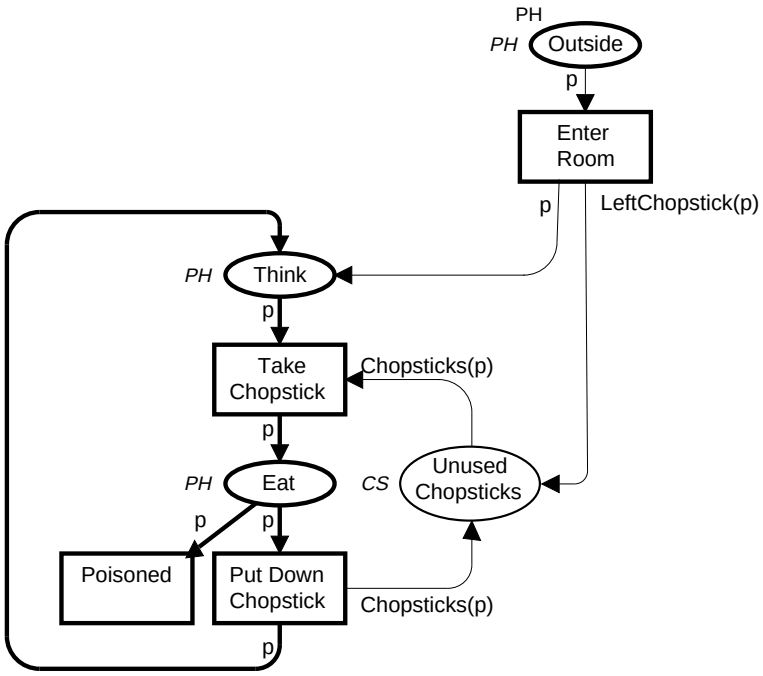
Model Checking Coloured Petri Nets Exploiting Strongly Connected Components

Allan Cheng
Søren Christensen
Kjeld H. Mortensen

University of Aarhus, Denmark

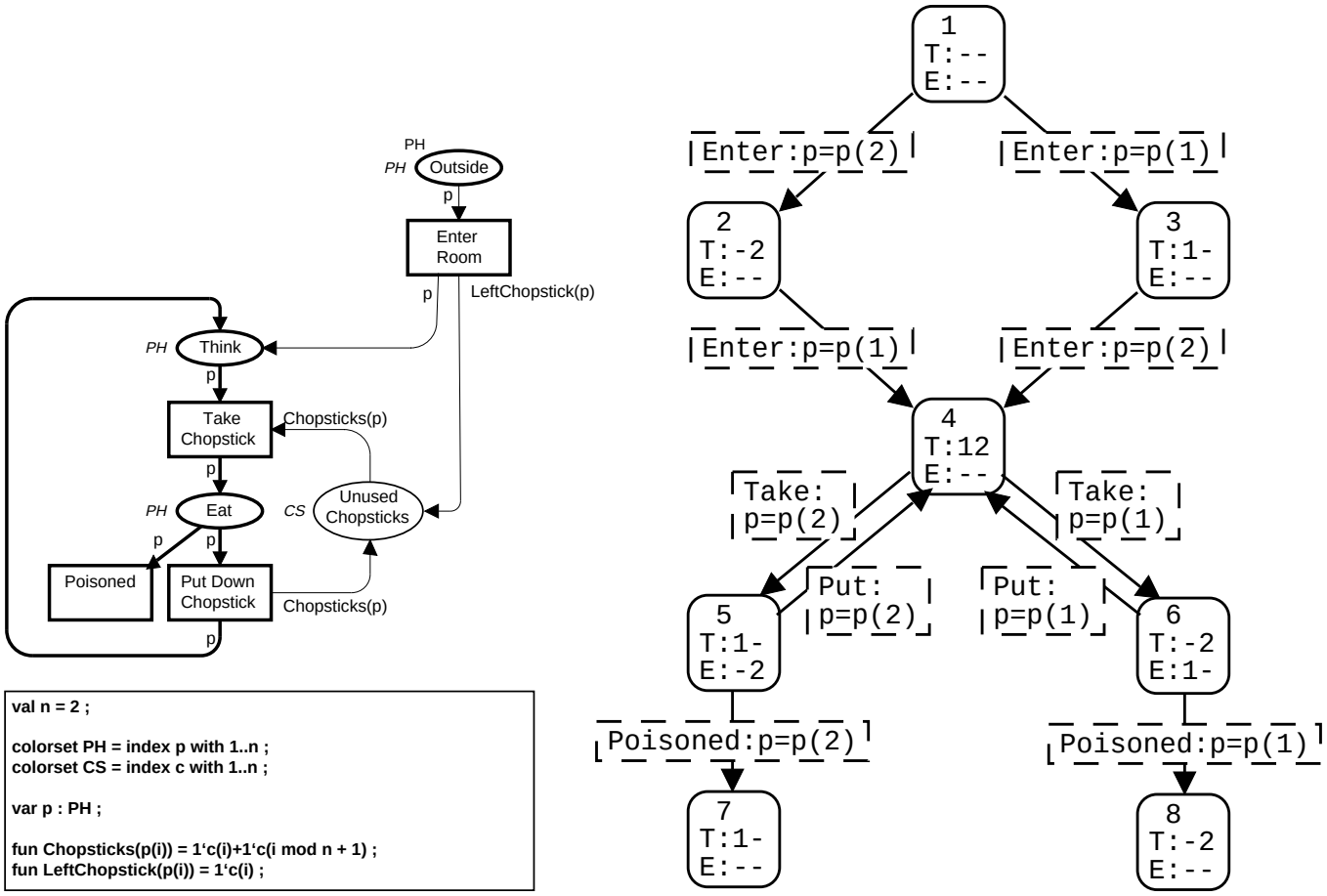
Coloured Petri Nets




```
val n = 2 ;

colorset PH = index p with 1..n ;
colorset CS = index c with 1..n ;

var p : PH ;

fun Chopsticks(p(i)) = 1‘c(i)+1‘c(i mod n + 1) ;
fun LeftChopstick(p(i)) = 1‘c(i) ;
```

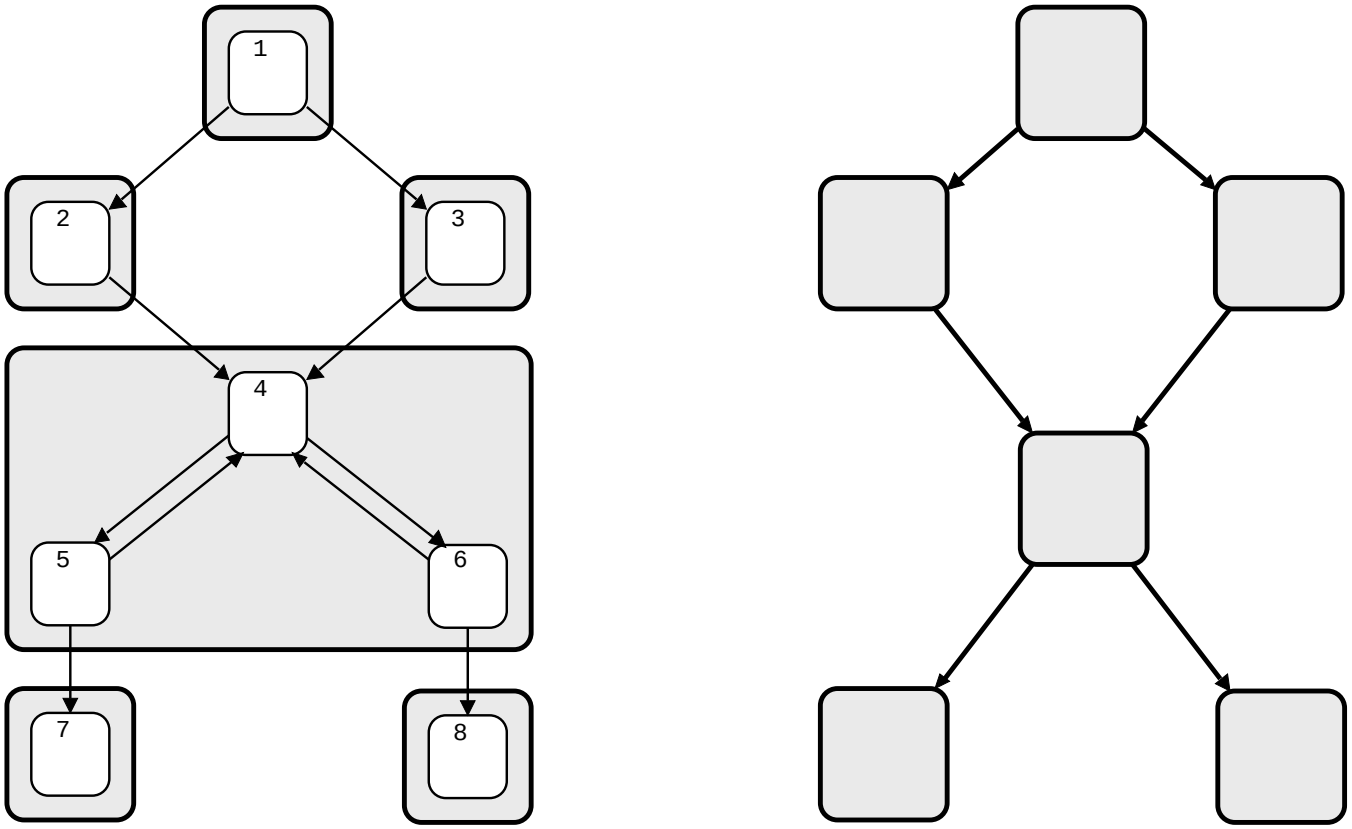
- A Coloured Petri Net (CP-net or CPN) is basically a Petri Net where
  - tokens can denote arbitrary values (colours),
  - places are typed (colour sets).

State Spaces



- A state space (occurrence graph, reachability graph/tree) is a representation of an exhaustive simulation of a given CP-net.
  - Nodes represent markings,
  - and edges represent transition occurrences.

SCC-graphs

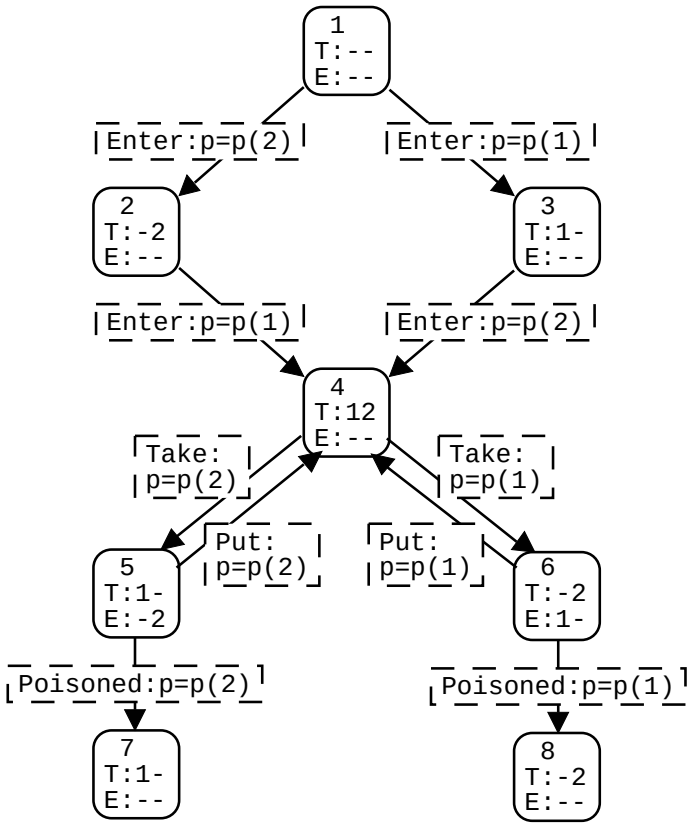


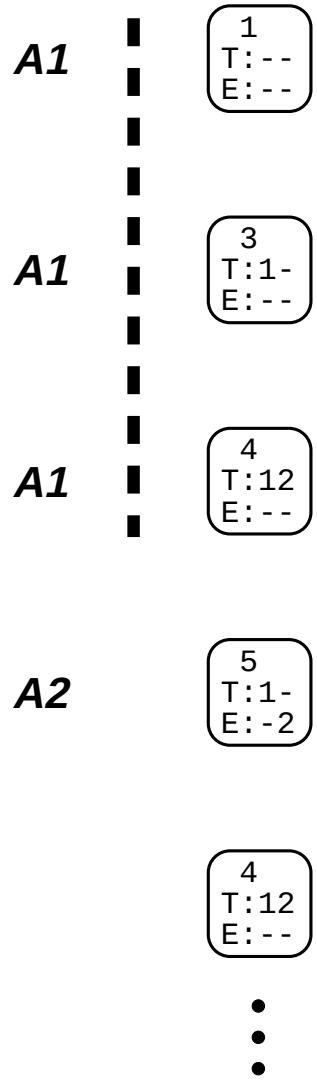
- A Strongly Connected Component (SCC) of a state space is a maximal set of nodes, where each node is reachable from any other node in the component.

- An SCC-graph is the SCC partitioning of a given state space, where edges reflect transition occurrences between components.

Temporal Logics

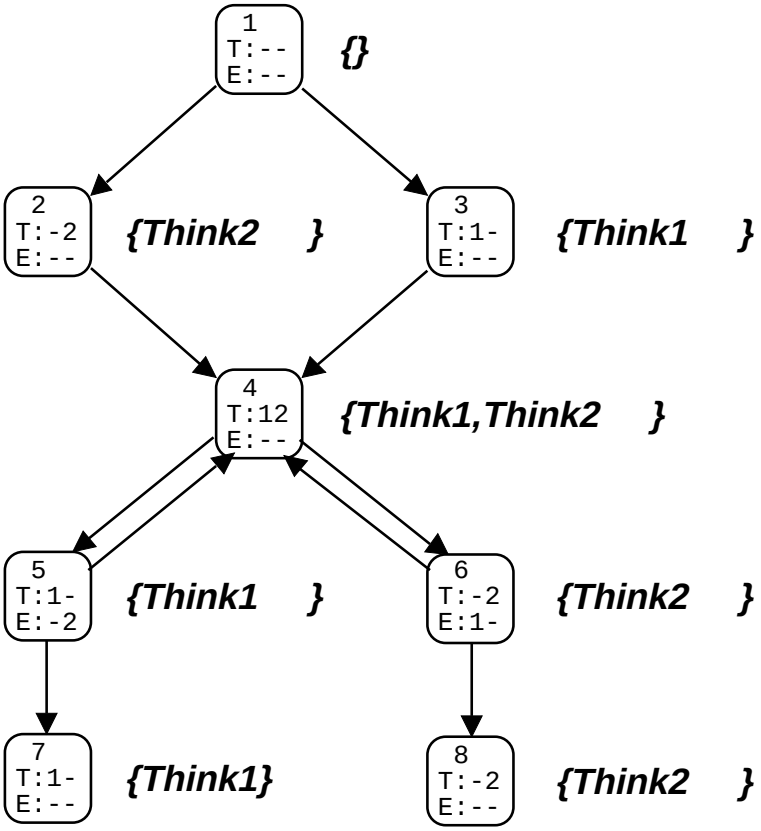
- We consider the branching time logic called CTL (by Clarke et al.).
- A temporal logic for reasoning about structures that are very similar to our state spaces.
- The subset of CTL formulas considered here is the *until* path quantification formulas $EU(\mathcal{A}_1, \mathcal{A}_2)$ and $AU(\mathcal{A}_1, \mathcal{A}_2)$.





Model Checking

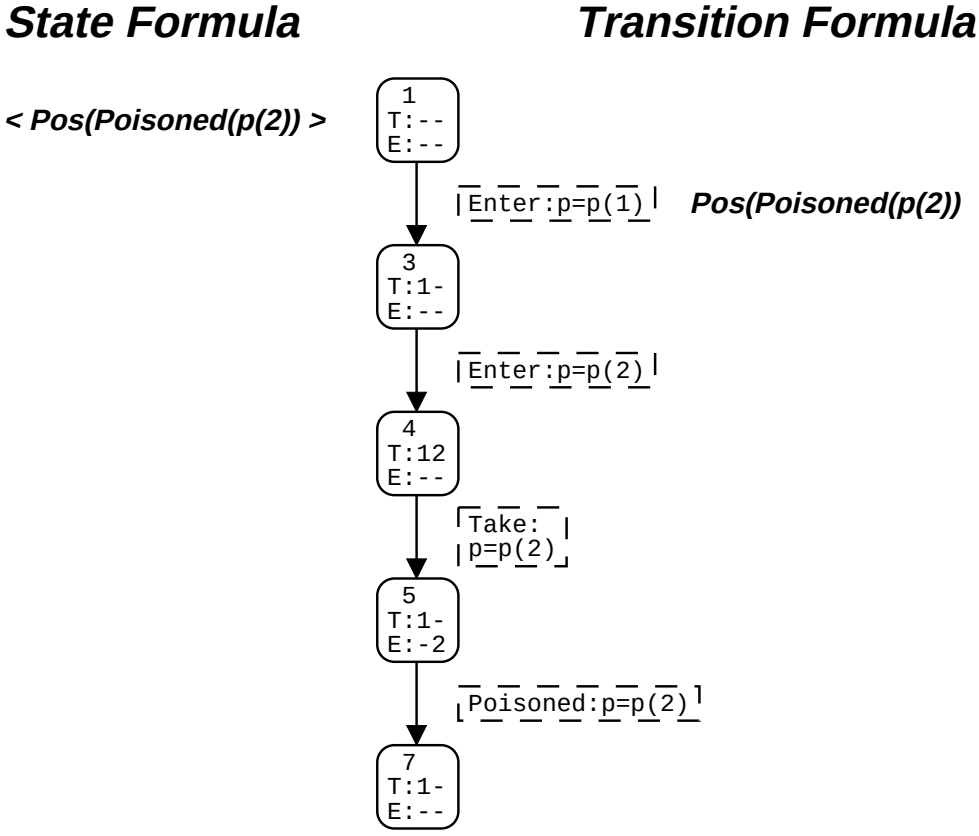
EU(Think1,Think2) ?



- Model checking is the act of checking the truth value of a given CTL formula for a given state space.
- When the algorithm terminates: The formula in question is true in every state of which it is also a label.
- Linear time complexity in the size of the formula times the size of the state space.

Extension of CTL

- Since our state spaces contain both information on nodes and edges, it is natural to consider an extension to CTL such that one can express properties about both state and transition information.
- We have studied an operator which supports our needs: An operator which switches between the two domains of states and transitions: $<\mathcal{B}>$
- As a consequence it makes sense to talk about both paths of nodes and paths of edges.



Special Cases

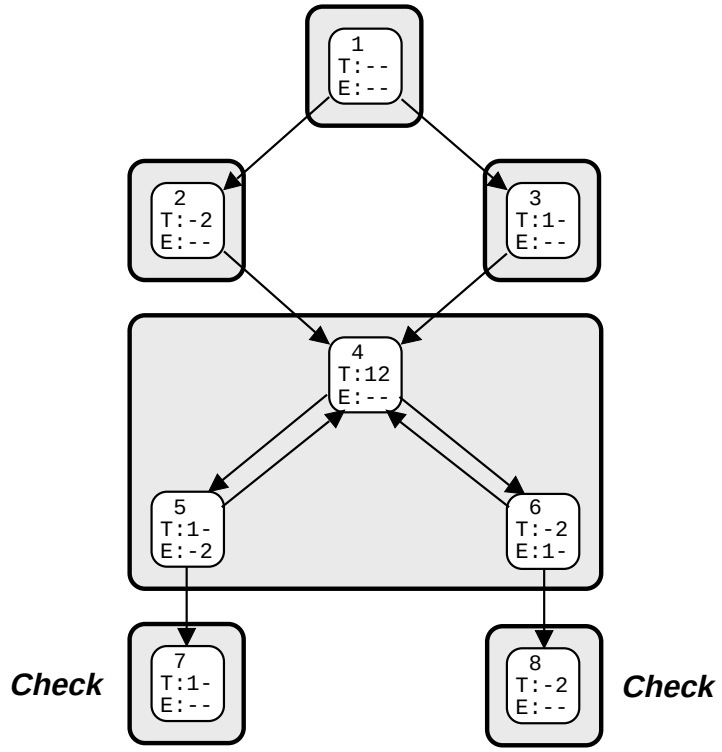
- Convenient notation (standard CTL):
  - $EU(tt, \mathcal{A}) \equiv Pos(\mathcal{A})$
  - $\neg EU(tt, \neg\mathcal{A}) \equiv Inv(\mathcal{A})$
  - $AU(tt, \mathcal{A}) \equiv Ev(\mathcal{A})$
  - $\neg AU(tt, \neg\mathcal{A}) \equiv Along(\mathcal{A})$

- We have studied efficient model check for special case combinations of CTL formulas.
  - $Pos(Inv(\mathcal{A}))$
  - $Ev(Inv(\mathcal{A}))$
  - $Pos(Along(\mathcal{A}))$
  - $Ev(Along(\mathcal{A}))$

Improved Model Checking

- The case: $Pos(Inv(\mathcal{A}))$
  - Consider its negation: $Inv(Pos(\mathcal{A}'))$
  - The same as the home space property — a standard Petri net property.
  - A set of states, X, is a home space iff the terminal SCCs are a subset of the SCCs which contain elements from X.
  - Thus in the model checking algorithm it is sufficient to search for elements of X in terminal SCCs only — when the special case above is encountered.

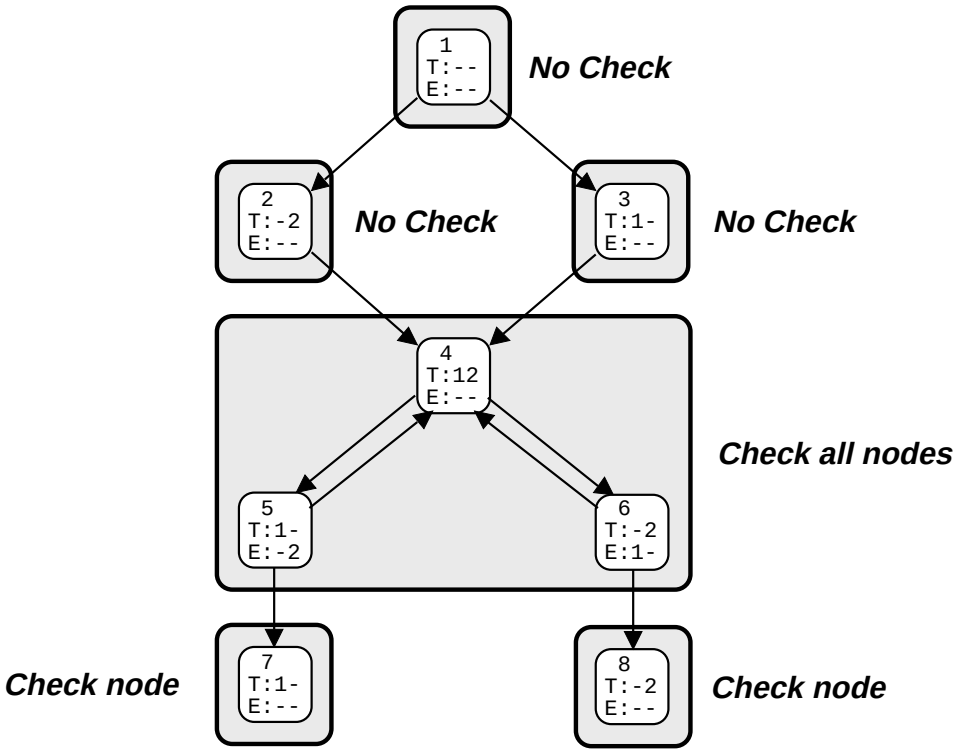
Inv(Pos(NoAvailChopsticks) ?



Improved Model Checking

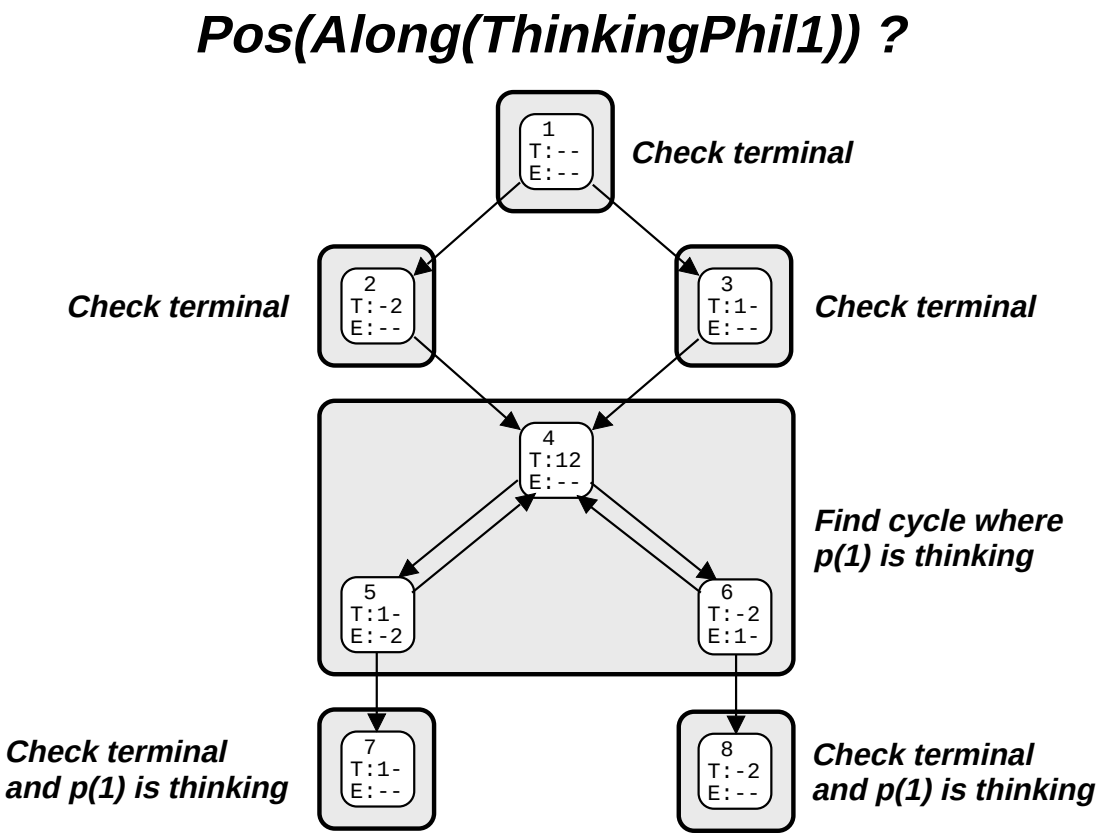
- The case: $Ev(Inv(\mathcal{A}))$
  - For all reachable SCC components, c, check:
  - If c is trivial then
  - If c is terminal then check $\mathcal{A}$
  - Else No check; skip to next component
  - Else
  - Check all nodes in c with $\mathcal{A}$

Ev(Inv(AllEatingOrThinking)) ?



Improved Model Checking

- The case: $Pos(Along(\mathcal{A}))$
  - There exists a reachable component, c, such that:
  - If c is trivial then
  - c is terminal and check it with $\mathcal{A}$
  - Else
  - c contains a $\mathcal{A}$ cycle



Improved Model Checking

- The case: $Ev(Along(\mathcal{A}))$

- Unfortunately it seems that the model checking algorithm cannot take advantage of the SCC-graph for this special case.

- The combination of universal path quantification together with the necessary existence of a path seems to cause the problem.

Summary

- We have developed an extension of CTL, the logic ASK-CTL, which makes it possible to reason about both states and transition occurrence information.

- We have improved the model checking algorithm by taking advantage of the SCC-graph information. Good performance even when the SCC-graph only has one SCC.

- We have made an implementation of ASK-CTL and the improved model checker in the CP-net tool Design/CPN. Experiments show performance improvements of factors in the range 2.4 to 1300.

Online Information

- http://www.daimi.aau.dk/designCPN/
  Design/CPN — free of charge computer tool for Coloured Petri nets

- http://www.daimi.aau.dk/PetriNets/
  Main resource for general information about Petri Nets