CIS 721 - Real-Time Systems

Lecture 31: Advanced UPPAAL Models

Mitch Neilsen
neilsen@ksu.edu

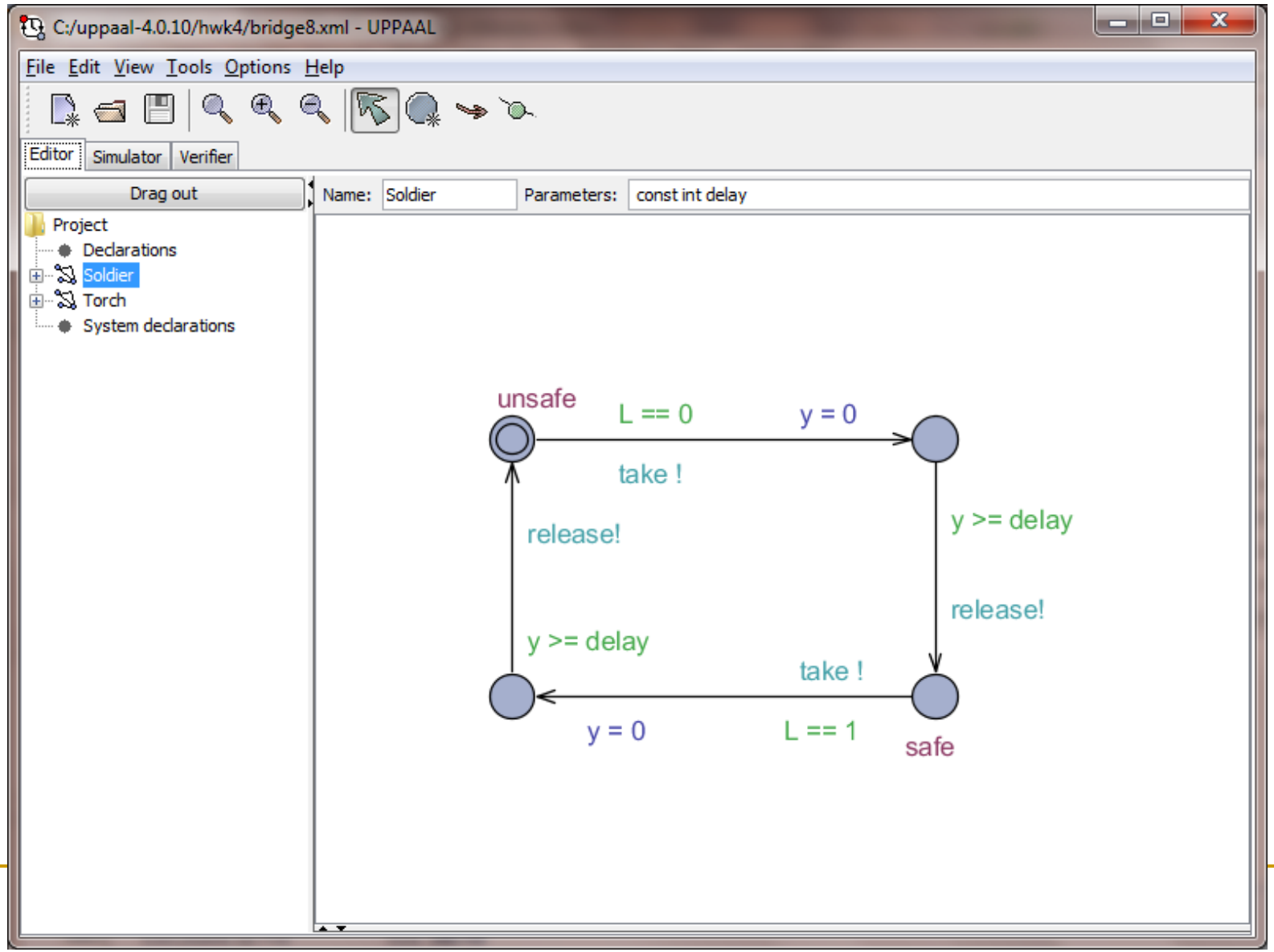
Uppaal Models

- Vikings' Bridge Problem
- Gossiping Girls' Problem
- Prioritized Token-Based Mutual Exclusion Algorithms

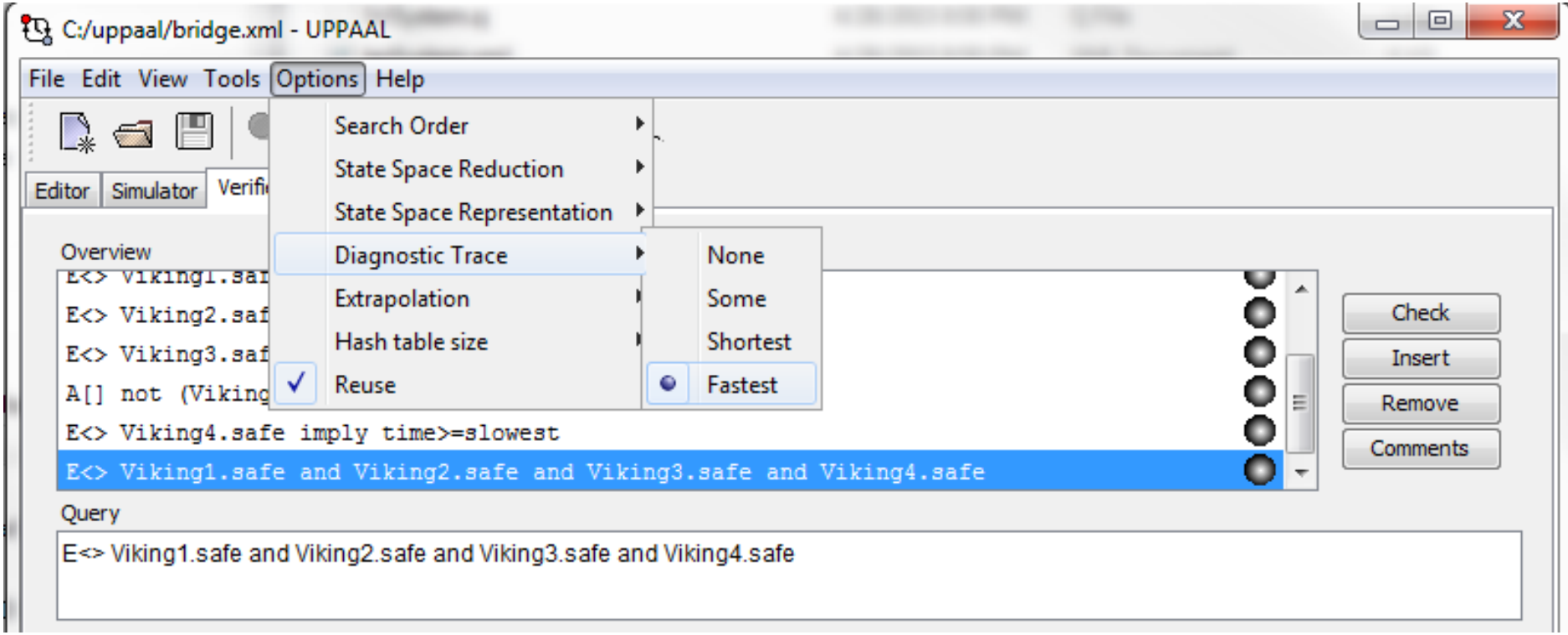
Vikings' Bridge Problem

- Adjust the solution to the Vikings' Problem given in bridge.xml to handle four vikings with crossing times of 3, 6, 12, and 15. Determine the minimum time required for all four vikings to cross the bridge.
- Repeat the problem, but vary the time required for the third viking (currently 12); e.g., try 3, 6, 9, and 15, etc.
- What range of values require the same amount of time as the first run with crossing times of 3, 6, 12, and 15?
- Finally, develop a general solution (on paper) for 4 vikings with crossing times of k, 2*k, 4*k, and 5*k, for any fixed k > 0.

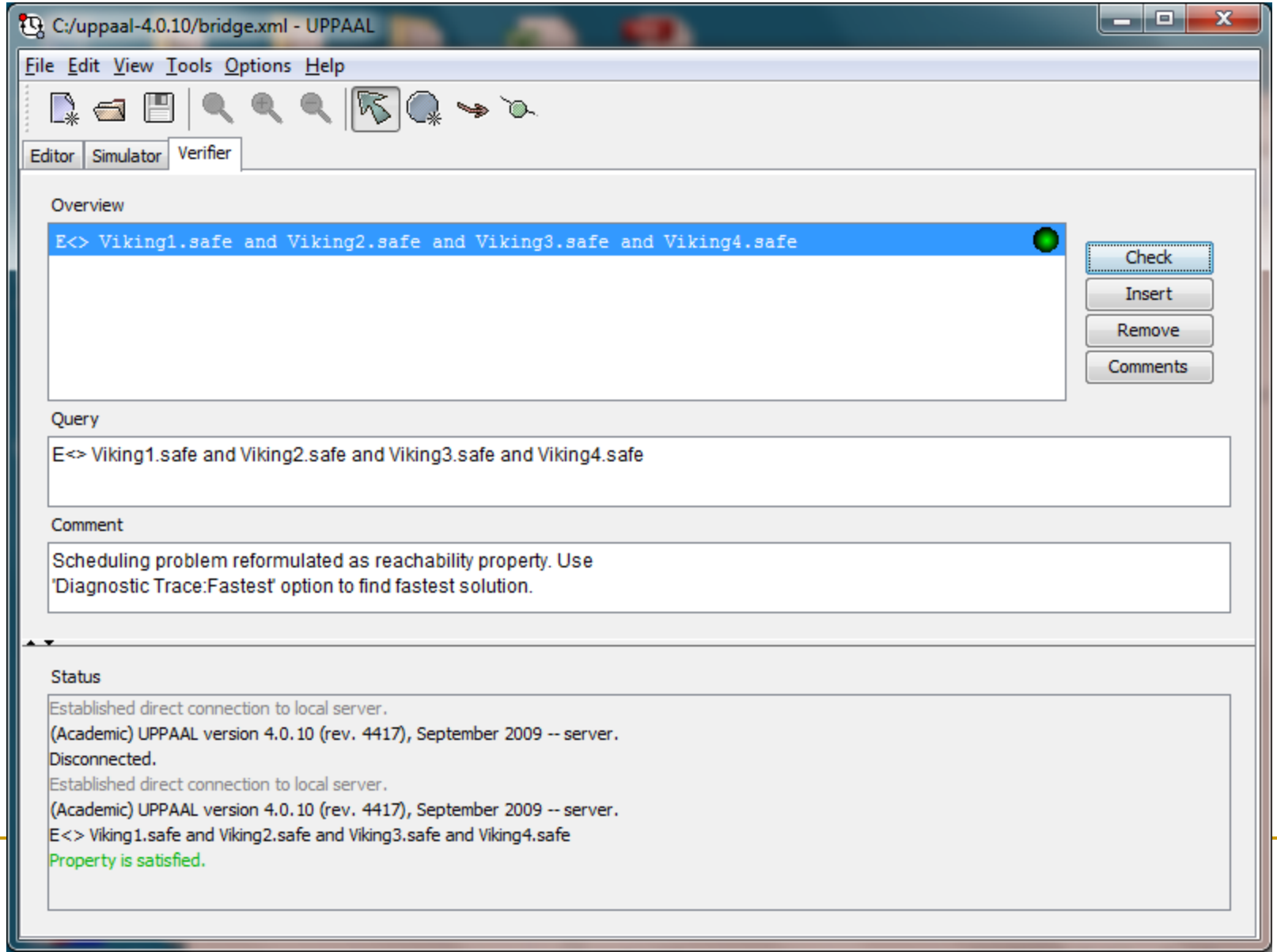
Vikings' Bridge Problem



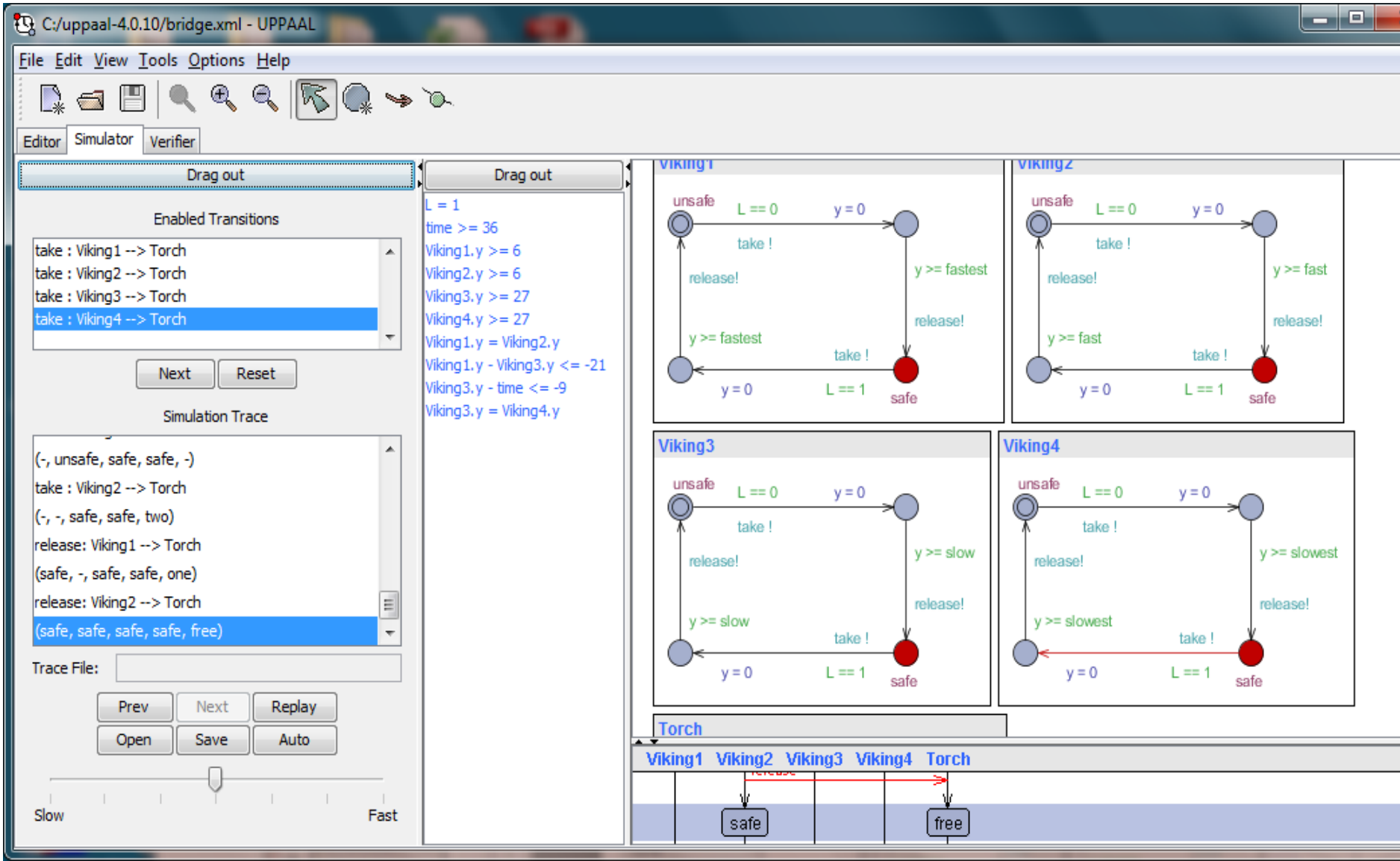
Diagnostic Trace + Fastest



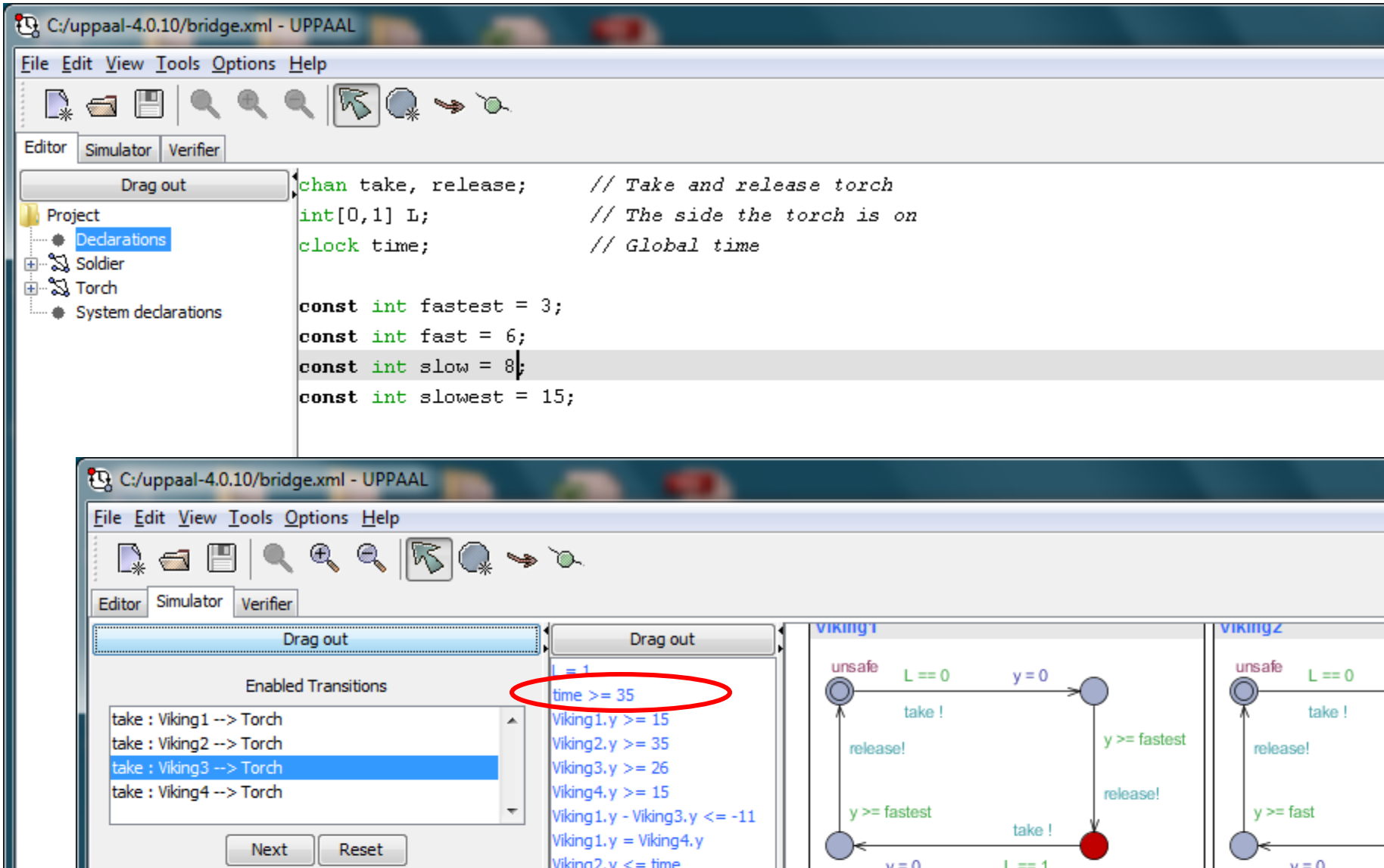
Vikings' Bridge Problem



Vikings' Bridge Problem



Vikings' Bridge Problem



Vikings' Bridge Problem

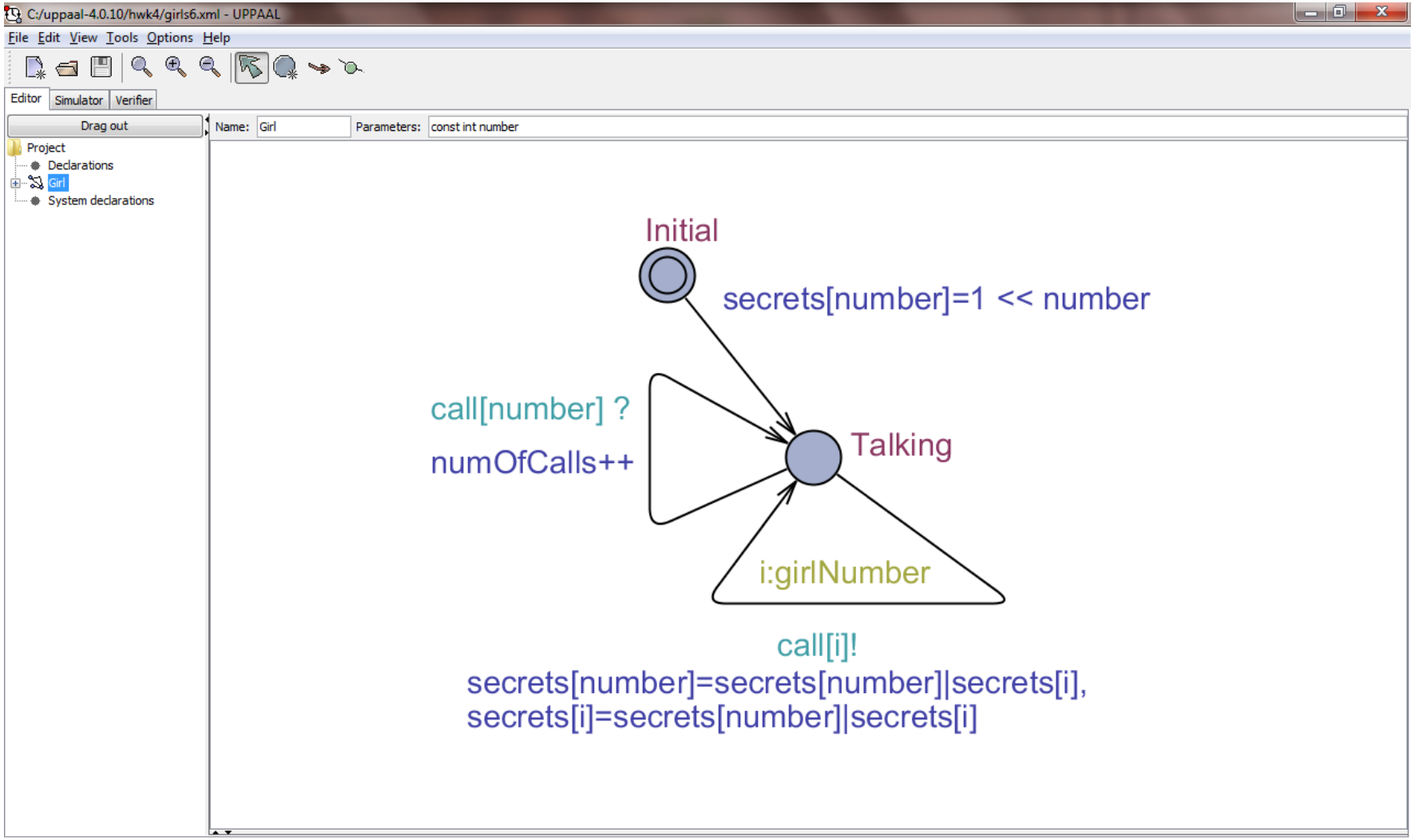
- Values from 9 to 15 for the third viking require the same time, 36, for all four to cross.
- Generalize:
  - k=3: 3, 6, 12, 15 => 36 crossing time
  - k=4: 4, 8, 16, 20 => 48 crossing time
  - k=5: 5, 10, 20, 25 => 60 crossing time

So, total time = 12*k, for crossing times of k, 2*k, 4*k, 5*k

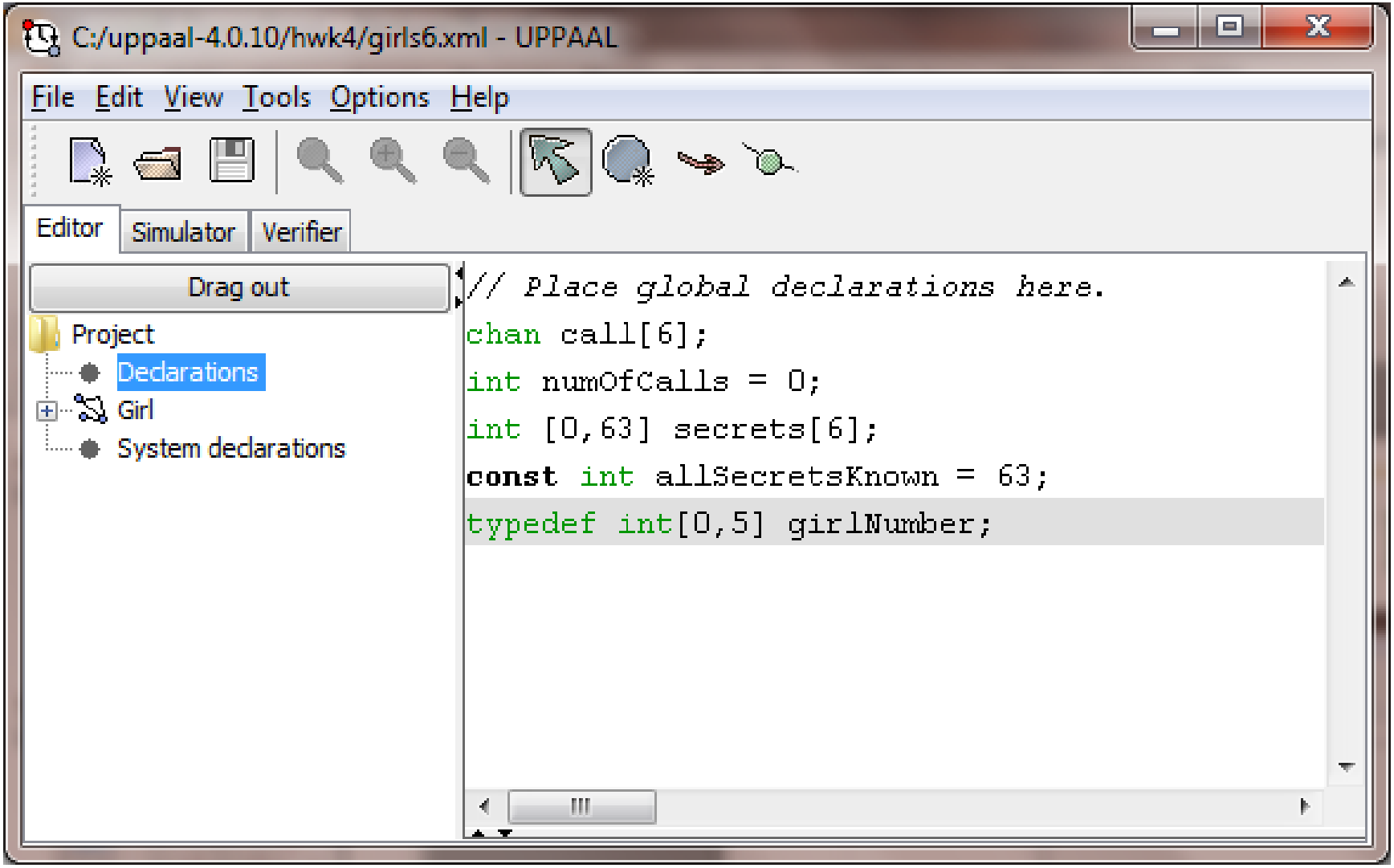
Gossiping Girls' Problem

- Model and analyze the following gossiping girls problem in UPPAAL. A number of girls initially know one distinct secret each. Each pair of girls has access to a phone line that can be used to share their secrets. Each time two girls talk to each other they always exchange all secrets that they currently know with each other (thus after the phone call they both know all secrets they knew together before the phone call). Only binary communication (between two girls in each phone call) is possible.
- (a) Use UPPAAL to find the minimal number of phone calls needed for **two**, **three**, or **four** girls to know all secrets.
- (b) Analytically determine how many phone calls are needed to solve the gossiping girls' problem for *n* girls. Hint: Use a recursive or inductive approach. Consider the solution for *n = 2, n = 3,* etc. Then, write a proof by induction.

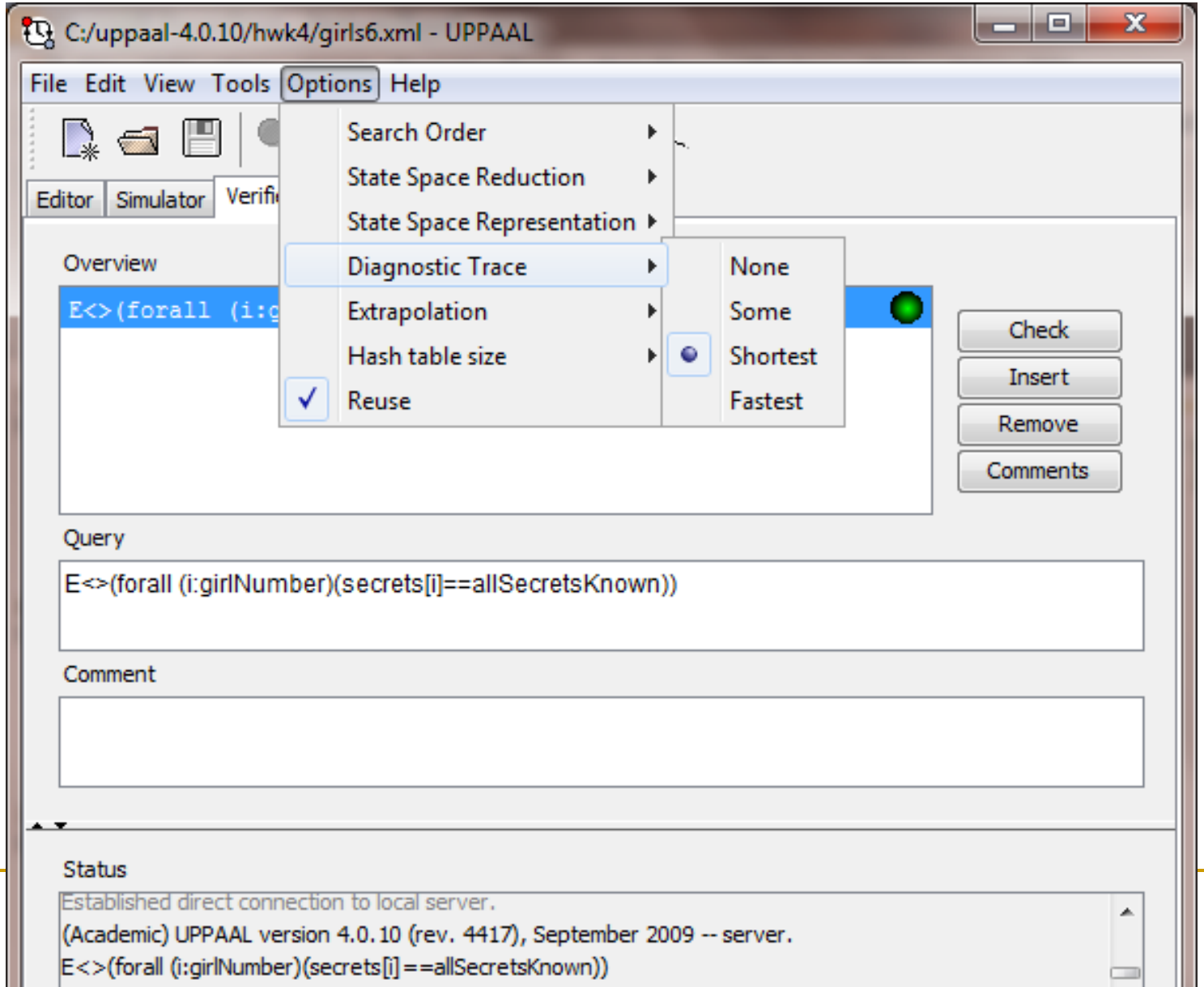
Gossiping Girls' Problem



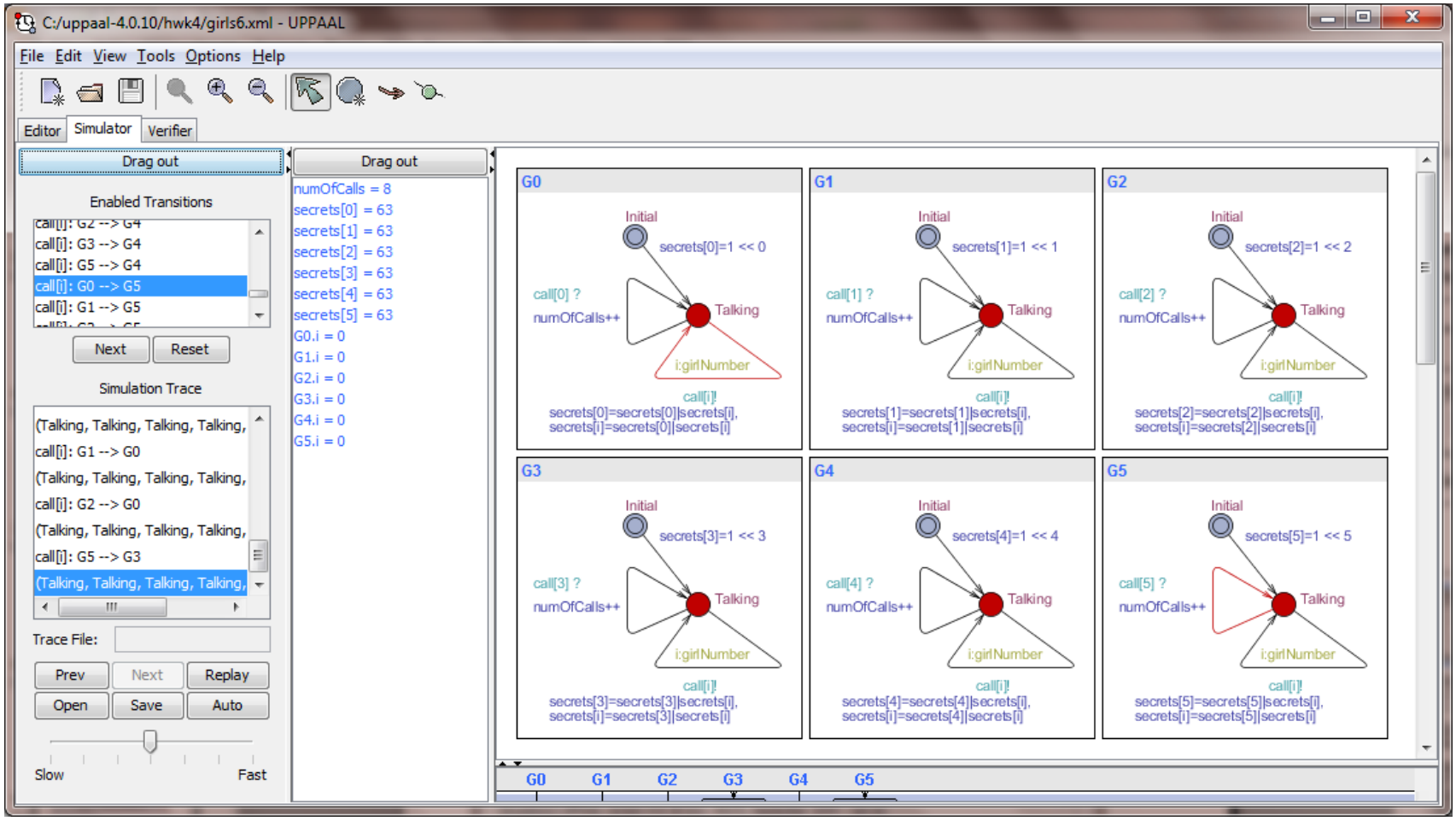
Gossiping Girls' Problem



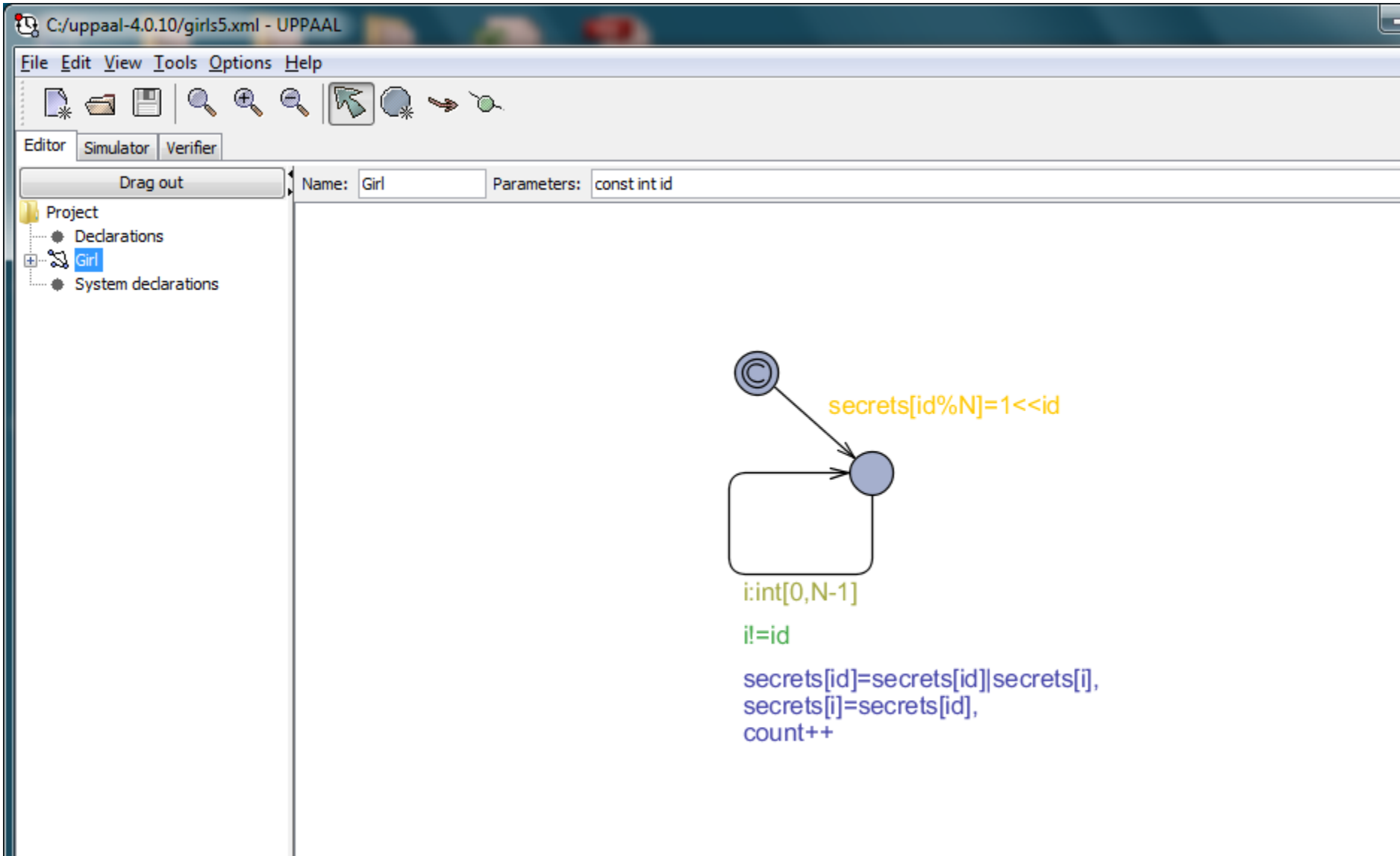
Gossiping Girls' Problem



Gossiping Girls' Problem

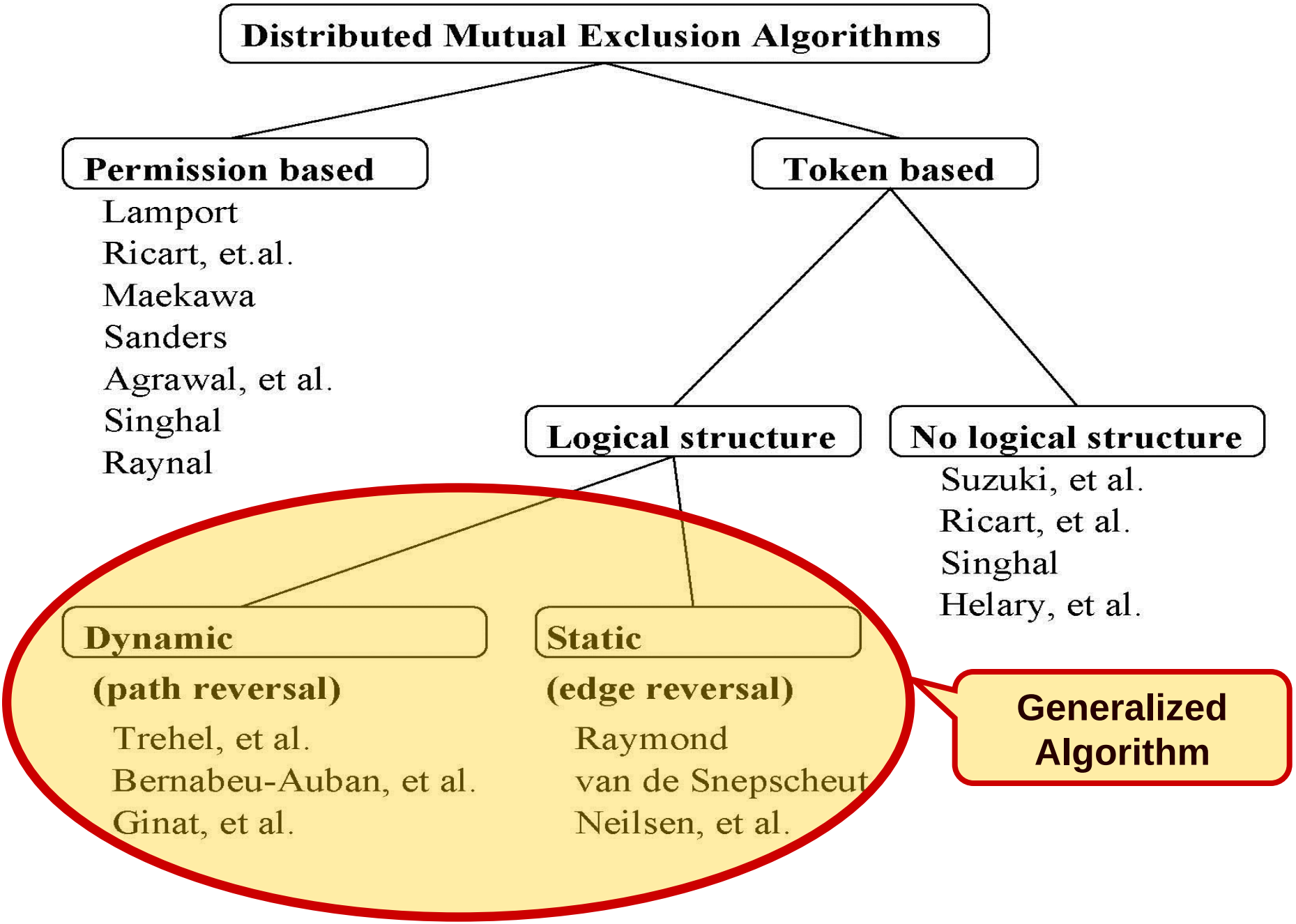


Gossiping Girls' Problem

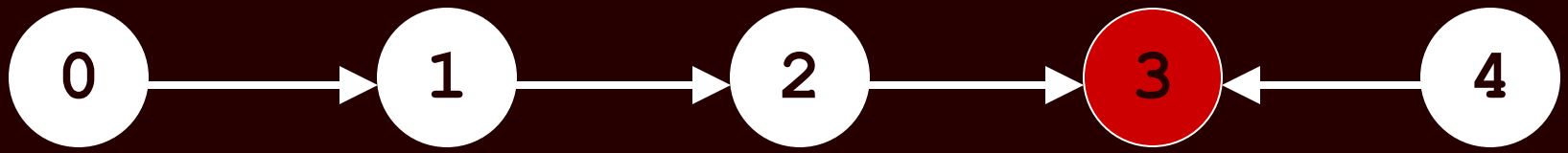


Gossiping Girls' Problem

- Special case, n = 4: 4 calls required.
- Other cases:
  - n = 2 => 1 call
  - n = 3 => 3 calls
  - n = 4 => 4 calls
  - n = 5 => 6 calls
  - n = 6 => 8 calls
  - n = 7 => 10 calls
  - n = 8 => 12 calls
  - For (n >= 4), 2*n – 4 calls


Distributed Mutual Exclusion Algorithms
Permission based
Lamport
Ricart, et.al.
Maekawa
Sanders
Agrawal, et al.
Singhal
Raynal
Token based
Logical structure
No logical structure
Suzuki, et al.
Ricart, et al.
Singhal
Helary, et al.
Dynamic
(path reversal)
Trehel, et al.
Bernabeu-Auban, et al.
Ginat, et al.
Static
(edge reversal)
Raymond
van de Snepscheut
Neilsen, et al.
Generalized Algorithm

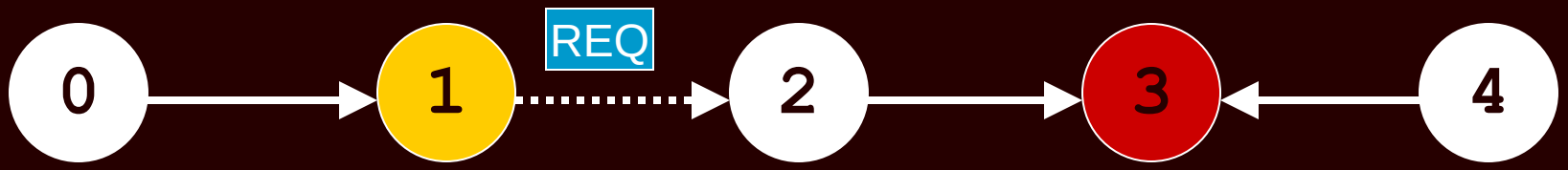
Dynamic Logical Structure – Path Reversal



= token holder

= requesting token

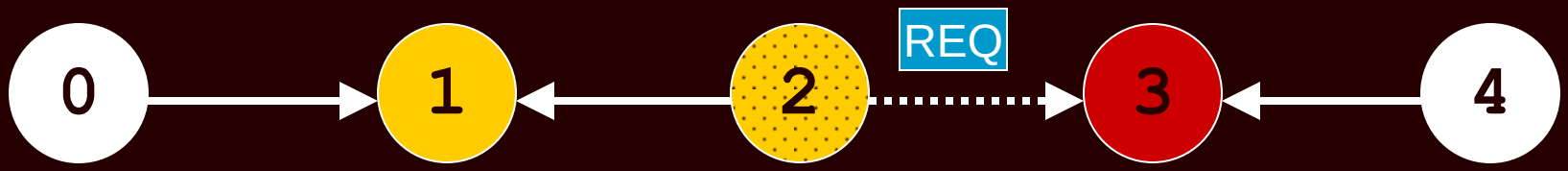
Dynamic Logical Structure – Path Reversal



= token holder

= requesting token

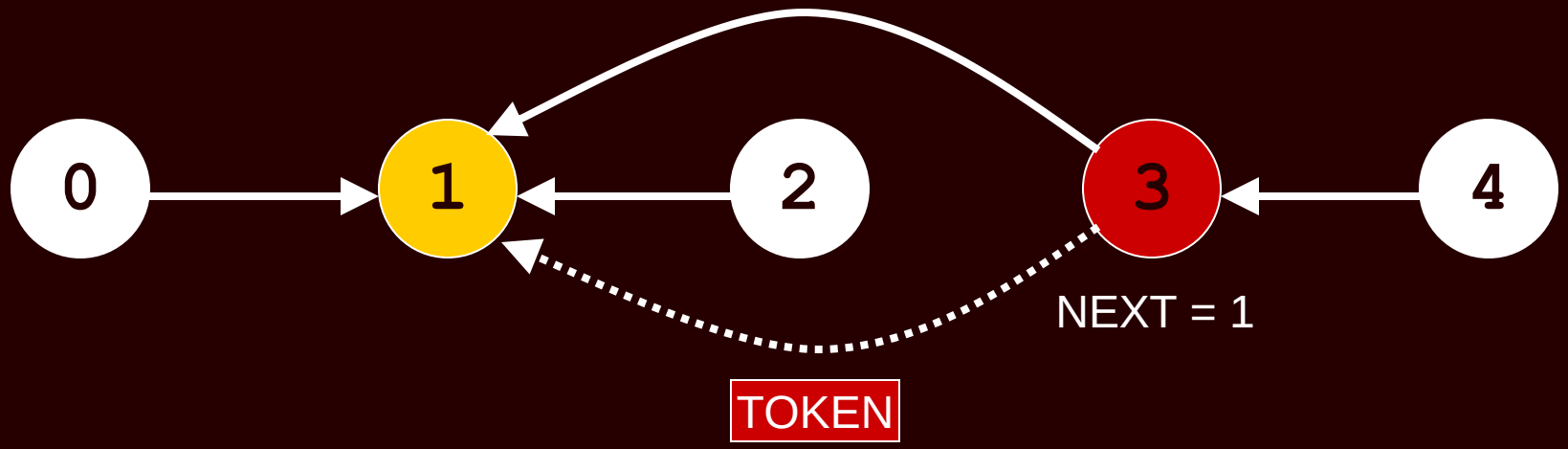
Dynamic Logical Structure – Path Reversal



= token holder

= requesting token

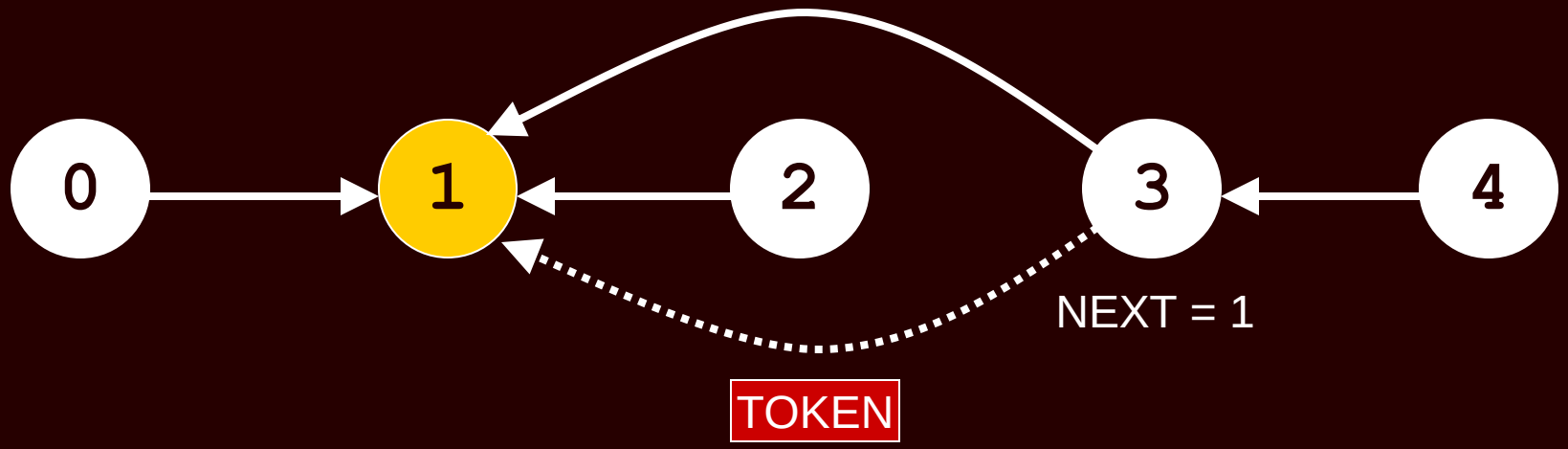
Dynamic Logical Structure – Path Reversal



= token holder

= requesting token

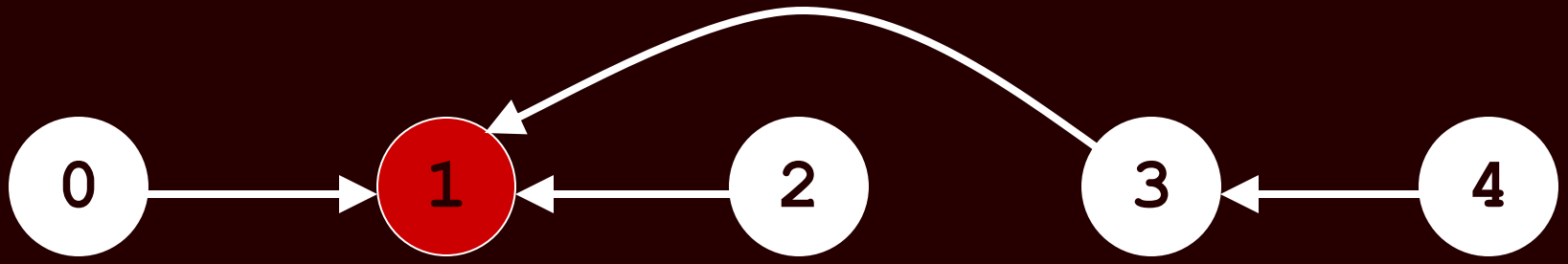
Dynamic Logical Structure – Path Reversal



= token holder

= requesting token

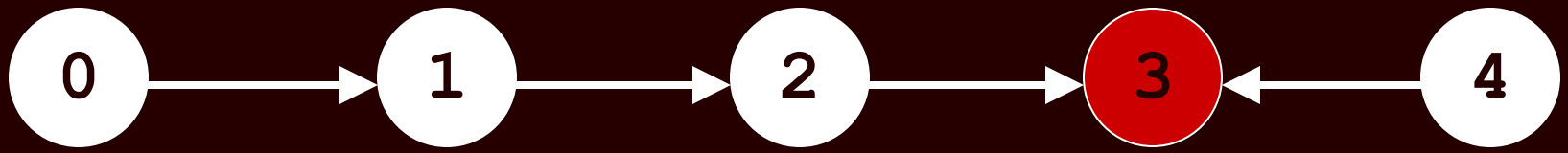
Dynamic Logical Structure – Path Reversal



= token holder

= requesting token

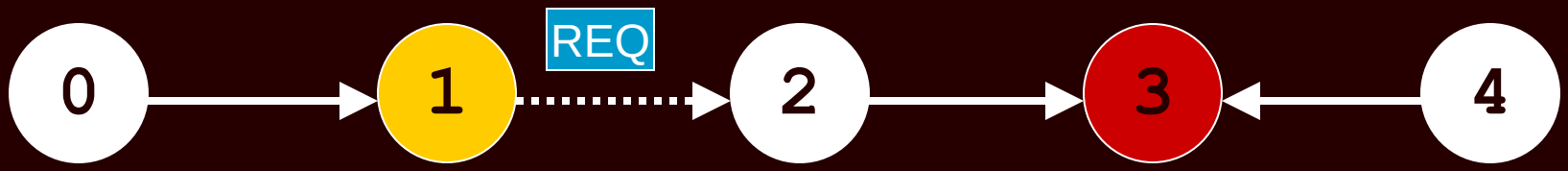
Static Logical Structure – Edge Reversal



= token holder

= requesting token

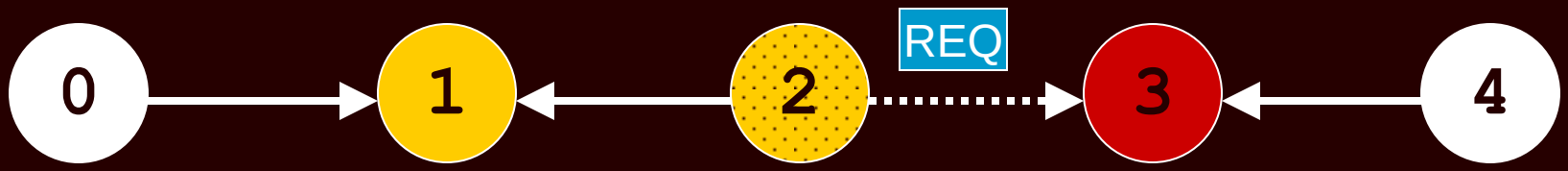
Static Logical Structure – Edge Reversal



= token holder

= requesting token

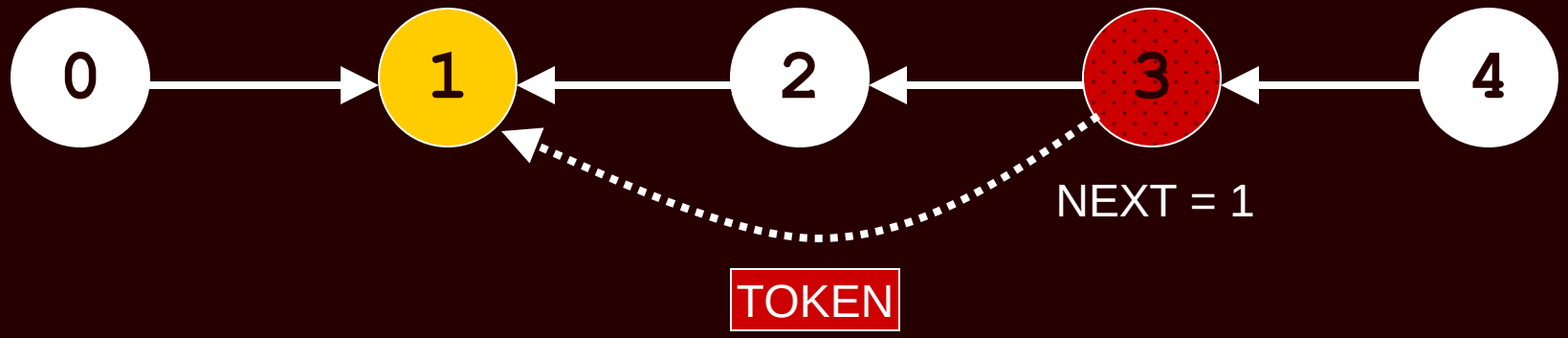
Static Logical Structure – Edge Reversal



= token holder

= requesting token

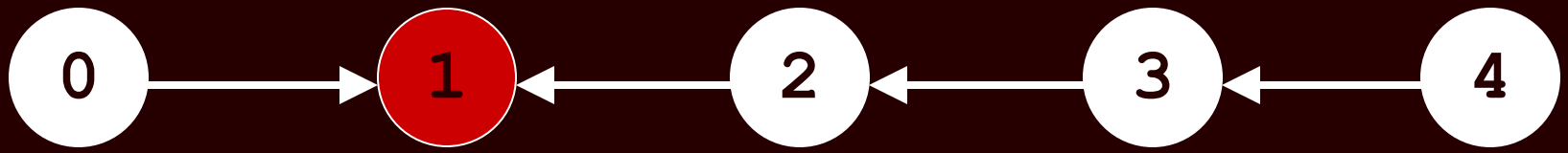
Static Logical Structure – Edge Reversal



= token holder

= requesting token

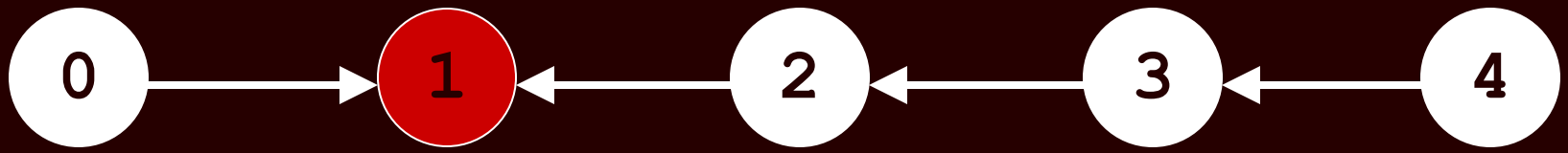
Static Logical Structure – Edge Reversal



= token holder

= requesting token

Static Logical Structure – Edge Reversal

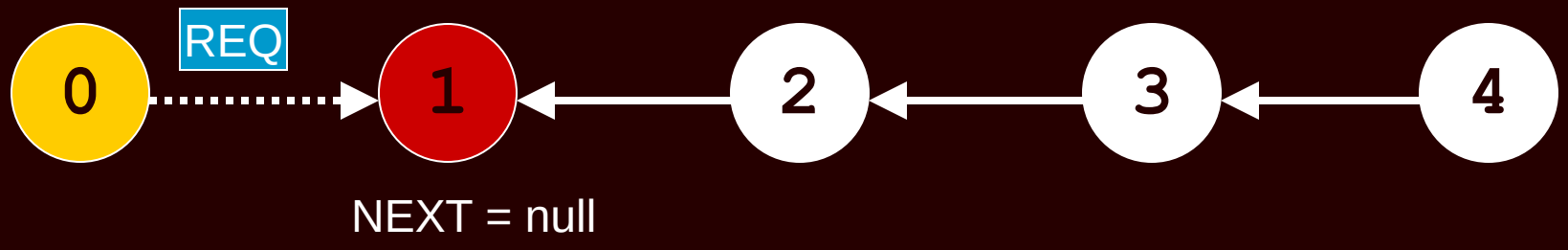


= token holder

= requesting token

Logical Wait Queue

Static Logical Structure – Edge Reversal

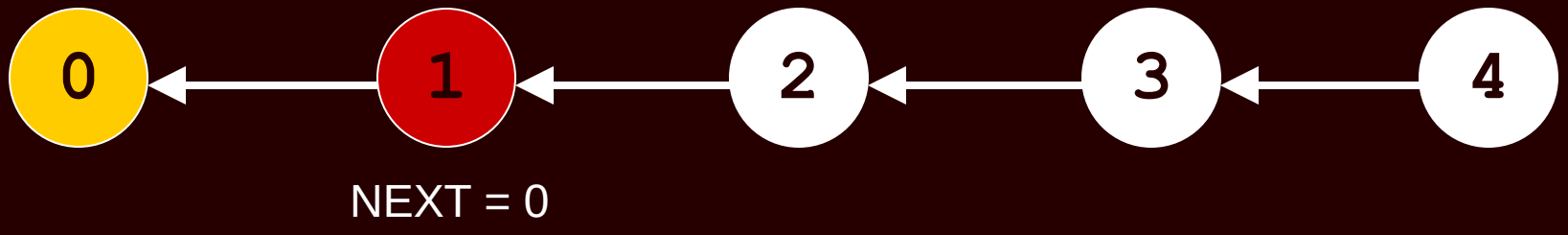


= token holder

= requesting token

Logical Wait Queue

Static Logical Structure – Edge Reversal

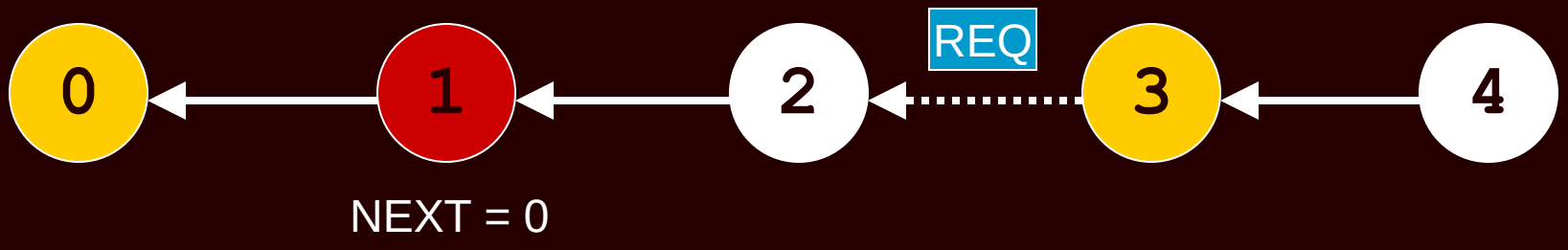


= token holder

= requesting token

Logical Wait Queue

Static Logical Structure – Edge Reversal

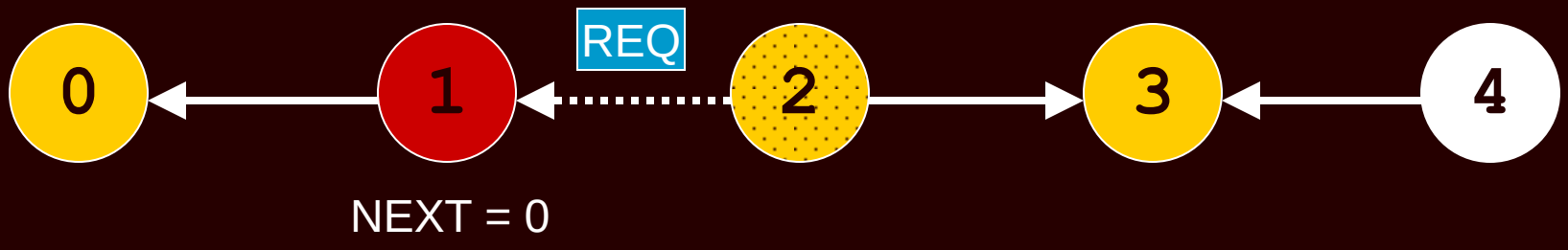


= token holder

= requesting token

Logical Wait Queue

Static Logical Structure – Edge Reversal

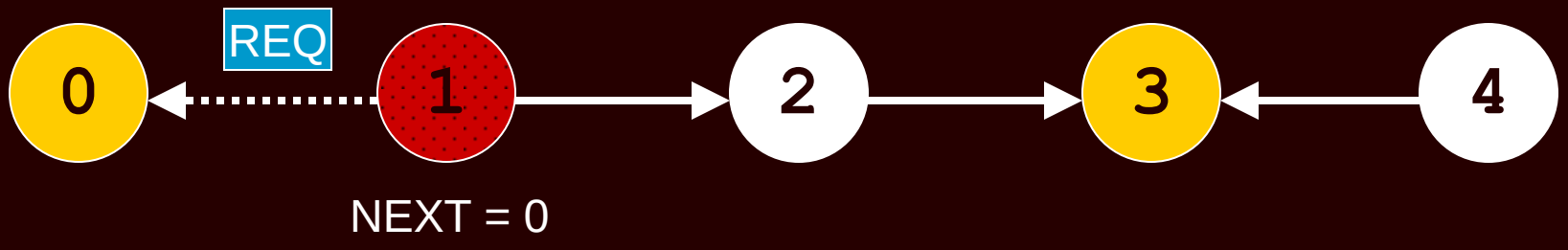


= token holder

= requesting token

Logical Wait Queue

Static Logical Structure – Edge Reversal

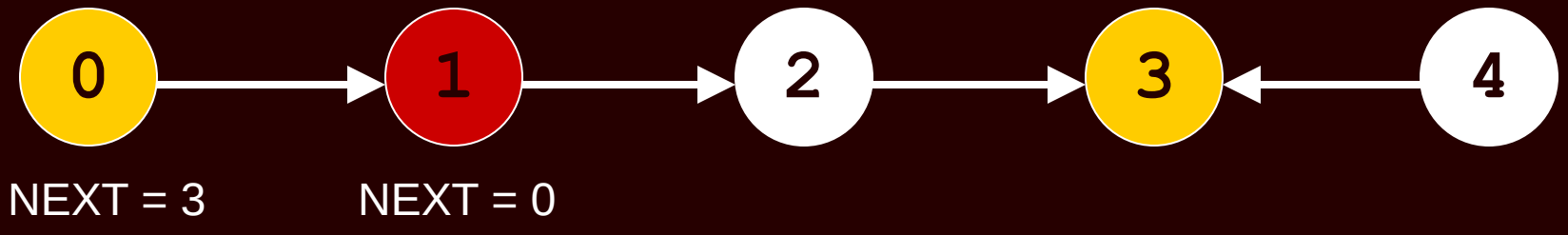


= token holder

= requesting token

Logical Wait Queue

Static Logical Structure – Edge Reversal

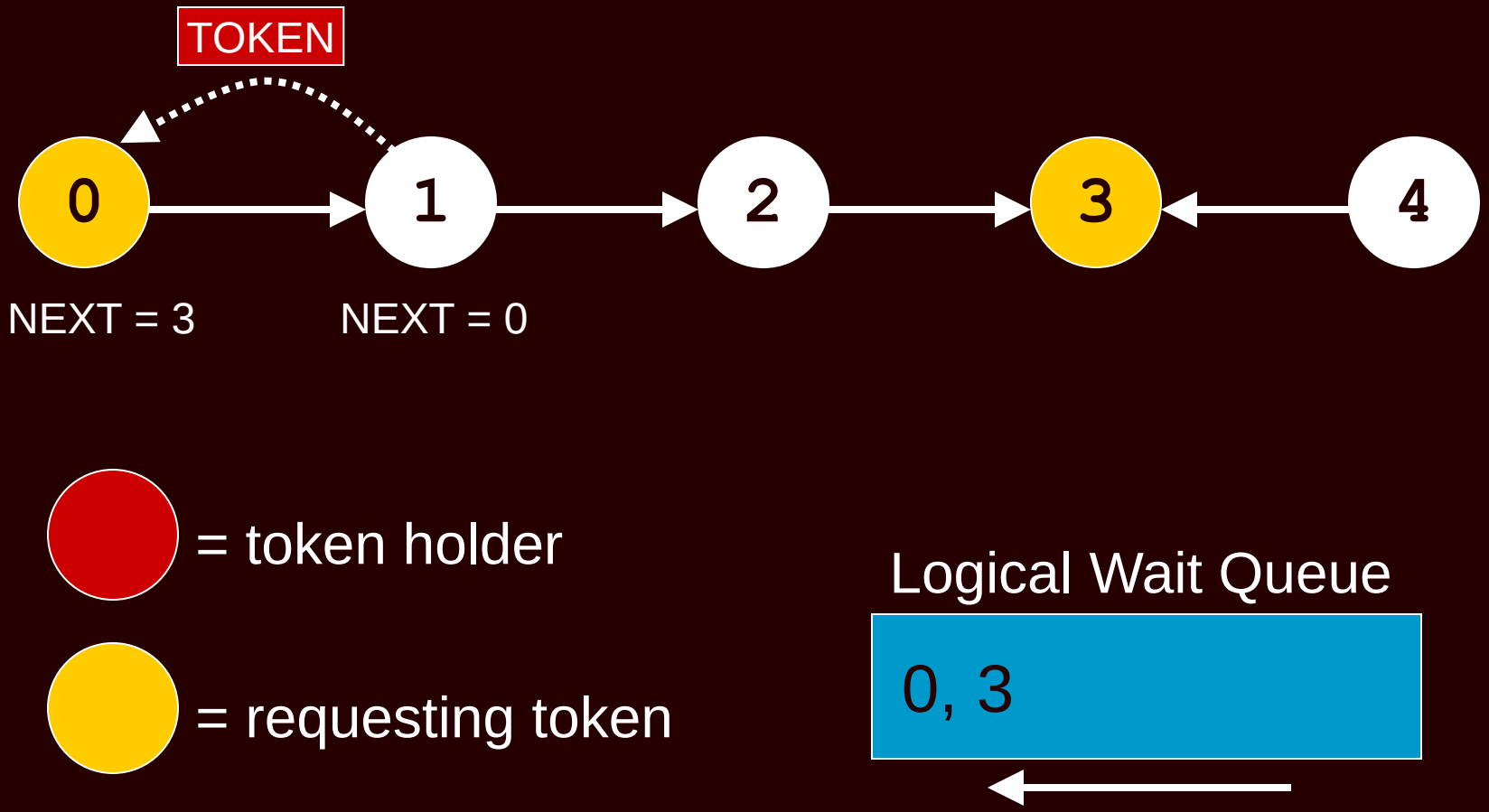


= token holder

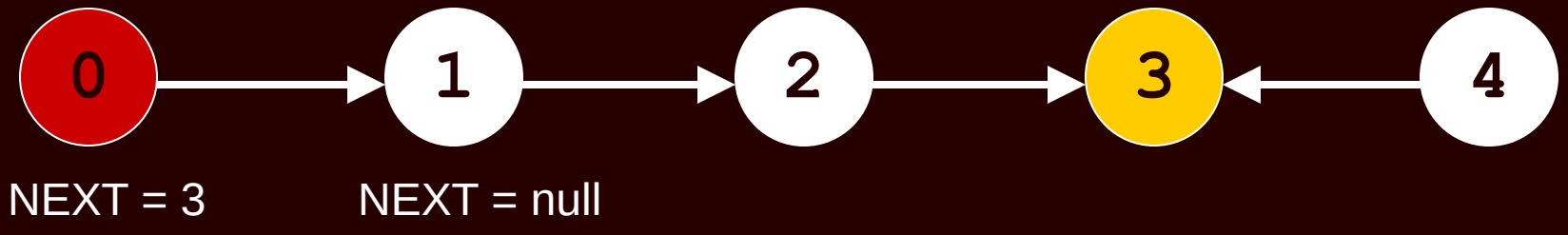
= requesting token

Logical Wait Queue

Static Logical Structure – Edge Reversal



Static Logical Structure – Edge Reversal

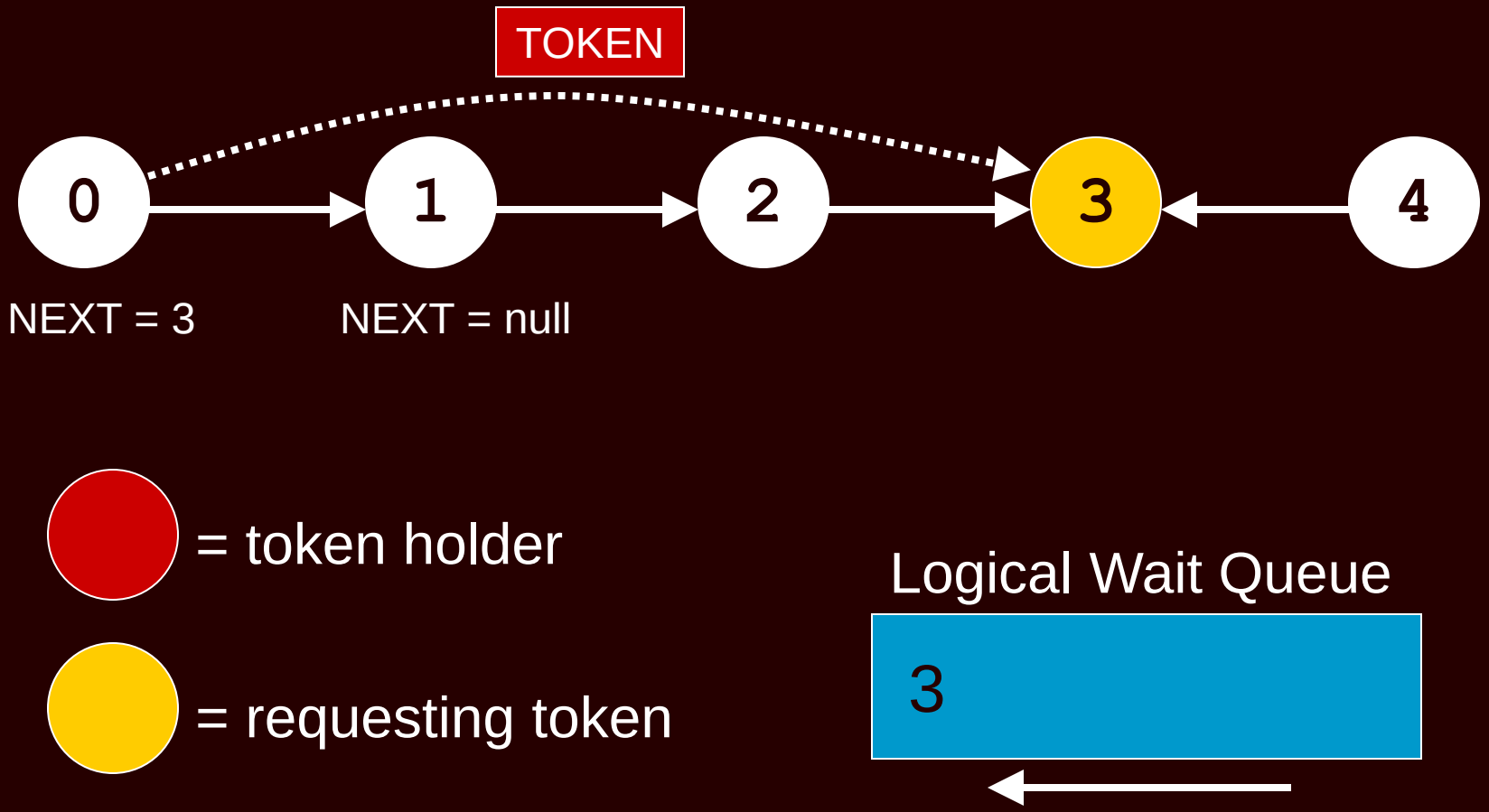


= token holder

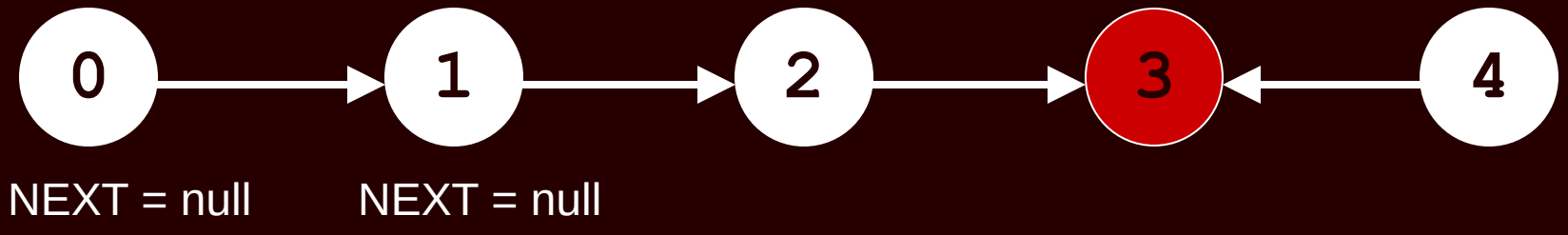
= requesting token

Logical Wait Queue

Static Logical Structure – Edge Reversal



Static Logical Structure – Edge Reversal

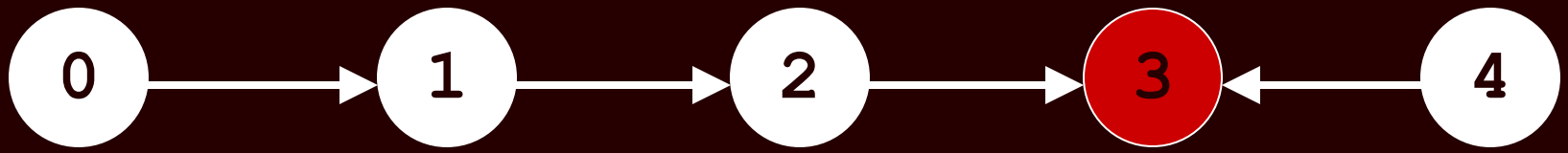


= token holder

= requesting token

Logical Wait Queue

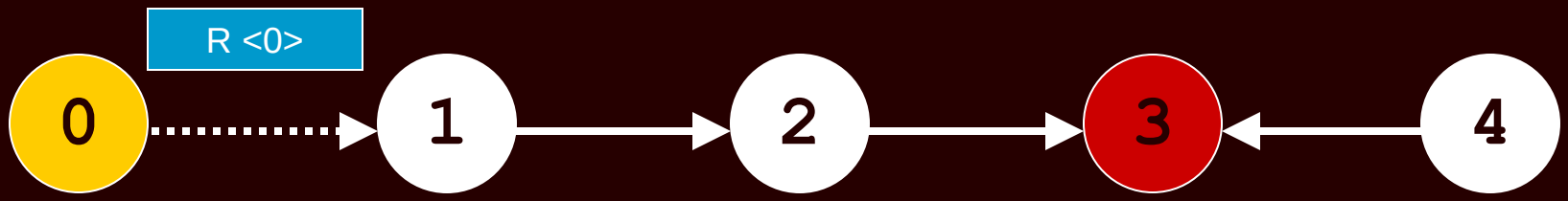
Generalized Algorithm: Edge+Path Reversal



= token holder

= requesting token

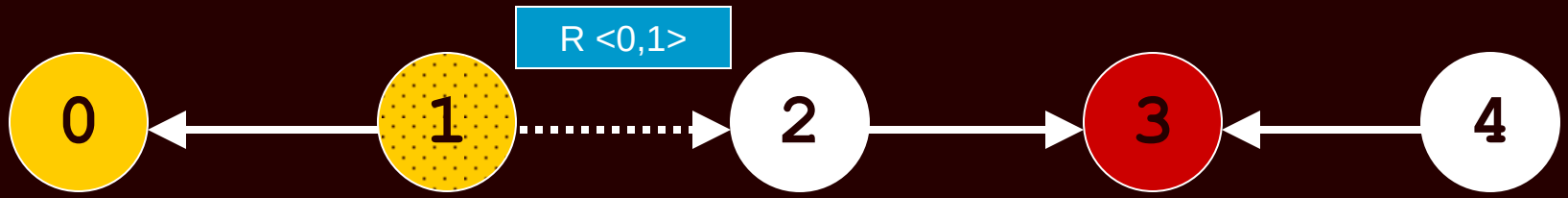
Generalized Algorithm: Edge+Path Reversal



= token holder

= requesting token

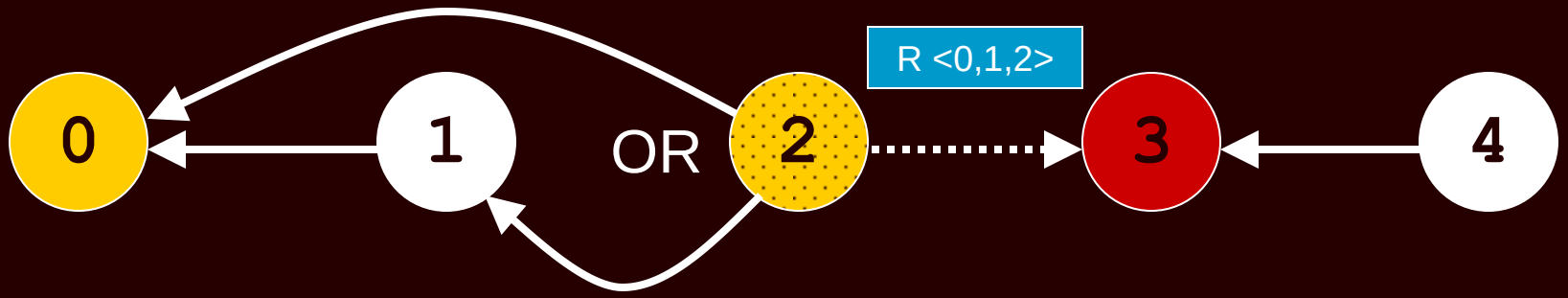
Generalized Algorithm: Edge+Path Reversal



= token holder

= requesting token

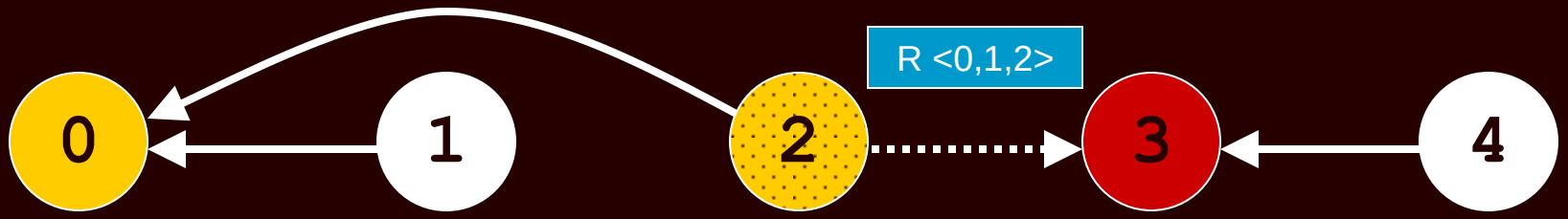
Generalized Algorithm: Edge+Path Reversal



= token holder

= requesting token

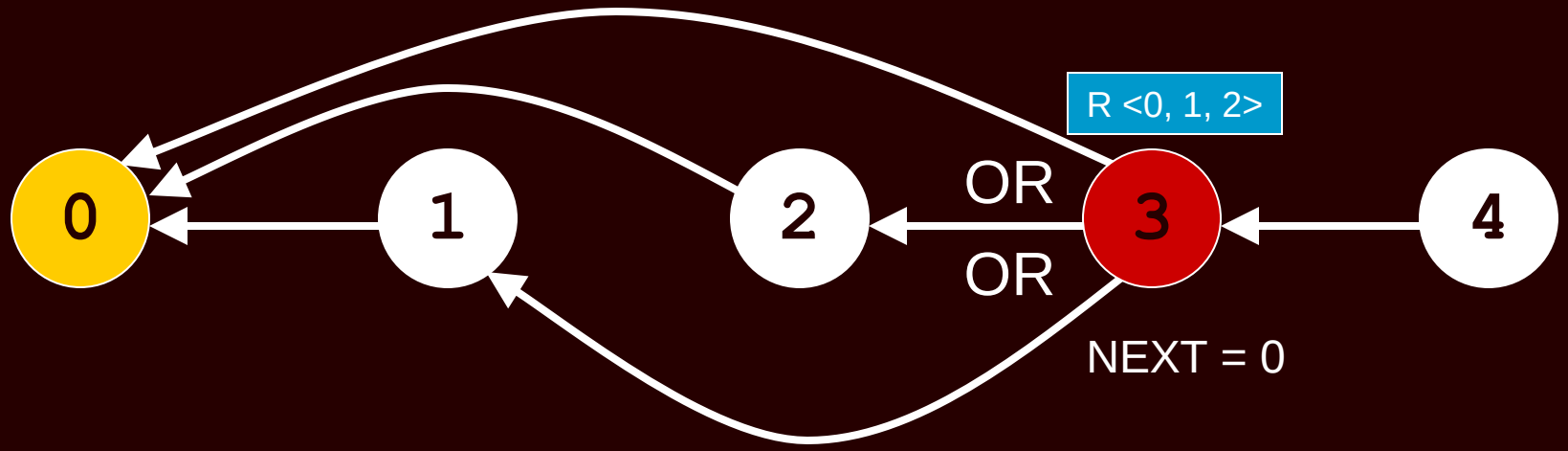
Generalized Algorithm: Edge+Path Reversal



= token holder

= requesting token

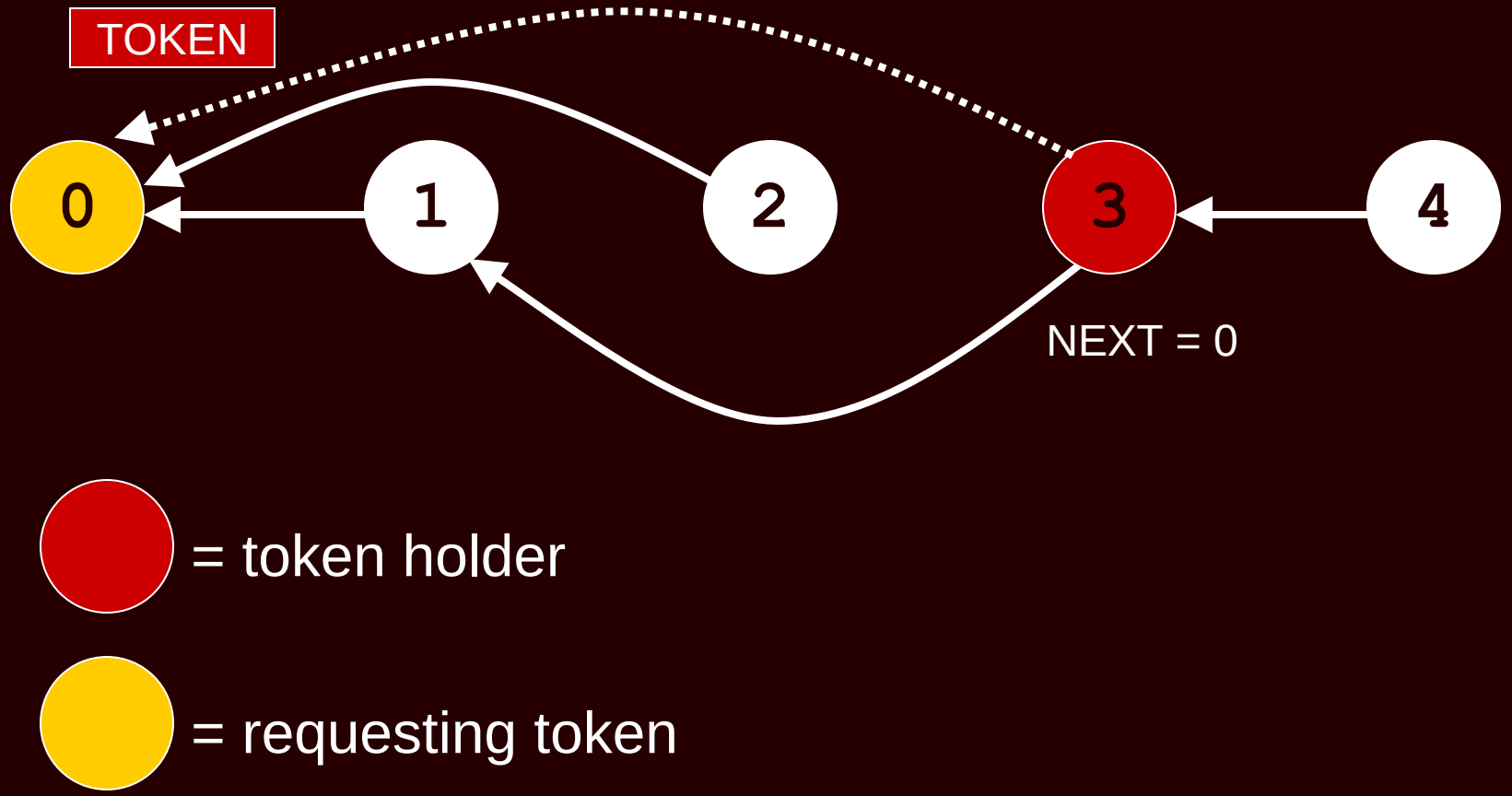
Generalized Algorithm: Edge+Path Reversal



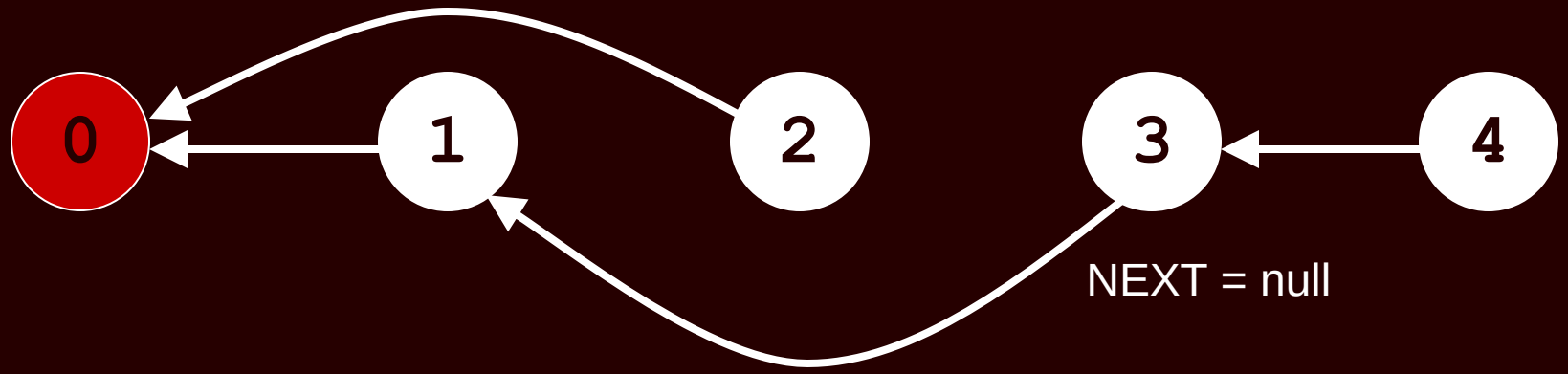
= token holder

= requesting token

Generalized Algorithm: Edge+Path Reversal



Generalized Algorithm: Edge+Path Reversal

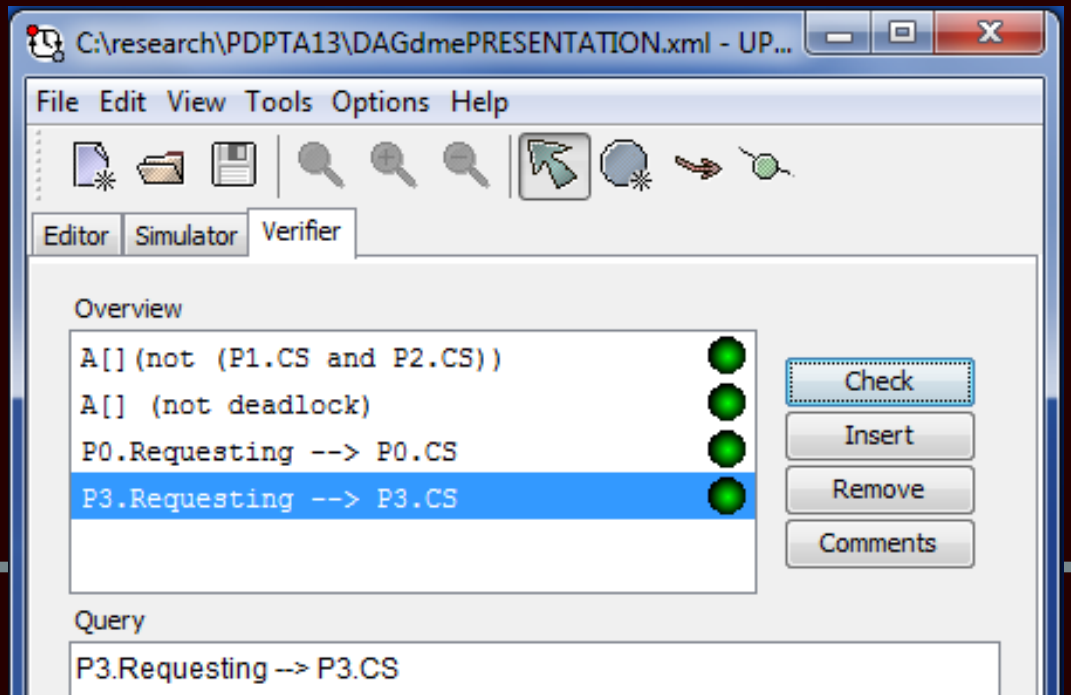


= token holder

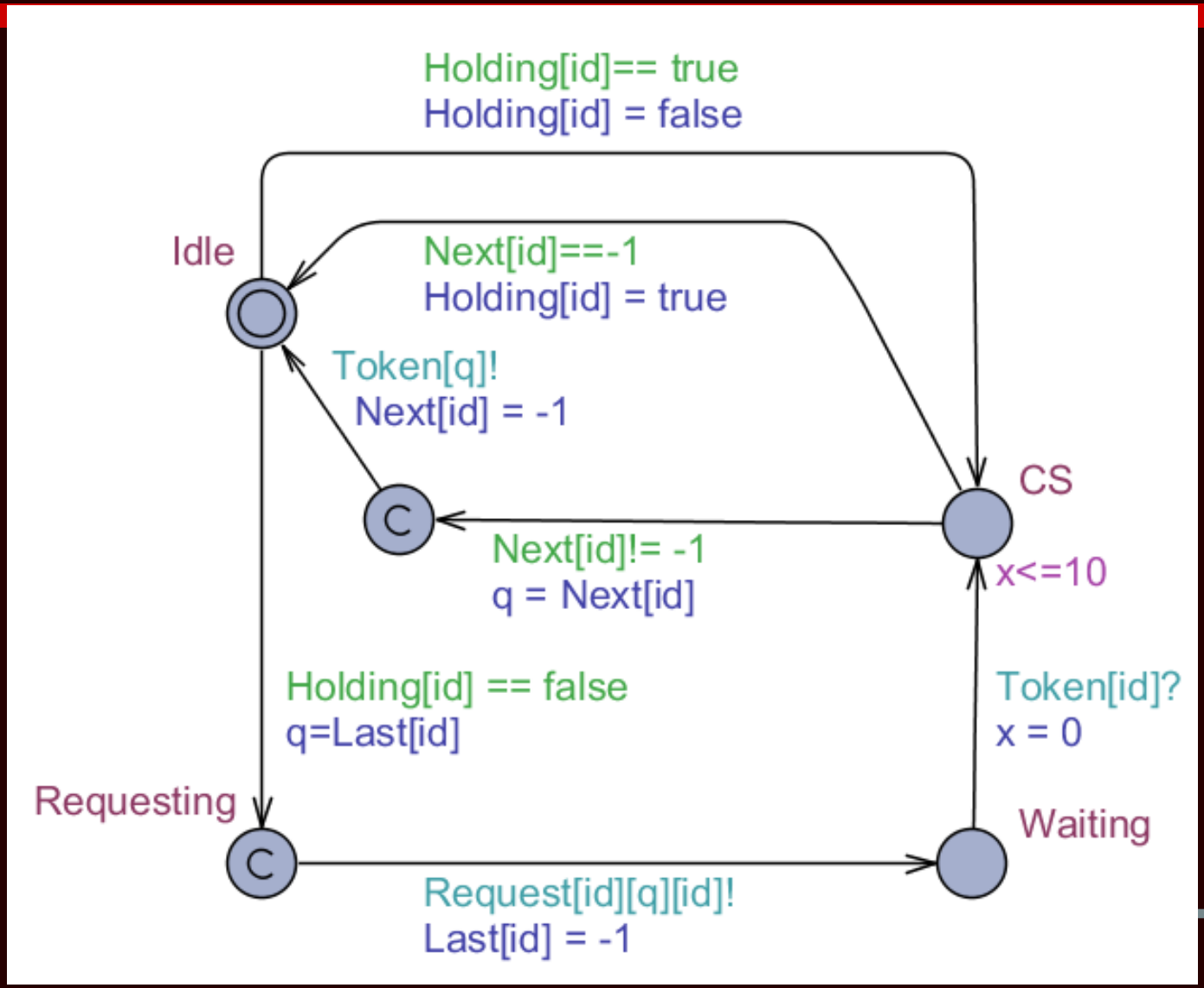
= requesting token

Properties to Verify

- Mutual exclusion is guaranteed.
- The algorithm is deadlock free.
- The algorithm is starvation free.



UPPAAL Model – ProcessWork(id)

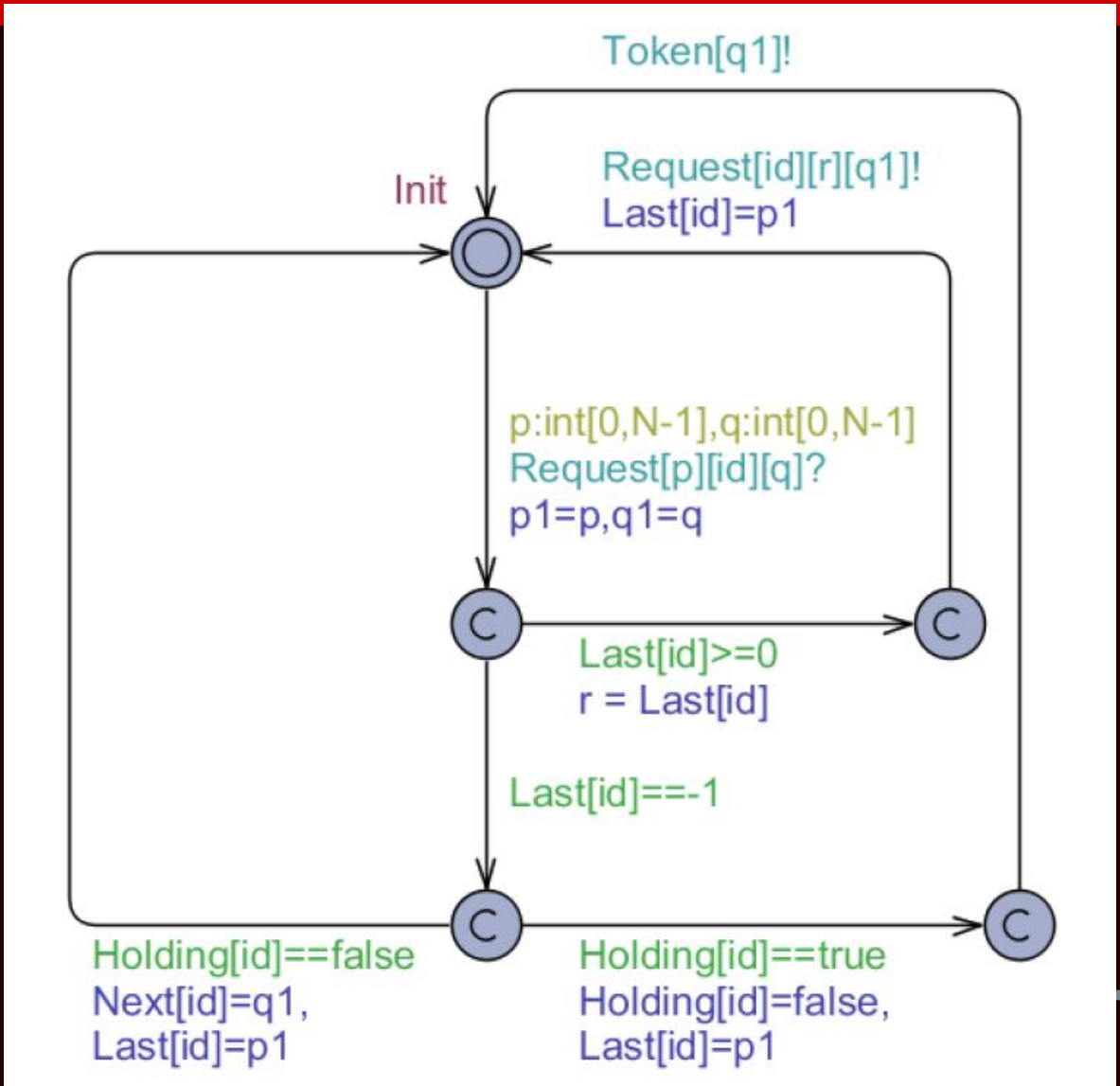


Holding – holding, but not using the token

Next – next node to receive token

Last – next node to send request to (node on path to last request)

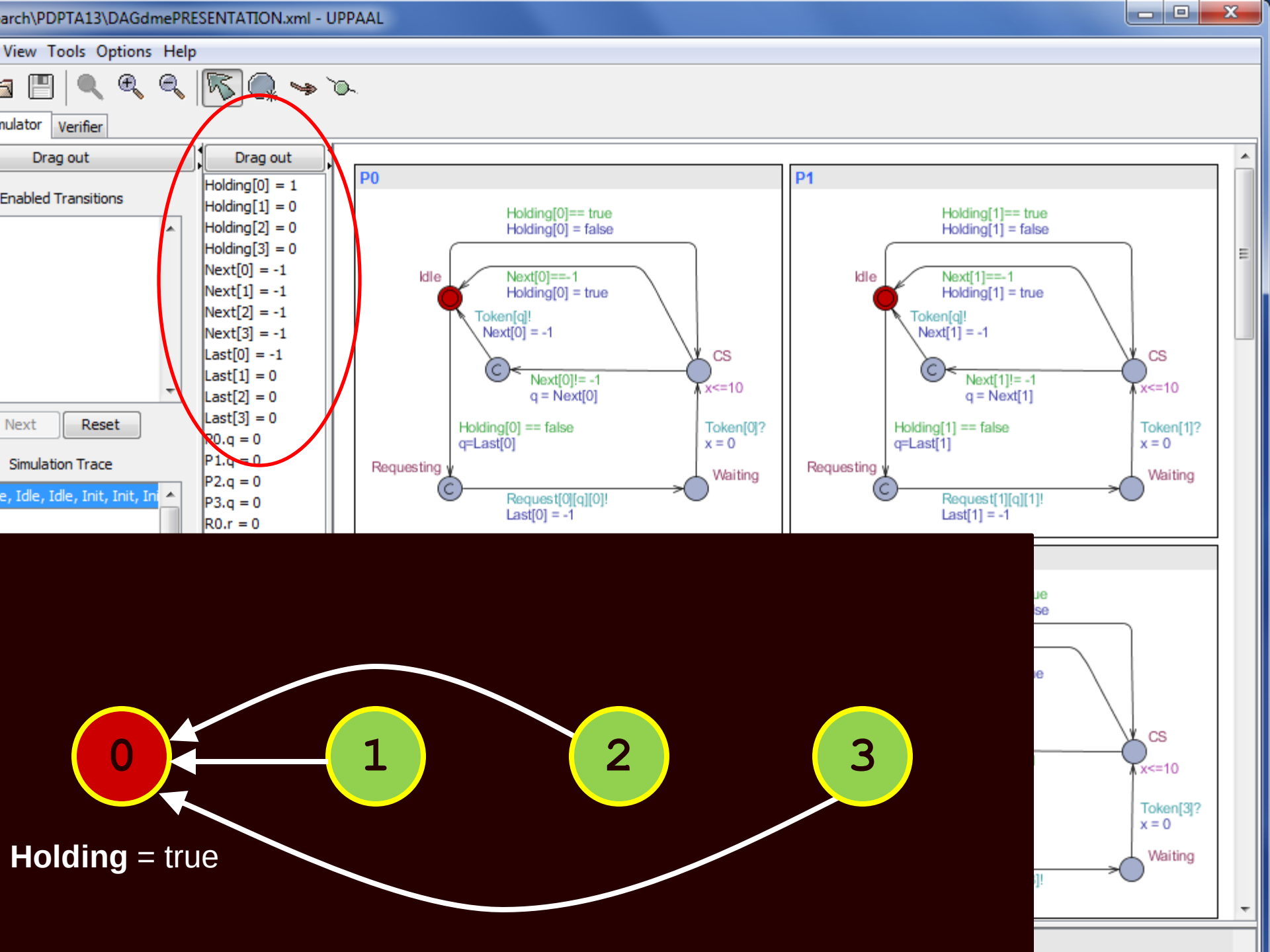
UPPAAL Model – ProcessRequest(id)



Holding – holding, but not using the token

Next – next node to receive token

Last – next node to send request to (node on path to last request)


arch\PDPTA13\DAGdmePRESENTATION.xml - UPPAAL
View Tools Options Help
Verifier
Drag out
Enabled Transitions
Next
Reset
Simulation Trace
Drag out
Holding[0] = 1
Holding[1] = 0
Holding[2] = 0
Holding[3] = 0
Next[0] = -1
Next[1] = -1
Next[2] = -1
Next[3] = -1
Last[0] = -1
Last[1] = 0
Last[2] = 0
Last[3] = 0
P0.q = 0
P1.q = 0
P2.q = 0
P3.q = 0
R0.r = 0
P0
Holding[0]== true
Holding[0] = false
Idle
Next[0]==-1
Holding[0] = true
Token[q]!
Next[0] = -1
CS
Next[0]!= -1
q = Next[0]
x<=10
Holding[0] == false
q=Last[0]
Token[0]?
x = 0
Requesting
Waiting
Request[0][q][0]!
Last[0] = -1
P1
Holding[1]== true
Holding[1] = false
Idle
Next[1]==-1
Holding[1] = true
Token[q]!
Next[1] = -1
CS
Next[1]!= -1
q = Next[1]
x<=10
Holding[1] == false
q=Last[1]
Token[1]?
x = 0
Requesting
Waiting
Request[1][q][1]!
Last[1] = -1
CS
x<=10
Token[3]?
x = 0
Waiting
0
1
2
3
Holding = true

C:\research\PDPTA13\DAGdmePRESENTATION.xml - UPPAAL

File Edit View Tools Options Help

Editor Simulator Verifier

Drag out

Enabled Transitions

P0
P1
P2
P3

Next Reset

Simulation Trace

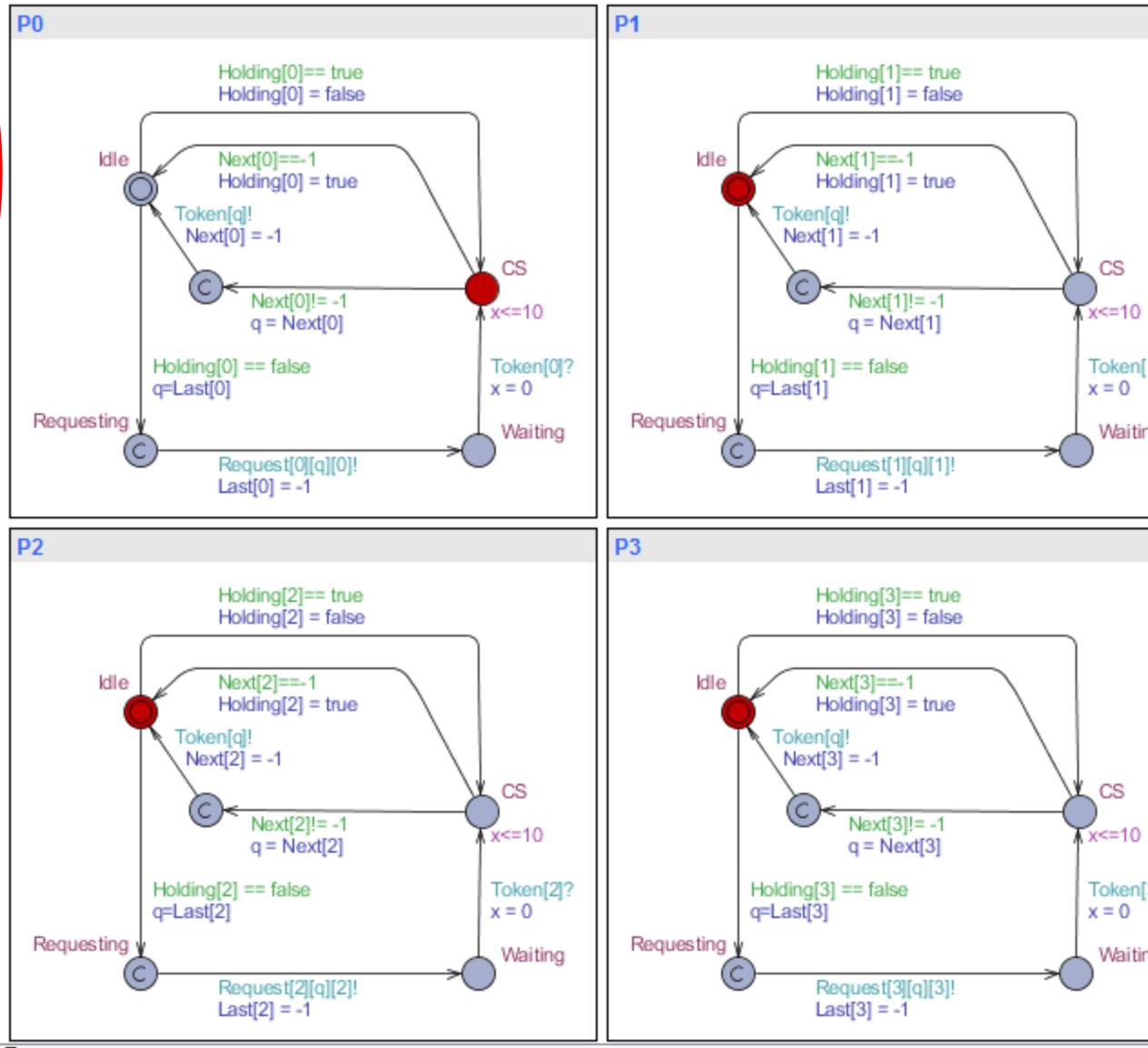
(Idle, Idle, Idle, Idle, Init, Init, Ini
P0
(CS, Idle, Idle, Idle, Init, Init, Init
P1
(CS, Requesting, Idle, Idle, Init, I
Request[1][q][1]: P1 --> R0
(CS, Waiting, Idle, Idle, -, Init, In
R0
(CS, Waiting, Idle, Idle, -, Init, In
R0
(CS, Waiting, Idle, Idle, Init, Init,
P2
(CS, Waiting, Requesting, Idle, Ir
Request[2][q][2]: P2 --> R0

Trace File:

Prev Next Replay

Drag out

Holding[0] = 0
Holding[1] = 0
Holding[2] = 0
Holding[3] = 0
Next[0] = -1
Next[1] = -1
Next[2] = -1
Next[3] = -1
Last[0] = -1
Last[1] = 0
Last[2] = 0
Last[3] = 0
P0.q = 0
P1.q = 0
P2.q = 0
P3.q = 0
R0.r = 0
R0.p1 = 0
R0.q1 = 0
R1.r = 0
R1.p1 = 0
R1.q1 = 0
R2.r = 0
R2.p1 = 0
R2.q1 = 0
R3.r = 0
R3.p1 = 0
R3.q1 = 0
P0.x in [0,10]
P1.x in [0,10]
P2.x in [0,10]
P3.x in [0,10]
P0.x = P1.x
P1.x = P2.x
P2.x = P3.x
P3.x = P0.x



C:\research\PDPTA13\DAGdmePRESENTATION.xml - UPPAAL

File Edit View Tools Options Help

Editor | Simulator | Verifier

Drag out

Enabled Transitions

Request[1][q][1]: P1 --> R0

Next | Reset

Simulation Trace

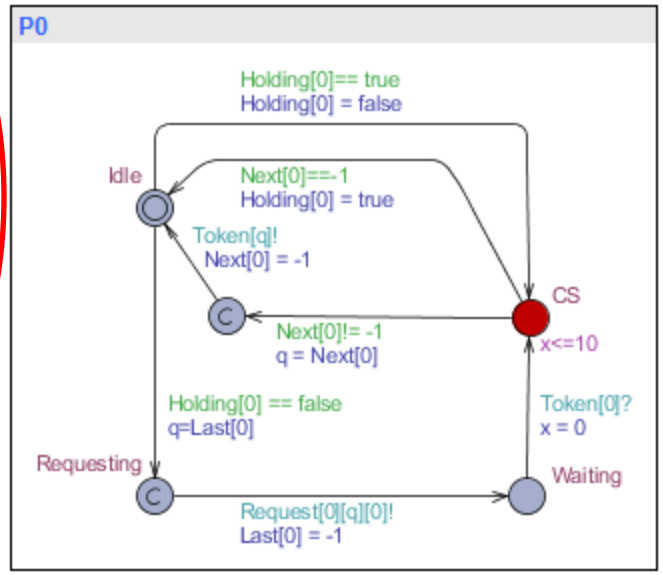
(Idle, Idle, Idle, Idle, Init, Init, Ini
P0
(CS, Idle, Idle, Idle, Init, Init, Init
P1
(CS, Requesting, Idle, Idle, Init, I
Request[1][q][1]: P1 --> R0
(CS, Waiting, Idle, Idle, -, Init, In
R0
(CS, Waiting, Idle, Idle, -, Init, In
R0
(CS, Waiting, Idle, Idle, Init, Init,
P2
(CS, Waiting, Requesting, Idle, Ir
Request[2][q][2]: P2 --> R0

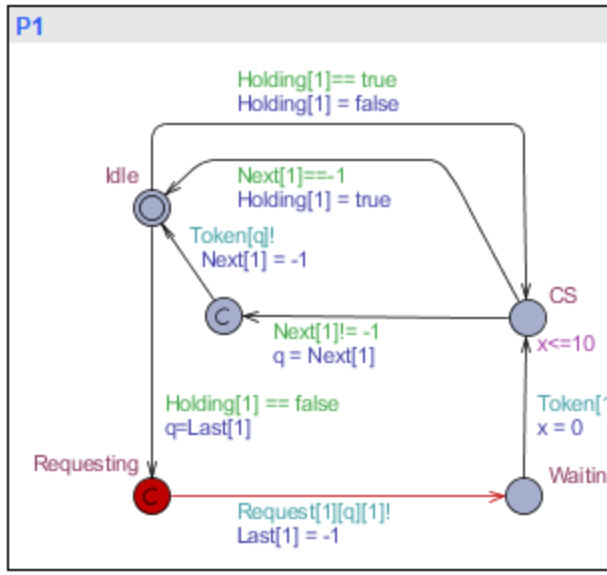
Trace File:

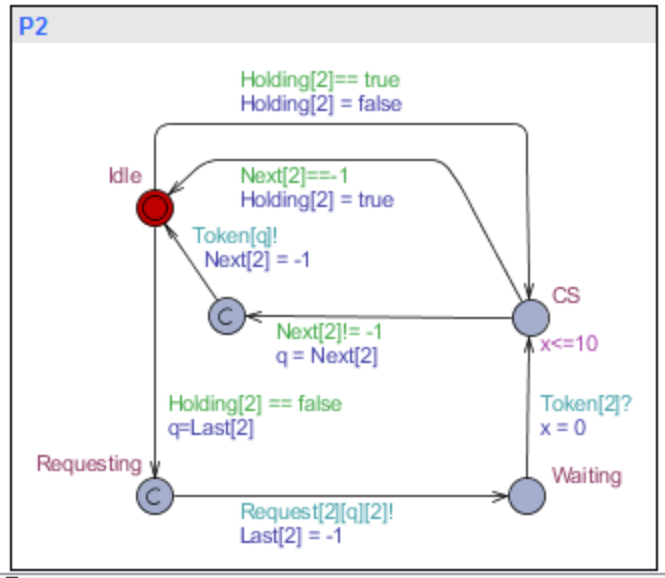
Prev | Next | Replay

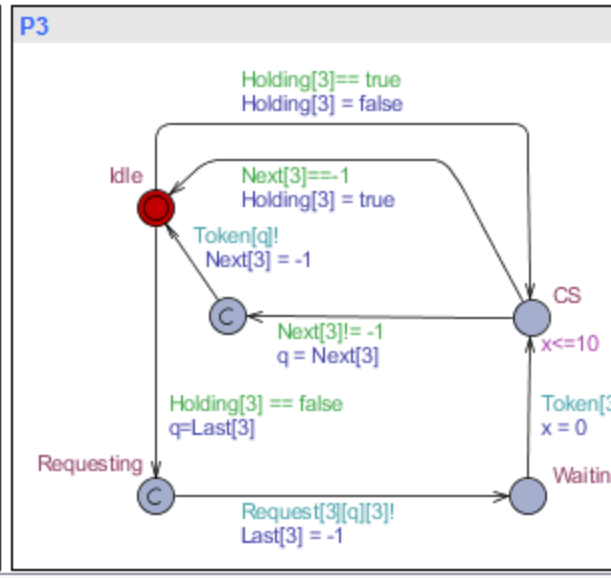
Drag out

Holding[0] = 0
Holding[1] = 0
Holding[2] = 0
Holding[3] = 0
Next[0] = -1
Next[1] = -1
Next[2] = -1
Next[3] = -1
Last[0] = -1
Last[1] = 0
Last[2] = 0
Last[3] = 0
P0.q = 0
P1.q = 0
P2.q = 0
P3.q = 0
R0.r = 0
R0.p1 = 0
R0.q1 = 0
R1.r = 0
R1.p1 = 0
R1.q1 = 0
R2.r = 0
R2.p1 = 0
R2.q1 = 0
R3.r = 0
R3.p1 = 0
R3.q1 = 0
P0.x in [0,10]
P1.x in [0,10]
P2.x in [0,10]
P3.x in [0,10]
P0.x = P1.x
P1.x = P2.x
P2.x = P3.x
P3.x = P0.x









P0 P1 P2 P3 R0 R1 R2 R3

File Edit View Tools Options Help

Editor | Simulator | Verifier

Drag out

Enabled Transitions

R0

Next | Reset

Simulation Trace

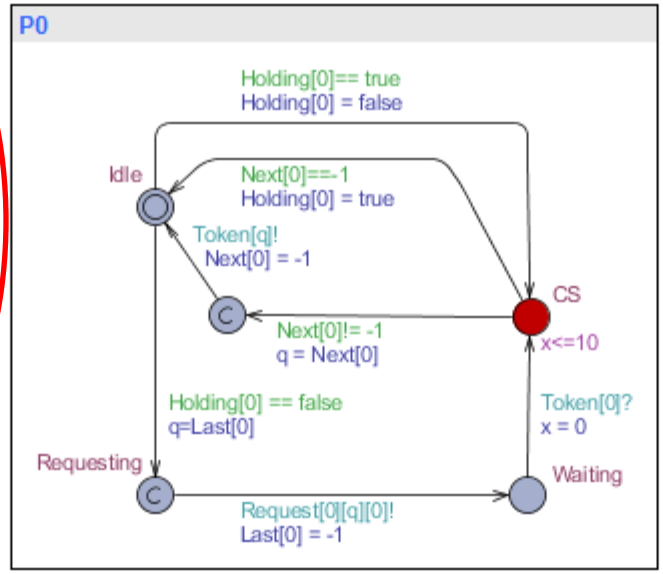
P1
(CS, Requesting, Idle, Idle, Init, I
Request[1][q][1]: P1 --> R0
(CS, Waiting, Idle, Idle, -, Init, In
R0
(CS, Waiting, Idle, Idle, -, Init, In
R0
(CS, Waiting, Idle, Idle, Init, Init,
P2
(CS, Waiting, Requesting, Idle, Ir
Request[2][q][2]: P2 --> R0
(CS, Waiting, Waiting, Idle, -, Init
R0
(CS, Waiting, Waiting, Idle, -, Init

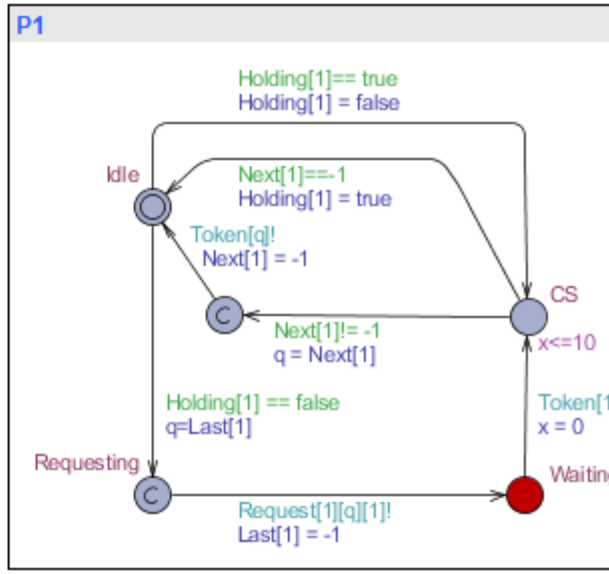
Trace File:

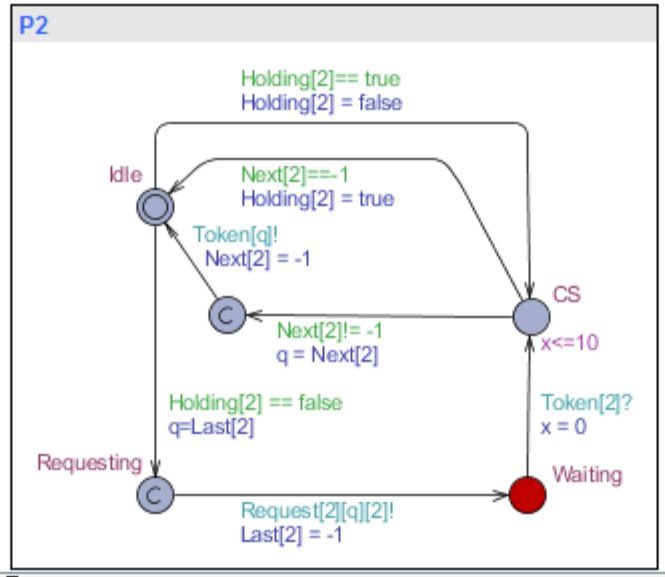
Prev | Next | Replay
Open | Save | Auto

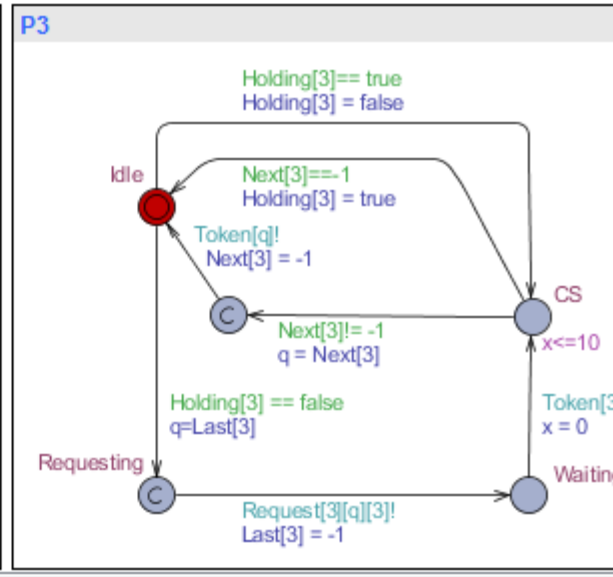
Drag out

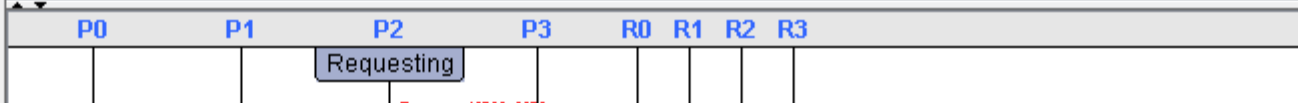
Holding[0] = 0
Holding[1] = 0
Holding[2] = 0
Holding[3] = 0
Next[0] = 1
Next[1] = -1
Next[2] = -1
Next[3] = -1
Last[0] = 1
Last[1] = -1
Last[2] = -1
Last[3] = 0
P0.q = 0
P1.q = 0
P2.q = 0
P3.q = 0
R0.r = 0
R0.p1 = 2
R0.q1 = 2
R1.r = 0
R1.p1 = 0
R1.q1 = 0
R2.r = 0
R2.p1 = 0
R2.q1 = 0
R3.r = 0
R3.p1 = 0
R3.q1 = 0
P0.x in [0,10]
P1.x in [0,10]
P2.x in [0,10]
P3.x in [0,10]
P0.x = P1.x
P1.x = P2.x
P2.x = P3.x
P3.x = P0.x

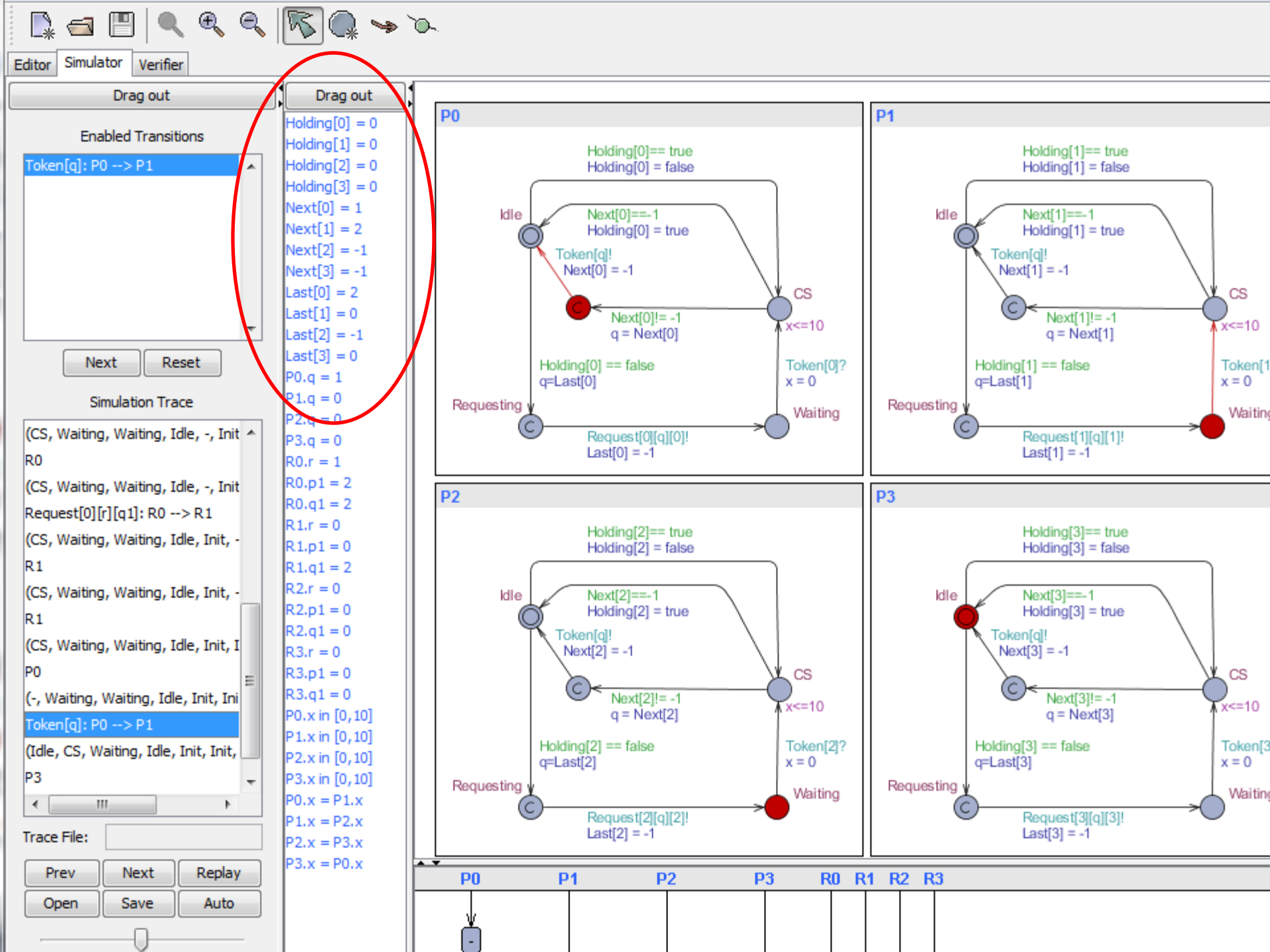











Editor
Simulator
Verifier
Drag out
Enabled Transitions
Token[q]: P0 --> P1
Next
Reset
Simulation Trace
(CS, Waiting, Waiting, Idle, -, Init
R0
(CS, Waiting, Waiting, Idle, -, Init
Request[0][r][q1]: R0 --> R1
(CS, Waiting, Waiting, Idle, Init, -
R1
(CS, Waiting, Waiting, Idle, Init, -
R1
(CS, Waiting, Waiting, Idle, Init, I
P0
(-, Waiting, Waiting, Idle, Init, Ini
Token[q]: P0 --> P1
(Idle, CS, Waiting, Idle, Init, Init,
P3
Trace File:
Prev
Next
Replay
Open
Save
Auto
Drag out
Holding[0] = 0
Holding[1] = 0
Holding[2] = 0
Holding[3] = 0
Next[0] = 1
Next[1] = 2
Next[2] = -1
Next[3] = -1
Last[0] = 2
Last[1] = 0
Last[2] = -1
Last[3] = 0
P0.q = 1
P1.q = 0
P3.q = 0
R0.r = 1
R0.p1 = 2
R0.q1 = 2
R1.r = 0
R1.p1 = 0
R1.q1 = 2
R2.r = 0
R2.p1 = 0
R2.q1 = 0
R3.r = 0
R3.p1 = 0
R3.q1 = 0
P0.x in [0,10]
P1.x in [0,10]
P2.x in [0,10]
P3.x in [0,10]
P0.x = P1.x
P1.x = P2.x
P2.x = P3.x
P3.x = P0.x
P0
Holding[0]== true
Holding[0] = false
Idle
Next[0]==-1
Holding[0] = true
Token[q]!
Next[0] = -1
CS
Next[0]!= -1
q = Next[0]
x<=10
Holding[0] == false
q=Last[0]
Token[0]?
x = 0
Requesting
Waiting
Request[0][q][0]!
Last[0] = -1
P1
Holding[1]== true
Holding[1] = false
Idle
Next[1]==-1
Holding[1] = true
Token[q]!
Next[1] = -1
CS
Next[1]!= -1
q = Next[1]
x<=10
Holding[1] == false
q=Last[1]
Token[1
x = 0
Requesting
Waiting
Request[1][q][1]!
Last[1] = -1
P2
Holding[2]== true
Holding[2] = false
Idle
Next[2]==-1
Holding[2] = true
Token[q]!
Next[2] = -1
CS
Next[2]!= -1
q = Next[2]
x<=10
Holding[2] == false
q=Last[2]
Token[2]?
x = 0
Requesting
Waiting
Request[2][q][2]!
Last[2] = -1
P3
Holding[3]== true
Holding[3] = false
Idle
Next[3]==-1
Holding[3] = true
Token[q]!
Next[3] = -1
CS
Next[3]!= -1
q = Next[3]
x<=10
Holding[3] == false
q=Last[3]
Token[3
x = 0
Requesting
Waiting
Request[3][q][3]!
Last[3] = -1
P0
P1
P2
P3
R0
R1
R2
R3

File Edit View Tools Options Help

Editor | Simulator | Verifier

Drag out

Enabled Transitions

P0
P1
P3

Next | Reset

Simulation Trace

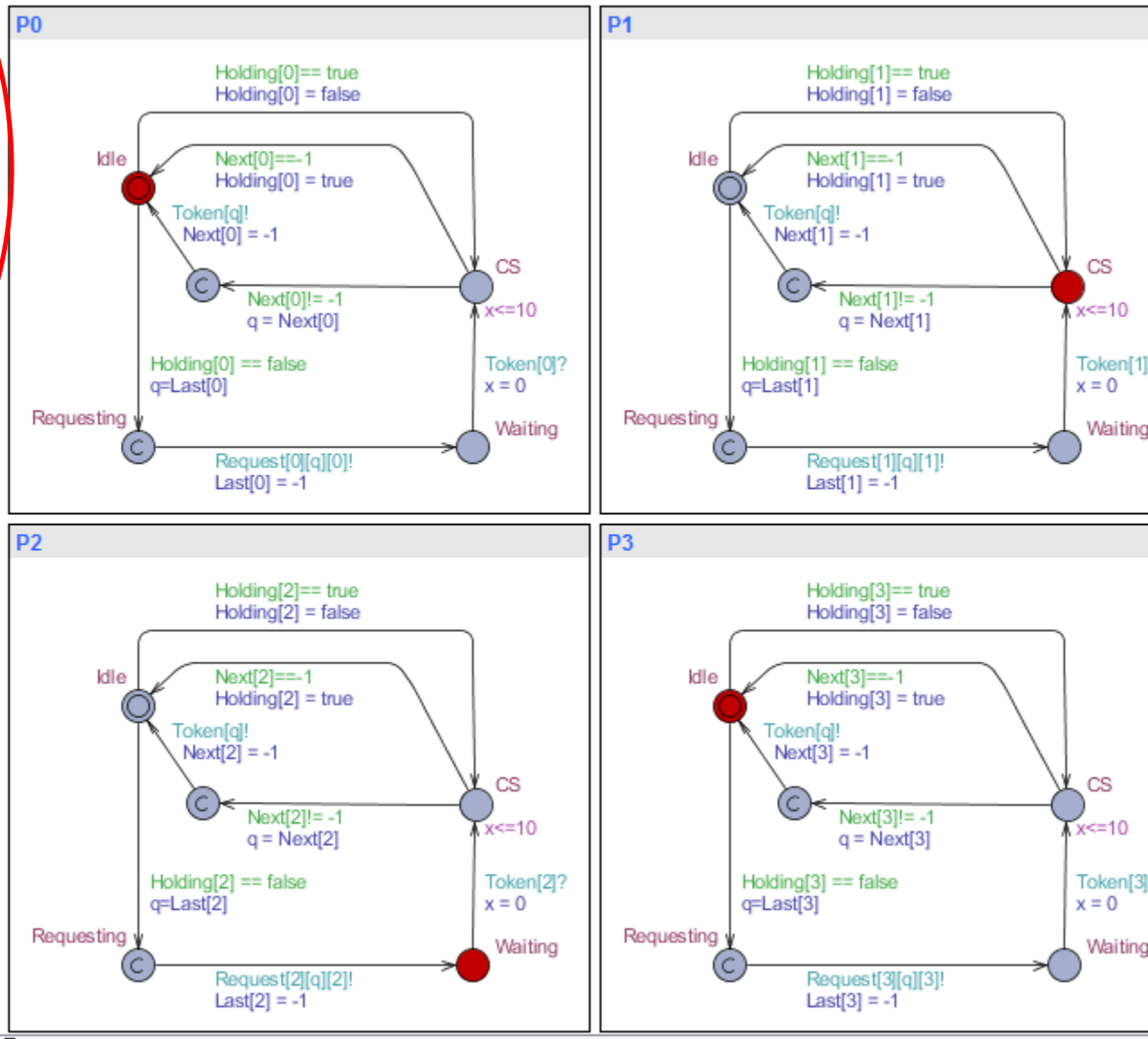
(CS, Waiting, Waiting, Idle, -, Init
R0
(CS, Waiting, Waiting, Idle, -, Init
Request[0][r][q1]: R0 --> R1
(CS, Waiting, Waiting, Idle, Init, -
R1
(CS, Waiting, Waiting, Idle, Init, -
R1
(CS, Waiting, Waiting, Idle, Init, I
P0
(-, Waiting, Waiting, Idle, Init, Ini
Token[q]: P0 --> P1
(Idle, CS, Waiting, Idle, Init, Init,
P3

Trace File:

Prev | Next | Replay
Open | Save | Auto

Drag out

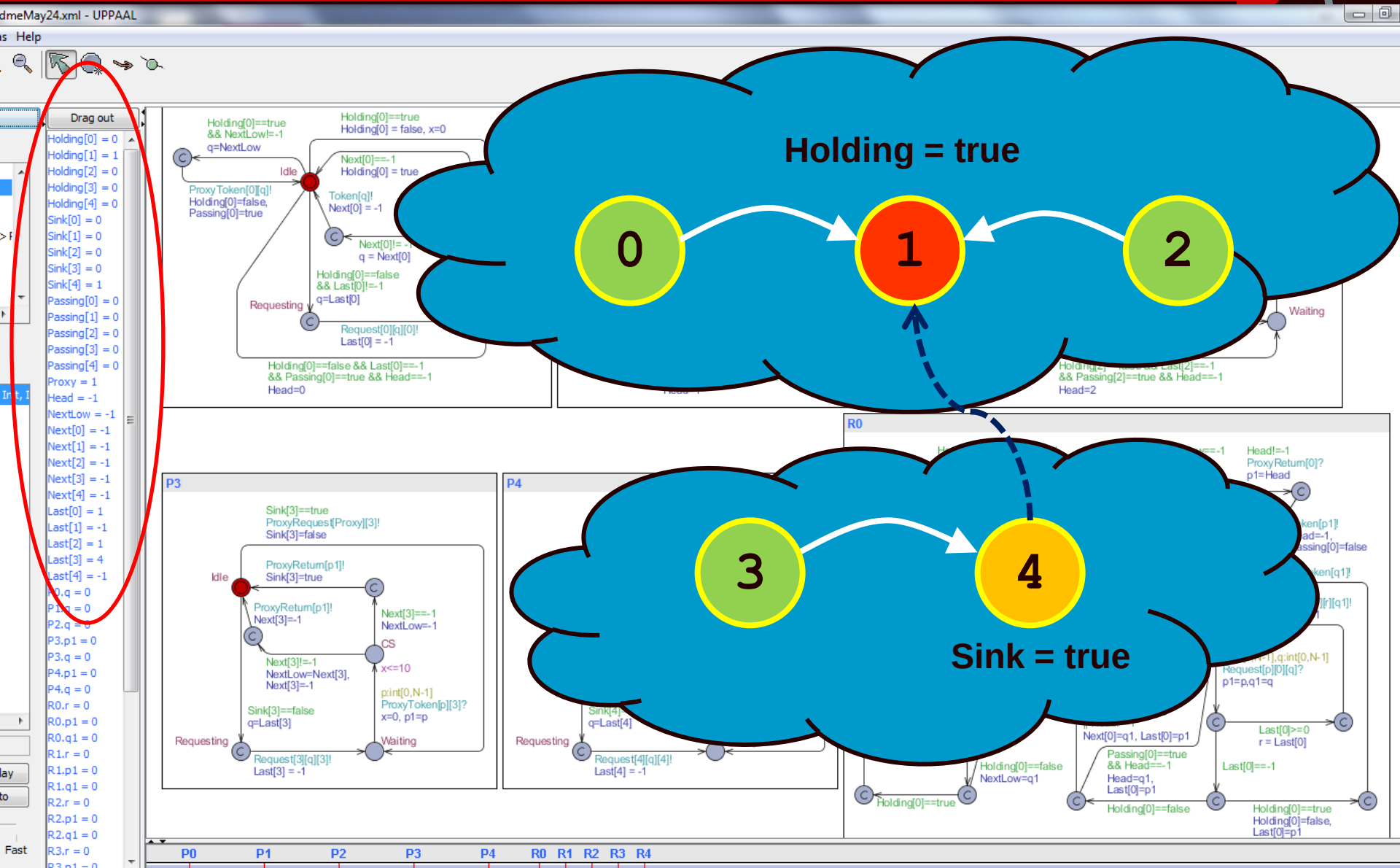
Holding[0] = 0
Holding[1] = 0
Holding[2] = 0
Holding[3] = 0
Next[0] = -1
Next[1] = 2
Next[2] = -1
Next[3] = -1
Last[0] = 2
Last[1] = 0
Last[2] = -1
Last[3] = 0
P0.q = 1
P1.q = 0
P2.q = 0
P3.q = 0
R0.r = 1
R0.p1 = 2
R0.q1 = 2
R1.r = 0
R1.p1 = 0
R1.q1 = 2
R2.r = 0
R2.p1 = 0
R2.q1 = 0
R3.r = 0
R3.p1 = 0
R3.q1 = 0
P0.x in [0,20]
P1.x in [0,10]
P2.x in [0,20]
P3.x in [0,20]
P0.x - P1.x in [0,1
P0.x = P2.x
P2.x = P3.x
P3.x = P0.x



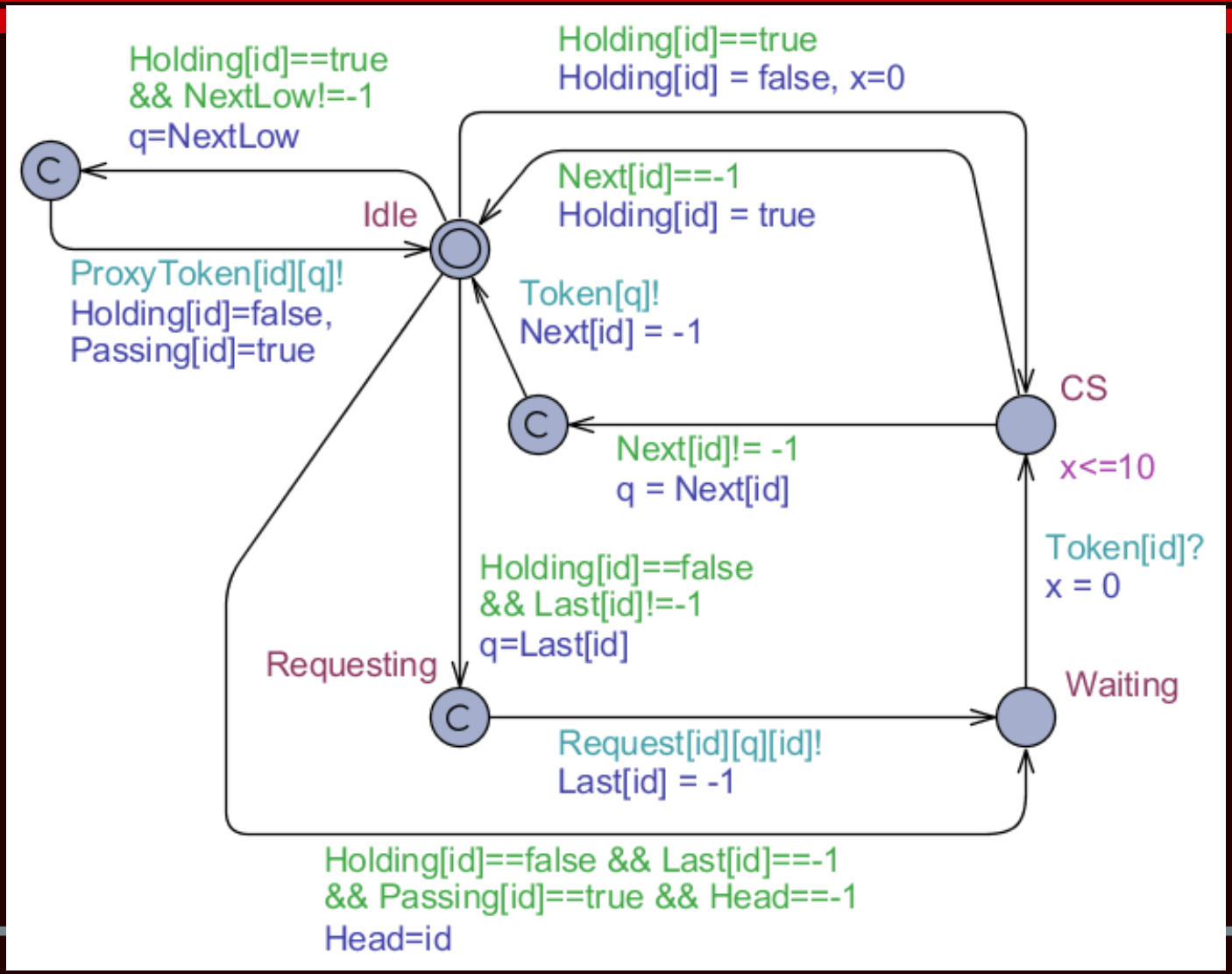
Prioritized Algorithms

- Modeled after real-time scheduling algorithm.
- Each priority level uses distributed algorithm for round-robin processing of requests at the same priority level.
- Proxy requests are passed up to reach a sink node at the highest priority level and to be enqueued in priority order in the token.
- Algorithm correctness is verified using same real-time verification tool, UPPAAL.
- Algorithm performance can also be determined using UPPAAL.

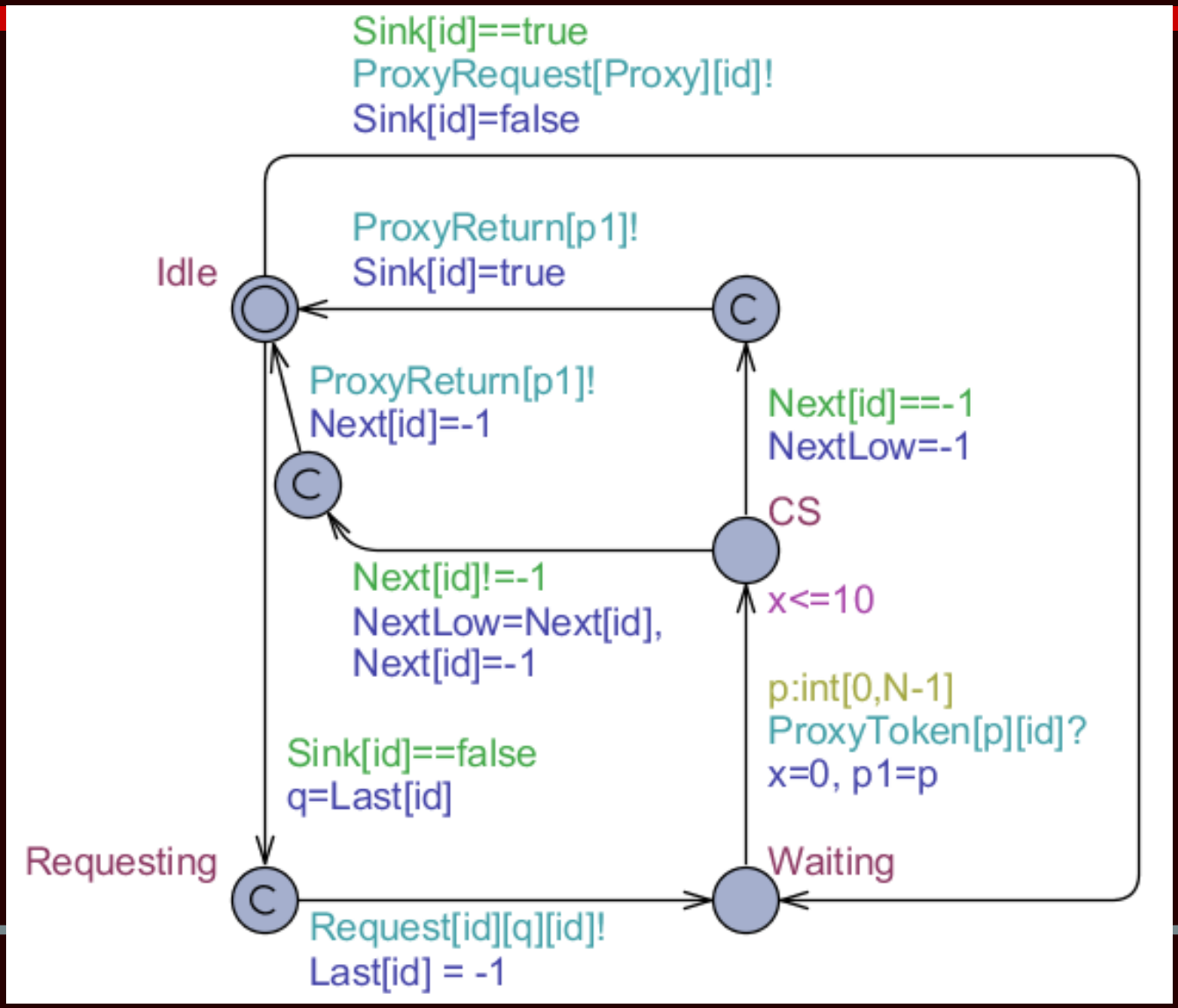
Example


dmeMay24.xml - UPPAAL
Help
Drag out
Holding[0] = 0
Holding[1] = 1
Holding[2] = 0
Holding[3] = 0
Holding[4] = 0
Sink[0] = 0
Sink[1] = 0
Sink[2] = 0
Sink[3] = 0
Sink[4] = 1
Passing[0] = 0
Passing[1] = 0
Passing[2] = 0
Passing[3] = 0
Passing[4] = 0
Proxy = 1
Head = -1
NextLow = -1
Next[0] = -1
Next[1] = -1
Next[2] = -1
Next[3] = -1
Next[4] = -1
Last[0] = 1
Last[1] = -1
Last[2] = 1
Last[3] = 4
Last[4] = -1
Holding = true
0
1
2
Sink = true
3
4
P3
P4
R0
Idle
Requesting
Waiting
CS
P0
P1
P2
P3
P4
R0
R1
R2
R3
R4

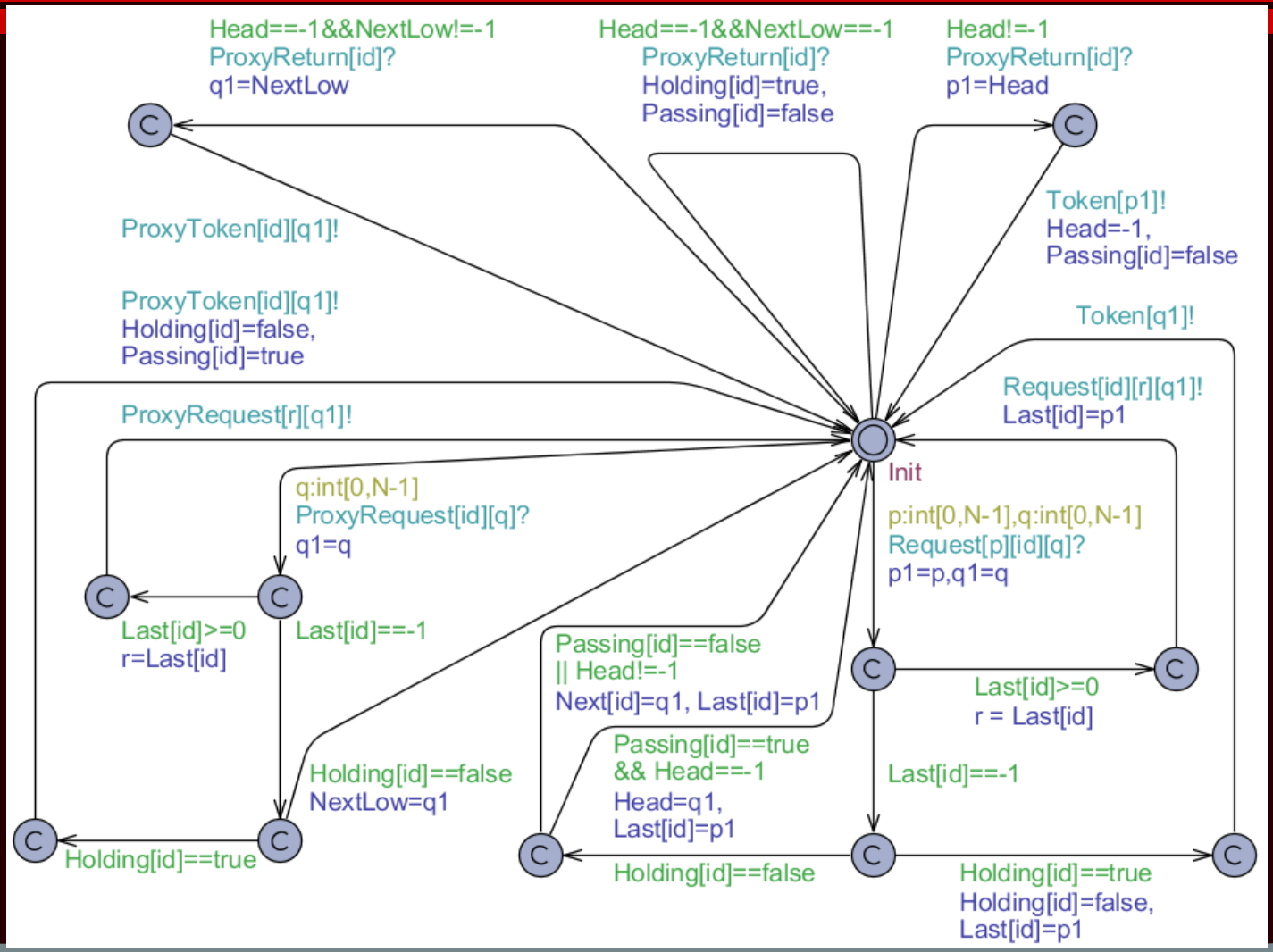
Prioritized ProcessWork(id)

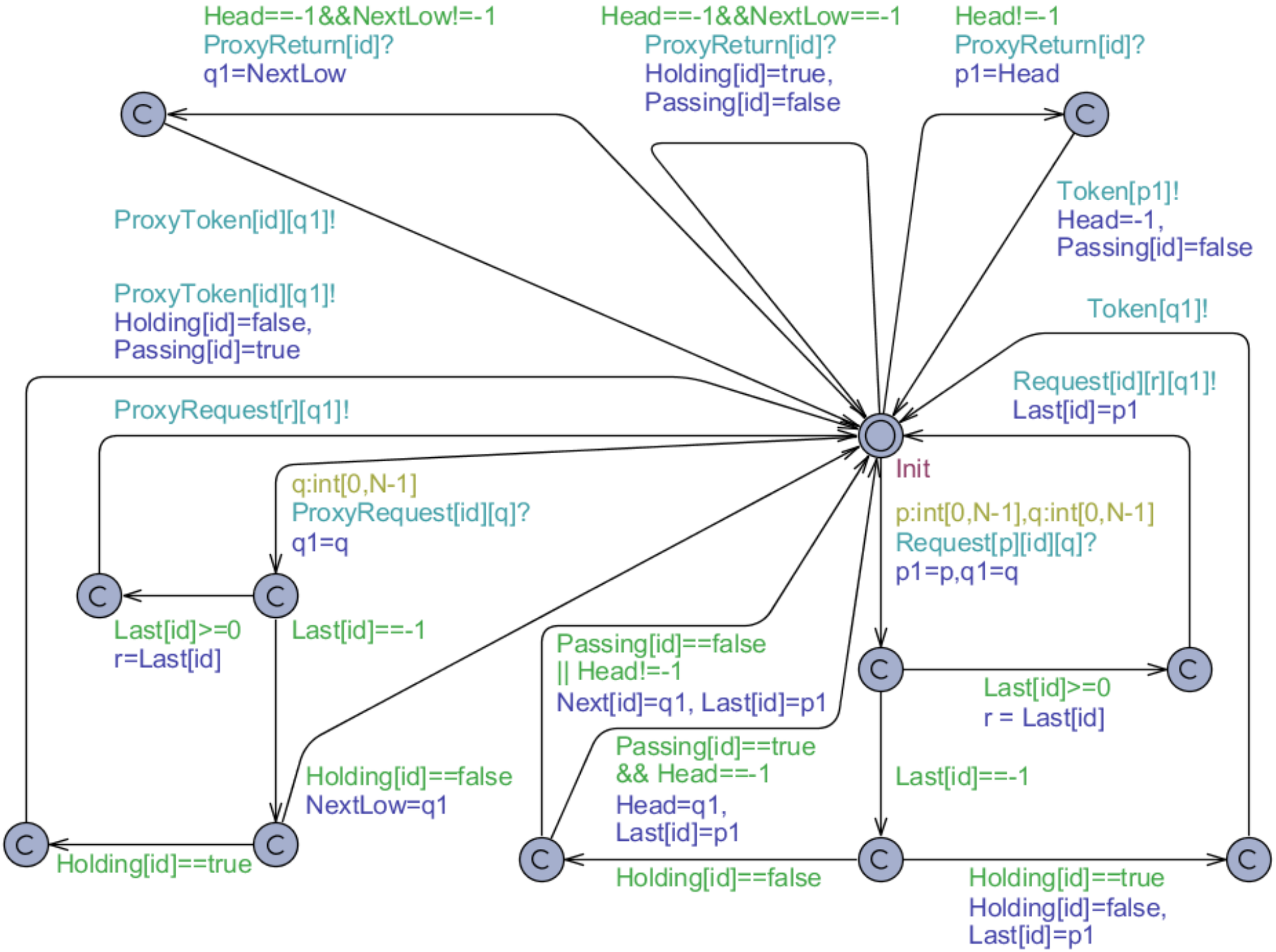


Prioritized ProcessWorkLow(id)

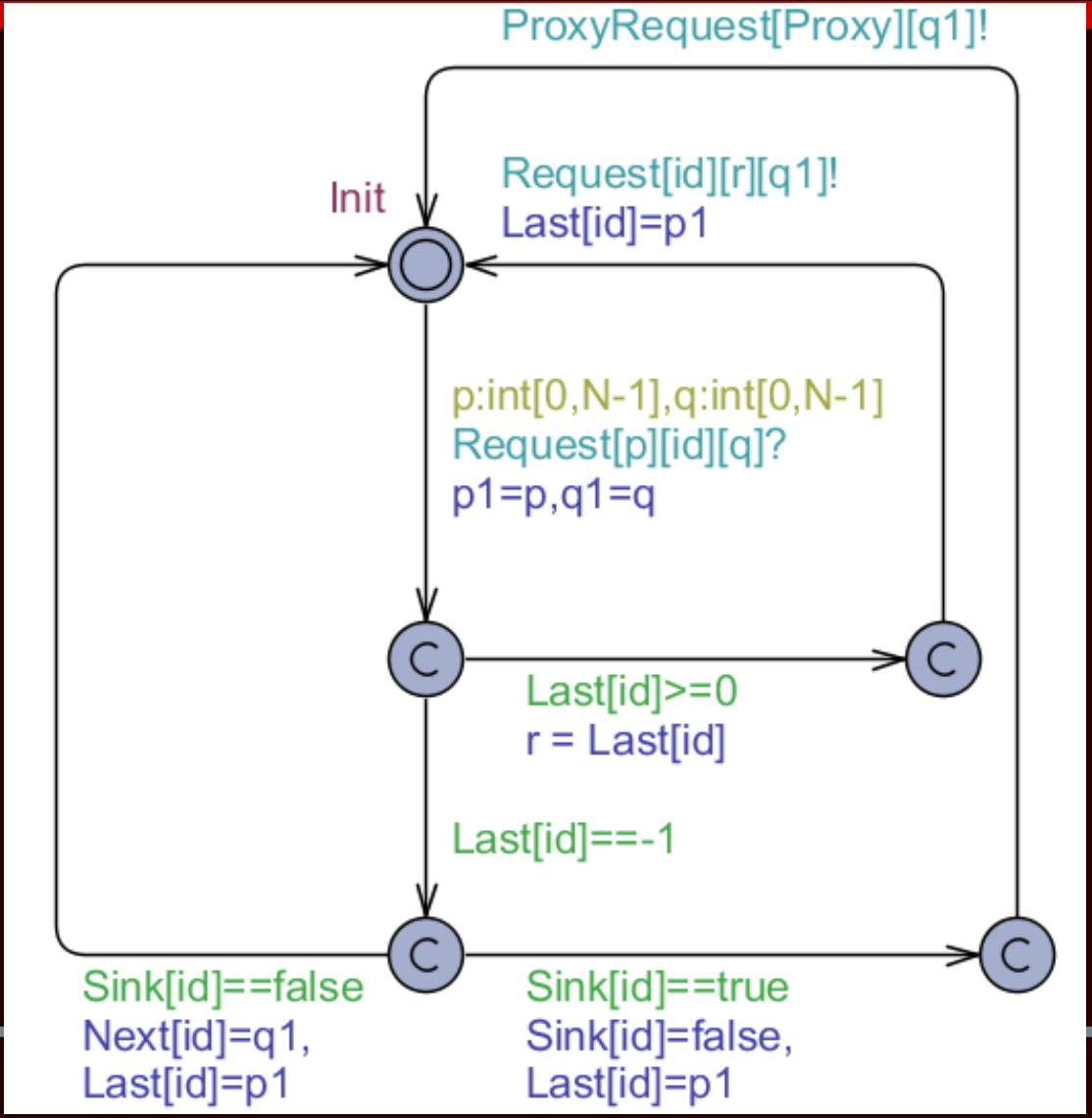


Prioritized ProcessRequest(id)

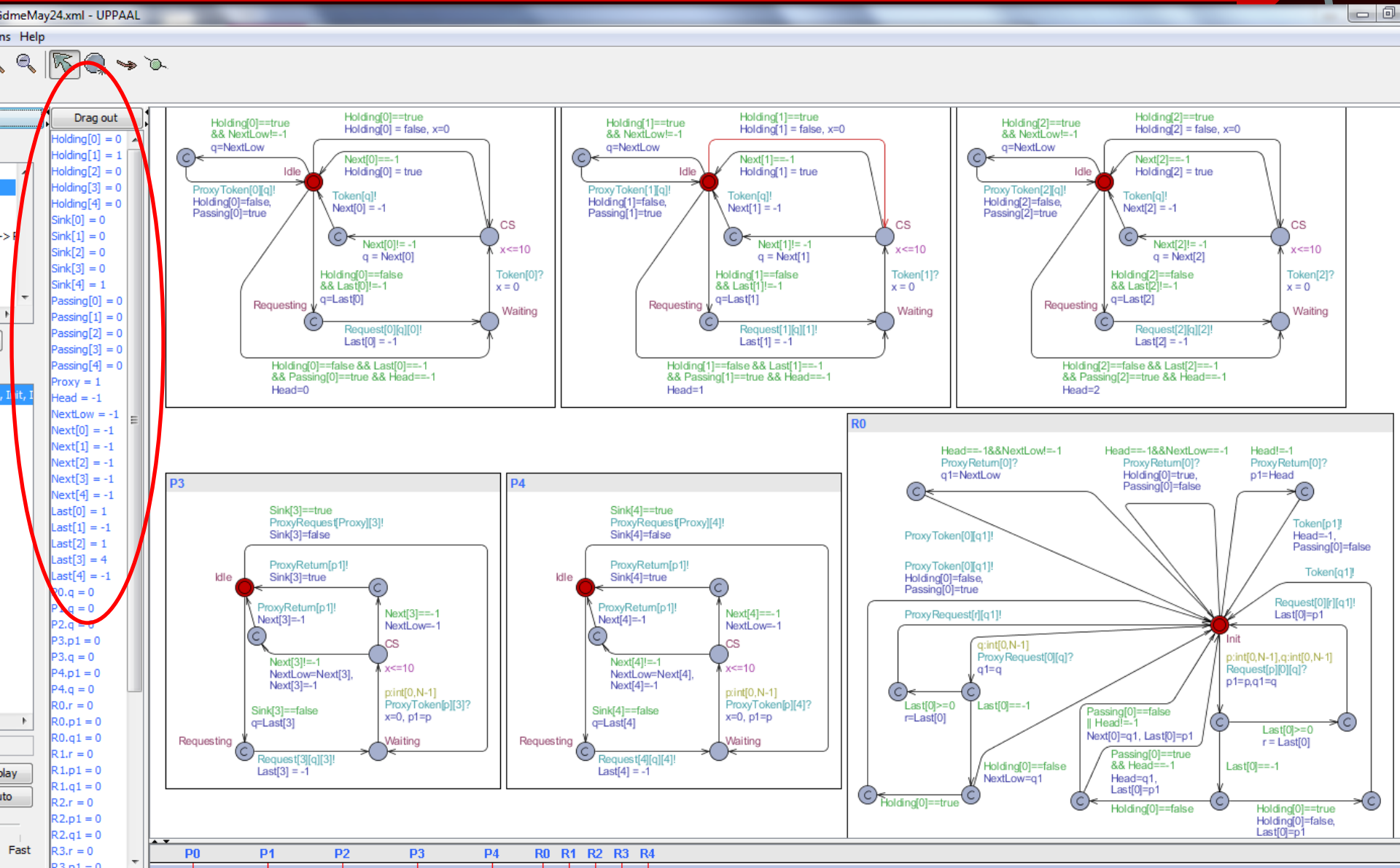



Head==-1&&NextLow!=-1
ProxyReturn[id]?
q1=NextLow
Head==-1&&NextLow==-1
ProxyReturn[id]?
Holding[id]=true,
Passing[id]=false
Head!=-1
ProxyReturn[id]?
p1=Head
ProxyToken[id][q1]!
Token[p1]!
Head=-1,
Passing[id]=false
ProxyToken[id][q1]!
Holding[id]=false,
Passing[id]=true
Token[q1]!
Request[id][r][q1]!
Last[id]=p1
ProxyRequest[r][q1]!
Init
q:int[0,N-1]
ProxyRequest[id][q]?
q1=q
p:int[0,N-1],q:int[0,N-1]
Request[p][id][q]?
p1=p,q1=q
Last[id]>=0
r=Last[id]
Last[id]==-1
Passing[id]==false
|| Head!=-1
Next[id]=q1, Last[id]=p1
Last[id]>=0
r = Last[id]
Passing[id]==true
&& Head==-1
Head=q1,
Last[id]=p1
Last[id]==-1
Holding[id]==false
NextLow=q1
Holding[id]==true
Holding[id]==false
Holding[id]==true
Holding[id]=false,
Last[id]=p1

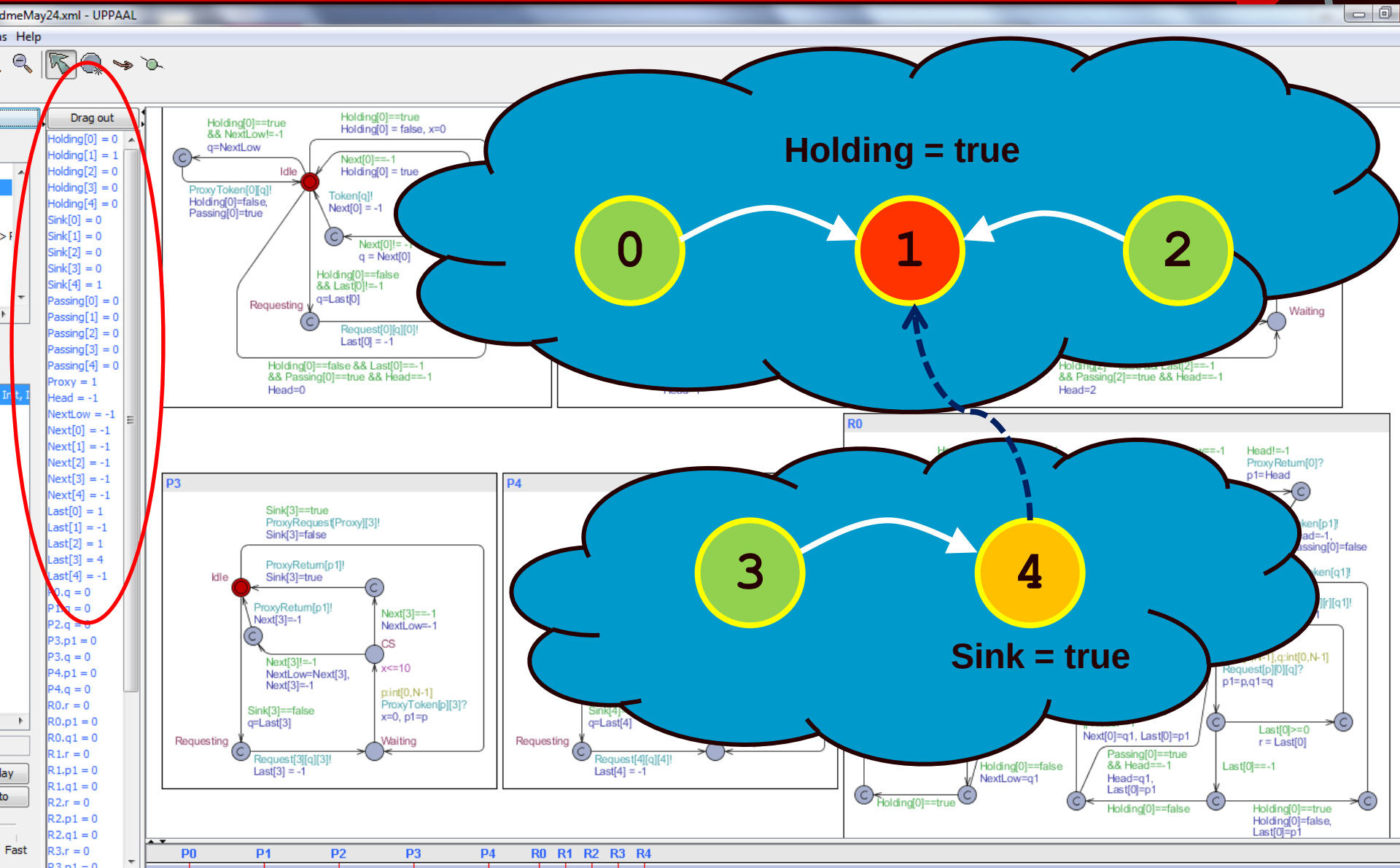
Prioritized ProcessRequestLow(id)



Example

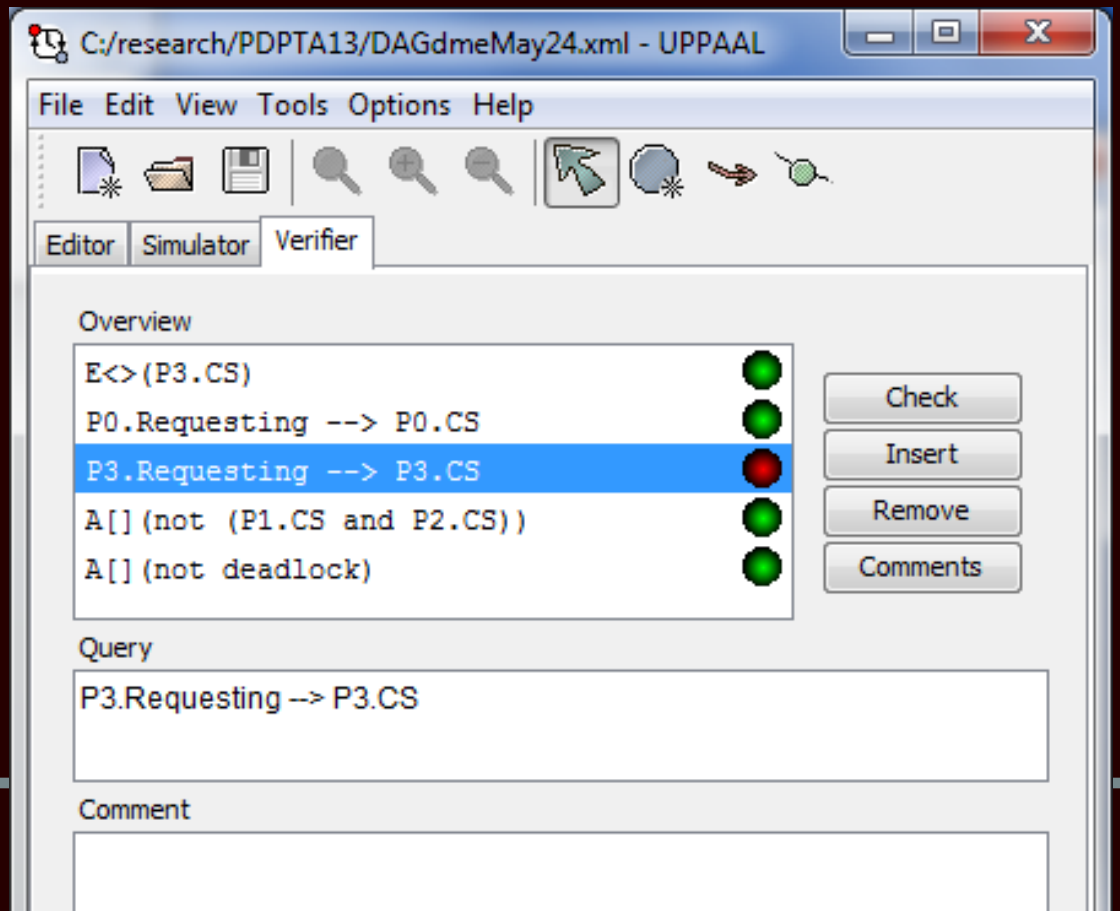

dmeMay24.xml - UPPAAL
ns Help
Drag out
Holding[0] = 0
Holding[1] = 1
Holding[2] = 0
Holding[3] = 0
Holding[4] = 0
Sink[0] = 0
Sink[1] = 0
Sink[2] = 0
Sink[3] = 0
Sink[4] = 1
Passing[0] = 0
Passing[1] = 0
Passing[2] = 0
Passing[3] = 0
Passing[4] = 0
Proxy = 1
Head = -1
NextLow = -1
Next[0] = -1
Next[1] = -1
Next[2] = -1
Next[3] = -1
Next[4] = -1
Last[0] = 1
Last[1] = -1
Last[2] = 1
Last[3] = 4
Last[4] = -1
P0.q = 0
P1.q = 0
P2.q = 0
P3.p1 = 0
P3.q = 0
P4.p1 = 0
P4.q = 0
R0.r = 0
R0.p1 = 0
R0.q1 = 0
R1.r = 0
R1.p1 = 0
R1.q1 = 0
R2.r = 0
R2.p1 = 0
R2.q1 = 0
R3.r = 0
Holding[0]==true && NextLow!=-1
q=NextLow
Holding[0]==true
Holding[0] = false, x=0
Idle
Next[0]==-1
Holding[0] = true
ProxyToken[0][q]!
Holding[0]=false,
Passing[0]=true
Token[q]!
Next[0] = -1
CS
x<=10
Next[0]!= -1
q = Next[0]
Holding[0]==false
&& Last[0]!=-1
q=Last[0]
Token[0]?
x = 0
Requesting
Waiting
Request[0][q][0]!
Last[0] = -1
Holding[0]==false && Last[0]==-1
&& Passing[0]==true && Head==-1
Head=0
Holding[1]==true && NextLow!=-1
q=NextLow
Holding[1]==true
Holding[1] = false, x=0
Idle
Next[1]==-1
Holding[1] = true
ProxyToken[1][q]!
Holding[1]=false,
Passing[1]=true
Token[q]!
Next[1] = -1
CS
x<=10
Next[1]!= -1
q = Next[1]
Holding[1]==false
&& Last[1]!=-1
q=Last[1]
Token[1]?
x = 0
Requesting
Waiting
Request[1][q][1]!
Last[1] = -1
Holding[1]==false && Last[1]==-1
&& Passing[1]==true && Head==-1
Head=1
Holding[2]==true && NextLow!=-1
q=NextLow
Holding[2]==true
Holding[2] = false, x=0
Idle
Next[2]==-1
Holding[2] = true
ProxyToken[2][q]!
Holding[2]=false,
Passing[2]=true
Token[q]!
Next[2] = -1
CS
x<=10
Next[2]!= -1
q = Next[2]
Holding[2]==false
&& Last[2]!=-1
q=Last[2]
Token[2]?
x = 0
Requesting
Waiting
Request[2][q][2]!
Last[2] = -1
Holding[2]==false && Last[2]==-1
&& Passing[2]==true && Head==-1
Head=2
P3
Sink[3]==true
ProxyRequest[Proxy][3]!
Sink[3]=false
Idle
ProxyReturn[p1]!
Sink[3]=true
ProxyReturn[p1]!
Next[3]=-1
Next[3]==-1
NextLow=-1
CS
x<=10
Next[3]!=-1
NextLow=Next[3],
Next[3]=-1
p:int[0,N-1]
ProxyToken[p][3]?
x=0, p1=p
Sink[3]==false
q=Last[3]
Requesting
Waiting
Request[3][q][3]!
Last[3] = -1
P4
Sink[4]==true
ProxyRequest[Proxy][4]!
Sink[4]=false
Idle
ProxyReturn[p1]!
Sink[4]=true
ProxyReturn[p1]!
Next[4]=-1
Next[4]==-1
NextLow=-1
CS
x<=10
Next[4]!=-1
NextLow=Next[4],
Next[4]=-1
p:int[0,N-1]
ProxyToken[p][4]?
x=0, p1=p
Sink[4]==false
q=Last[4]
Requesting
Waiting
Request[4][q][4]!
Last[4] = -1
R0
Head==-1&&NextLow!=-1
ProxyReturn[0]?
q1=NextLow
Head==-1&&NextLow==-1
ProxyReturn[0]?
Holding[0]=true,
Passing[0]=false
Head!=-1
ProxyReturn[0]?
p1=Head
ProxyToken[0][q1]!
ProxyToken[0][q1]!
Holding[0]=false,
Passing[0]=true
Token[p1]!
Head=-1,
Passing[0]=false
Token[q1]!
ProxyRequest[r][q1]!
Request[0][r][q1]!
Last[0]=p1
Init
p:int[0,N-1],q:int[0,N-1]
Request[p][0][q]?
p1=p,q1=q
q:int[0,N-1]
ProxyRequest[0][q]?
q1=q
Last[0]>=0
r=Last[0]
Last[0]==-1
Passing[0]==false
|| Head!=-1
Next[0]=q1, Last[0]=p1
Last[0]>=0
r = Last[0]
Last[0]==-1
Holding[0]==false
NextLow=q1
Passing[0]==true
&& Head==-1
Head=q1,
Last[0]=p1
Holding[0]==true
Holding[0]==false
Holding[0]==true
Holding[0]=false,
Last[0]=p1
P0
P1
P2
P3
P4
R0
R1
R2
R3
R4

Example


dmeMay24.xml - UPPAAL
Help
Drag out
Holding[0] = 0
Holding[1] = 1
Holding[2] = 0
Holding[3] = 0
Holding[4] = 0
Sink[0] = 0
Sink[1] = 0
Sink[2] = 0
Sink[3] = 0
Sink[4] = 1
Passing[0] = 0
Passing[1] = 0
Passing[2] = 0
Passing[3] = 0
Passing[4] = 0
Proxy = 1
Head = -1
NextLow = -1
Next[0] = -1
Next[1] = -1
Next[2] = -1
Next[3] = -1
Next[4] = -1
Last[0] = 1
Last[1] = -1
Last[2] = 1
Last[3] = 4
Last[4] = -1
Holding = true
0
1
2
Sink = true
3
4
P3
P4
R0
Idle
Requesting
Waiting
CS
P0
P1
P2
P3
P4
R0
R1
R2
R3
R4

Properties Checked

- Note: Starvation Freedom is not guaranteed for low-priority nodes.



Summary

- Generalized Algorithm generalizes all algorithms that impose a logical structure on the nodes.
- Using a star topology, the dag-based algorithm achieves performance that exceeds any known algorithm, even the centralized algorithm.
- Generalized Algorithm can be prioritized, and correctness is verified.
- Prioritized Algorithm is realized by passing token between priority levels and enqueuing low-priority requests in the token.

Summary

- Quiz #2 – this Friday, 4/25
- Final – take-home
- Project Presentations – start next week