
Timeline



Visited states:

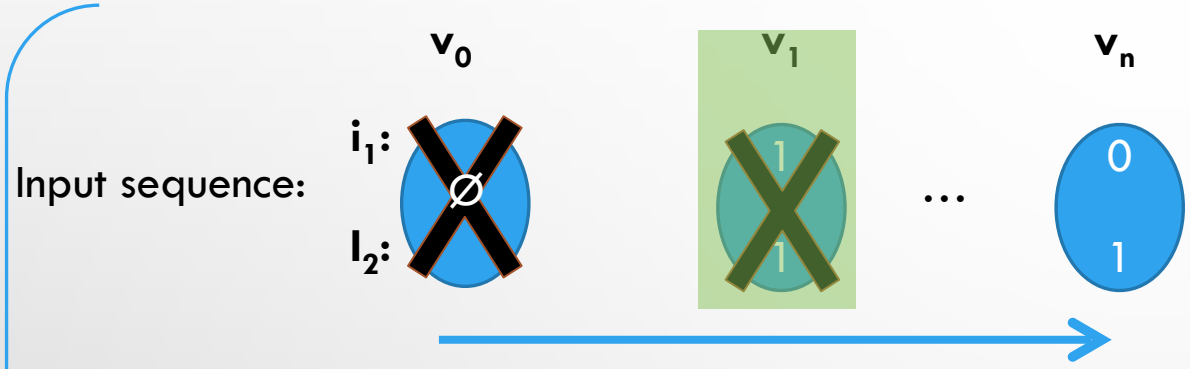


Generated output sequence:

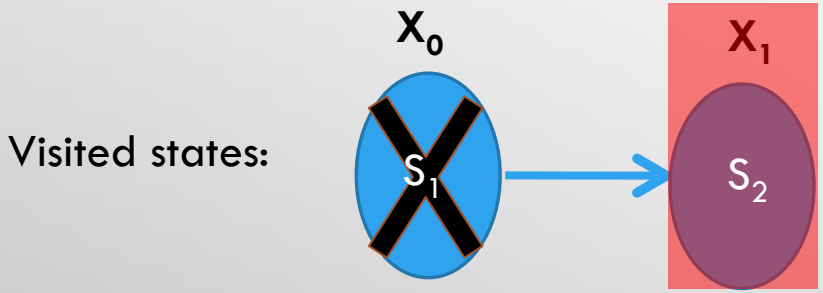


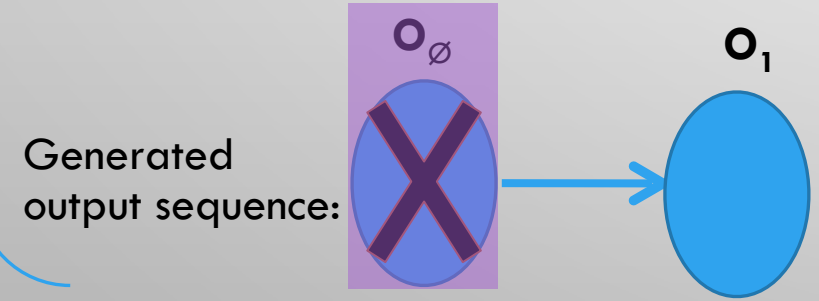
KB Representation

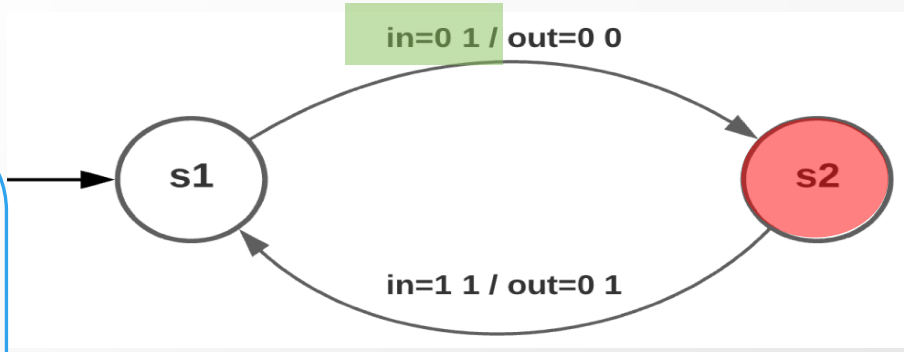
Fact Base:



Timeline

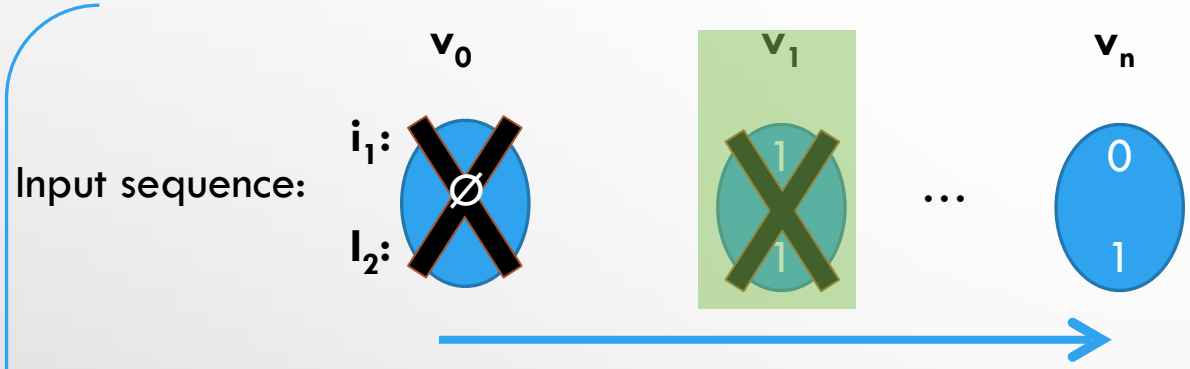






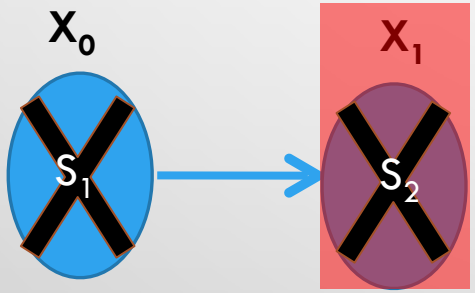
KB Representation

Fact Base:

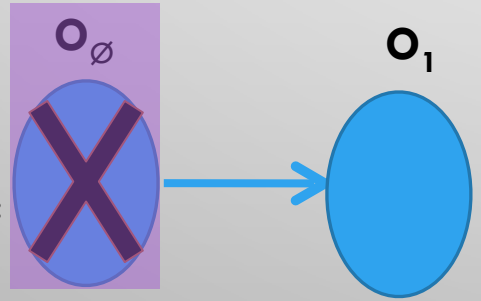


Input sequence:

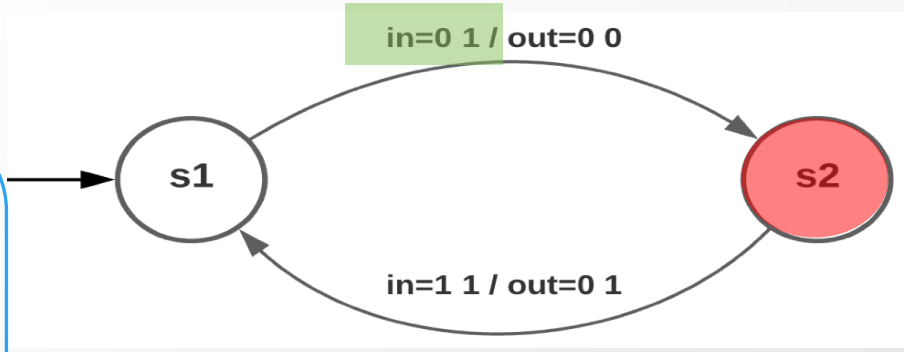
Timeline



Visited states:

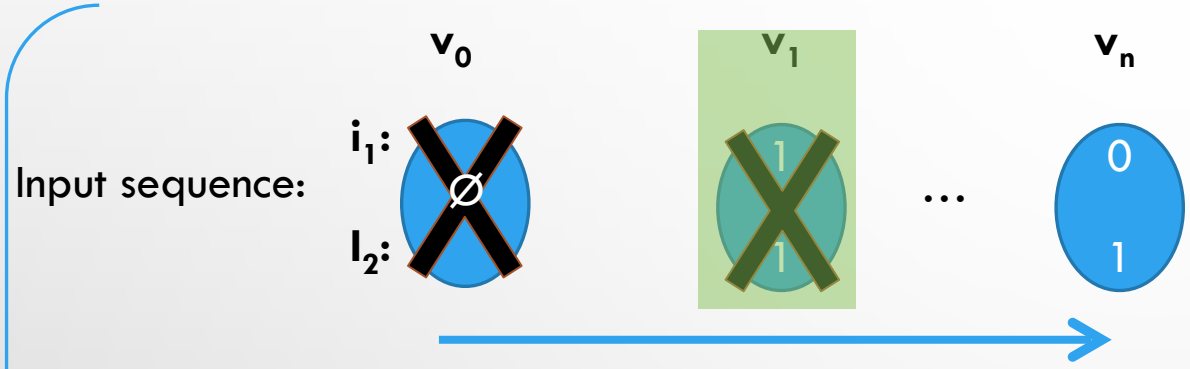


Generated output sequence:

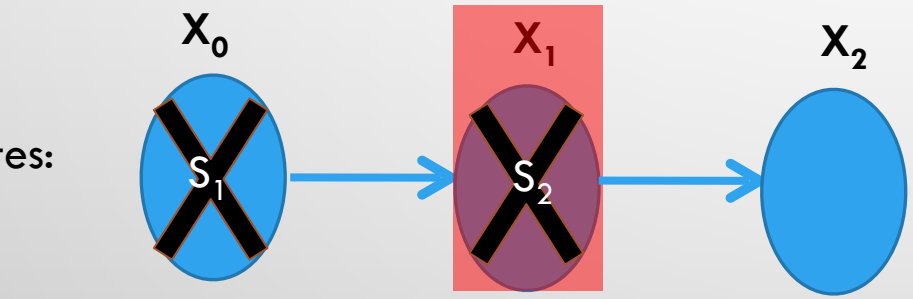


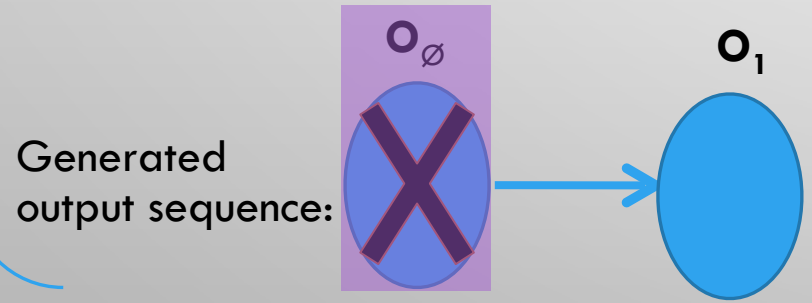
KB Representation

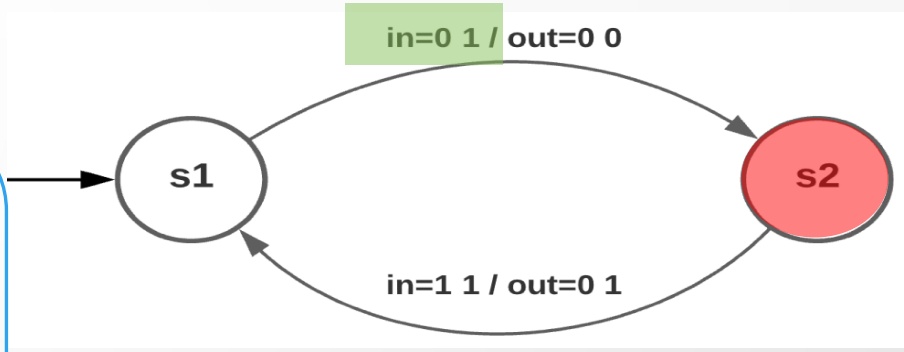
Fact Base:



Timeline

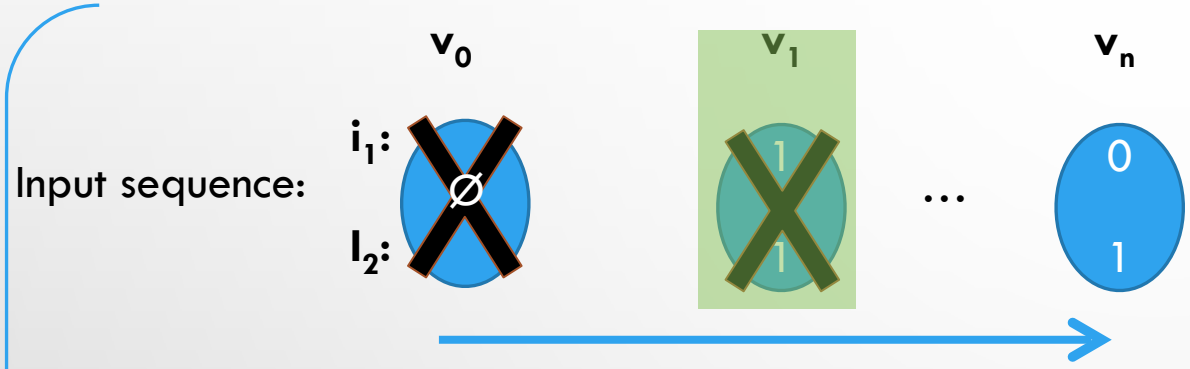






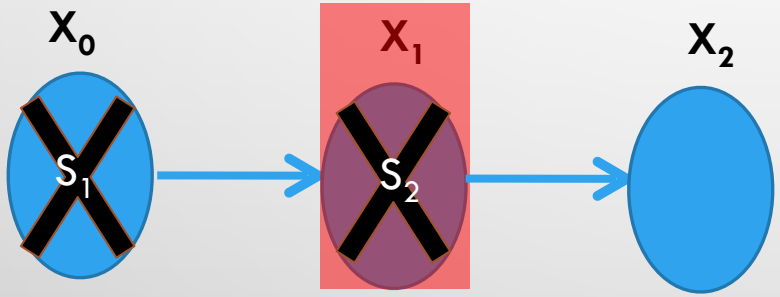
KB Representation

Fact Base:

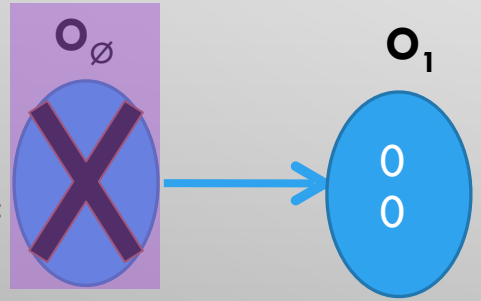


Input sequence:

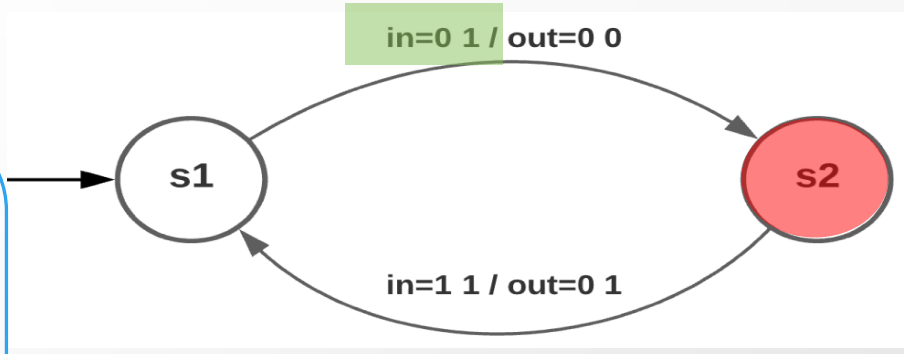
Timeline



Visited states:

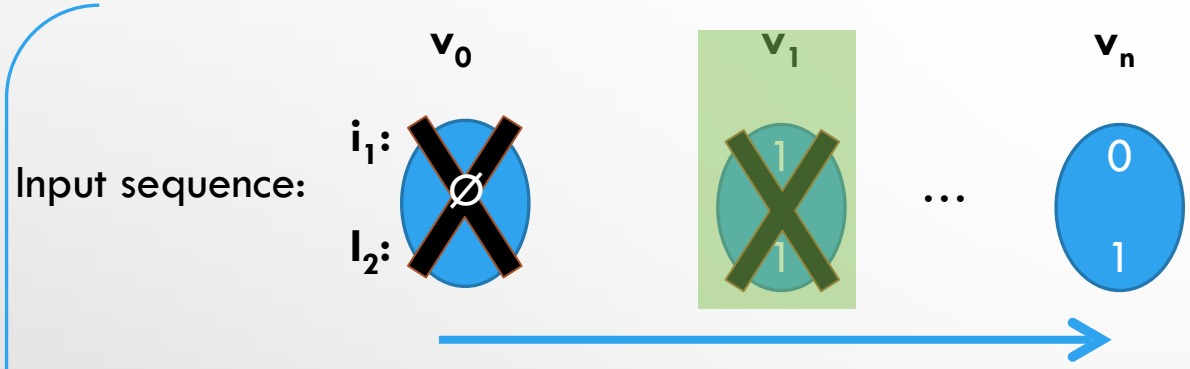


Generated output sequence:

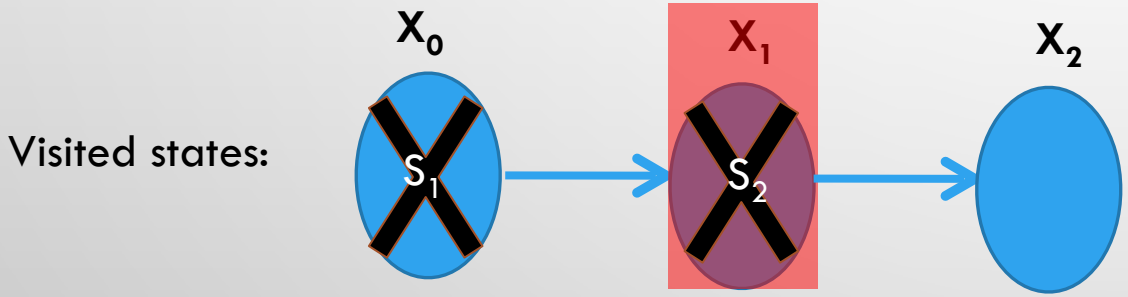


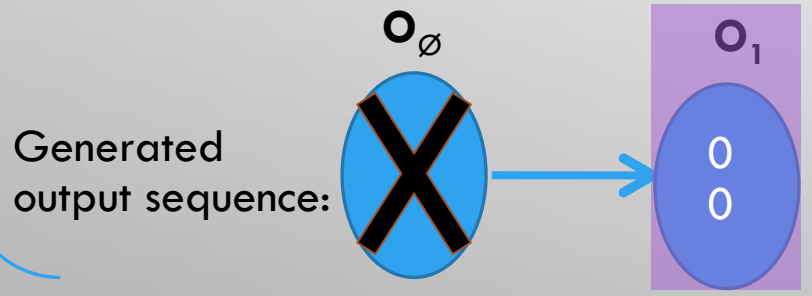
KB Representation

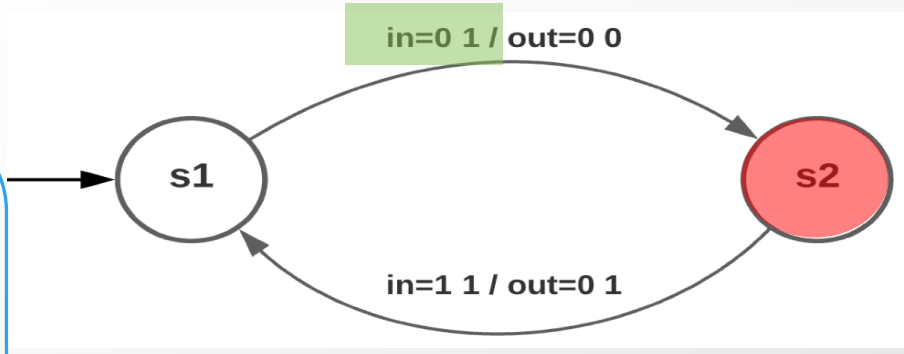
Fact Base:



Timeline

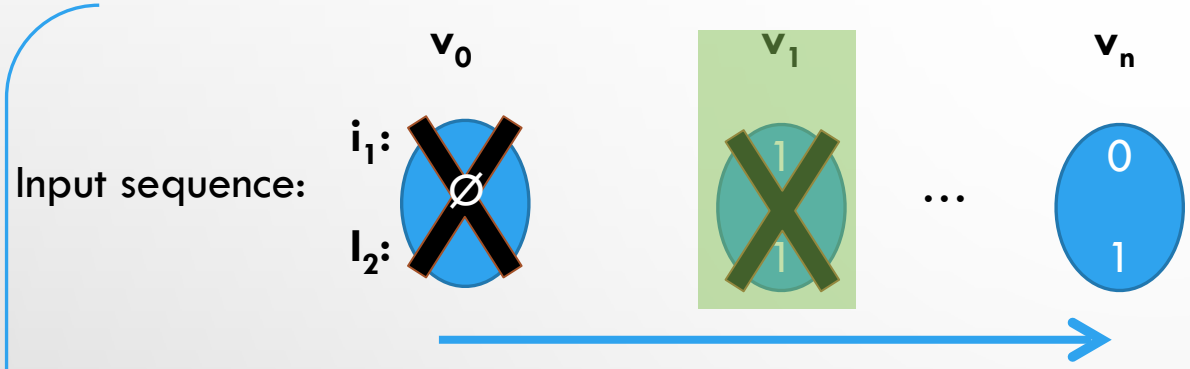




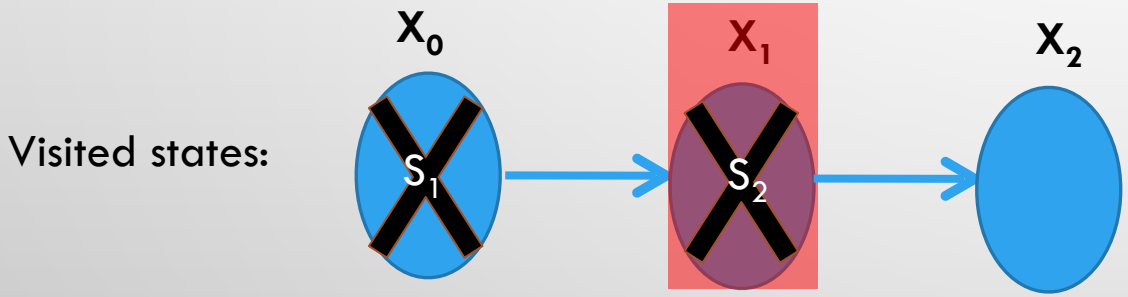


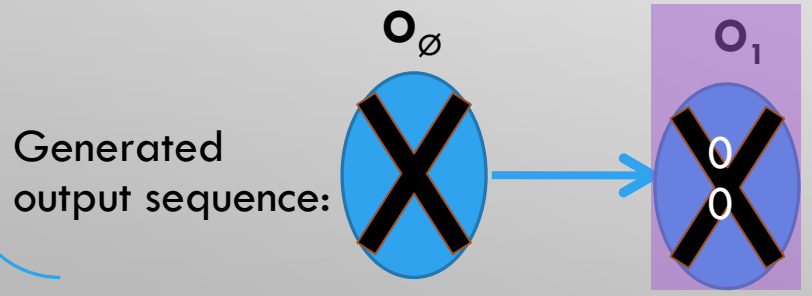
KB Representation

Fact Base:



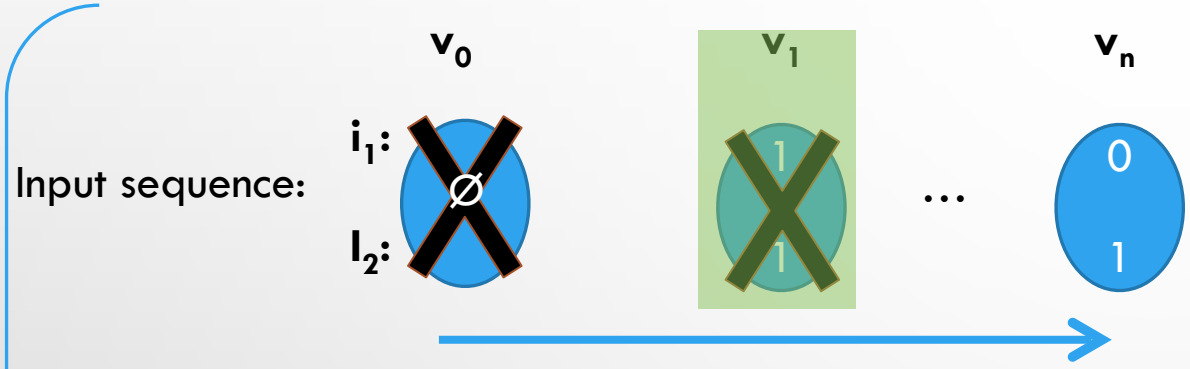
Timeline



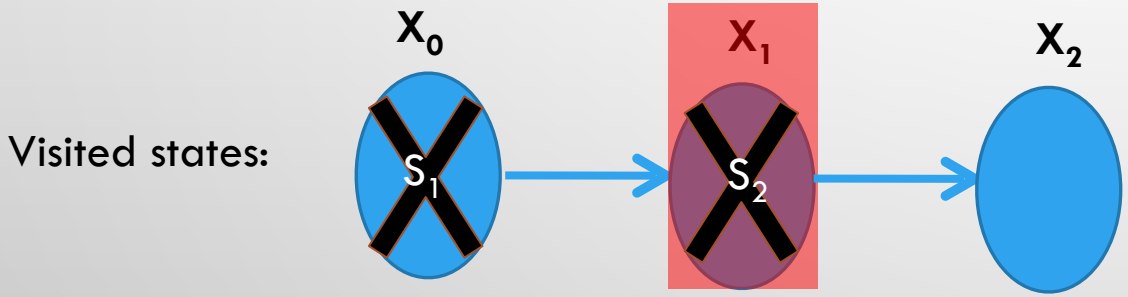


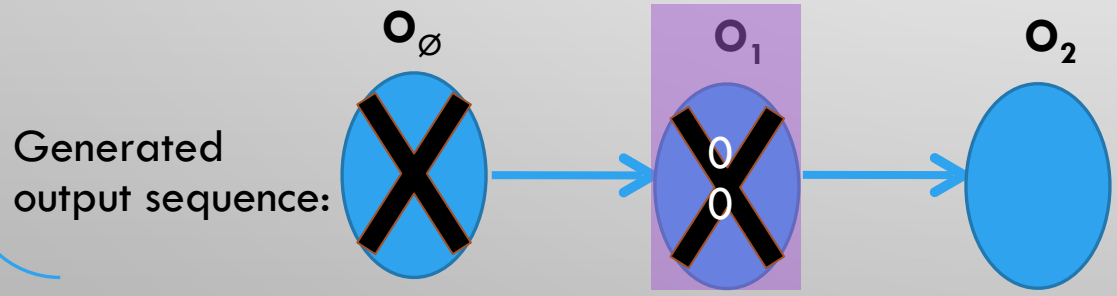
KB Representation

Fact Base:

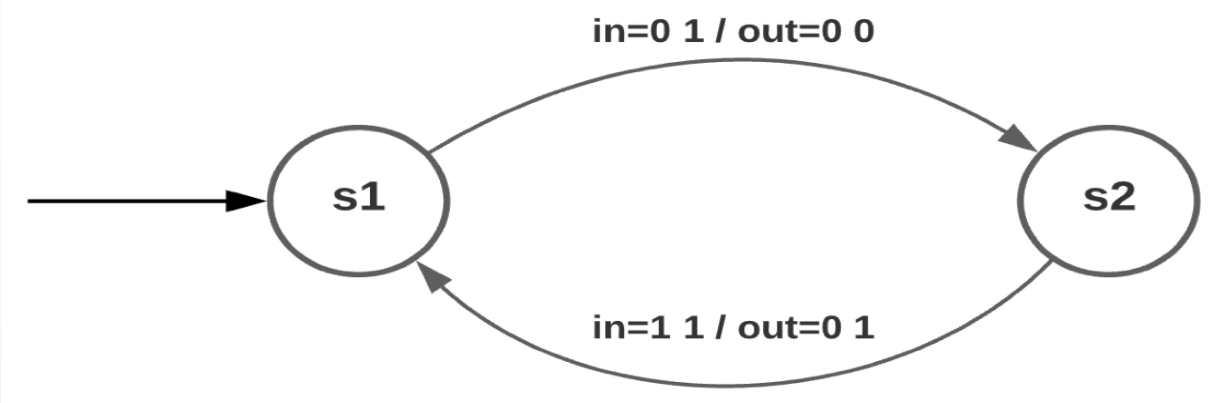


Timeline

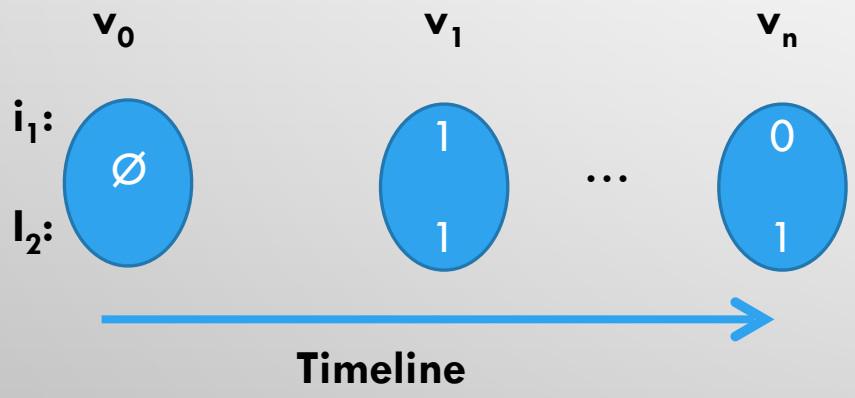




KB Representation



Fact Base: input sequence



input(v_0)

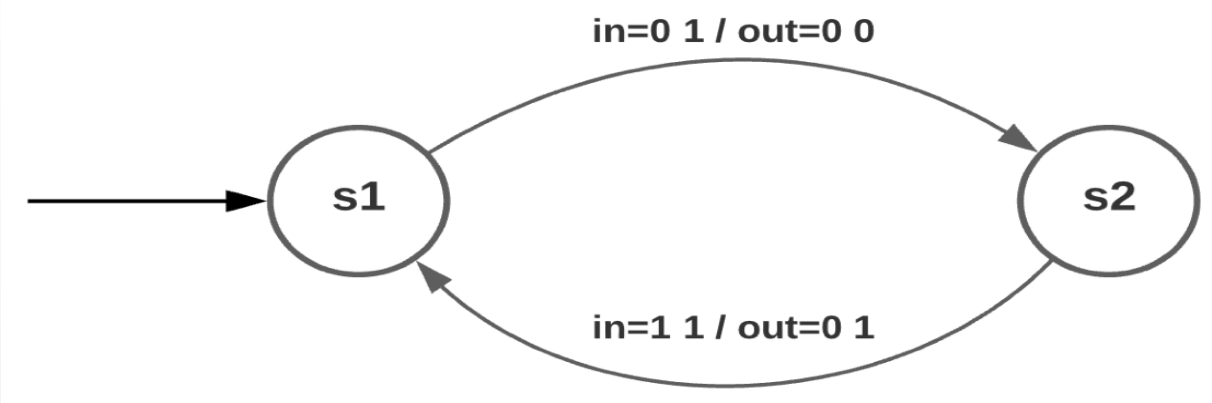
input(v_1)
…
input(v_n)

inputValue(v_1, '01')
…
inputValue(v_n, '01')

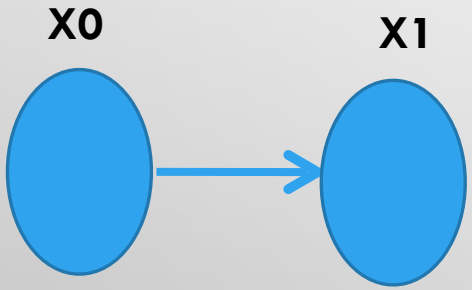
next(v_0, v_1)

next(v_1, v_2)
…
next(v_{n-1}, v_n)

KB Representation



Fact Base: initial state



state(X_0)

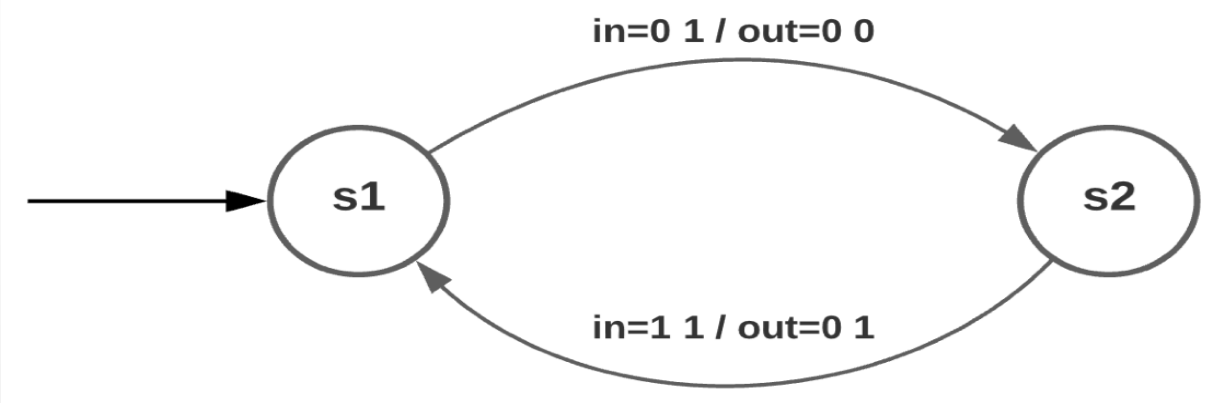
state(X_1)

next(X_0, X_1)

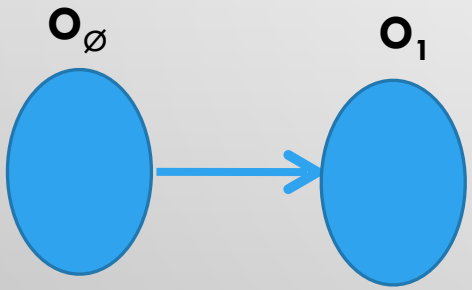
done(X_0)

value(X_0, 's_1')

KB Representation



Fact Base: initial output



output ($O_\emptyset$)

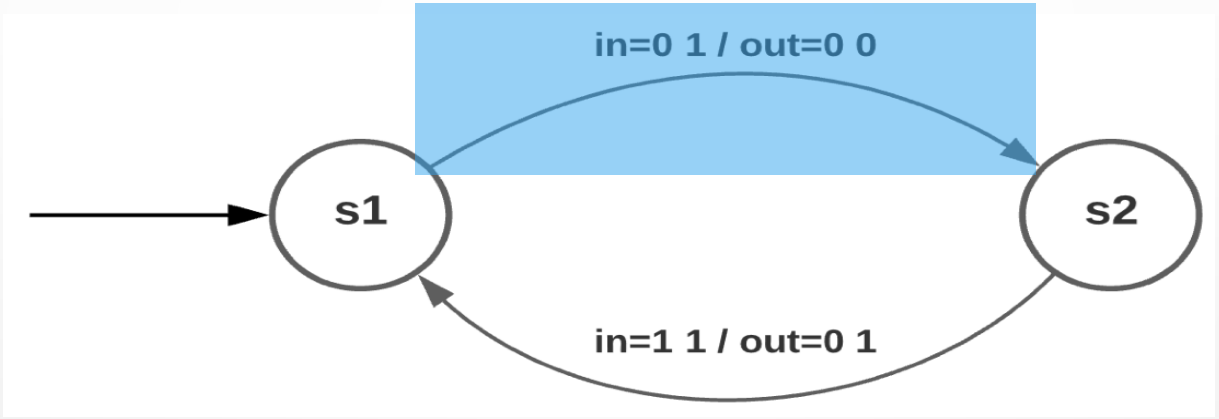
output(O_1)

next($O_\emptyset$, O_1)

done($O_\emptyset$)

value(O_0, 'NOT ACTIVE')

KB Representation



Rules: transitions

Next state in the chain:

state(X_{next}), next($X_{current}$, X_{next})

done($X_{current}$), not done(X_{next})

Find next output in the chain:

output(O_{next}), next($O_{current}$, O_{next})

done($O_{current}$), not done(Onext)

Find next input vector in the chain:

input(v_{next}), next($v_{current}$, v_{next})

done($v_{current}$), not done(v_{next})

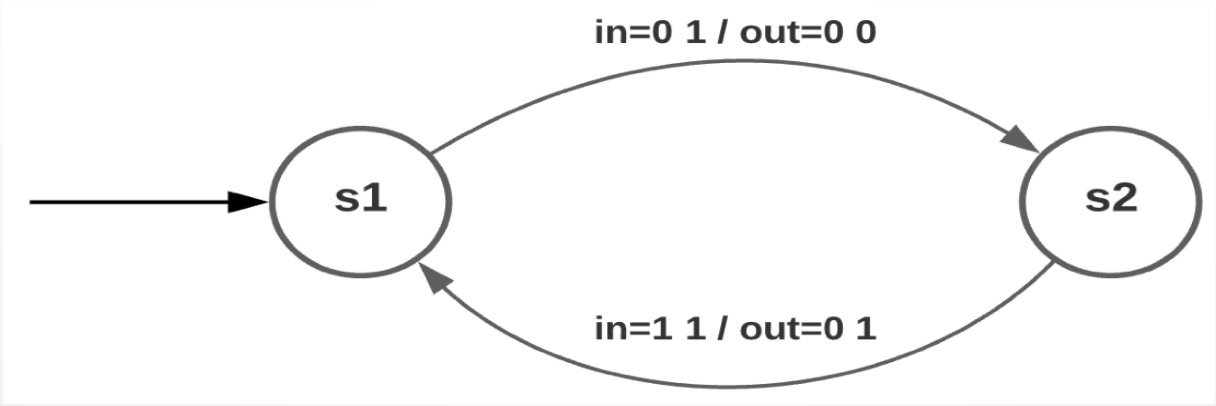
Transition (s_1 01 00 s_2):

Value($X_{current}$, 's_1 ')

Value(v_{next}, '01')

Transition (s1 01 00 s2)

KB Representation



Rules: transitions

Next state in the chain:

state(X_{next}), next($X_{current}$, X_{next})

done($X_{current}$), not done(X_{next})

Find next output in the chain:

output(O_{next}), next($O_{current}$, O_{next})

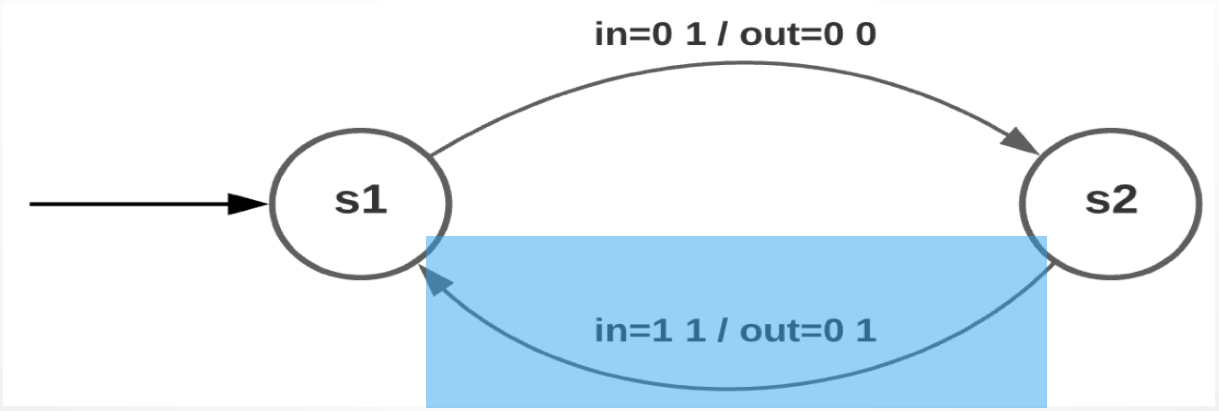
done($O_{current}$), not done(Onext)

Find next input vector in the chain:

input(v_{next}), next($v_{current}$, v_{next})

done($v_{current}$), not done(v_{next})

KB Representation



Rules: transitions

Next state in the chain:

state(X_{next}), next($X_{current}$, X_{next})

done($X_{current}$), not done(X_{next})

Find next output in the chain:

output(O_{next}), next($O_{current}$, O_{next})

done($O_{current}$), not done(Onext)

Find next input vector in the chain:

input(v_{next}), next($v_{current}$, v_{next})

done($v_{current}$), not done(v_{next})

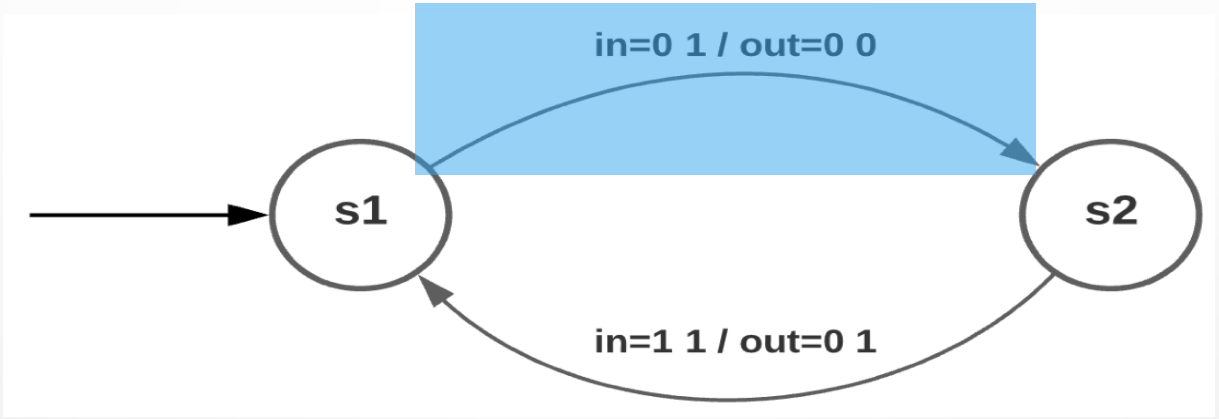
Transition (s_2 11 01 s_1):

Value($X_{current}$, 's_2 ')

Value(v_{next}, '11')

Transition (s_2 11 01 s_1)

KB Representation



Rules: transitions

Transition (s_1 01 00 s_2)

$value(X_{next}, `s_2')$

$value(O_{next}, `00')$

$done(X_{next})$

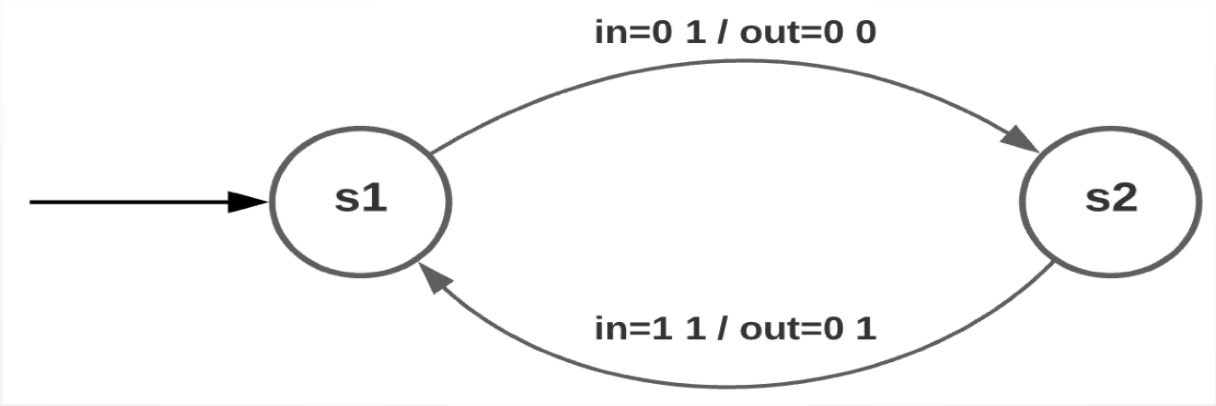
$done(O_{next})$

$done(v_{next})$

$\exists\ X_{following}\ next(X_{next}, X_{following})$

$\exists\ O_{following}\ next(O_{next}, O_{following})$

KB Representation



Rules: transitions

$done(X_{next})$

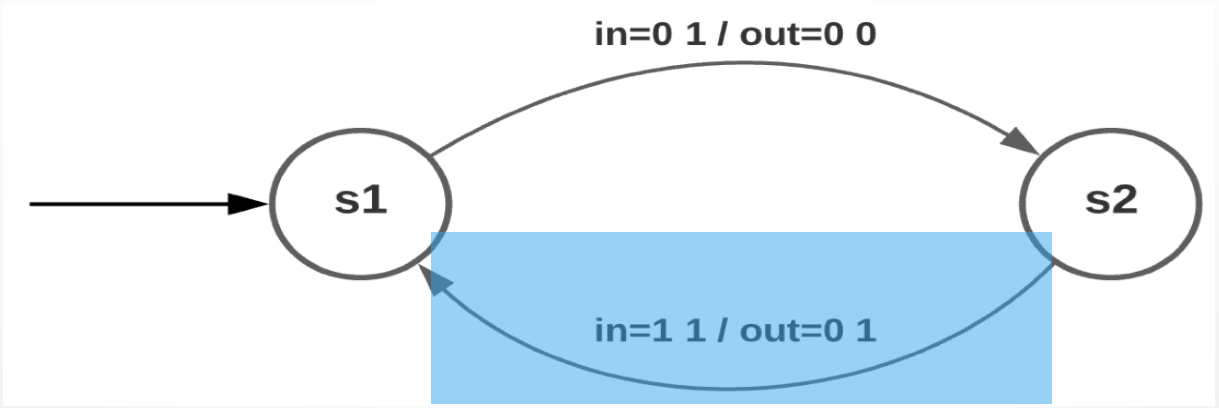
$done(O_{next})$

$done(v_{next})$

$\exists\ X_{following}\ next(X_{next}, X_{following})$

$\exists\ O_{following}\ next(O_{next}, O_{following})$

KB Representation



Rules: transitions

Transition (s_2 01 00 s_1)

$value(X_{next}, `s_1')$

$value(O_{next}, `01')$

$done(X_{next})$

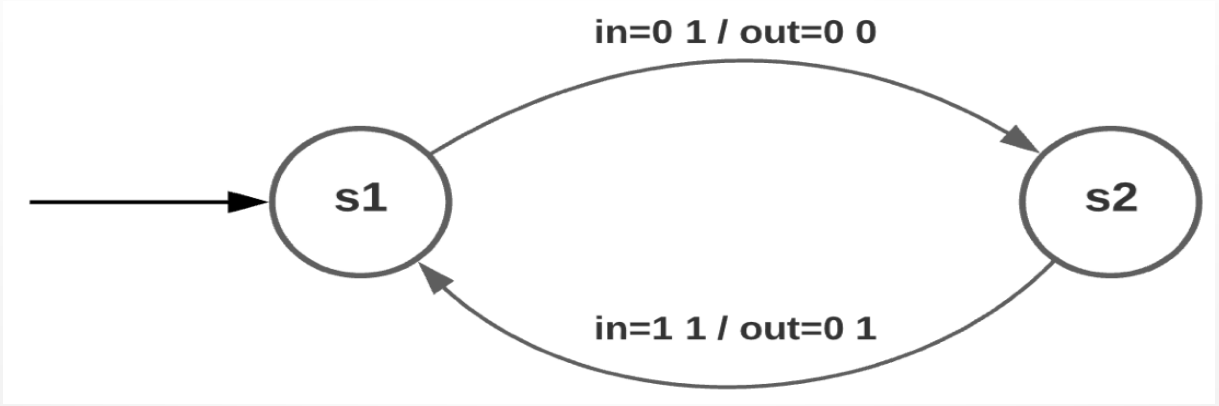
$done(O_{next})$

$done(v_{next})$

$\exists\ X_{following}\ next(X_{next}, X_{following})$

$\exists\ O_{following}\ next(O_{next}, O_{following})$

KB Representation



Queries:

Fact base:

Input sequence

Input sequence

...

Input sequence

We want to find a test sequence that verifies that the pump is always deactivated (o1=0) when the emergency button is on (i1=1)

Thank you !