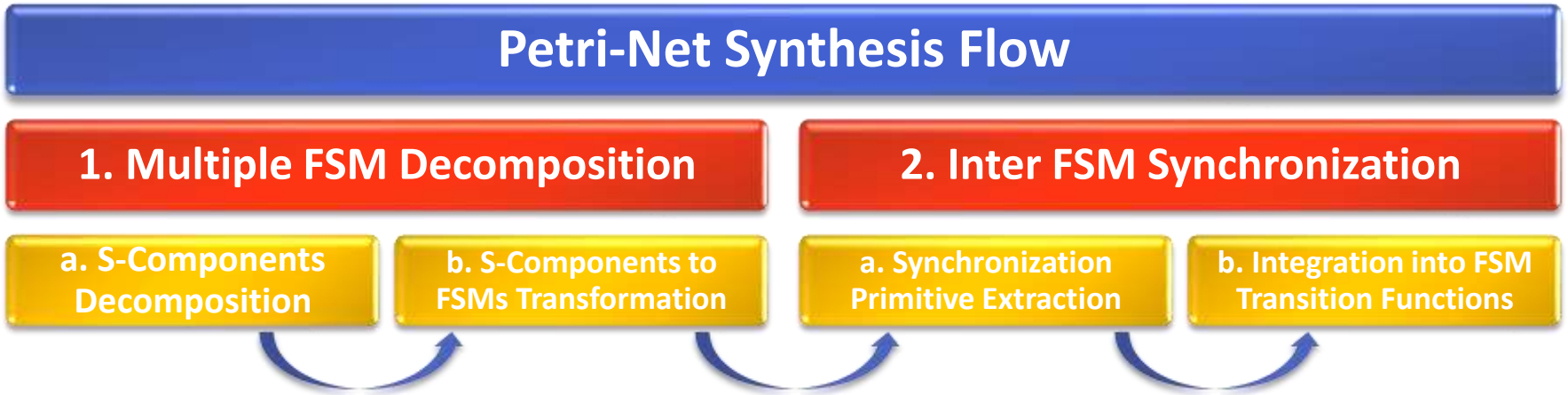
MSFSMs Synthesis Tool Flow Presentation

Circuit and Systems Lab,
University of Thessaly, Greece

Original Work

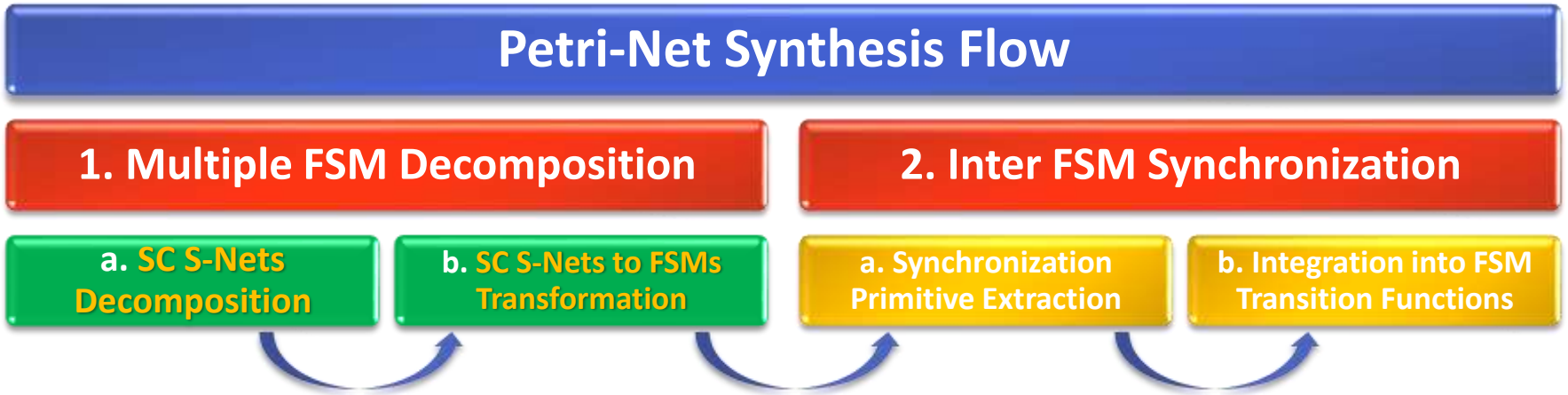
- Logic synthesis of concurrent control specifications
  - *P.M. Mattheakis, PhD Theses at* University of Crete (UOC), *Greece*
- **A Polynomial Time Flow for Implementing Free-choice Petri-nets,**
  - *P.M. Mattheakis, C.P. Sotiriou, and P.A. Beerel,*
  - *in 2012 IEEE 30th International Conference on Computer Design (ICCD).*
    - *IEEE, 2012, pp. 227–234.*
- All steps have linear complexity for
  - **Well-formed Free-Choice Petri-Nets**



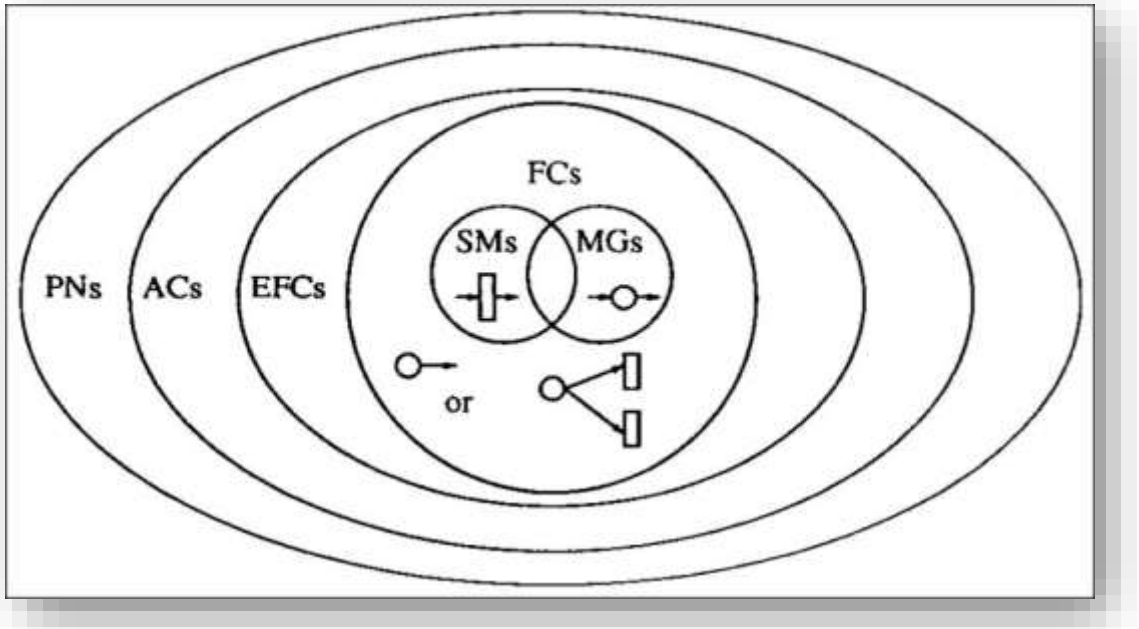
Our Contribution

- Improved S-Components Decomposition for
  - **ALL Well-formed Petri-Net Classes,**
    - withholding Linear Complexity, and
    - decomposing Strongly Connected S-Nets.

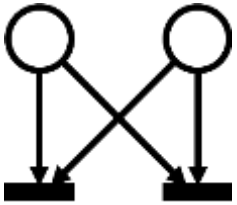
- Provide Community *the Complete* **MSFSMs Synthesis Tool**
  - *Multiple Synchronised FSMs (**MSFSMs**)*
  - *as closed-source freeware*
    - on GitHub



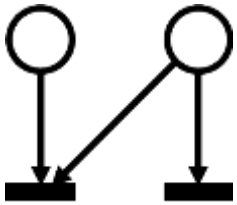
Structural Characterisation of Petri-Nets



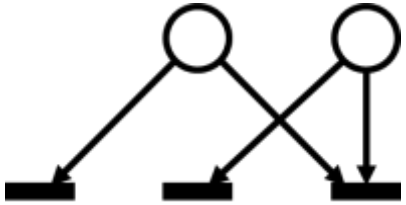
Free-Choice Net (a.k.a. Extended-Free Choice)



Asymmetric Choice Net



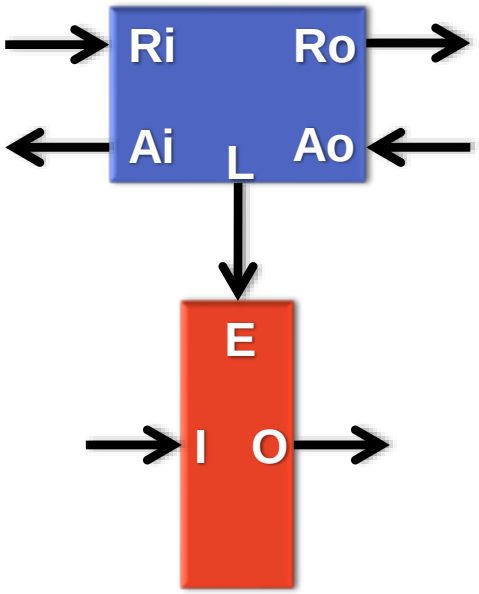
General Net

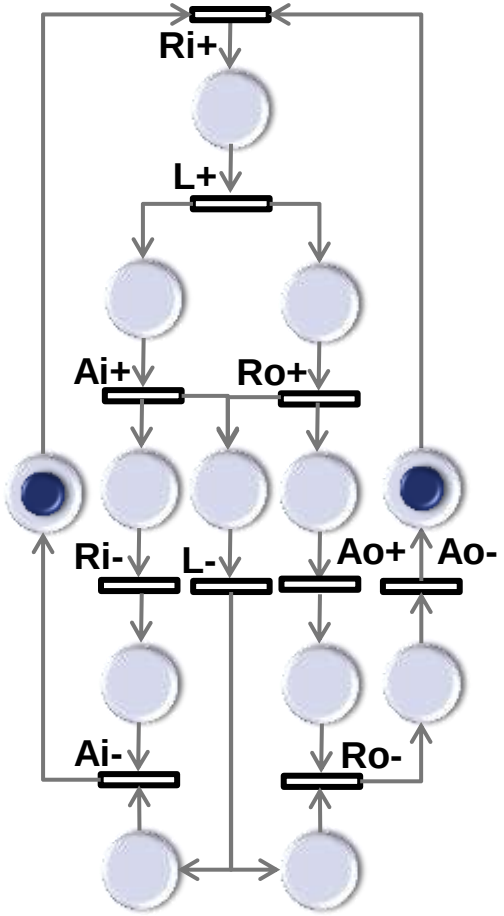


Complexity

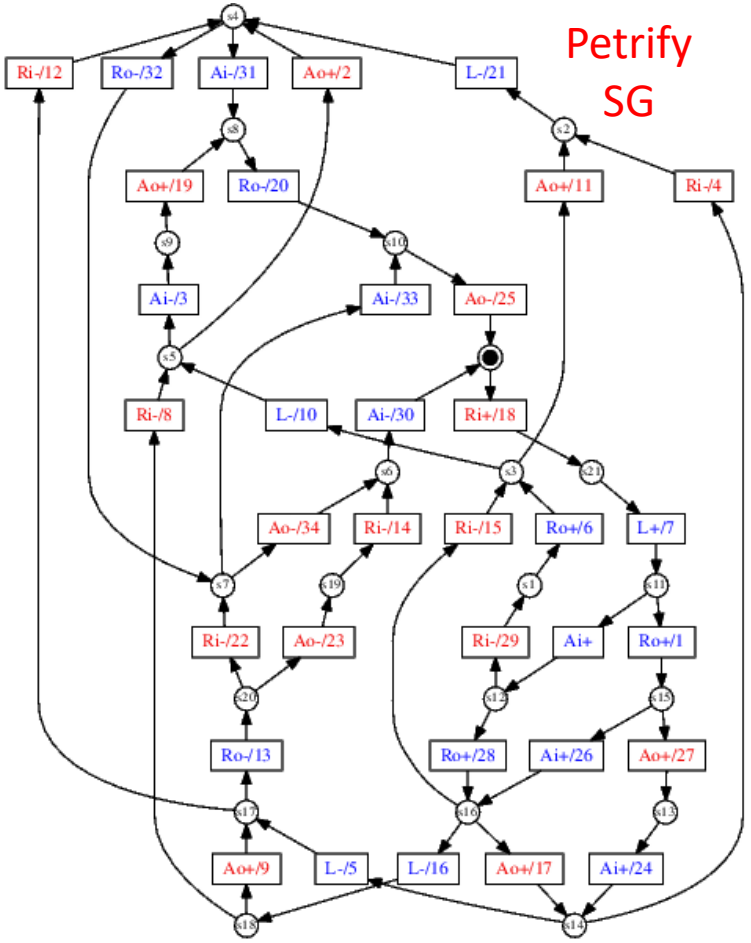
Theoretical Example

- State Graph Construction





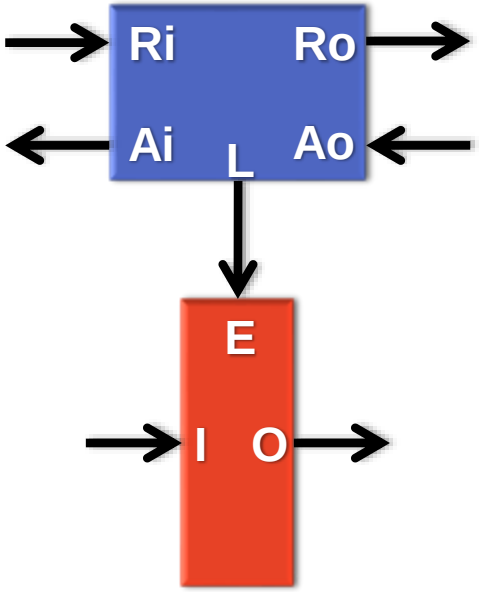
Well formed Petri-Net

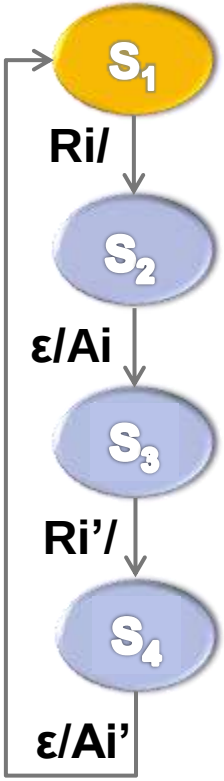


State Graph (FSM)

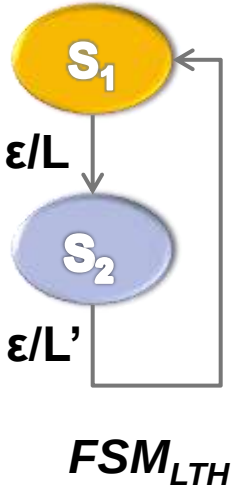
Theoretical Example

- SC S-Nets Decomposition

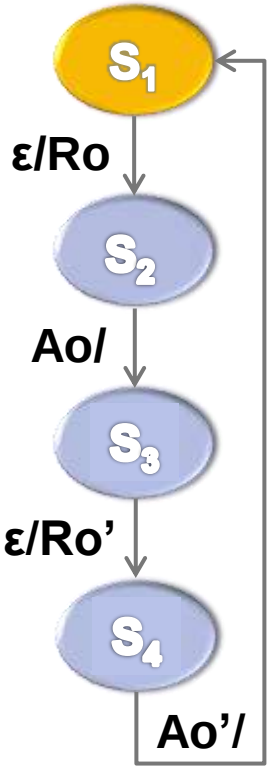




FSM_{LHS}



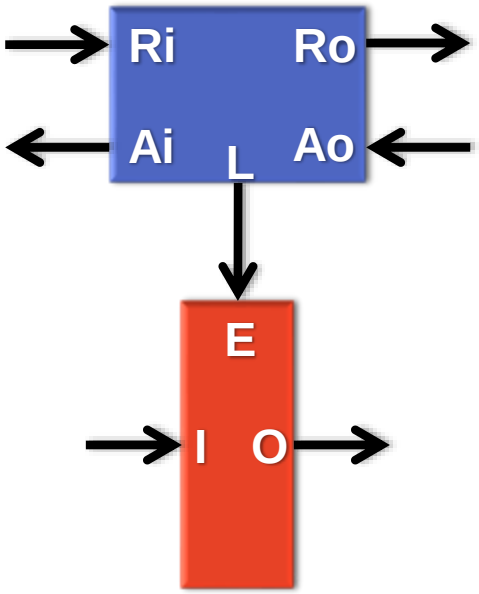
FSM_{LTH}

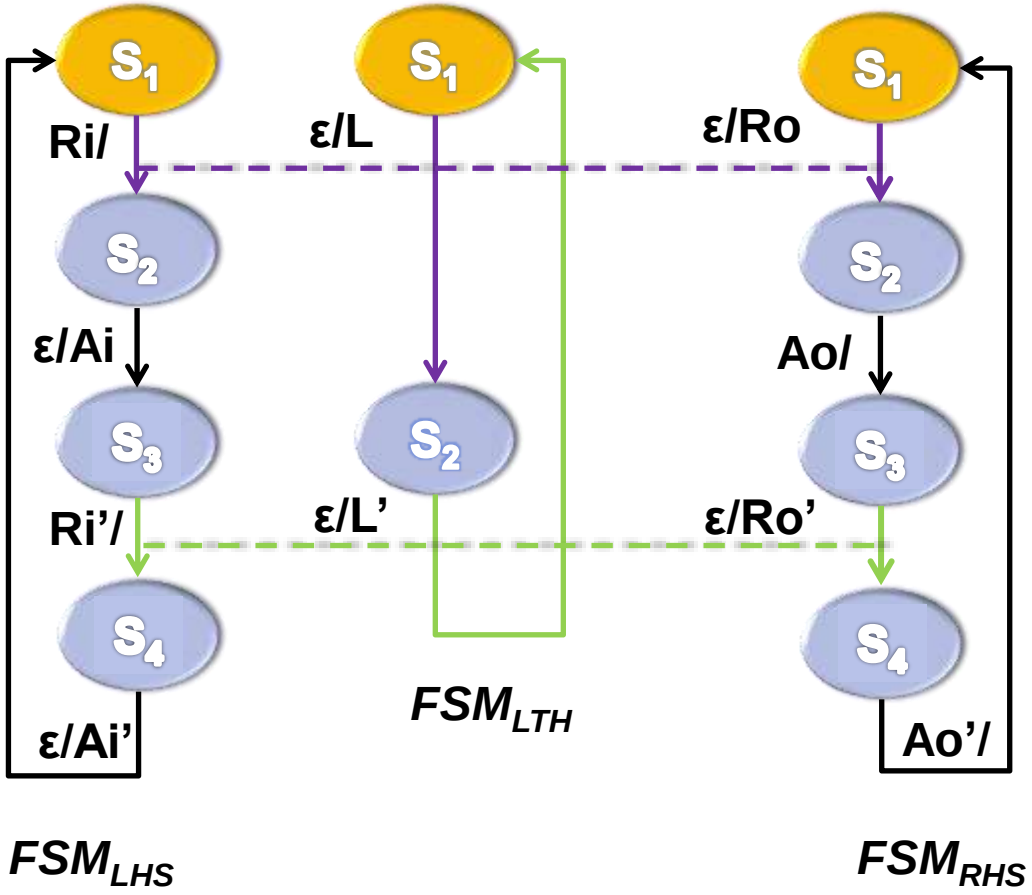


FSM_{RHS}

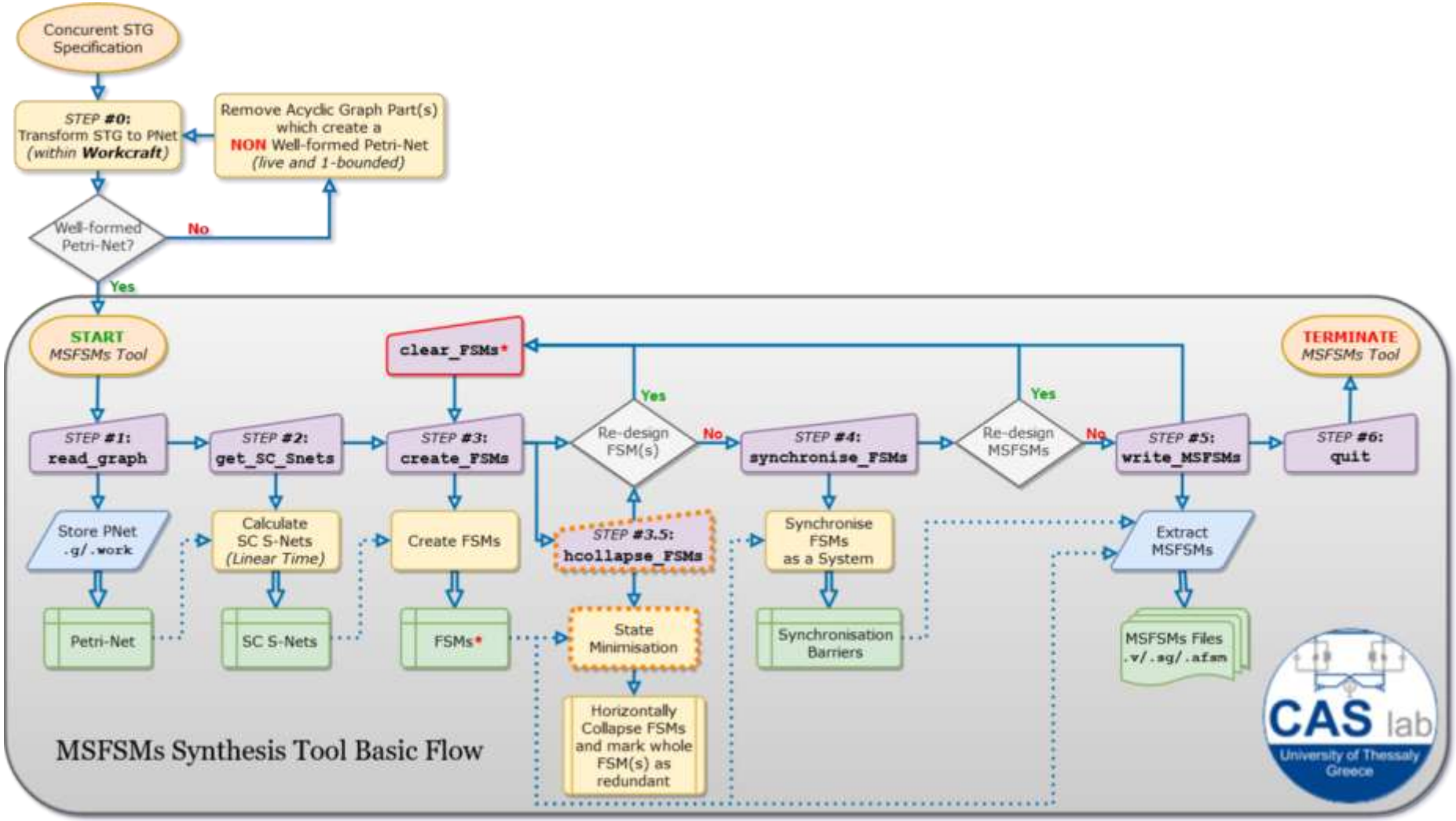
Theoretical Example

- FSMs Synchronisation





Flow Overview



STEP #1: `read_graph`

- run TCL command,
  - to load Petri-Net structure into MSFSMs tool

`read_graph`

option(s): `"?-format <g (default) | work> <filename>?"`

command type: **mandatory** flow command [*STEP#1*]

synopsys: loads an input graph file

description: Input graph must be a single well-formed Petri-Net. In case the graph is an STG should be transformed to Petri-Net before is loaded (feature available within Workcraft), meaning:

- place nodes,
  1) are connected only with transition nodes,
  2) is forbidden to connect with any other place node
- transition nodes,
  1) are connected only with place nodes,
  2) is forbidden to connect with any other transition node

Moreover, we do NOT support, and we do NOT filter out (yet):

- labels which indicate multiple instances of the same signal (i.e: req/1, req/2 etc.)
- or +/- keywords, for signals rise or fall events

STEP #2: `get_SC_snets`

- run TCL command,
  - to calculate an S-cover, and obtain an array of Strongly Connected S-Nets (SC-SNets) for the Petri-Net

`get_SC_snets`

option(s): ""

command type: **mandatory** flow command [*STEP#2*]

synopsys: calculates Strongly Connected S-Nets for all well formed Petri-Net Class

description: Petri-Net(PTNet) Classes are, Finite State Machine (FSM), Marked-Graph (MG), Free-Choice PTNet (FC-PTNet), Extended Free-Choice PTNet (EFC-PTNet), Asymmetric Choice PTNet (AC-PTNet) and General PTNet (GN-PTNet)

STEP #3: `create_FSMs`

- run TCL command,
  - to implement FSM mapping of SC-SNets and obtain an array of MSFSMs

`create_FSMs`

option(s): `"?-vertical_collapse? ?-horizontal_collapse? ?-cross_compatible?"`

command type: **mandatory** flow command [*STEP#3 and #3.5*]

synopsys: creates FSMs based on extracted S-Nets

description: -

STEP #3.5: `hcollapse_FSMs`

- run TCL command,
  - to implement vertical state collapsing between different FSMs
    - Meaning, compares FSMs with each other in order to check
      - if they are equivalent
      - to avoid mapping of redundant state

`hcollapse_FSMs` (or `horizontal_collapse_FSMs`)

option(s): ""

command type: **optional** flow command [**#3.5**]

synopsys: horizontally collapses extracted FSMs

description: -

STEP #3 & #3.5: `<flags>` of `'create_FSMs'`

- run TCL command,
  - to implement FSM mapping and vertical state collapsing

`create_FSMs`

option(s): ""

command type: **mandatory** flow command [*STEP#3 and #3.5*]

synopsys: executes both "`fsm_create`" and "`fsm_collapse`" commands in succession

description: -

STEP #4: synchronise_FSMs

- run TCL command,
  - to synchronise FSMs together and obtain the final array of the MSFSMs system

```
synchronise_FSMs
```

option(s): ""

command type: **mandatory** flow command [*STEP#4*]

synopsys: synchronises extracted "minimised" FSMs, in order to be able to implement wrapper logic

description: -

STEP #5: write_MSFSMs

- run TCL command,
  - to generate verilog RTL code per FSM and for the whole MSFSMs wrapper logic

write_MSFSMs

option(s): "?-format <syncmealy_behav (default) | syncmealy_synth | afsm_format>?"
"?-createcollapsed?"
"?-fulloutputstate?"
"?-timescale <string>?"

command type: **mandatory** flow command [*STEP#5*]

synopsys: writes MSFSMs systems in the requested format in the tool's execution directory

description: In total 3 formats are currently available with option "-format":

- syncmealy_behav (default), extracts 2 Verilog files, fsm_behav_mealy.v and msfsm_behav_mealy.v. The 1st defines the information of each FSM in **behavioral RTL** and the 2nd instantiates and interconnects the FSMs as a system.
- syncmealy_synth, extracts 2 Verilog files, fsm_synth_mealy.v and msfsm_synth_mealy.v. The 1st defines the information of each fsm in **synthesizable RTL** and the 2nd instantiates and interconnects the FSMs as a system.
- afsm_format, extracts 1 AFSM (.afsm) file and 1 Verilog file, fsm_afsm.afsm and msfsm_afsm.v. The 1st defines the information of each fsm structuraly for an internal tool which generates ***Asynchronous FSMs, using SR latches*** for each state and the 2nd instantiates and interconnects the FSMs as a system *(contact us for more info)*.

Option "-createcollapsed" creates only the collapsed FSMs

Option "-fulloutputstate" produces the full state of each FSM as output for debugging purposes

Option "-timescale " let the user to contumely specify the required timescale

List of all Available TCL commands

- TCL Command Line Interface
  - Supports TAB auto-completion

- read_graph
- list_petrinet
- list_places
- list_transitions
- list_petrinet_node_info
- list_petrinet_node_connections
- create_complimentary_set_of_output_signal
- list_complimentary_output_set_membership
- list_complimentary_output_sets
- list_complimentary_output_sets
- delete_complimentary_output_set
- get_transition_region_of_complimentary_output_set
- get_SC_Snets
- list_SC_Snet

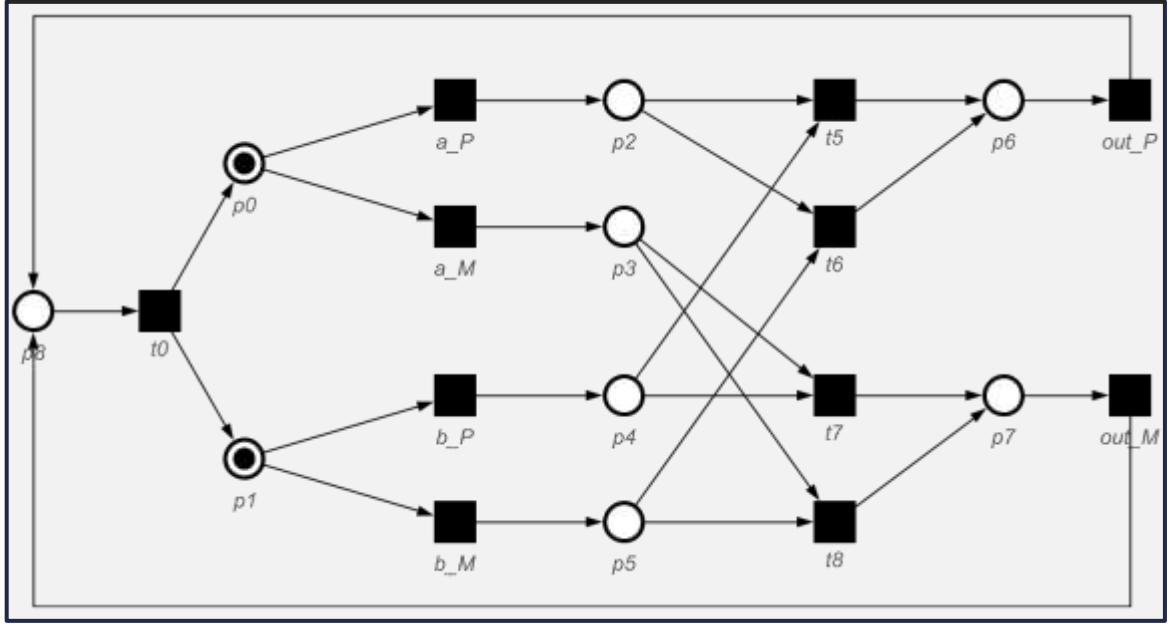
- create_FSMs[*]
- list_FSM
- clear_FSMs
- horizontal_collapse_FSMs
- list_horizontal_collapsed_FSMs[*]
- vertical_collapse_FSMs[*]
- list_vertical_collapsed_FSMs[*]
- synchronise_FSMs
- list_FSM_syncrhonisation_barriers[*]
- setvar_write_level_sensitisation_of_output_signals
- getvar_write_level_sensitisation_of_output_signals
- write_MSFSMs
- write_SC_Snet_to_dot
- write_FSM_to_dot
- quit

[*] Some feature*(s)* are not supported yet.

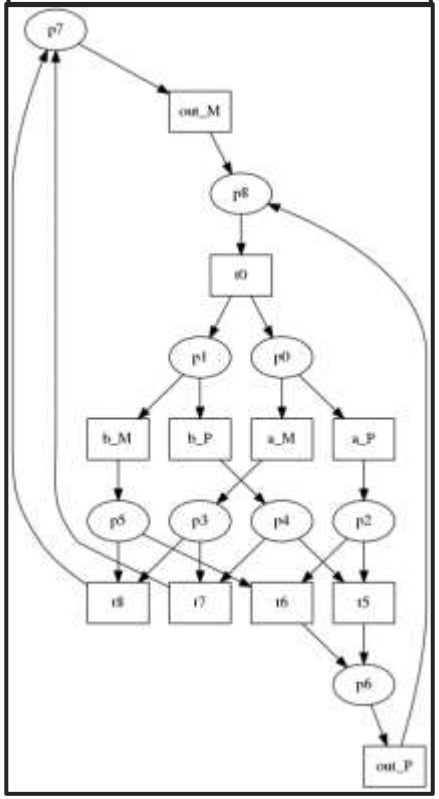
Example: xor-gate-PTnet.g

- General Class of a Well-formed Petri-Net
  - Behavior a XOR gate

Original Petri-Net *(representation within Workcraft)*



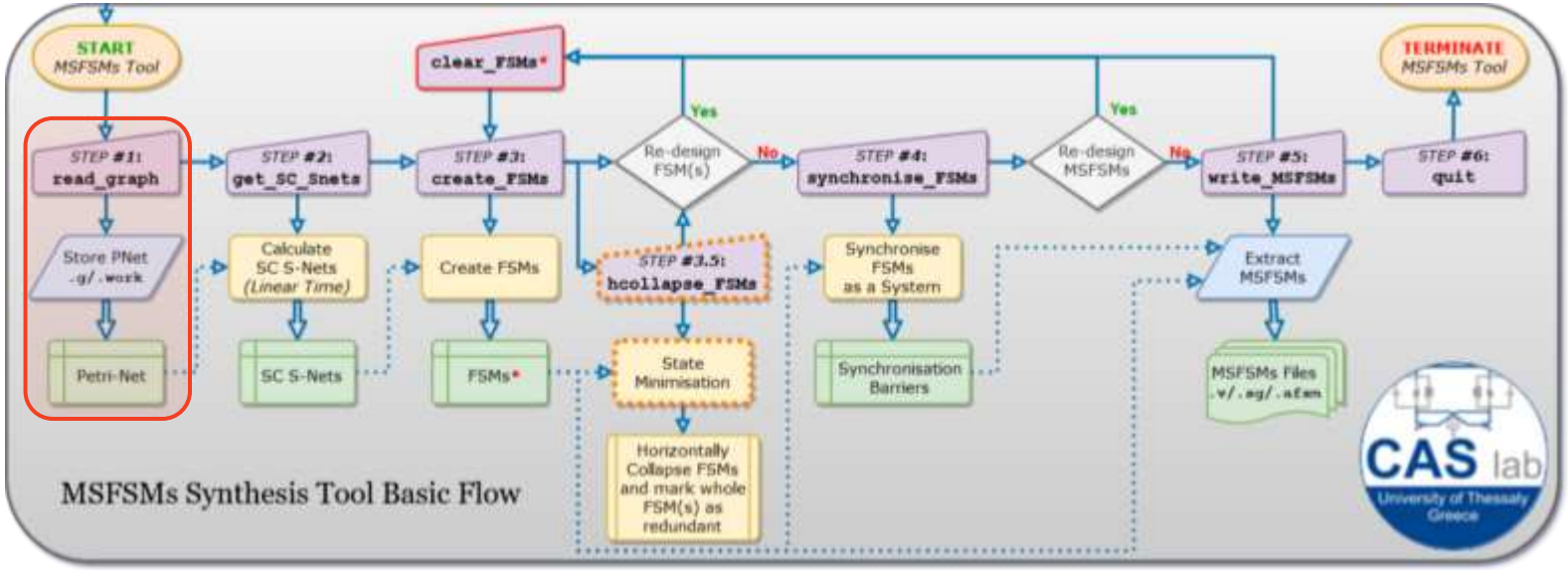
Original Petri-Net



read_graph

- FlowSTEP **#1**
  - Load a well-formed Petri-Net

```
[MSFSMs Syntehsis Tool v1.00]%> read_graph xor-gate-PTnet.g
INFO: Total Nodes       : 20
INFO: Total Transitions : 11
INFO: Total Places      : 9
INFO: Total Edges       : 27
INFO-TCL: Successfully loaded Petri-Net.
[MSFSMs Syntehsis Tool v1.00]%>
```



list_petrinet

```
PT-Net [0][0]: Label = p7, Type = Place (is Empty)
                Predecessors: t8[9][1], t7[16][1]
                Successors: out_M[17][0]
PT-Net [1][0]: Label = p4, Type = Place (is Empty)
                Predecessors: b_P[2][1]
                Successors: t5[10][0], t7[16][1]
PT-Net [2][0]: Label = p1, Type = Place (is Marked)
                Predecessors: t0[5][0]
                Successors: b_M[3][0], b_P[2][1]
PT-Net [2][1]: Label = b_P, Type = Transition (is Input)
                Predecessors: p1[2][0]
                Successors: p4[1][0]
PT-Net [3][0]: Label = b_M, Type = Transition (is Input)
                Predecessors: p1[2][0]
                Successors: p5[14][0]
PT-Net [3][1]: Label = t6, Type = Transition (is Input)
                Predecessors: p5[14][0], p2[15][0]
                Successors: p6[7][0]
PT-Net [5][0]: Label = t0, Type = Transition (is Input)
                Predecessors: p8[13][0]
                Successors: p0[9][0], p1[2][0]
PT-Net [5][1]: Label = a_P, Type = Transition (is Input)
                Predecessors: p0[9][0]
                Successors: p2[15][0]
PT-Net [6][0]: Label = a_M, Type = Transition (is Input)
                Predecessors: p0[9][0]
                Successors: p3[8][0]
PT-Net [7][0]: Label = p6, Type = Place (is Empty)
                Predecessors: t5[10][0], t6[3][1]
                Successors: out_P[16][0]
PT-Net [8][0]: Label = p3, Type = Place (is Empty)
                Predecessors: a_M[6][0]
                Successors: t7[16][1], t8[9][1]
```

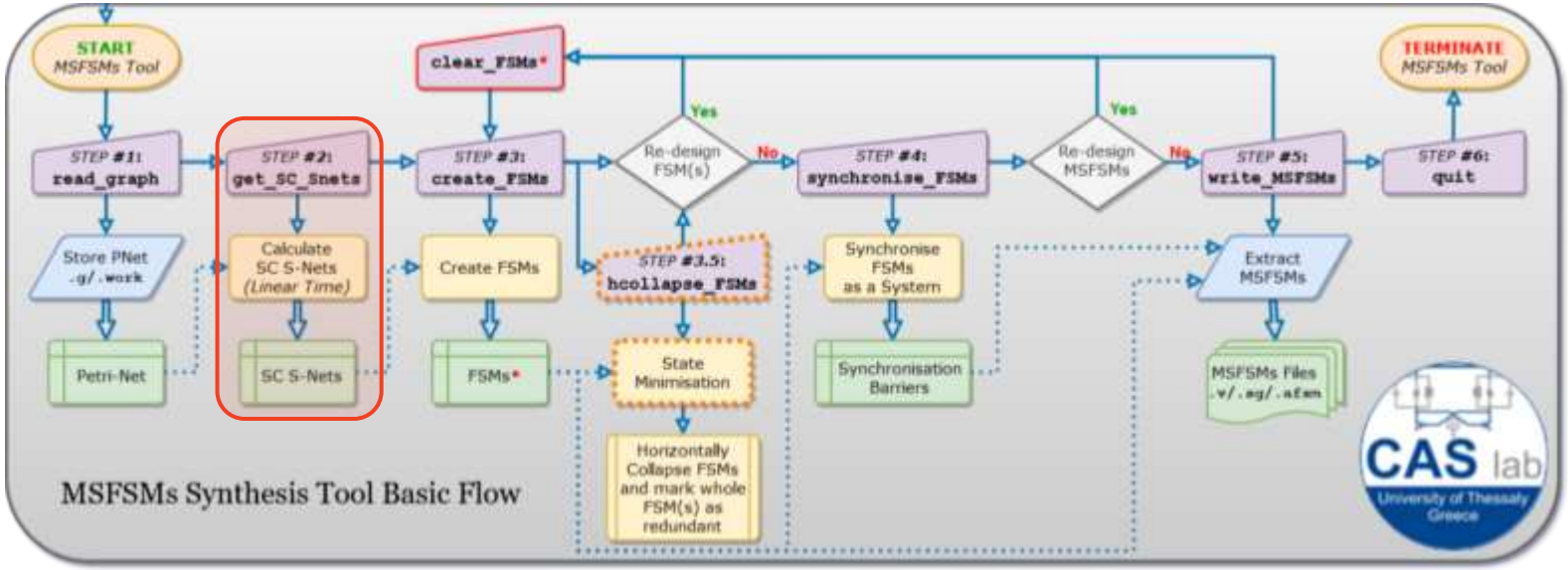
```
PT-Net [8][0]: Label = p3, Type = Place (is Empty)
                Predecessors: a_M[6][0]
                Successors: t7[16][1], t8[9][1]
PT-Net [9][0]: Label = p0, Type = Place (is Marked)
                Predecessors: t0[5][0]
                Successors: a_M[6][0], a_P[5][1]
PT-Net [9][1]: Label = t8, Type = Transition (is Input)
                Predecessors: p3[8][0], p5[14][0]
                Successors: p7[0][0]
PT-Net [10][0]: Label = t5, Type = Transition (is Input)
                Predecessors: p4[1][0], p2[15][0]
                Successors: p6[7][0]
PT-Net [13][0]: Label = p8, Type = Place (is Empty)
                Predecessors: out_P[16][0], out_M[17][0]
                Successors: t0[5][0]
PT-Net [14][0]: Label = p5, Type = Place (is Empty)
                Predecessors: b_M[3][0]
                Successors: t6[3][1], t8[9][1]
PT-Net [15][0]: Label = p2, Type = Place (is Empty)
                Predecessors: a_P[5][1]
                Successors: t5[10][0], t6[3][1]
PT-Net [16][0]: Label = out_P, Type = Transition (is Input)
                Predecessors: p6[7][0]
                Successors: p8[13][0]
PT-Net [16][1]: Label = t7, Type = Transition (is Input)
                Predecessors: p4[1][0], p3[8][0]
                Successors: p7[0][0]
PT-Net [17][0]: Label = out_M, Type = Transition (is Input)
                Predecessors: p7[0][0]
                Successors: p8[13][0]

INFO-TCL: Successfully displayed Petri-Net.
[MSFSMs Syntehsis Tool v1.00]%>
```

get_SC_Snets

- FlowSTEP **#2**
  - Calculate Strongly Connected S-Nets

```
[MSFSMs Syntehsis Tool v1.00]%> get_SC_Snets
INFO-TCL: Successfully extracted #3 Strongly Connected S-Nets.
[MSFSMs Syntehsis Tool v1.00]%>
```



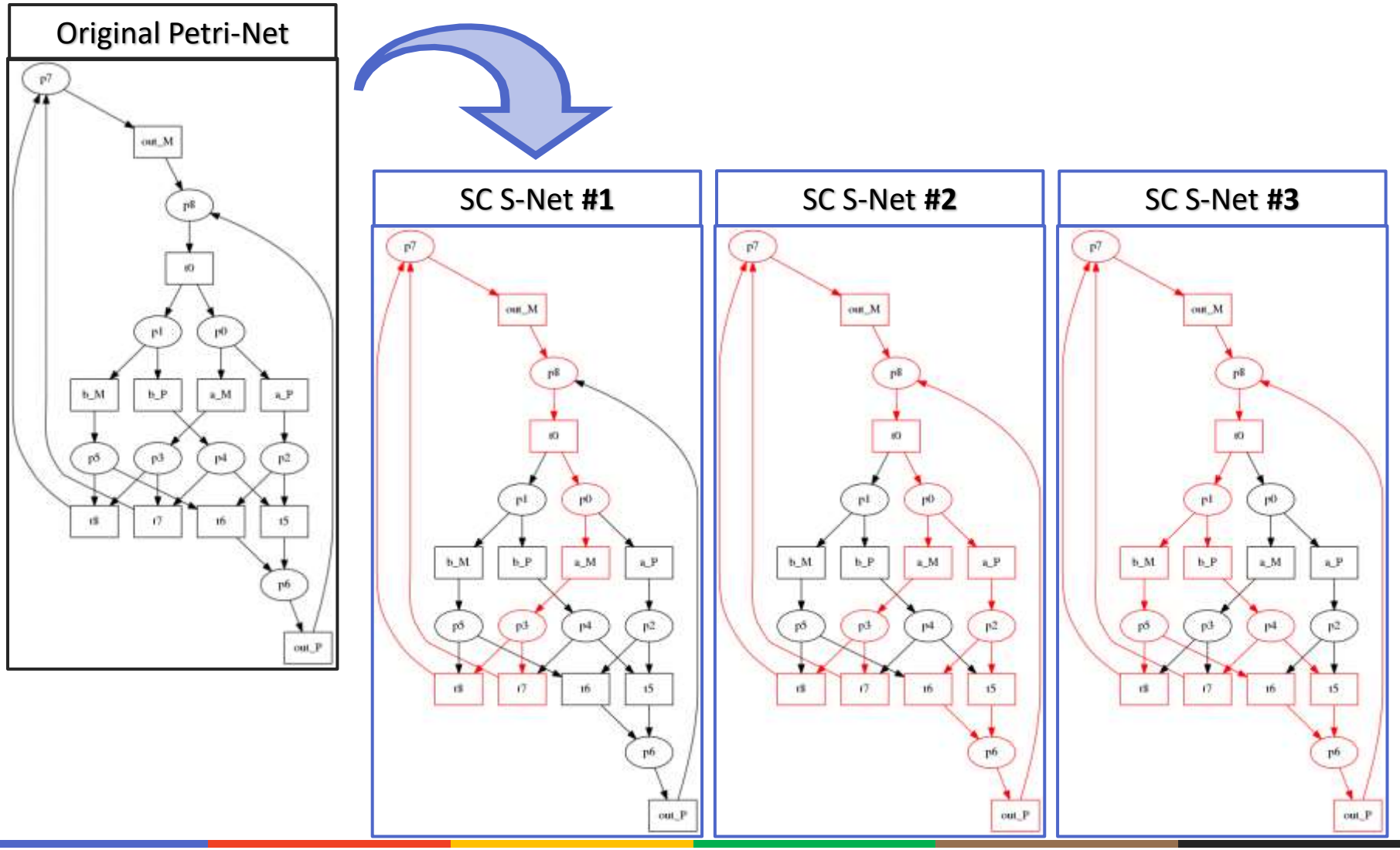
list_SC_Snet

```
[MSFSMs Syntehsis Tool v1.00]%> list_SC_Snet
ERROR-TCL: wrong # args: should be "list_SC_Snet -all | -index <SC S-Net number>"
[MSFSMs Syntehsis Tool v1.00]%> list_SC_Snet -all
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
*** SC S-net #1, Total Nodes = 9, H-collapsed = 'false' ***
SC S-net (1,1): p7[0][0]
SC S-net (1,2): t0[5][0]
                Predecessor Place: p8(1,7)[13,0]
                Successor Place: p0(1,5)[9,0]
SC S-net (1,3): a_M[6][0]
                Predecessor Place: p0(1,5)[9,0]
                Successor Place: p3(1,4)[8,0]
SC S-net (1,4): p3[8][0]
SC S-net (1,5): p0[9][0]
SC S-net (1,6): t8[9][1]
                Predecessor Place: p3(1,4)[8,0]
                Successor Place: p7(1,1)[0,0]
SC S-net (1,7): p8[13][0]
SC S-net (1,8): t7[16][1]
                Predecessor Place: p3(1,4)[8,0]
                Successor Place: p7(1,1)[0,0]
SC S-net (1,9): out_M[17][0]
                Predecessor Place: p7(1,1)[0,0]
                Successor Place: p8(1,7)[13,0]
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
*** SC S-net #2, Total Nodes = 15, H-collapsed = 'false' ***
SC S-net (2,1): p7[0][0]
SC S-net (2,2): p4[1][0]
```

```
SC S-net (3,15): out_M[17][0]
                 Predecessor Place: p7(3,1)[0,0]
                 Successor Place: p8(3,11)[13,0]
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

INFO-TCL: Successfully displayed all SC S-Nets.
[MSFSMs Syntehsis Tool v1.00]%>
```

write_SC_Snet_to_dot

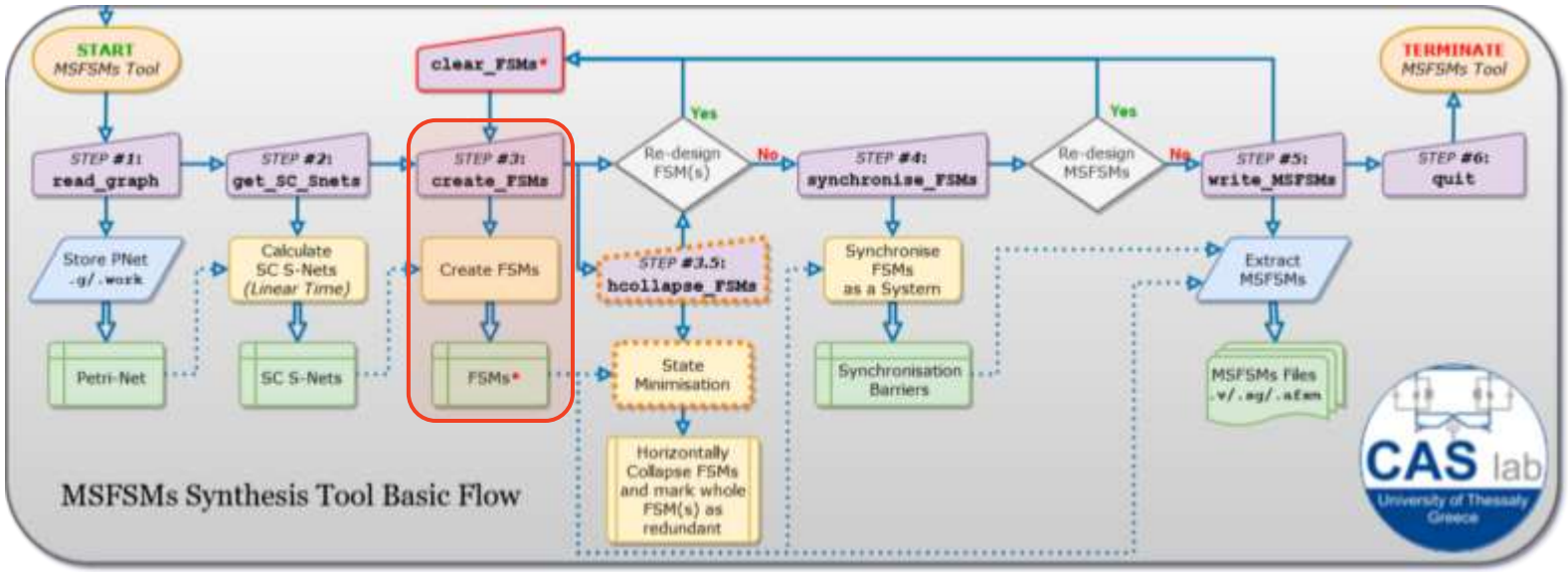


create_FSMs

- FlowSTEP **#3**
  - Create FSMs based on S-Nets

```
[MSFSMs Syntehsis Tool v1.00]%> create_FSMs
---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---
INFO: Creation of FSM #1 (of #3 total).
---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---
INFO: Creation of FSM #2 (of #3 total).
---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---
INFO: Creation of FSM #3 (of #3 total).
---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---

INFO-TCL: Created FSMs (without Collapsing redundant FSMs).
[MSFSMs Syntehsis Tool v1.00]%>
```



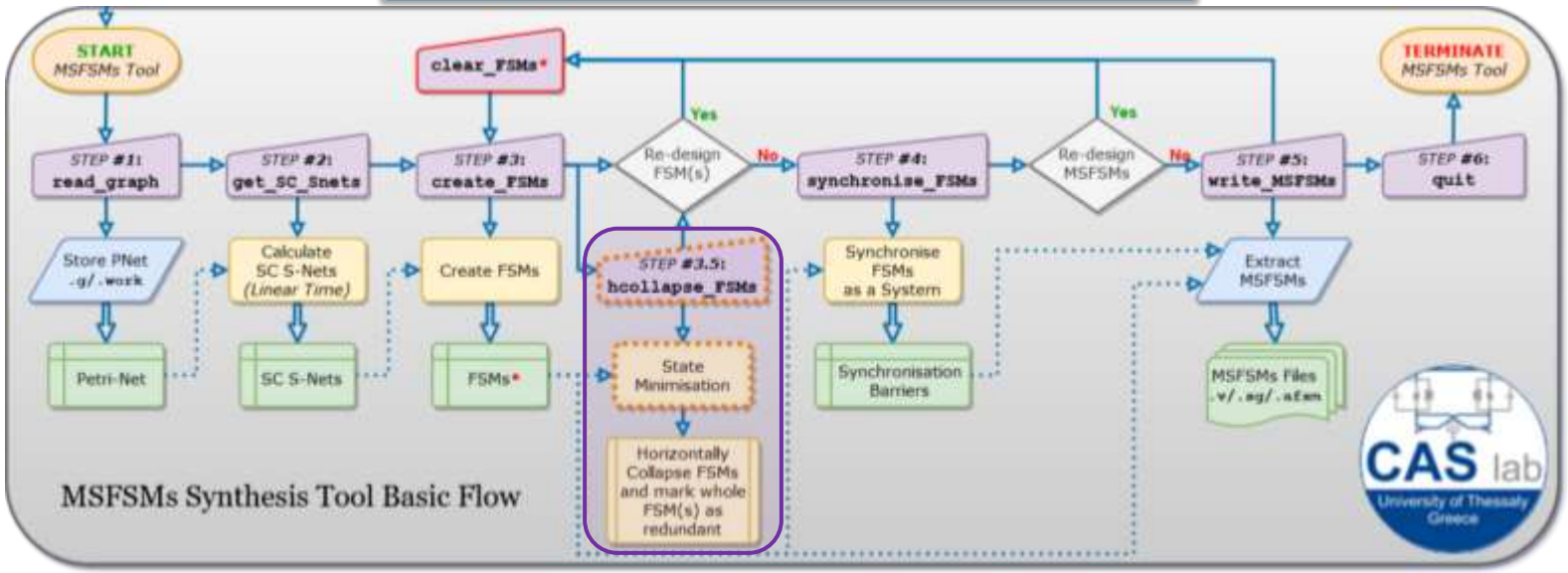
horizontal_collapse_FSMs

- FlowSTEP #3.5
  - Minimise FSMs by horizontally collapsing equivalent ones

```
[MSFSMs Syntehsis Tool v1.00]%> horizontal_collapse_FSMs
--- --- --- --- --- --- --- --- --- --- --- ---
INFO: Creation of FSM #1 (of #3 total).
--- --- --- --- --- --- --- --- --- --- --- ---
INFO: Creation of FSM #2 (of #3 total).
--- --- --- --- --- --- --- --- --- --- --- ---
INFO: Creation of FSM #3 (of #3 total).
--- --- --- --- --- --- --- --- --- --- --- ---

--- --- --- --- --- --- --- --- --- --- --- ---

INFO-TCL: Successfully horizontally Collapsed FSMs.
[MSFSMs Syntehsis Tool v1.00]%>
```



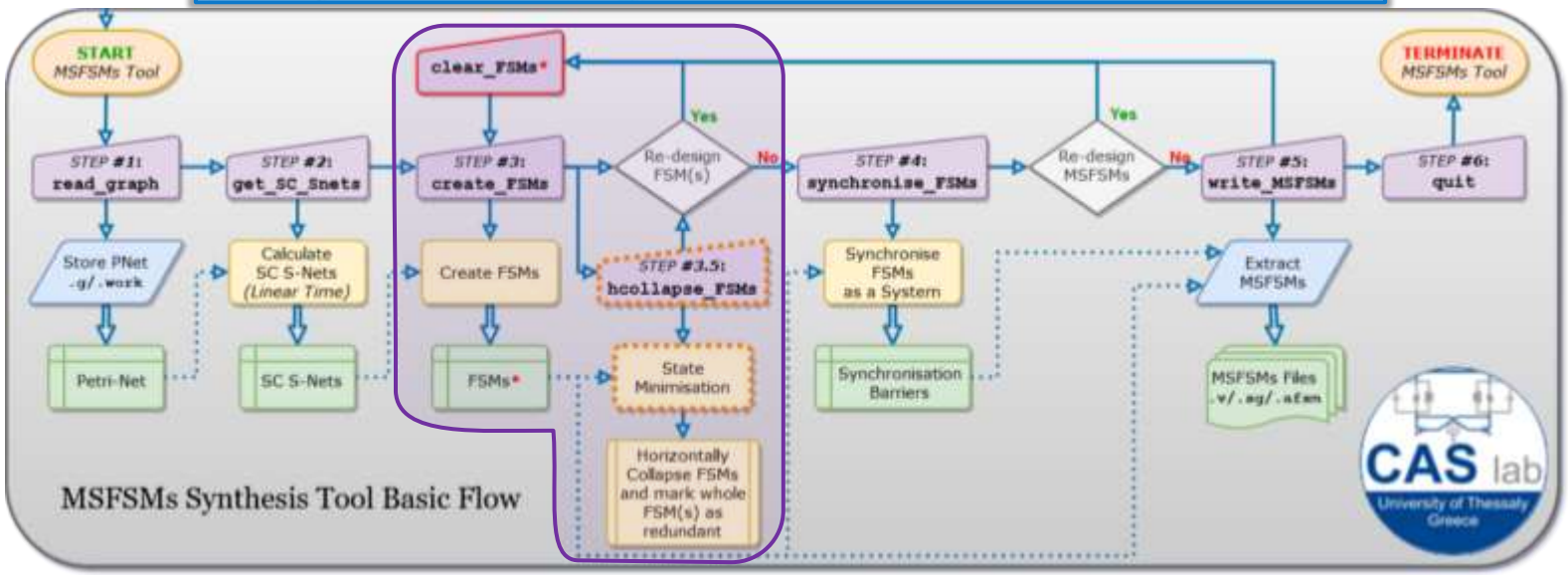
`create_FSMs` *(with arguments)*

- FlowSTEP **#3** *(alternative)*
  - Create and minimize FSMs based on S-Nets

```
WARNING: Argument "-horizontal_collapse | -hcollapse" is not supported yet. It is ignored.
WARNING: Argument "-cross_compatible | -xcompatible" is not supported yet. It is ignored.
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
INFO: Creation of FSM #1 (of #3 total).
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
INFO: Creation of FSM #2 (of #3 total).
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
INFO: Creation of FSM #3 (of #3 total).
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

INFO-TCL: Created FSMs and Horizontally Collapsed redundant FSMs.
[MSFSMs Syntehsis Tool v1.00]%>
```

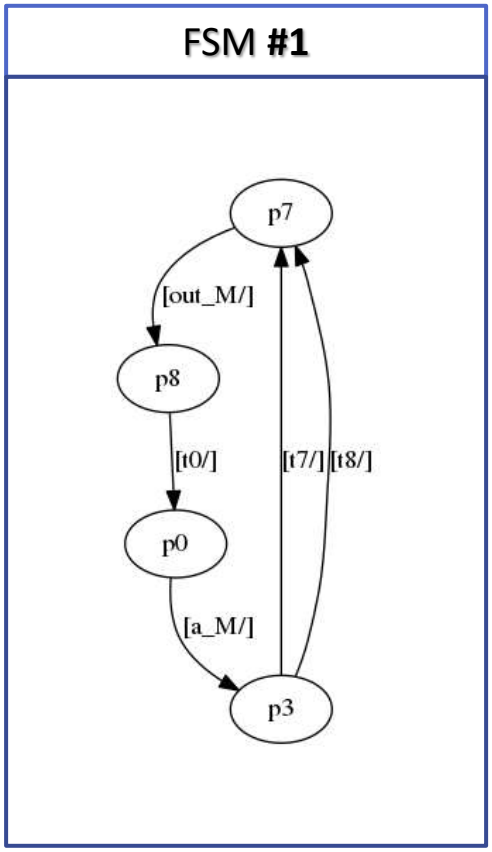


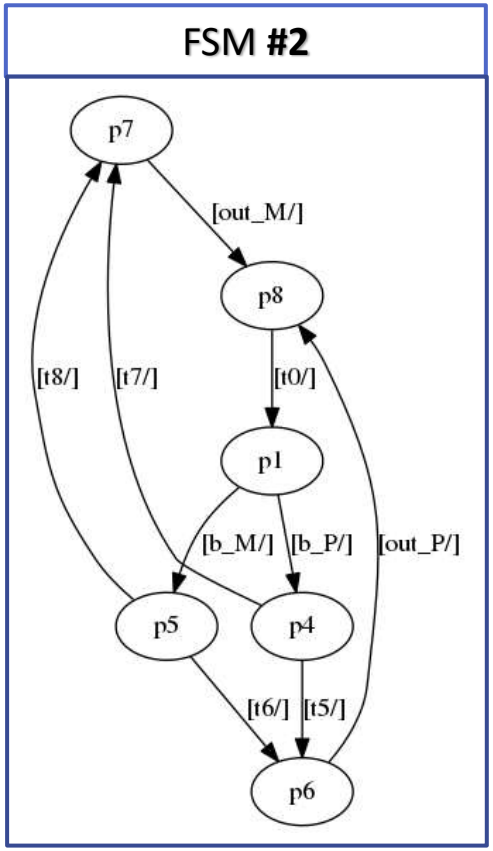
list_FSM

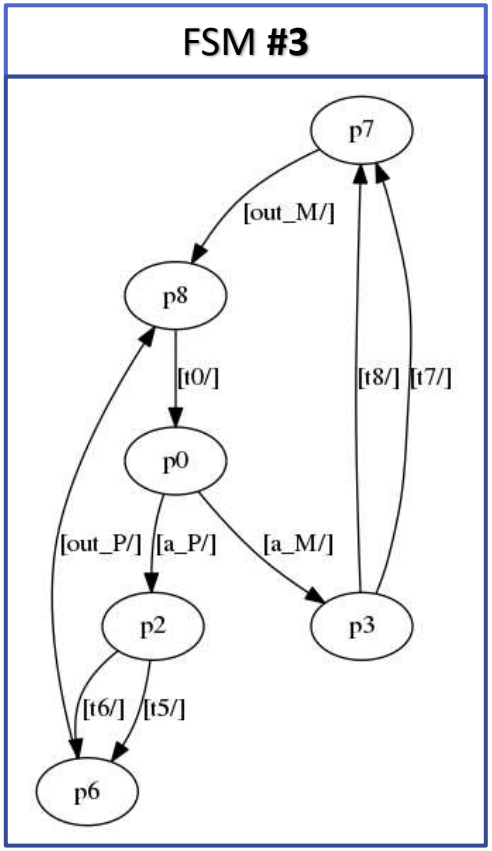
```
[MSFSMs Syntehsis Tool v1.00]%> list_FSM -all
--- --- --- --- --- --- --- --- --- --- --- --- ---
*** FSM 1, Total list Entries = 9, H-Collapsed = 'false' ***
FSM (1,1): Label = p7, Type = State (is Initially Inactive)
                Successor(s): out_M/(0,9)
                Predecessor(s): t8/(0,6) t7/(0,8)
FSM (1,2): Label = t0/, Type = Trans. Function (is Input)
                Successor(s): p0(0,5)
                Predecessor(s): p8(0,7)
FSM (1,3): Label = a_M/, Type = Trans. Function (is Input)
                Successor(s): p3(0,4)
                Predecessor(s): p0(0,5)
FSM (1,4): Label = p3, Type = State (is Initially Inactive)
                Successor(s): t7/(0,8) t8/(0,6)
                Predecessor(s): a_M/(0,3)
FSM (1,5): Label = p0, Type = State (is Initially Active)
                Successor(s): a_M/(0,3)
                Predecessor(s): t0/(0,2)
FSM (1,6): Label = t8/, Type = Trans. Function (is Input)
                Successor(s): p7(0,1)
                Predecessor(s): p3(0,4)
FSM (1,7): Label = p8, Type = State (is Initially Inactive)
                Successor(s): t0/(0,2)
                Predecessor(s): out_M/(0,9)
FSM (1,8): Label = t7/, Type = Trans. Function (is Input)
                Successor(s): p7(0,1)
                Predecessor(s): p3(0,4)
FSM (1,9): Label = out_M/, Type = Trans. Function (is Input)
                Successor(s): p8(0,7)
                Predecessor(s): p7(0,1)
--- --- --- --- --- --- --- --- --- --- --- --- ---
*** FSM 2, Total list Entries = 15, H-Collapsed = 'false' ***
FSM (2,1): Label = p7, Type = State (is Initially Inactive)
                Successor(s): out_M/(1,15)
                Predecessor(s): t8/(1,9) t7/(1,14)
```

```
FSM (3,15): Label = out_M/, Type = Trans. Function (is Input)
                Successor(s): p8(2,11)
                Predecessor(s): p7(2,1)
--- --- --- --- --- --- --- --- --- --- --- --- ---
INFO-TCL: Successfully displayed all FSMs.
[MSFSMs Syntehsis Tool v1.00]%>
```

write_FSMs_to_dot



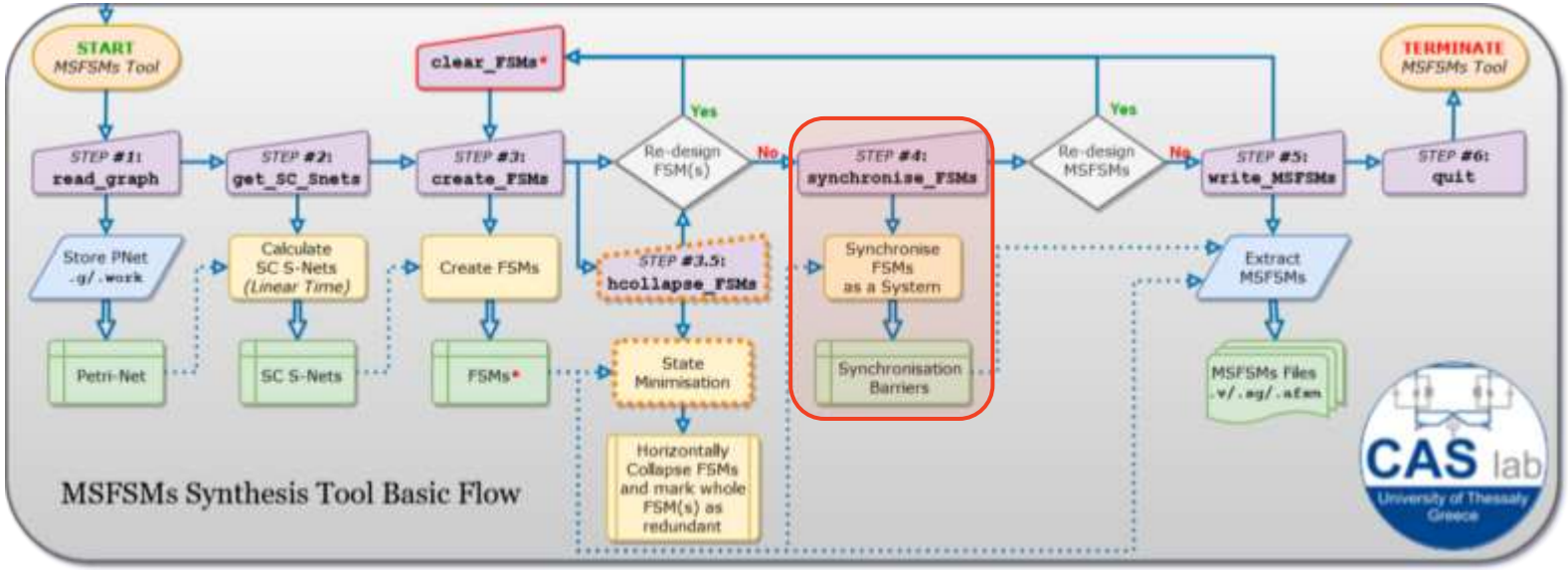




synchronise_FSMs

- FlowSTEP **#4**
  - Synchronisation between FSMs in order to form MSFSMs system

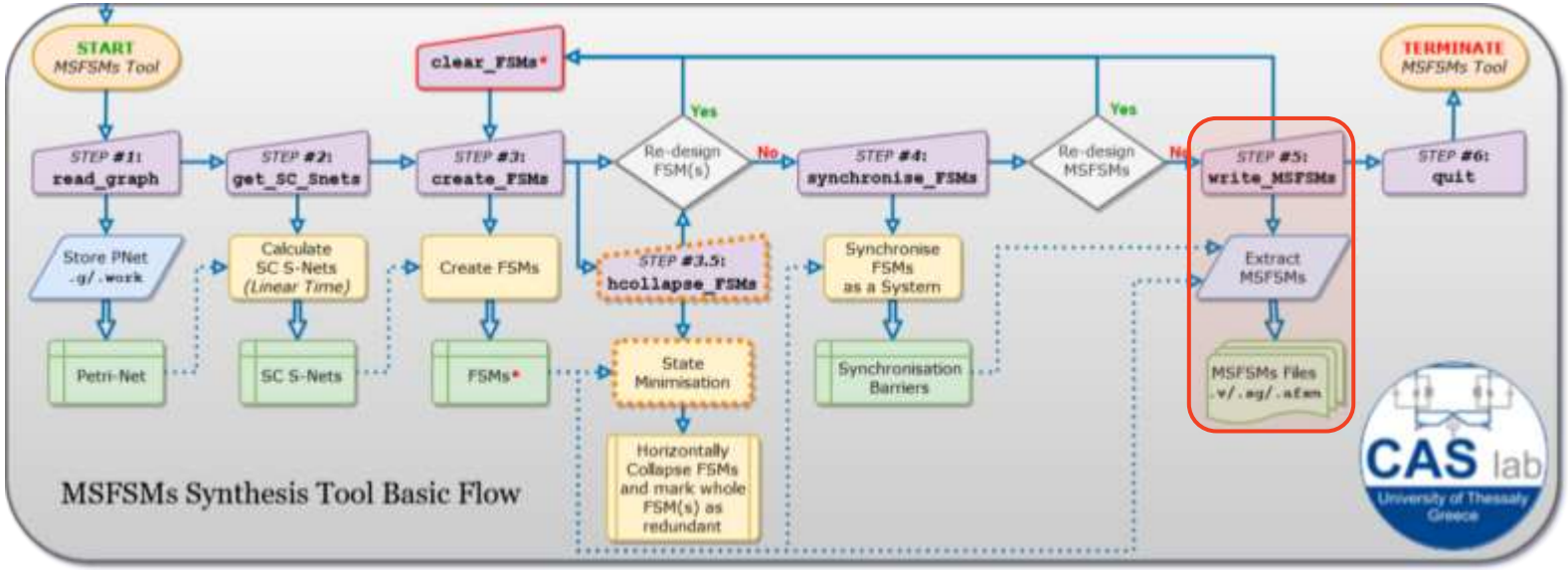
```
[MSFSMs Syntehsis Tool v1.00]%> synchronise_FSMs
INFO-TCL: Successfully Synchronised FSMs.
[MSFSMs Syntehsis Tool v1.00]%>
```



write_MSFSMs

- FlowSTEP #5
  - Write Verilog RTL Files of a complete MSFSMs System.

```
[MSFSMs Syntehsis Tool v1.00]%> write_MSFSMs -
ERROR-TCL: wrong # args: should be "write_MSFSMs ?-format <syncmealy_behav (default) | syncmealy_synth | afsm_format>?
                                                ?-createcollapsed?
                                                ?-fulloutputstate?
                                                ?-timescale <string>?
                                                ?-dlatchimplementation clk <string> data <string>?"
[MSFSMs Syntehsis Tool v1.00]%> write_MSFSMs
INFO-TCL: FSMs Verilog Code generated for "-format syncmealy_behav".
[MSFSMs Syntehsis Tool v1.00]%>
```



Contact Us