Modely konkurentných systémov
Formálne metódy tvorby softvéru

Damas Gruska

Katedra aplikovanej informatiky, I20, gruska@fmph.uniba.sk

Prednáška 12.

CCS, synatx

Dané Act a procesové premenné $X, Y, Z, \ldots$.

Množina CCS termov:

$P ::=$	Nil	prázdny proces
$\mid$	X	procesová premenná
$\mid$	$x.P$	$x \in Act$ operácia prefixu
$\mid$	$P + Q$	nedeterministický výber P alebo Q
$\mid$	$P \mid Q$	paralelná kompozícia
$\mid$	$P \setminus L$	reštrikcia $L \subseteq A$
$\mid$	$P[f]$	premenovanie funkciou $f : A \rightarrow A$
$\mid$	μXP	rekurzia X je procesová premenná

Premonovávacia funkcia: $f : Act \rightarrow Act$ taká, že $f(\bar{a}) = \overline{f(a)}, f(\tau) = \tau$.

CCS, operačná sémantika

$$\frac{}{x.P \xrightarrow{x} P} \qquad \frac{P \xrightarrow{x} P'}{P + Q \xrightarrow{x} P',\; Q + P \xrightarrow{x} P'}$$

$$\frac{P \xrightarrow{u} P'}{P \mid Q \xrightarrow{u} P' \mid Q,\; Q \mid P \xrightarrow{u} Q \mid P'} \qquad \frac{P \xrightarrow{a} P',\, Q \xrightarrow{\bar{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{P \xrightarrow{x} P'}{P \setminus L \xrightarrow{x} P' \setminus L}, (x, \bar{x} \notin L) \qquad \frac{P \xrightarrow{x} P'}{P[f] \xrightarrow{f(x)} P'[f]}$$

$$\frac{P[\mu X P/X] \xrightarrow{x} P'}{\mu X P \xrightarrow{x} P'}$$

Bisimulácia

Dva procesy sa správajú rovnako, ak to, čo vie urobiť jeden vie urobiť aj druhý a výsledné procesy sa opäť správajú rovnako.

Definition

Binárna relácia $S \subseteq CCS \times CCS$ je (silná) bisimulácia, ak $(P, Q) \in S$ implikuje
1) ak $P \xrightarrow{x} P'$ tak existuje Q' také, že $Q \xrightarrow{x} Q'$ a platí $(P', Q') \in S$
2) ak $Q \xrightarrow{x} Q'$ tak existuje P' také, že $P \xrightarrow{x} P'$ a platí $(P', Q') \in S$

Slabá bisimulácia

Definition

Binárna relácia $S \subseteq CCS \times CCS$ je slabá bisimulácia, ak $(P, Q) \in S$ implikuje pre každé $x \in Act$

1) ak $P \xrightarrow{x} P'$ tak existuje Q' také, že $Q \overset{\hat{x}}{\Rightarrow} Q'$ a platí $(P', Q') \in S$

2) ak $Q \xrightarrow{x} Q'$ tak existuje P' také, že $P \overset{\hat{x}}{\Rightarrow} P'$ a platí $(P', Q') \in S$

Stopová ekvivalencia

Dva procesy sú stopovo ekvivalentné ($\sim_{trace}$) ak majú rovnaké stopy (stopa/trace procesu je postupnosť akcií, ktorú vie vykonať).

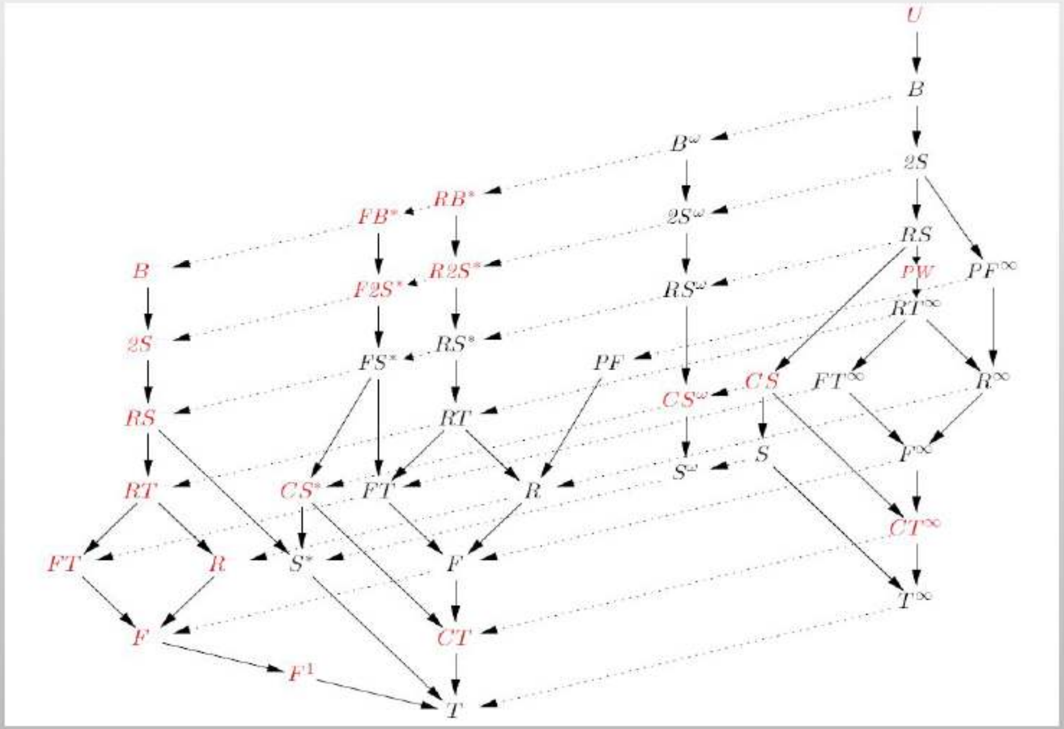
$Tr(P) = \{s | s \in Act^*$ také, že $P \xrightarrow{s}\}$

$Tr(Nil) = \{\epsilon\}$
$Tr(a.(.b.Nil + c.Nil)) = \{\epsilon, a, ab, ac\}$

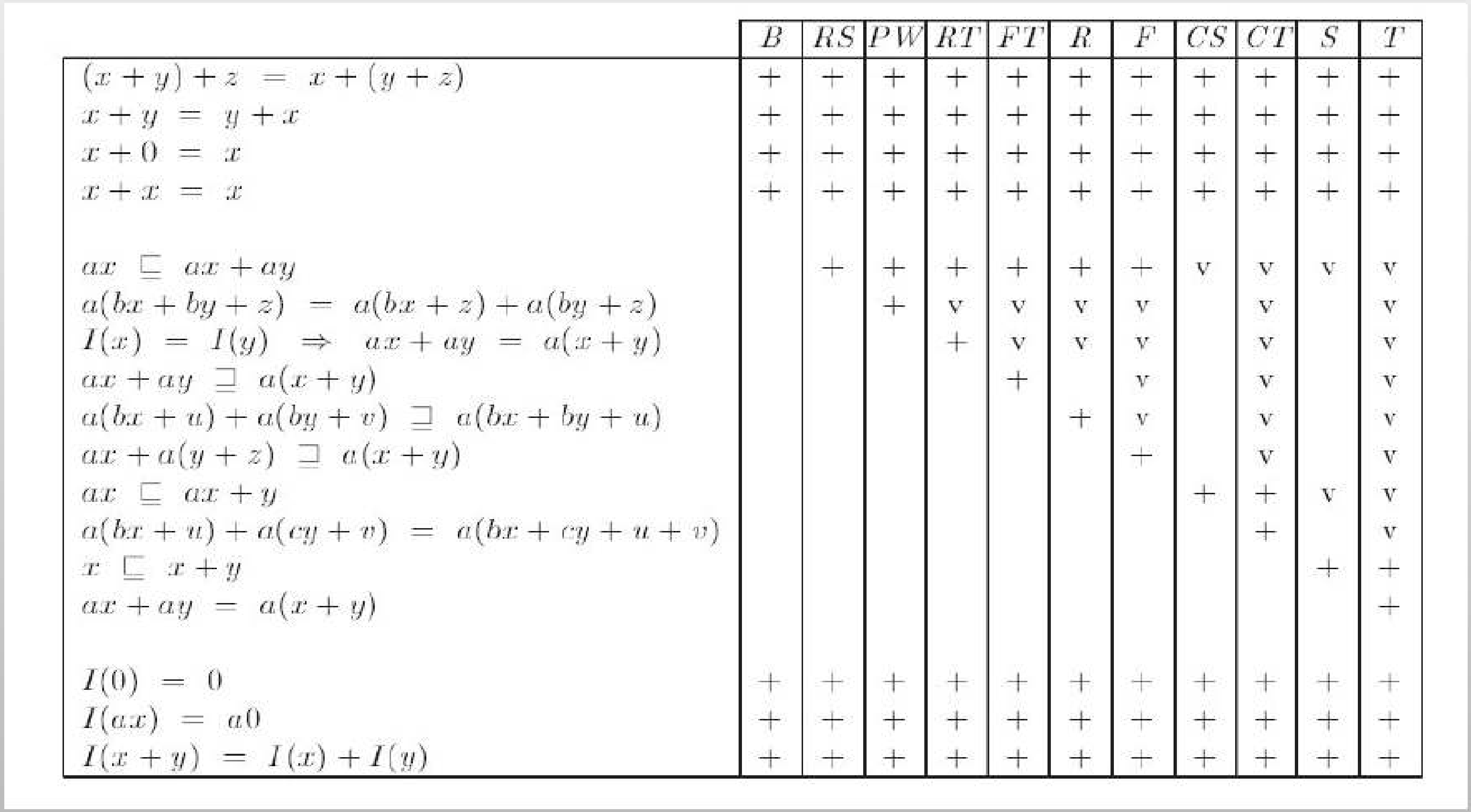
Definition

$P \sim_{trace} Q$ iff $Tr(P) = Tr(Q)$

Iné ekvivalencie

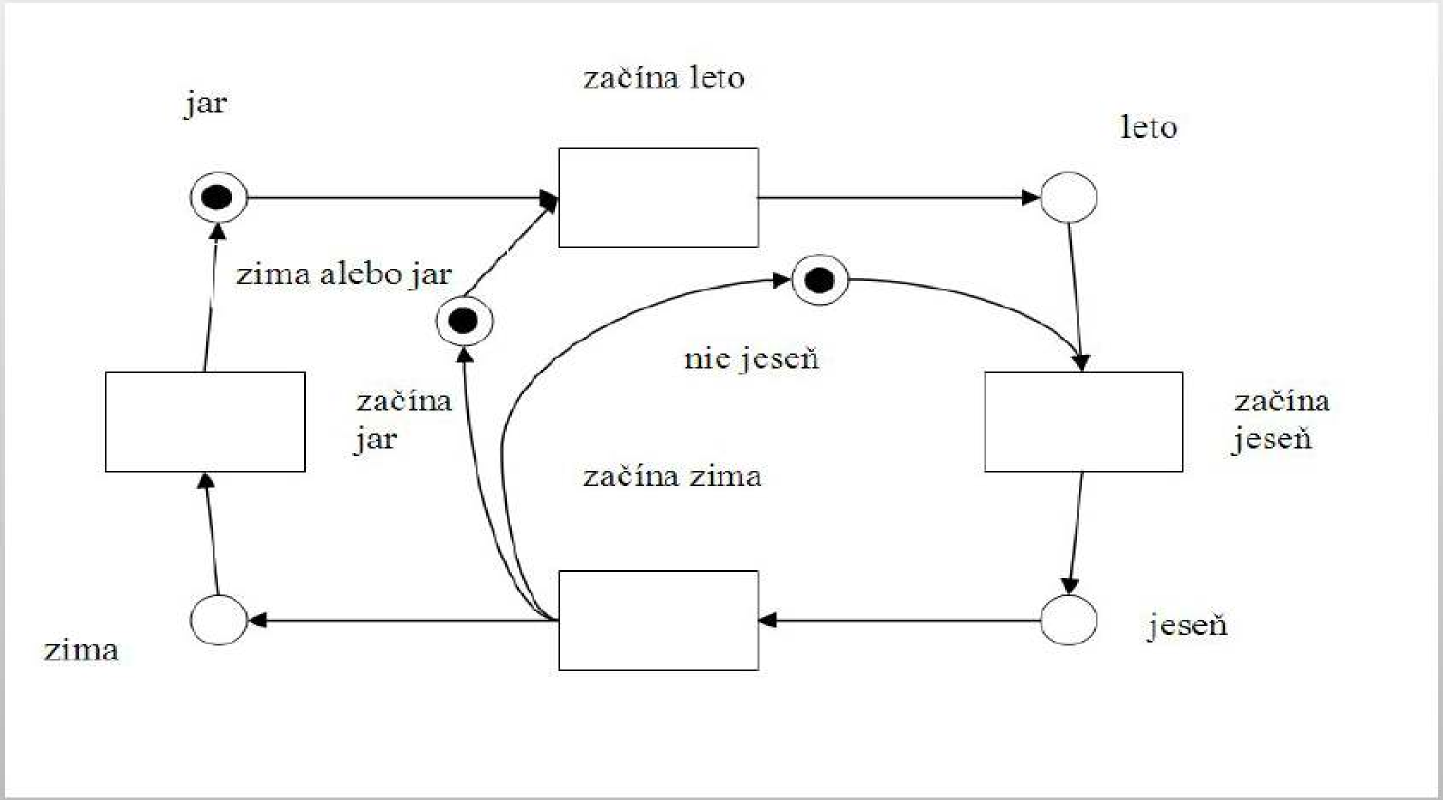


Iné ekvivalencie

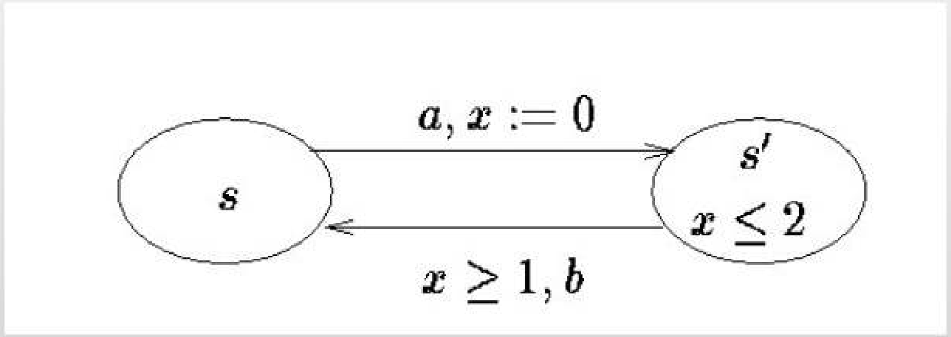


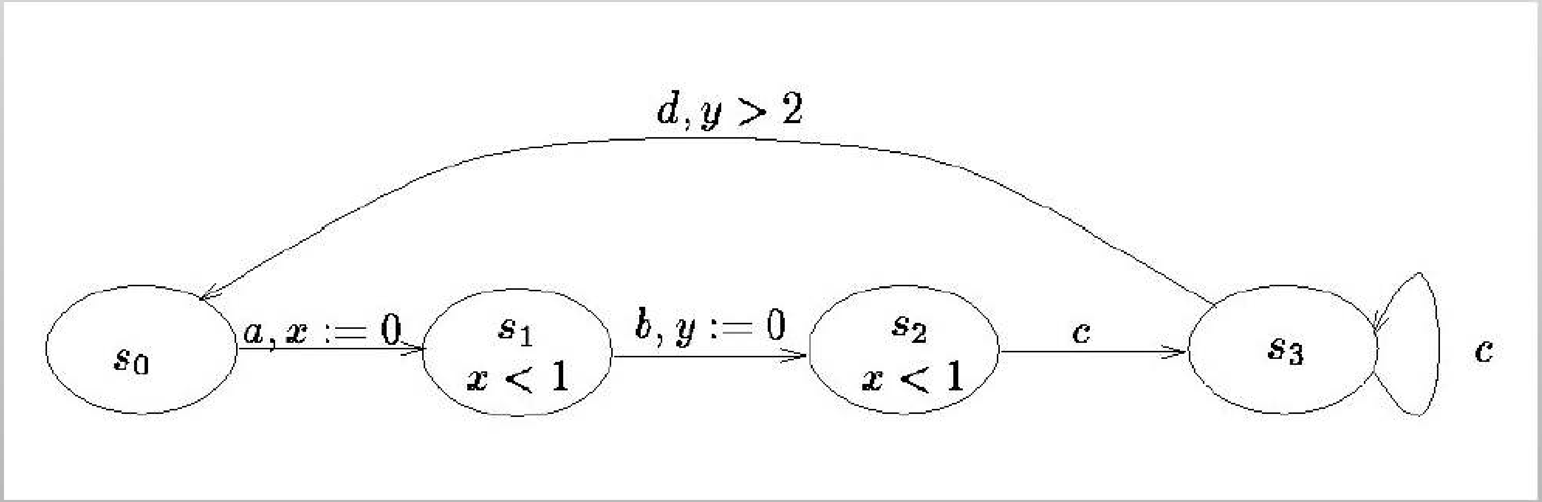
	B	RS	PW	RT	FT	R	F	CS	CT	S	T
$(x+y)+z \;=\; x+(y+z)$	+	+	+	+	+	+	+	+	+	+	+
$x+y \;=\; y+x$	+	+	+	+	+	+	+	+	+	+	+
$x+0 \;=\; x$	+	+	+	+	+	+	+	+	+	+	+
$x+x \;=\; x$	+	+	+	+	+	+	+	+	+	+	+
$ax \;\sqsubseteq\; ax+ay$		+	+	+	+	+	+	v	v	v	v
$a(bx+by+z) \;=\; a(bx+z)+a(by+z)$			+	v	v	v	v		v		v
$I(x) \;=\; I(y) \;\Rightarrow\; ax+ay \;=\; a(x+y)$				+	v	v	v		v		v
$ax+ay \;\sqsupseteq\; a(x+y)$					+		v		v		v
$a(bx+u)+a(by+v) \;\sqsupseteq\; a(bx+by+u)$						+	v		v		v
$ax+a(y+z) \;\sqsupseteq\; a(x+y)$							+		v		v
$ax \;\sqsubseteq\; ax+y$								+	+	v	v
$a(bx+u)+a(cy+v) \;=\; a(bx+cy+u+v)$									+		v
$x \;\sqsubseteq\; x+y$										+	+
$ax+ay \;=\; a(x+y)$											+
$I(0) \;=\; 0$	+	+	+	+	+	+	+	+	+	+	+
$I(ax) \;=\; a0$	+	+	+	+	+	+	+	+	+	+	+
$I(x+y) \;=\; I(x)+I(y)$	+	+	+	+	+	+	+	+	+	+	+

Petriho siete



Časové automaty

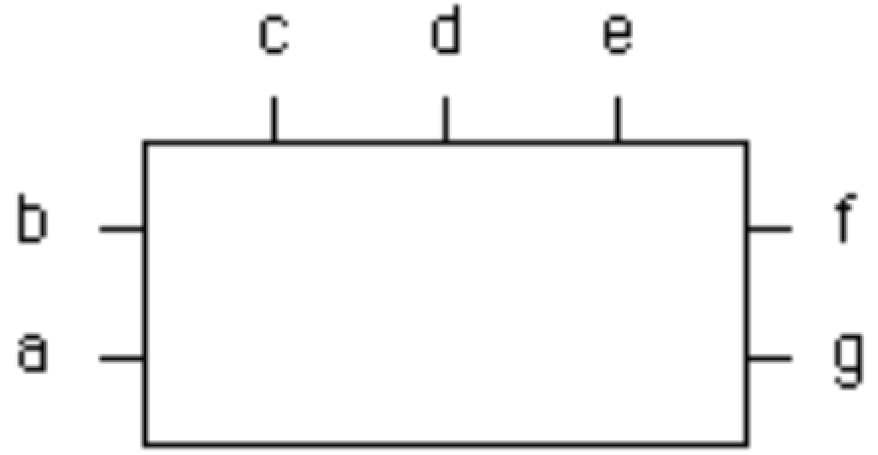




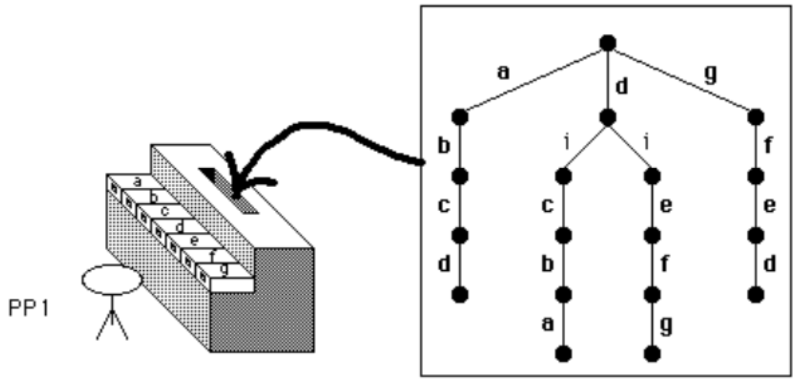
Name	Model Checking			Equivalence checking		GUI			Availability		
	Plain, Probabilistic, Stochastic,...	Modelling language	Properties language	Supported equivalences	Counter example generation	GUI	Graphical Specification	Counter example visualization	Software license	Programming language used	Platform / OS
APMC	Approximate Probabilistic	Reactive modules	PCTL, PLTL		No	Yes	No	No	FUSC	C	Unix & related
ARC	Plain	AltaRica	μ-calculus CTL*		Yes	Yes	No	No	FUSC	ANSI C	Unix & related
BANDERA	Code analysis	Java	CTL, LTL		Yes	Yes	Yes	Yes	Free	Java	Windows and Unix related
BLAST	Code analysis	C	Monitor automata		Yes	No	No	No	Free	OCaml	Windows and Unix related
CADENCE SMV	Plain	Cadence SMV, SMV, Verilog	CTL, LTL		Yes	Yes	No	No	FUSC	?	Windows and Unix related
CADP	Probabilistic	LOTOS, FSP, LOTOS NT	AFMC	SB, WB, BB, OE, STE, WTE, SE, tau*E	Yes	Yes	Yes	Yes	FUSC	?	MacOS, Linux, Solaris, Windows
CBMC	Code analysis	C, C++	Assertions		Yes	Yes	No	No	Free	C++	Windows and Unix related
CPAchecker	Code analyses	C	Monitor automata		Yes	Yes	No	Yes	Free	Java	Any
CWB-NC	Plain and Timed	CCS, CSP, LOTOS, TCCS	AFMC, CTL, GCTL	SB, WB, me, ME	Yes	Yes	No	No	FUSC	SML	Windows and Unix related, MacOS
DBRover	Timed	Ada, C, C++, Java, VHDL, Verilog	LTL, MTL		No	Yes	Yes	Yes	Non-free/Commercial use only	?	Windows and Unix related
DiVinE Tool	Plain	DVE input language, C/C++ (via LLVM bitcode), Timed automata	LTL		Yes	Yes	No	Yes	Free	C/C++	Unix, Windows
DREAM	Real-time	C++, Timed automata	Monitor automata		Yes	No	No	No	Free	C++	Windows and Unix related
Edinburgh CWB	Plain	CCS, TCCS, SCCS	μ-calculus	SB, WB, BB, me, ME, OE	Yes	No	No	No	FUSC	SML	Windows and Unix related
EmbeddedValidator	Hybrid	Simulink/Stateflow/TargetLink/C	Monitor automata		Yes	Yes	Yes	Yes	Non-free/Commercial use only	?	Windows
Expander2	Hybrid		AFMC, CTL	SB, OE	No	Yes	No	No	Free	O'Haskell	Unix related
Fc2Tools	Plain	FC2	?	SB, WB, BB	Yes	No	Yes	Yes	Free	?	Unix related
GEAR	Plain	?	AFMC, CTL, μ-calculus		Yes	Yes	Yes	Yes	Free	Java	Windows and Unix related
ImProve	Plain	Haskell	Assertions		Yes	No	No	No	Free	Haskell	Linux, Windows, MacOS
Java Pathfinder	Plain and timed	Java	unknown		No	Yes	No	No	NOSA	Java	MacOS, Windows, Linux
LLBMC	Code analysis	C (, C++, all languages supported by LLVM)	Assertions		Yes	No	No	No	FUSC	C++	Windows and Unix related
LTSA	Plain	FSP	LTL		Yes	Yes	No	Yes	Free	Java	Windows and Unix related
LTSmin	Plain, Real-time	Promela, μCRL, mCRL2, DVE Input Language	μ-calculus, LTL, CTL*	SB, BB	Yes	No	No	No	Free	C, C++	Unix, MacOSX, Windows
MCMAS	Plain, Epistemic	ISPL	CTL, CTLK		Yes	Yes	No	Yes	Free	C++	Unix, Windows, MacOS
mCRL2	Plain, Real-time	mCRL2	mCRL2 mu-calculus	SB, BB, HE, STE, WTE	Yes	Yes	No	Yes	Free	C++	MacOS, Linux, Solaris, Windows
MRMC	Real-time, Probabilistic	Plain MC	CSL, CSRL, PCTL, PRCTL	SB	No	No	No	No	Free	C	Windows, Linux, MacOS
NuSMV	Plain	SMV	CTL, LTL, PSL		Yes	No	No	No	Free	C	Unix, Windows, MacOSX
ompca, OpenMP C Analyzer	software symbolic simulation with API control	C/C++ programs with OpenMP directives	logic predicates or flexible procedures through API		Yes	Yes	No	Yes	Free	C, C++	Ubuntu Linux, Windows version available soon
PAT	Plain,Real-time,Probabilistic	CSP#, Timed CSP, Probabilistic CSP	LTL, Assertions		Yes	Yes	Yes	Yes	Free	C#	Windows, other OS with Mono
PRISM	Probabilistic	PEPA, PRISM language, Plain MC	CSL, PLTL, PCTL		No	Yes	No	No	Free	C++, Java	Windows, Linux, MacOS
Reactis Tester	Hybrid	Simulink/Stateflow	?		No	Yes	Yes	No	Non-free/Commercial use only	SML	Windows, Linux
RED	dense-time, linear-hybrid, fully symbolic	communicating timed automata (CTA), linear-hybrid automata (LHA)	TCTL with fairness assumptions, CTA with fairness assumptions	timed simulation, fair simulation	Yes	Yes	Yes	Yes	Free	C/C++	Ubuntu Linux
SATABS	Code analysis	C, C++	Assertions		Yes	Yes	No	No	Free	C++	Windows and Unix related
SLMC	Plain	pi-calculus	CGL		Yes	No	No	No	Free	OCAML	Windows and Unix related
SPIN	Plain	Promela	LTL		Yes	Yes	No	Yes	FUSC	C, C++	Windows and Unix related
Spot	Plain	Petri nets, DVE Input Language	LTL, PSL subset		Yes	No	No	No	Free	C, C++	Unix & related
TAPAAL	Real-time	Timed-Arc Petri Nets, age invariants, inhibitor arcs, transport arcs	TCTL subset		No	Yes	Yes	Yes	Free	C++, Java	MacOS, Windows, Linux
TAPAs	Plain	CCSP	CTL, μ-calculus	SB, WB, BB, STE, WTE, me, ME, OE	Yes	Yes	Yes	Yes	Free	Java	Windows, MacOS and Unix related
UPPAAL	Real-time	Timed automata, C subset	TCTL subset		Yes	Yes	Yes	Yes	FUSC	C++, Java	MacOS, Windows, Linux
ROMEO	Real-time	Time Petri Nets, stopwatch parametric Petri nets	TCTL subset		Yes	Yes	Yes	No	Free	C++, tcl/tk	MacOS, Windows, Linux
TLC	Plain	TLA+, PlusCal	TLA		Yes	Yes	Yes	No	Free	Java	Windows, Linux

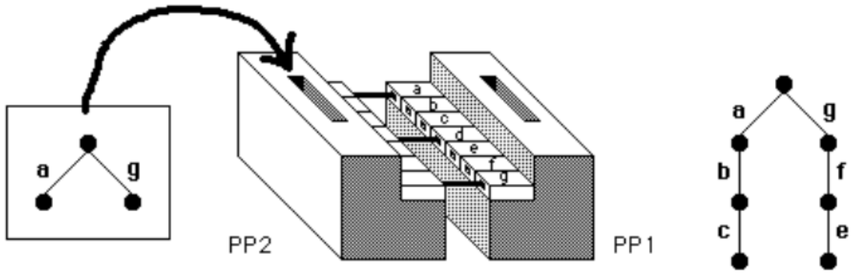
LOTOS

Proces ako “black box” a jeho porty

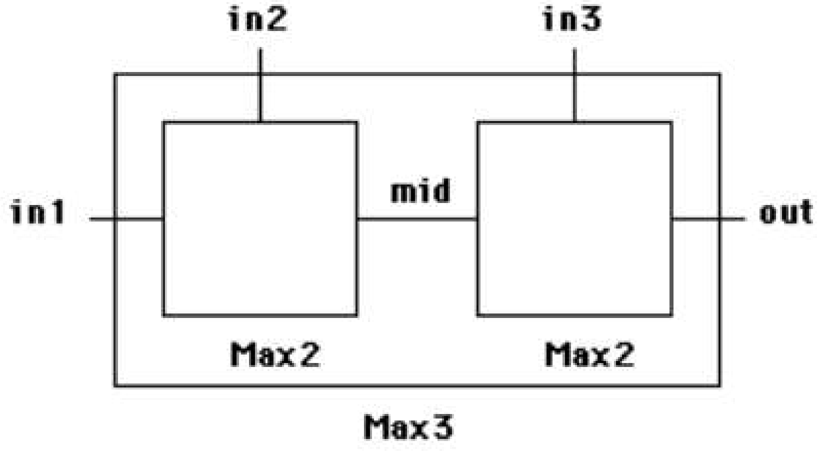


PP1[a,b,c,d,e,f,g]

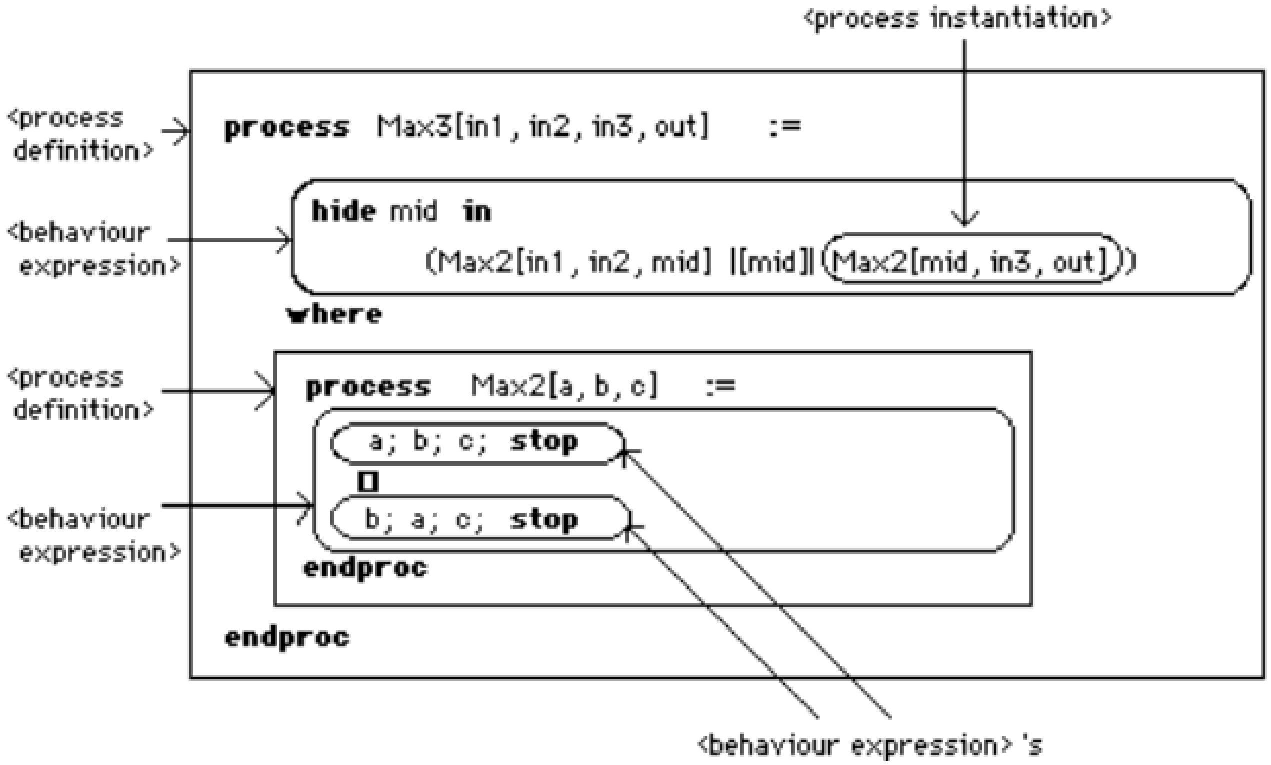




PP1[a,b,c,d,e,f,g] |[a,d,g] PP2[a,b,c,d,e,f,g]



```
process Max3 [in1, in2, in3, out] :=
        hide mid in
                (Max2[in1, in2, mid]
                | [mid] |
                Max2[mid, in3, out]
                )
        where ...
endproc (* Max3 *)
```



<process instantiation>
<process definition>
process Max3[in1, in2, in3, out] :=
<behaviour expression>
hide mid in
(Max2[in1, in2, mid] |[mid]| (Max2[mid, in3, out]))
where
<process definition>
process Max2[a, b, c] :=
a; b; c; stop
[]
<behaviour expression>
b; a; c; stop
endproc
endproc
<behaviour expression> 's

NAME	SYNTAX
inaction	**stop**
action prefix	
- unobservable (internal)	**i** ; B
- observable	g; B
choice	B1 **[]** B2
parallel composition	
- general case	B1 |[g_1, ... , g_n]| B2
- pure interleaving	B1 ||| B2
- full synchronization	B1 || B2
hiding	**hide** g_1, ... , g_n **in** B
process instantiation	p [g_1, ... ,g_n]
successful termination	**exit**
sequential composition (enabling)	B1 >> B2
disabling	B1 [> B2

$$B \xrightarrow{x} B'$$

G	denote the set of *user-definable* gates;
$g, g_1,\ldots,g_n$	range over G;
i	denote the unobservable action;
Act	denote the set $G \cup \{i\}$ of*user definable* actions;
μ	range over Act.
δ	be the successful termination action
G^+	be the set $G \cup \{\delta\}$ of observable actions
g^+	range over G^+
Act^+	be the set $Act \cup \{\delta\}$ of actions
μ^+	range over Act^+.

==============

$\mu;B \xrightarrow{\mu} B$

==============

===

$B1 \xrightarrow{\mu^+} B1'$	*implies*	$B1 \ [] \ B2 \xrightarrow{\mu^+} B1'$
$B2 \xrightarrow{\mu^+} B2'$	*implies*	$B1 \ [] \ B2 \xrightarrow{\mu^+} B2'$

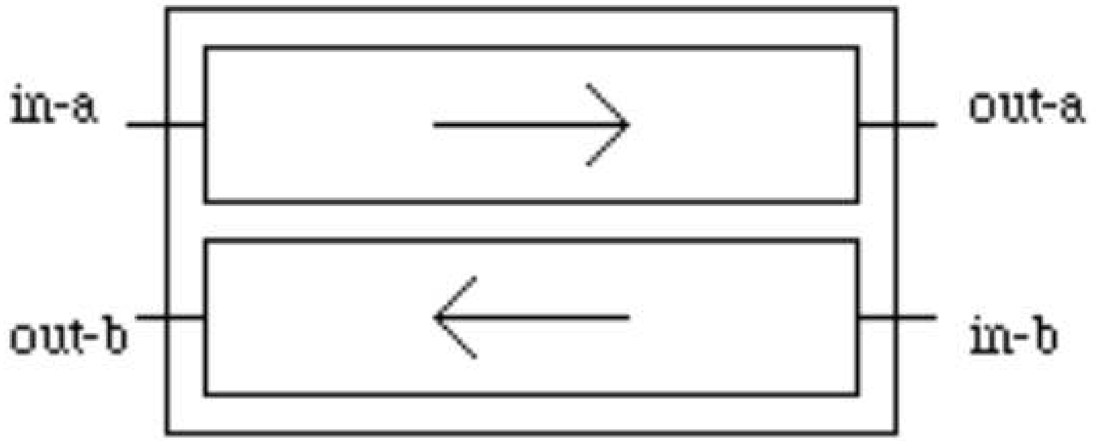
===

a; b; c; **stop** $\xrightarrow{a}$ b; c; **stop**

a; b; c; **stop** $\xrightarrow{a}$ b; c; **stop**

implies

a; b; c; **stop** [] b; a; c; **stop** $\xrightarrow{a}$ b; c; **stop**



A simple, full-duplex buffer

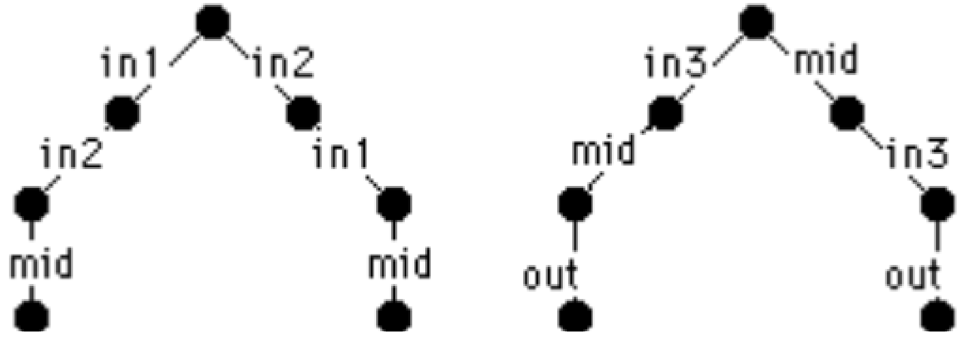
```
process duplex-buffer [in-a, in-b, out-a, out-b] :=
            in-a;     (in-b;    (  out-a; out-b; stop
                                [] out-b; out-a; stop)
                      [] out-a; in-b; out-b; stop)
     []     in-b;     (in-a;    (  out-a; out-b; stop
                                [] out-b; out-a; stop)
                      [] out-b; in-a; out-a; stop)
endproc
```

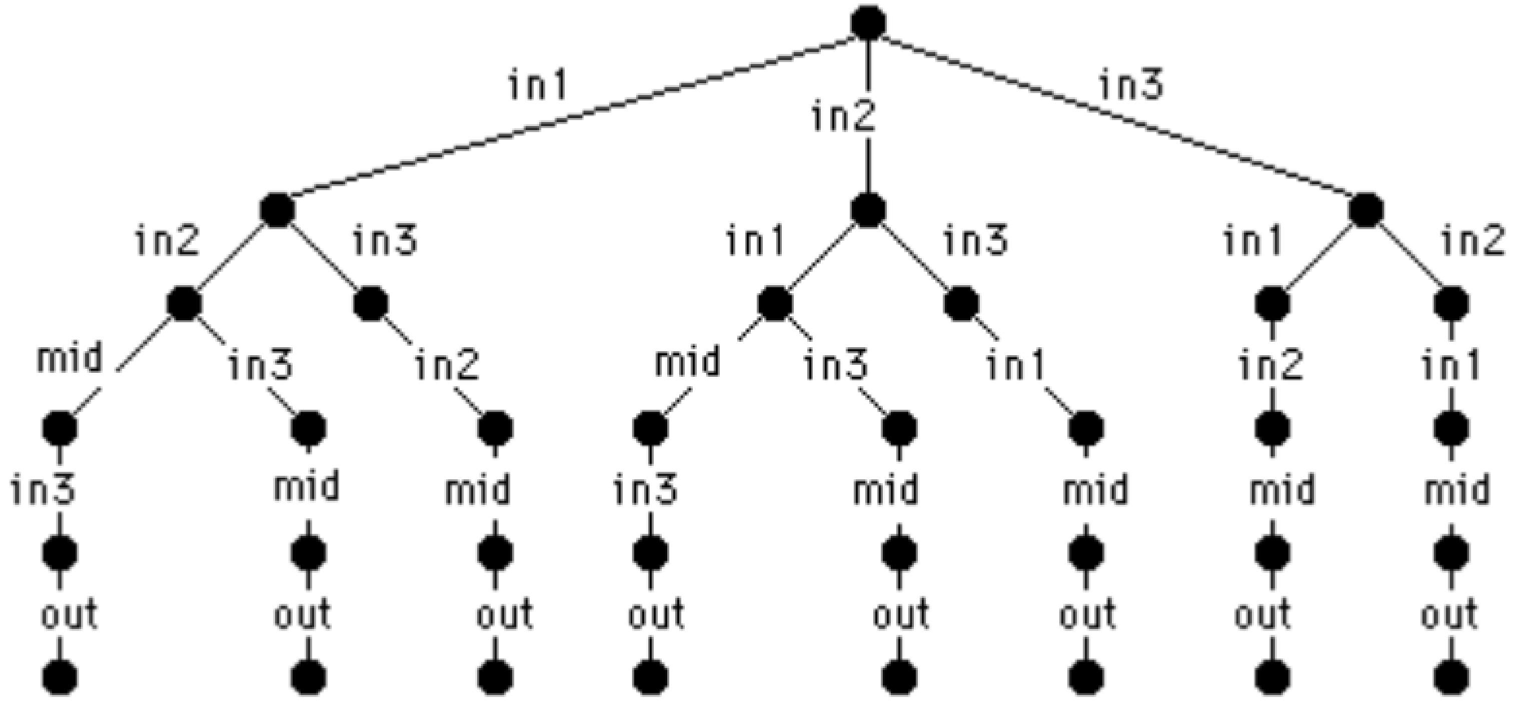
$$B1\ ||[\mathbf{g_1, ..., g_n}]||\ B2$$

$S = [g_1, ..., g_n]$

$B1 \xrightarrow{\mu} B1'$ and $\mu \notin S$	*implies*	$B1\|S\|B2 \xrightarrow{\mu} B1\|S\|B2$
$B2 \xrightarrow{\mu} B2'$ and $\mu \notin S$	*implies*	$B1\|S\|B2 \xrightarrow{\mu} B1\|S\|B2'$
$B1 \xrightarrow{g^+} B1'$ and $B2 \xrightarrow{g^+} B2'$ and $g^+ \in S \cup \{\delta\}$	*implies*	$B1\|S\|B2 \xrightarrow{g^+} B1'\|S\|B2'$

'Max2[in1, in2, mid] |[mid]| Max2[mid, in3, out]'

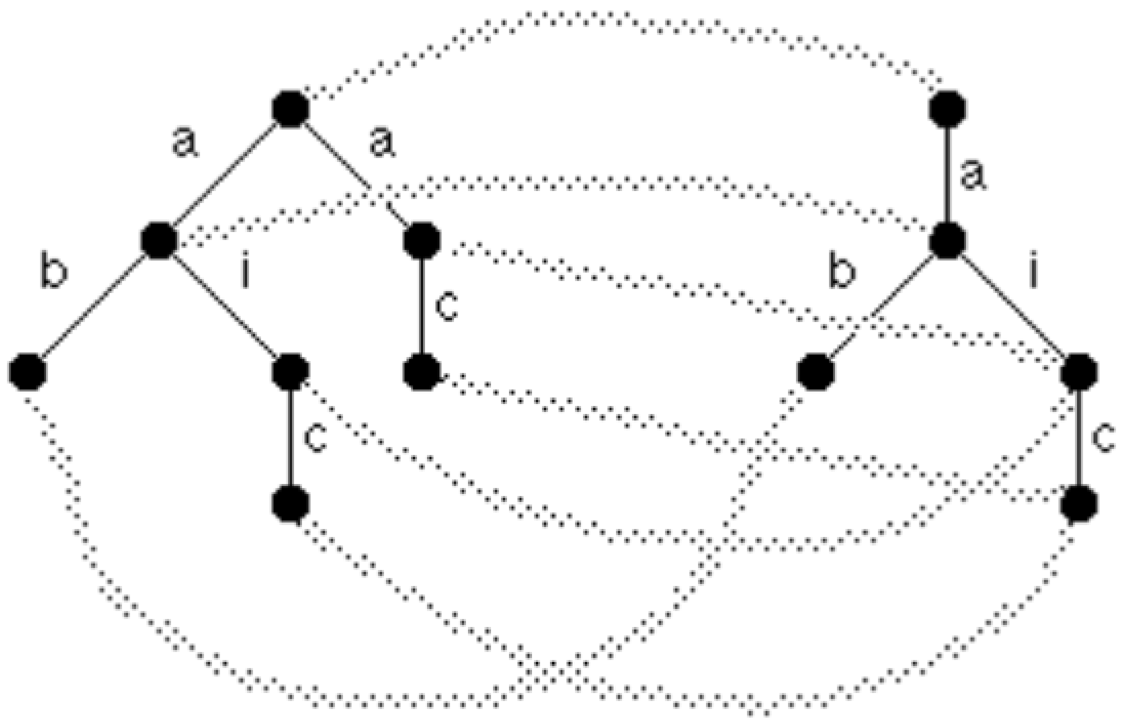




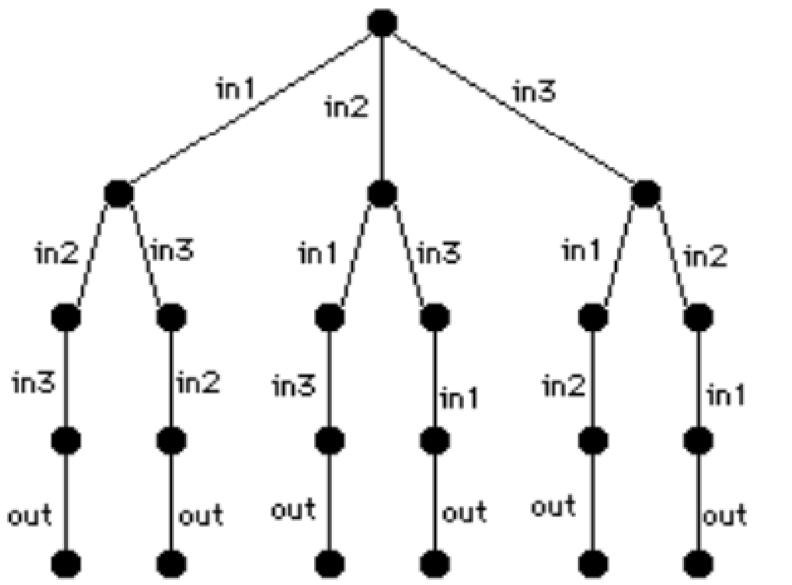
```
process reusable-simplex-buffer [in, out] :=
                    in; out; reusable-simplex-buffer [in, out]
endproc

process same-simplex-buffer [in, out] :=
                    in; same-simplex-buffer [out, in]
endproc
```

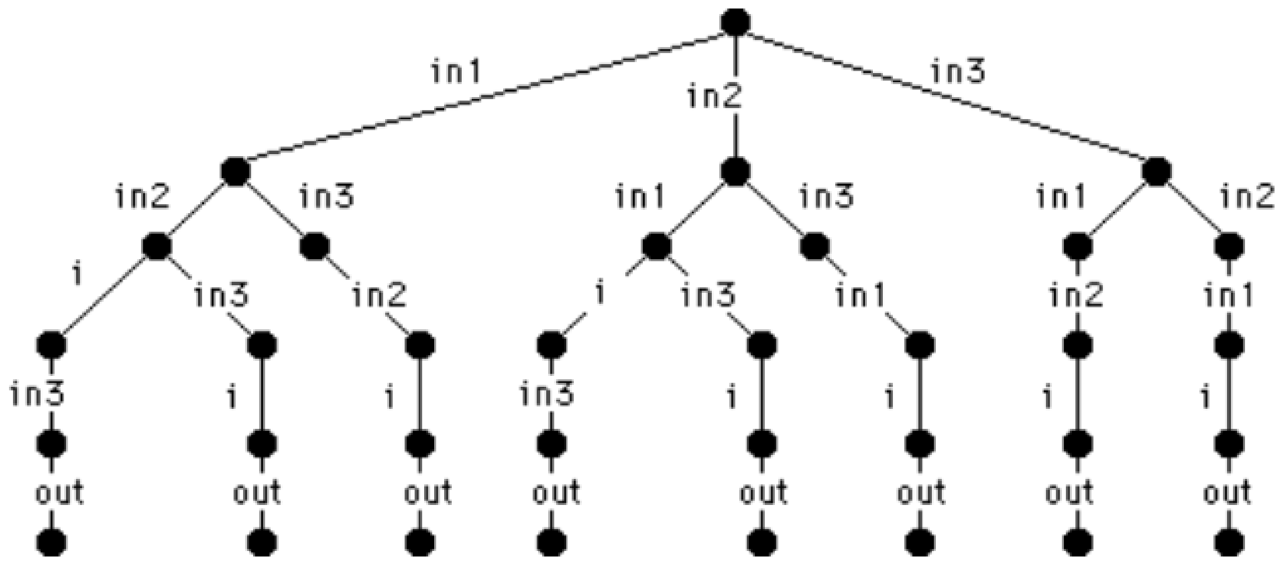
$B1 \xrightarrow{\mu} B1'$	*implies*	$B1 >> B2 \xrightarrow{\mu} B1' >> B2$
$B1 \xrightarrow{\delta} B1'$	*implies*	$B1 >> B2 \xrightarrow{i} B2$

$B1 \xrightarrow{\mu} B1'$	*implies*	$B1\ [>\ B2 \xrightarrow{\mu} B1'\ [>\ B2$
$B1 \xrightarrow{\delta} B1'$	*implies*	$B1\ [>\ B2 \xrightarrow{\delta} B1'$
$B2 \xrightarrow{\mu^+} B2'$	*implies*	$B1\ [>\ B2 \xrightarrow{\mu^+} B2'$


a
a
a
b
i
c
b
i
c
c

```
Process  Max3-Spec [in1, in2, in3, out] :=
      in1;   (   in2, in3, out, stop
                 []  in3, in2, out, stop )
[]    in2;   (   in1, in3, out, stop
                 []  in3, in1, out, stop )
[]    in3;   (   in1, in2, out, stop
                 []  in2, in1, out, stop )
endproc


Process Max3 [in1, in2, in3, out] :=

            hide mid in

                       (Max2[in1, in2, mid] |[mid]| Max2[mid, in3, out])
where
           process Max2 [a, b, c] :=
                       a; b; c; stop
                       []
                       b; a; c; stop
           endproc
endproc
```


in1
in2
in3
in2
in3
in1
in3
in1
in2
in3
in2
in3
in1
in2
in1
out
out
out
out
out
out

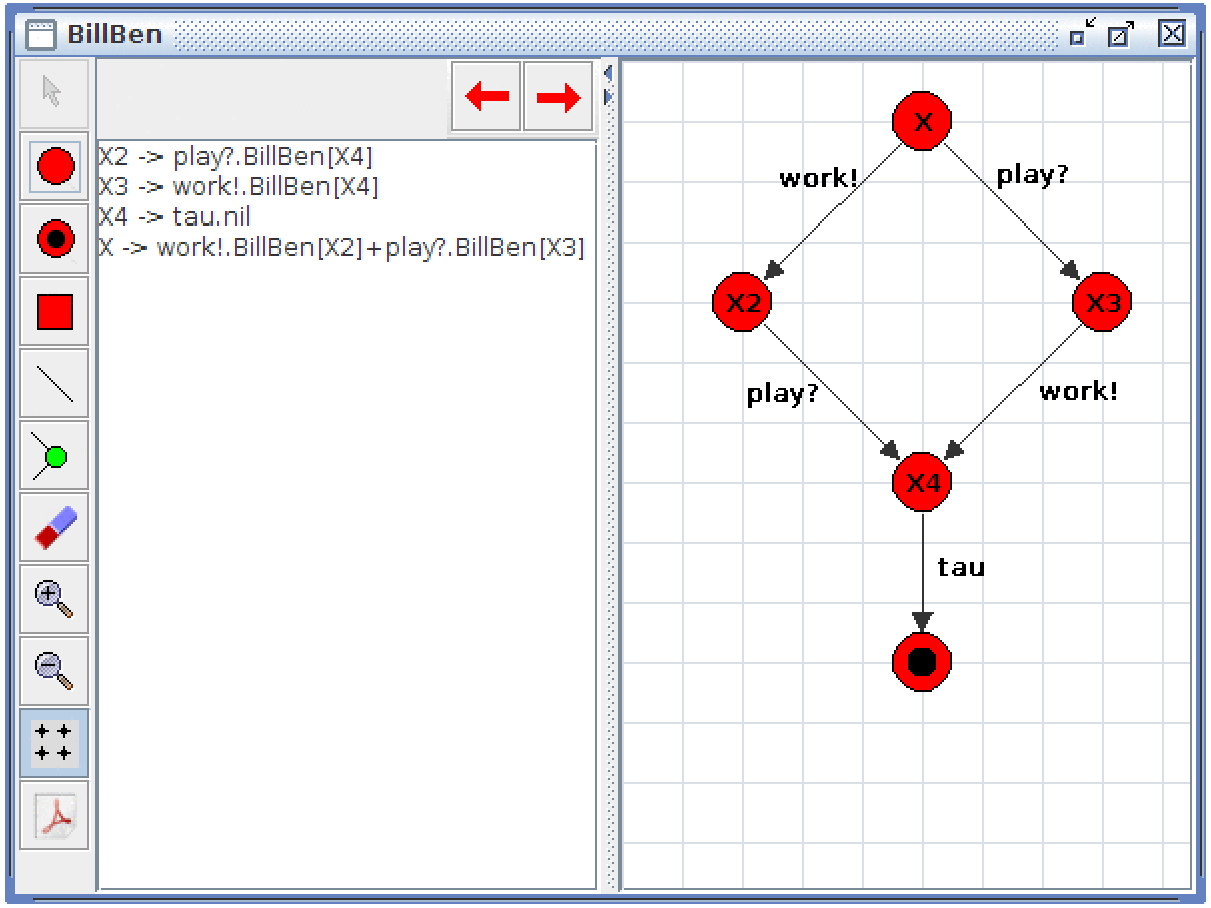

in1
in2
in3
in2
in3
in1
in3
in1
in2
i
in3
in2
i
in3
in1
in2
in1
in3
i
i
in3
i
i
i
i
out
out
out
out
out
out
out
out

```
Specification Max3 [in1, in2, in3, out]:noexit

 type natural is
        sorts    nat
        opns     zero:      → nat
                 succ:      nat → nat
                 largest: nat, nat → nat
        eqns     ofsort nat
                 forall x:nat
                           largest(zero, x) = x
                           largest(x, y) = largest(y, x)
                           largest(succ(x), succ(y)) = succ(largest(x, y))
 endtype (* natural *)

 behaviour

        hide mid in
                 (Max2[in1, in2, mid] |[mid]|  Max2[mid, in3, out])

 where

        process Max2[a, b, c] : noexit :=
                 a ?x:nat;   b ?y:nat;  c !largest(x,y); stop
                 []
                 b ?y:nat;   a ?x:nat;  c !largest(x,y); stop
        endproc (*Max2*)

 endspec (*Max3*)
```

CADP

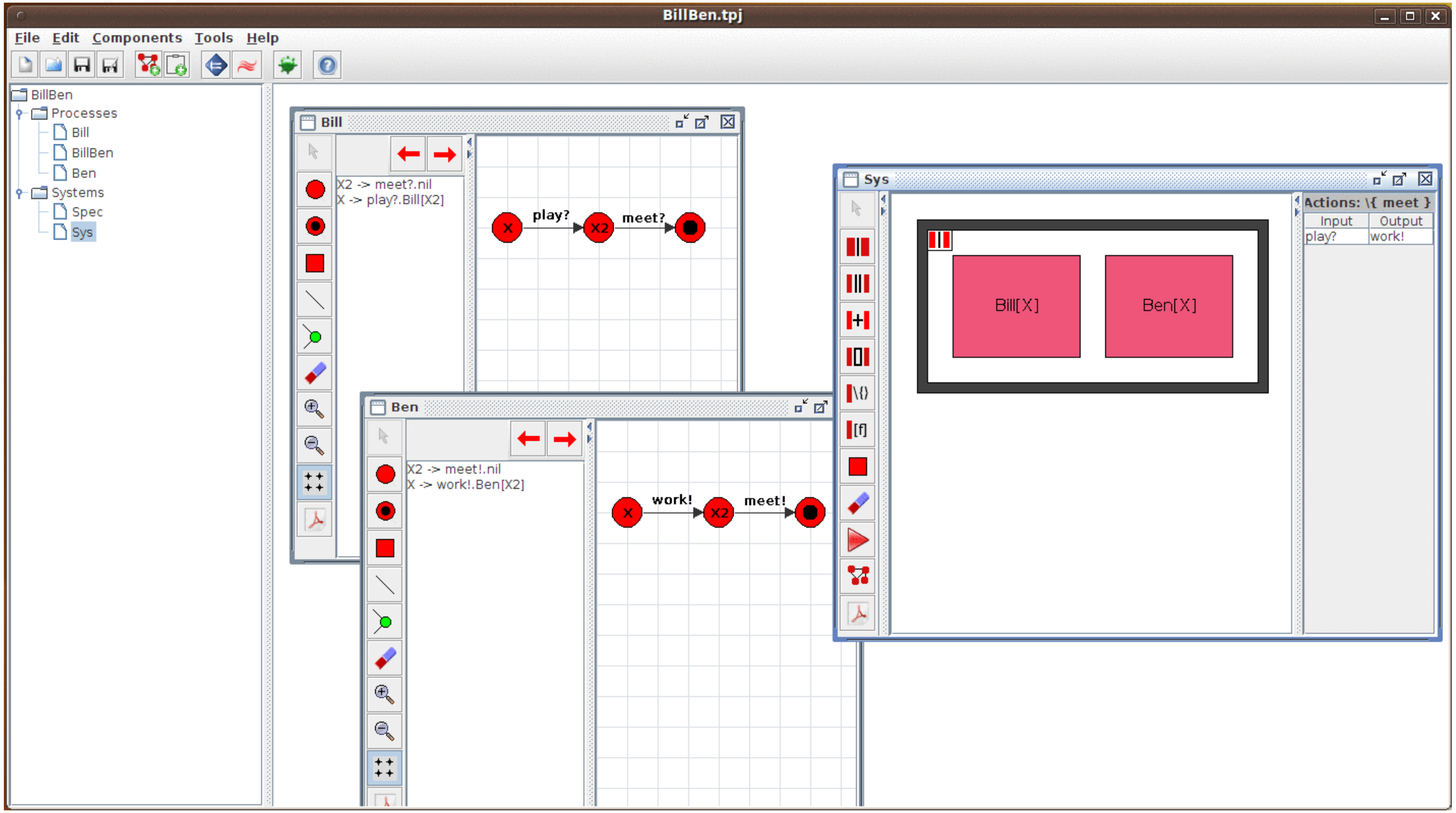
- Formal specification languages
- Verification paradigms:
  - Model checking (modal μ-calculus)
  - Equivalence checking (bisimulations)
  - Visual checking (graph drawing)
- Verification techniques:
  - Reachability analysis
  - On-the-fly verification
  - Compositional verification
  - Distributed verification
  - Static analysis
- Other features:
  - Step-by-step simulation
  - Rapid prototyping
  - Test-case generation
  - Performance evaluation

TAPAS



```
X2 -> play?.BillBen[X4]
X3 -> work!.BillBen[X4]
X4 -> tau.nil
X -> work!.BillBen[X2]+play?.BillBen[X3]
```

TAPAS

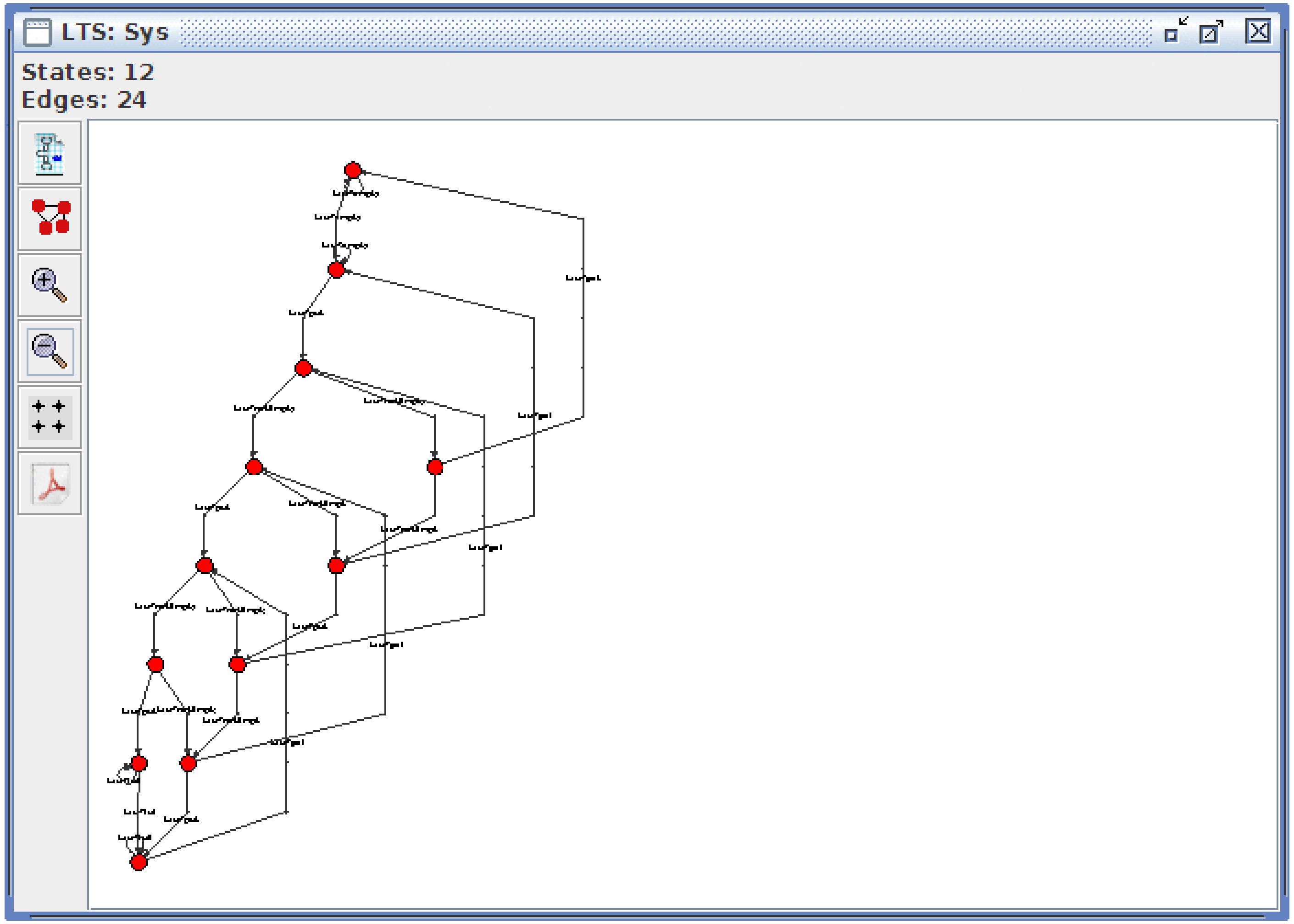


```
X2 -> meet?.nil
X -> play?.Bill[X2]
```

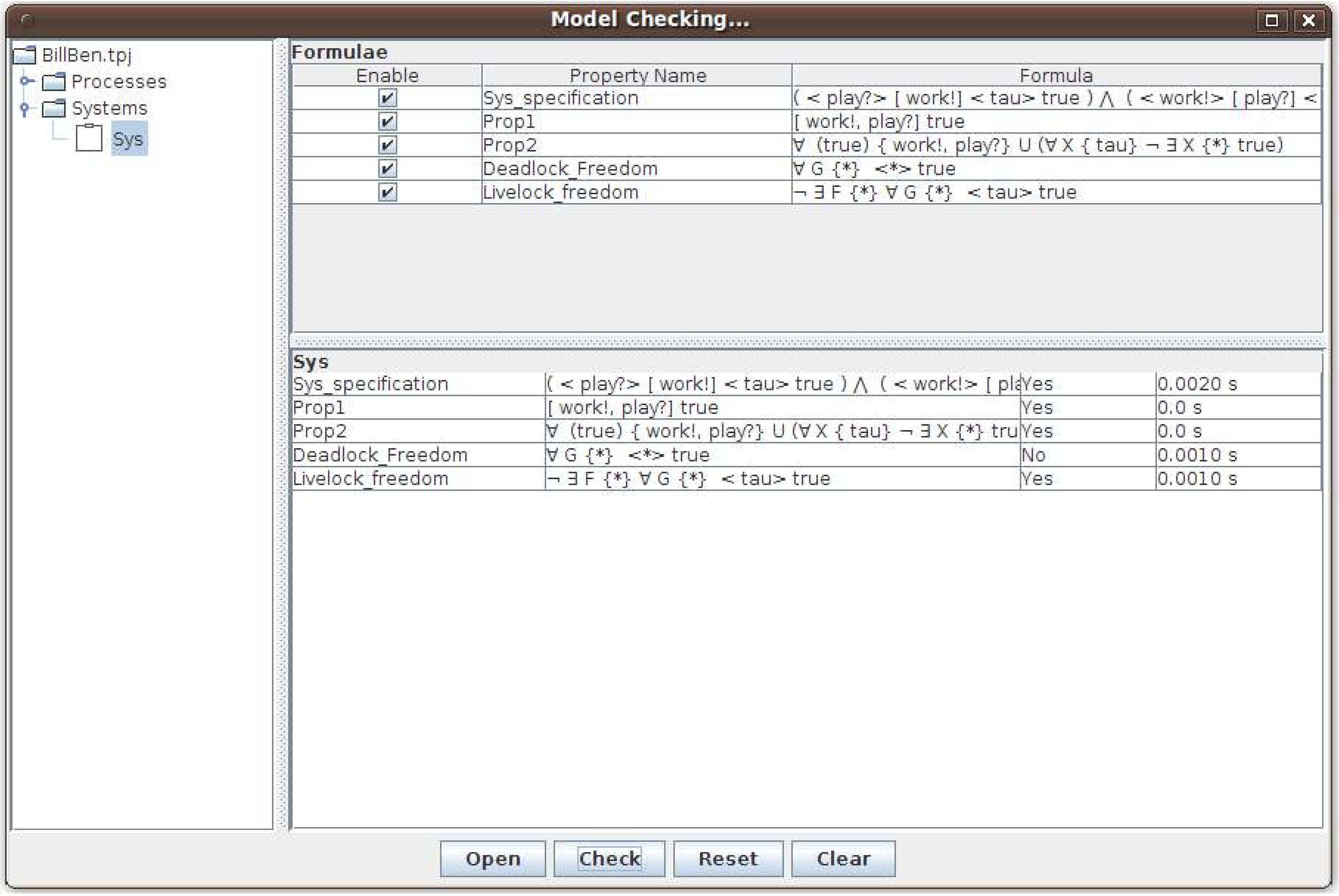
```
X2 -> meet!.nil
X -> work!.Ben[X2]
```

Actions: \{ meet }

Input	Output
play?	work!



States: 12
Edges: 24

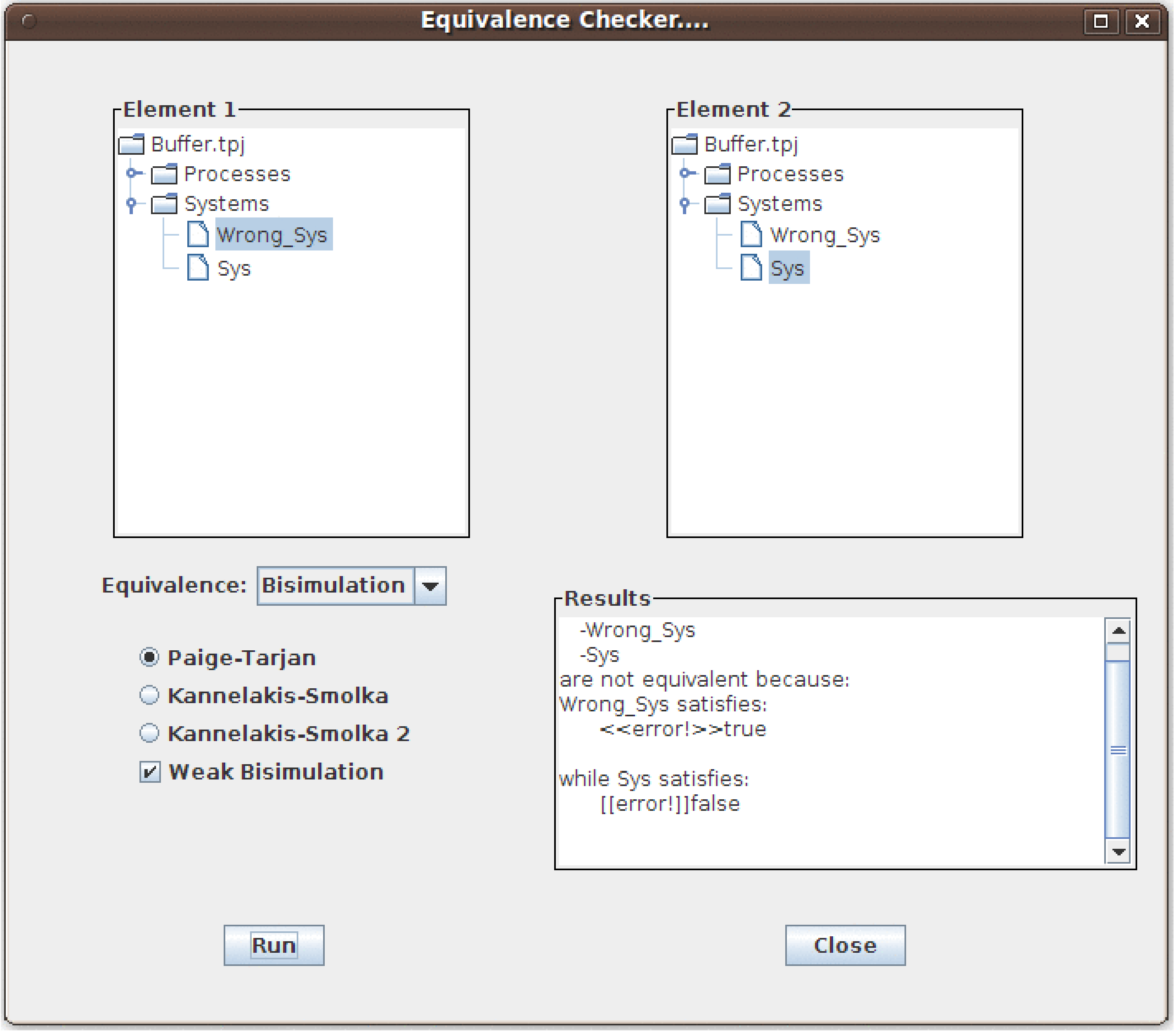


Formulae

Enable	Property Name	Formula
✓	Sys_specification	(< play?> [work!] < tau> true) ∧ (< work!> [play?] <
✓	Prop1	[work!, play?] true
✓	Prop2	∀ (true) { work!, play?} U (∀ X { tau} ¬ ∃ X {*} true)
✓	Deadlock_Freedom	∀ G {*} <*> true
✓	Livelock_freedom	¬ ∃ F {*} ∀ G {*} < tau> true

Sys

Sys_specification	(< play?> [work!] < tau> true) ∧ (< work!> [pla	Yes	0.0020 s
Prop1	[work!, play?] true	Yes	0.0 s
Prop2	∀ (true) { work!, play?} U (∀ X { tau} ¬ ∃ X {*} tru	Yes	0.0 s
Deadlock_Freedom	∀ G {*} <*> true	No	0.0010 s
Livelock_freedom	¬ ∃ F {*} ∀ G {*} < tau> true	Yes	0.0010 s

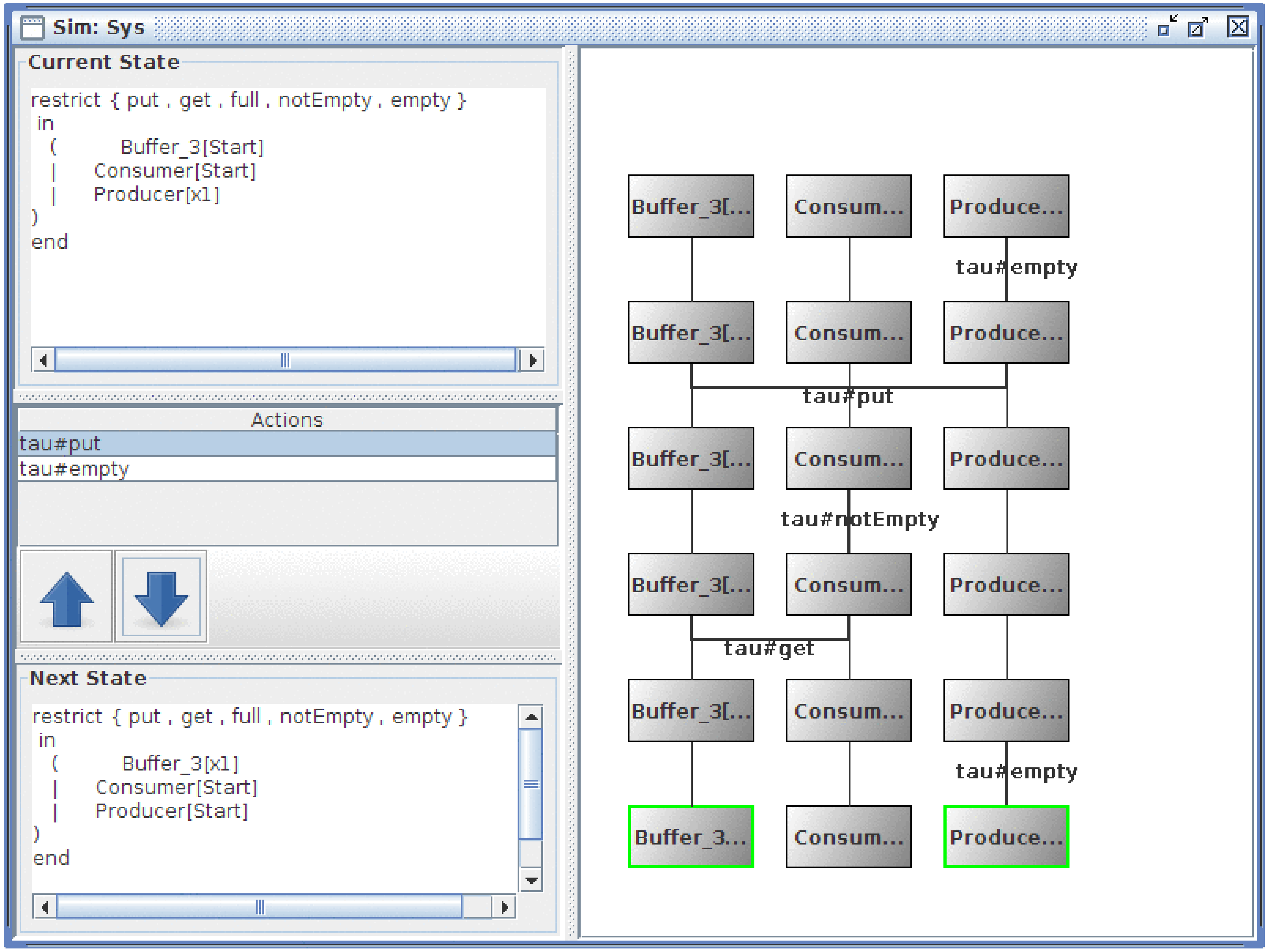


Equivalence: Bisimulation

[✓ Paige-Tarjan]
Kannelakis-Smolka
Kannelakis-Smolka 2
[✓ Weak Bisimulation]

Results

```
  -Wrong_Sys
  -Sys
are not equivalent because:
Wrong_Sys satisfies:
     <<error!>>true

while Sys satisfies:
     [[error!]]false
```


Sim: Sys

Current State

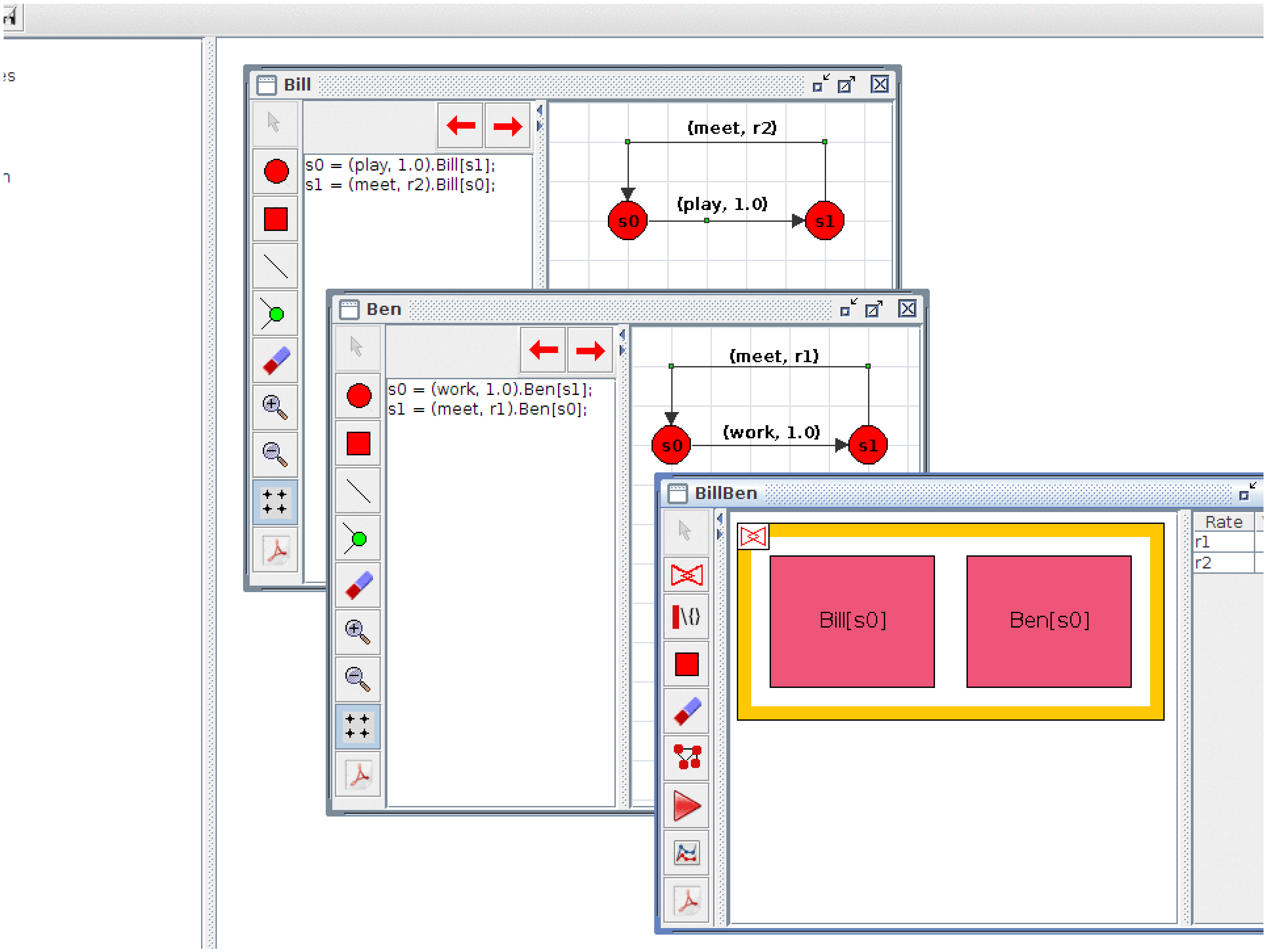
```
restrict { put , get , full , notEmpty , empty }
 in
  (        Buffer_3[Start]
  |     Consumer[Start]
  |     Producer[x1]
)
end
```

Actions

tau#put
tau#empty

Next State

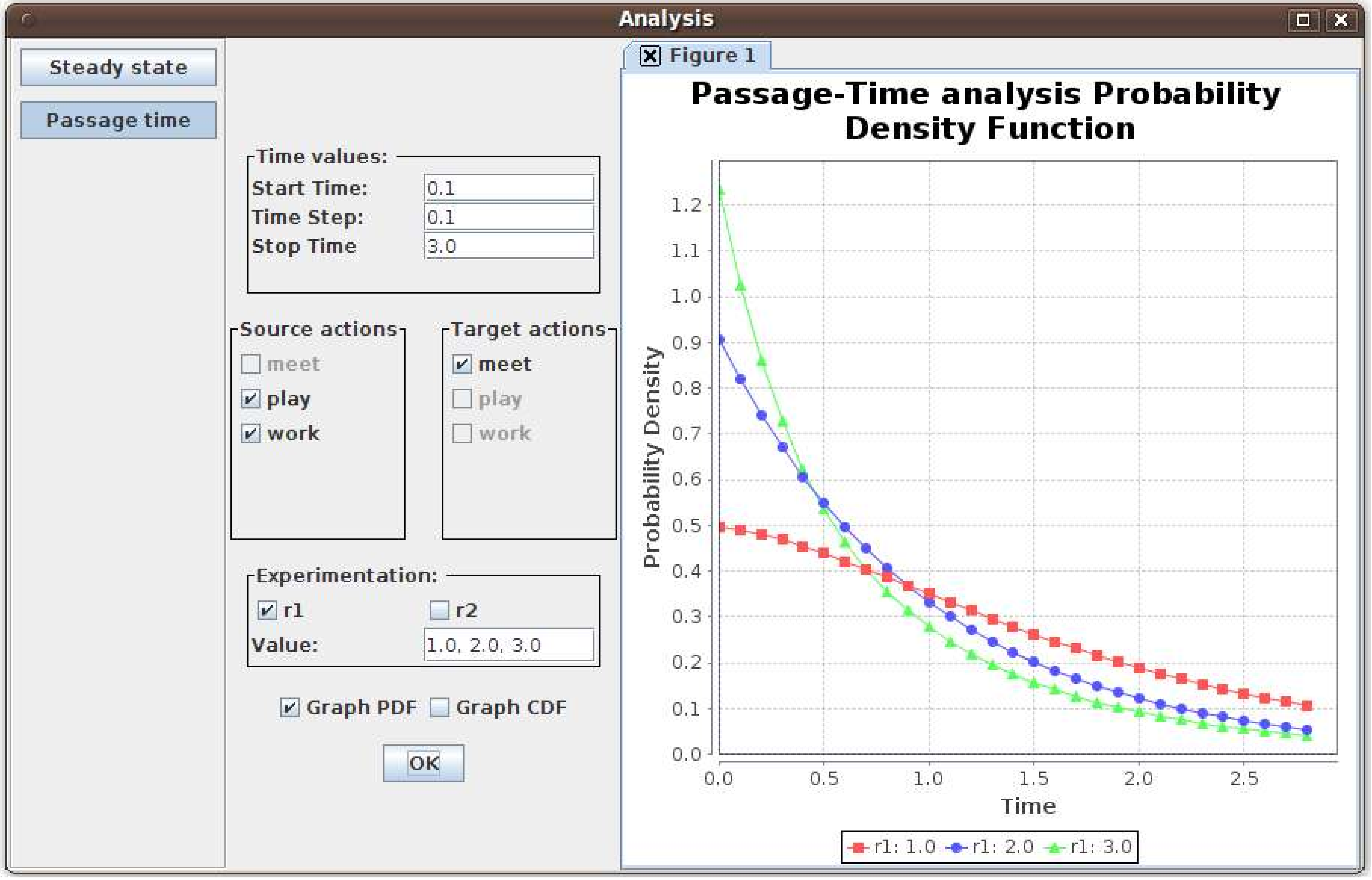
```
restrict { put , get , full , notEmpty , empty }
 in
  (        Buffer_3[x1]
  |     Consumer[Start]
  |     Producer[Start]
)
end
```



```
s0 = (play, 1.0).Bill[s1];
s1 = (meet, r2).Bill[s0];
```

```
s0 = (work, 1.0).Ben[s1];
s1 = (meet, r1).Ben[s0];
```

Rate
r1
r2



Time values:

Start Time: 0.1

Time Step: 0.1

Stop Time 3.0

Source actions:

- [ ] meet
- [✓] play
- [✓] work

Target actions:

- [✓] meet
- [ ] play
- [ ] work

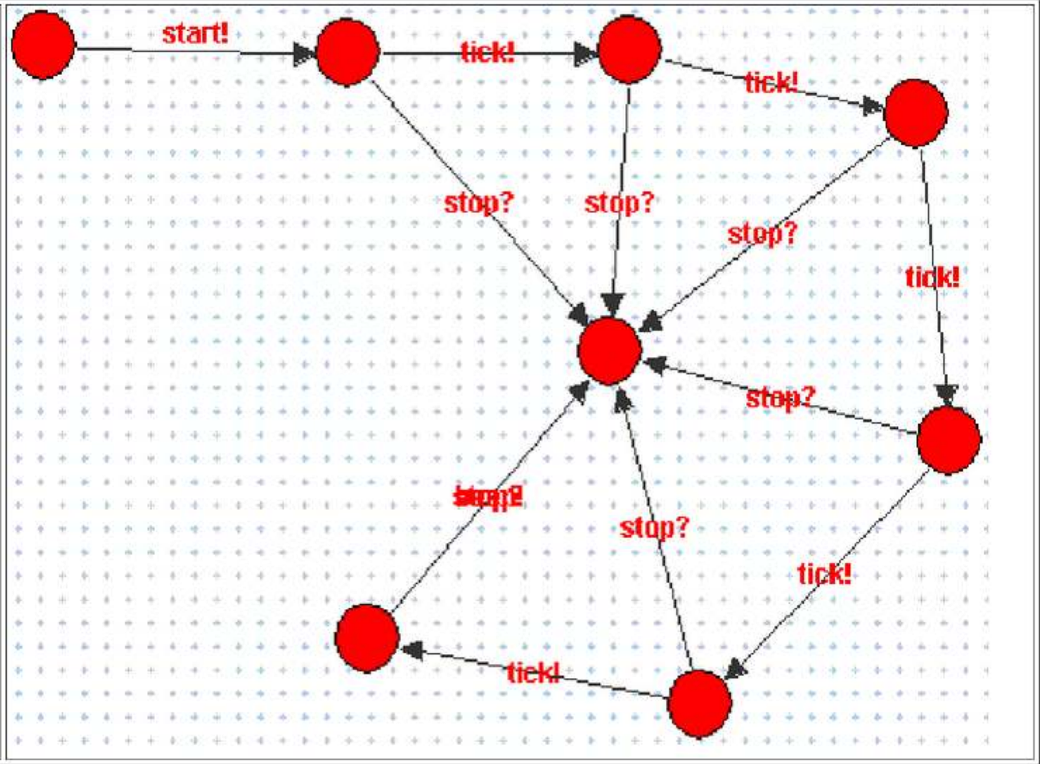
Experimentation:

- [✓] r1
- [ ] r2

Value: 1.0, 2.0, 3.0

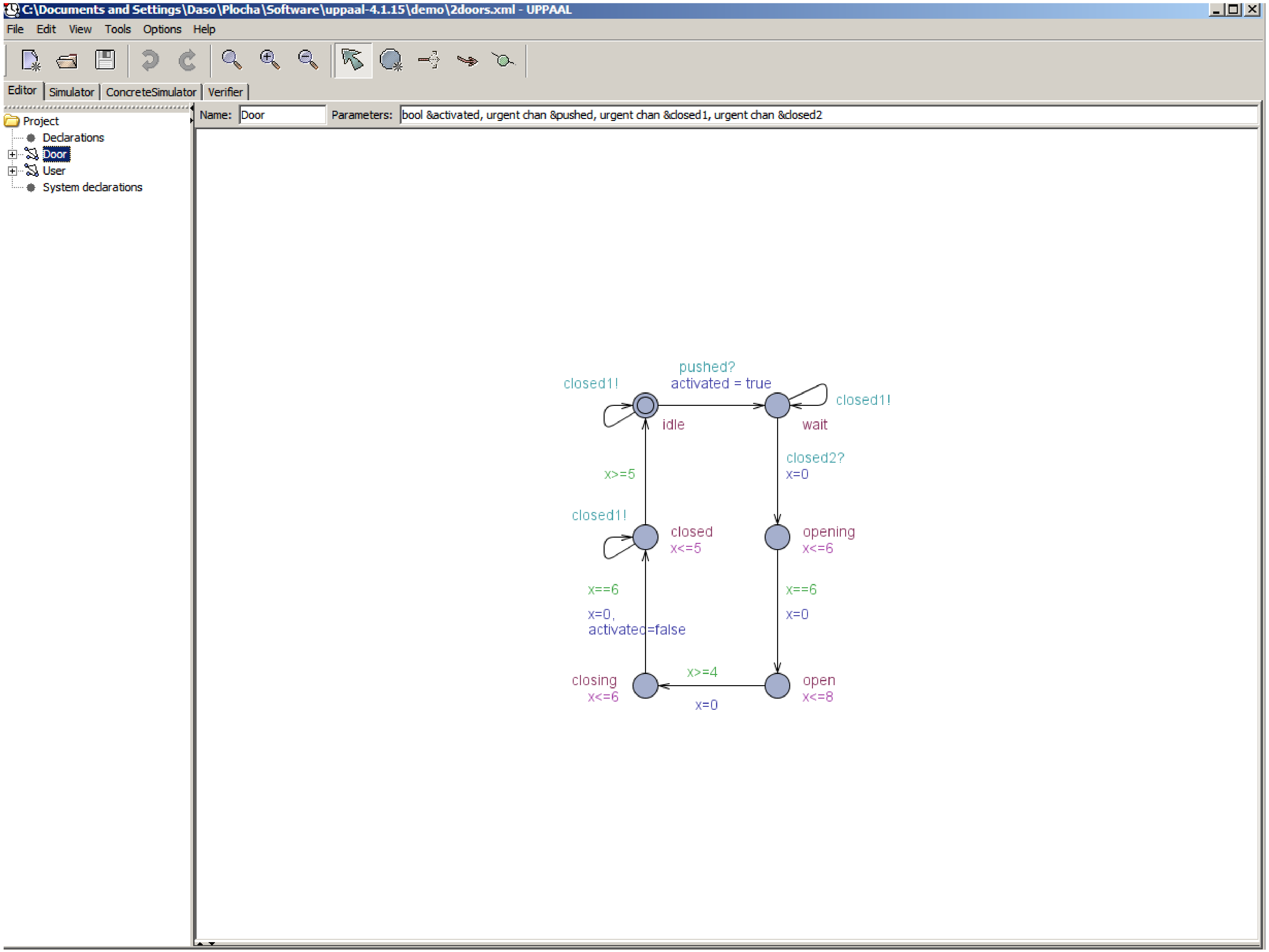
[✓] Graph PDF [ ] Graph CDF

```
Countdown -> start!.Countdown_5
Countdown_5 -> tick!.Countdown_4 + stop?.nil
Countdown_4 -> tick!.Countdown_3 + stop?.nil
Countdown_3 -> tick!.Countdown_2 + stop?.nil
Countdown_2 -> tick!.Countdown_1 + stop?.nil
Countdown_1 -> tick!.Countdown_0 + stop?.nil
Countdown_0 -> beep!.nil + stop?.nil
```



UPAALL

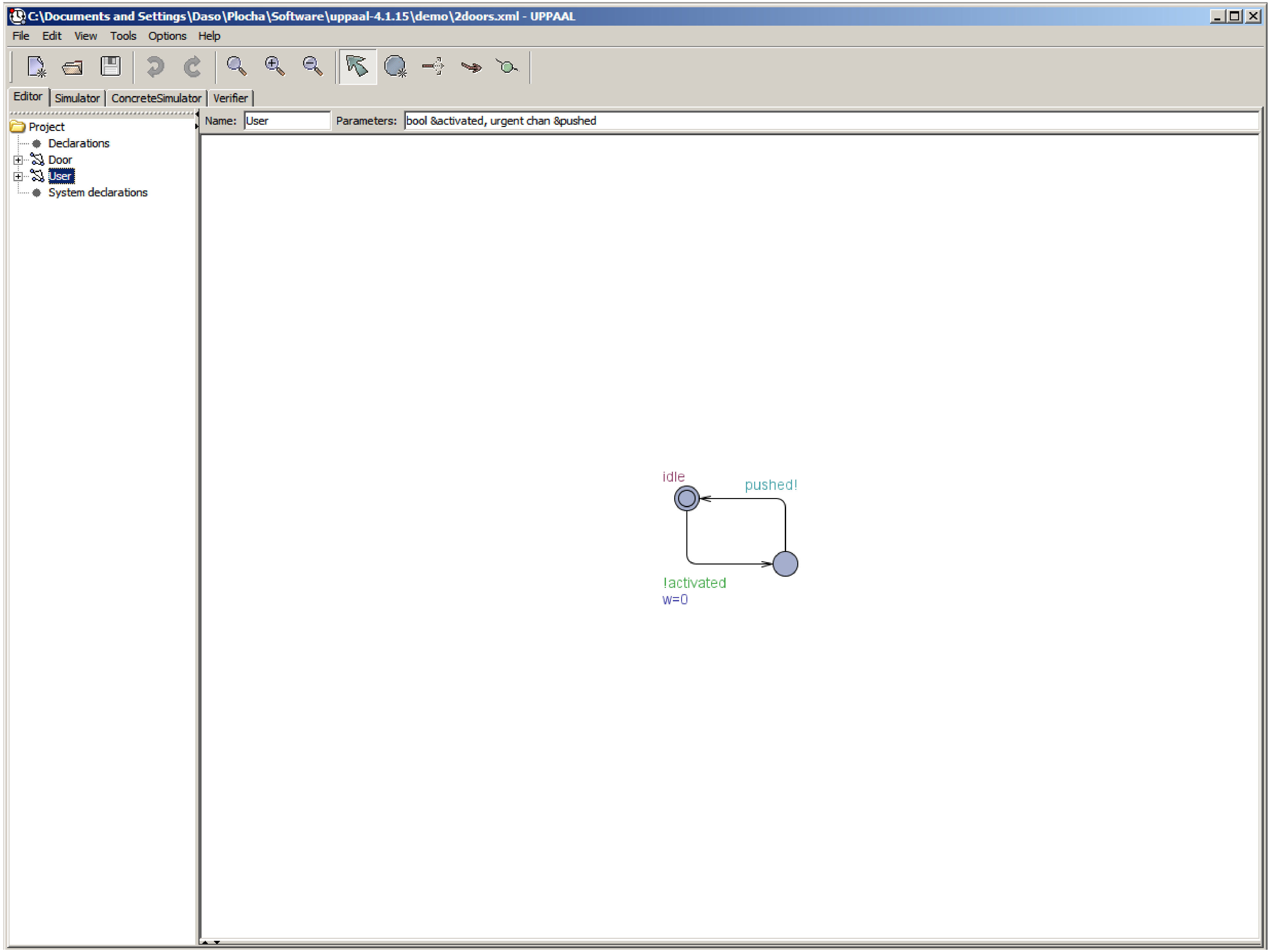
- Timed Automata
- Simulation
- Verification



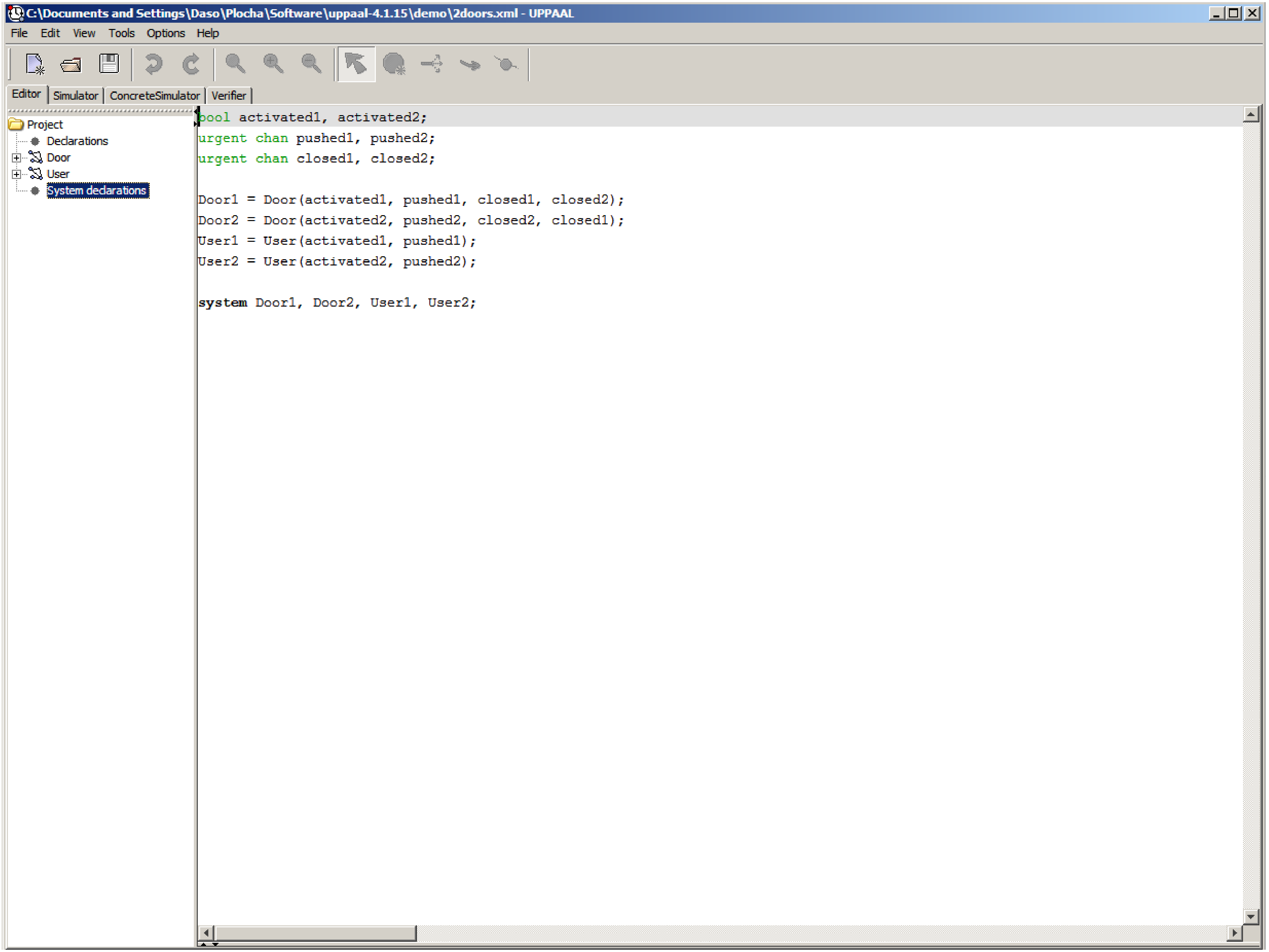
C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\2doors.xml - UPPAAL

Name: Door

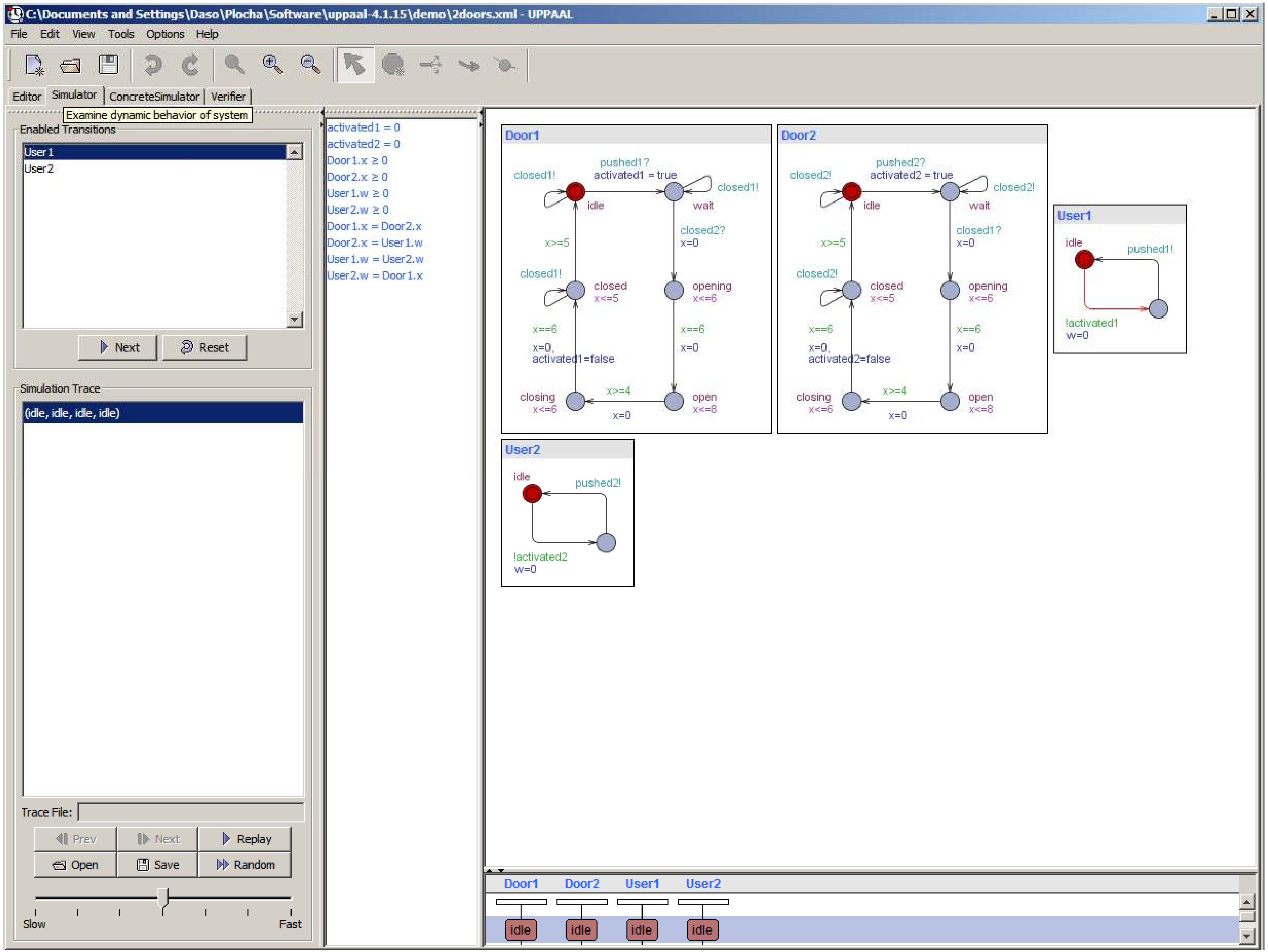
Parameters: bool &activated, urgent chan &pushed, urgent chan &closed1, urgent chan &closed2



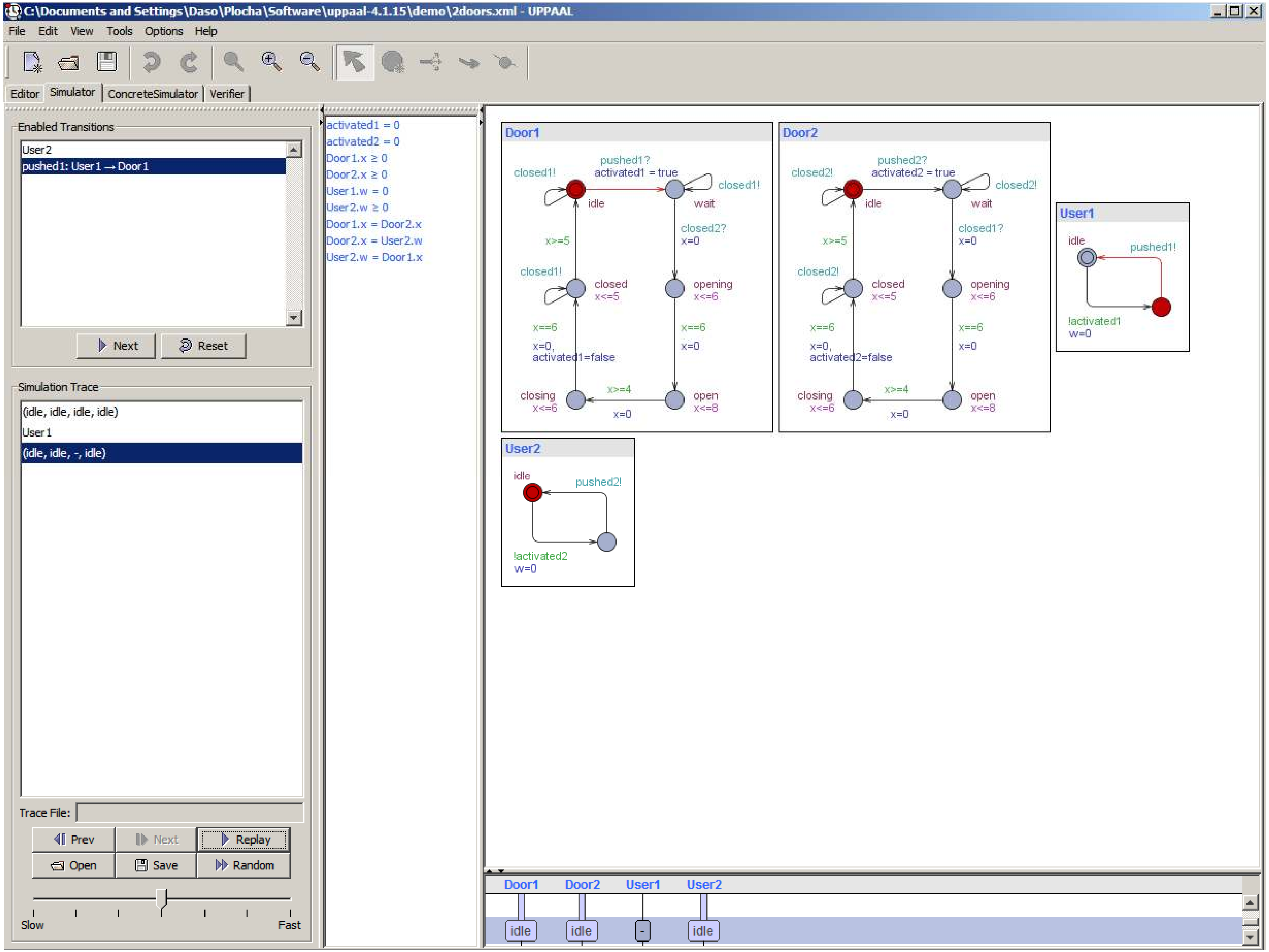
Name: User Parameters: bool &activated, urgent chan &pushed



```
bool activated1, activated2;
urgent chan pushed1, pushed2;
urgent chan closed1, closed2;

Door1 = Door(activated1, pushed1, closed1, closed2);
Door2 = Door(activated2, pushed2, closed2, closed1);
User1 = User(activated1, pushed1);
User2 = User(activated2, pushed2);

system Door1, Door2, User1, User2;
```



activated1 = 0
activated2 = 0
Door1.x ≥ 0
Door2.x ≥ 0
User1.w ≥ 0
User2.w ≥ 0
Door1.x = Door2.x
Door2.x = User1.w
User1.w = User2.w
User2.w = Door1.x



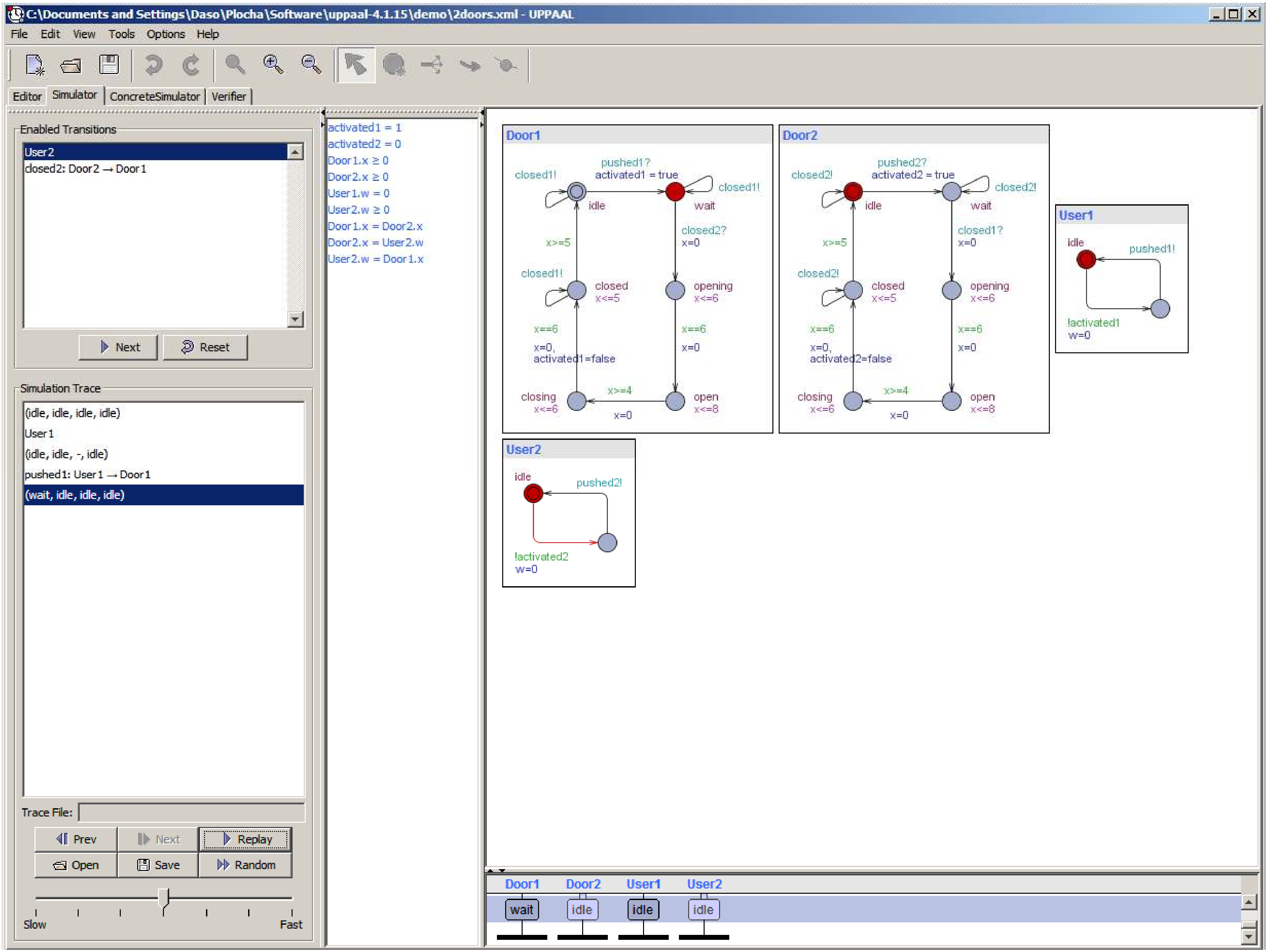
Enabled Transitions:

User2
pushed1: User1 → Door1

Simulation Trace:

(idle, idle, idle, idle)
User1
(idle, idle, -, idle)

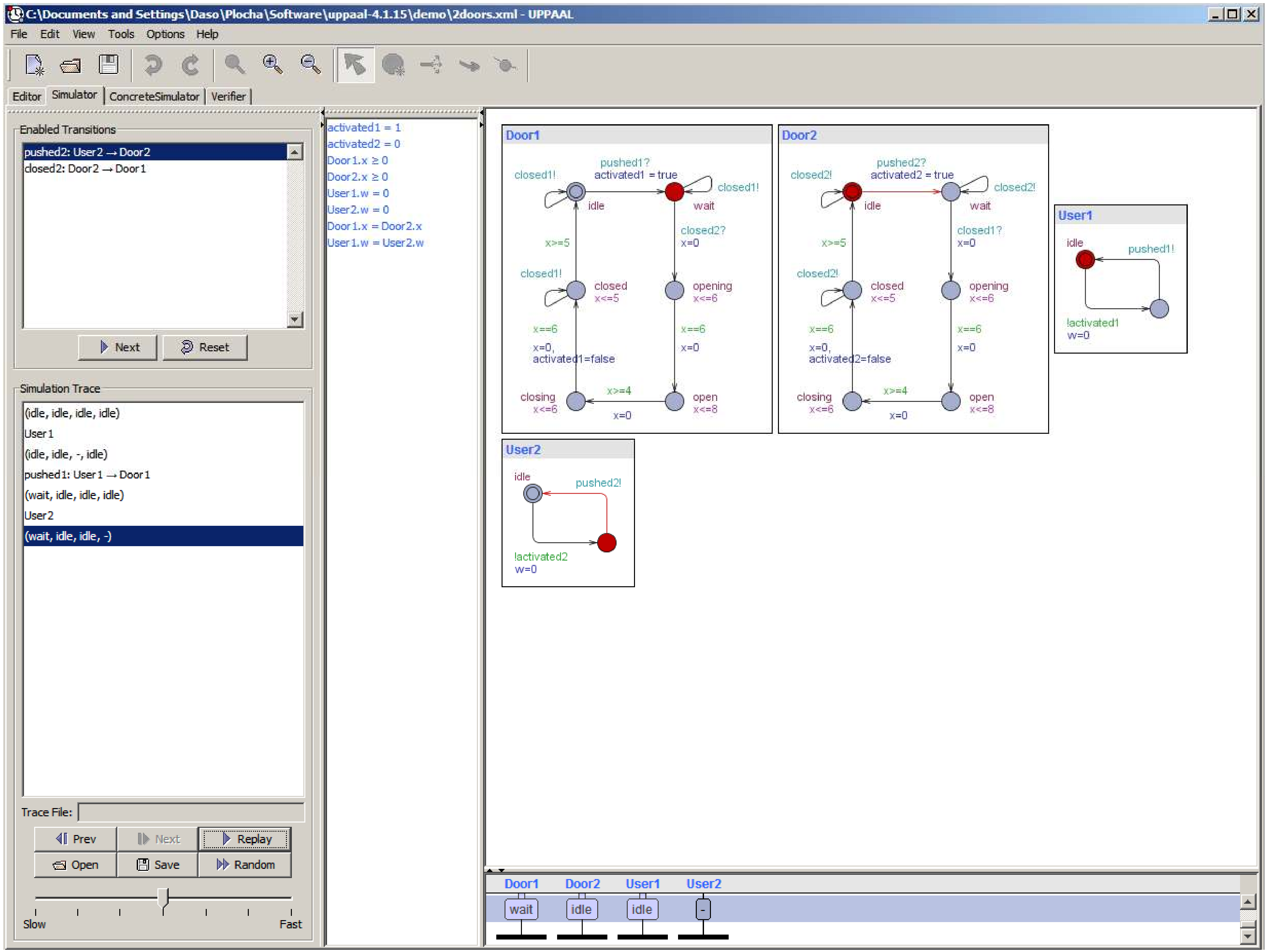
```
activated1 = 0
activated2 = 0
Door1.x ≥ 0
Door2.x ≥ 0
User1.w = 0
User2.w ≥ 0
Door1.x = Door2.x
Door2.x = User2.w
User2.w = Door1.x
```


C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\2doors.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier
Enabled Transitions
Next Reset
Simulation Trace
Trace File:
Prev Next Replay
Open Save Random
Slow Fast
Door1
closed1! pushed1? activated1 = true closed1!
idle wait
x>=5 closed2? x=0
closed1! closed x<=5 opening x<=6
x==6 x==6
x=0, activated1=false x=0
closing x<=6 x>=4 x=0 open x<=8
Door2
closed2! pushed2? activated2 = true closed2!
idle wait
x>=5 closed1? x=0
closed2! closed x<=5 opening x<=6
x==6 x==6
x=0, activated2=false x=0
closing x<=6 x>=4 x=0 open x<=8
User1
idle pushed1!
!activated1 w=0
User2
idle pushed2!
!activated2 w=0
Door1 Door2 User1 User2
wait idle idle idle

```
User2
closed2: Door2 → Door1
```

```
activated1 = 1
activated2 = 0
Door1.x ≥ 0
Door2.x ≥ 0
User1.w = 0
User2.w ≥ 0
Door1.x = Door2.x
Door2.x = User2.w
User2.w = Door1.x
```

```
(idle, idle, idle, idle)
User1
(idle, idle, -, idle)
pushed1: User1 → Door1
(wait, idle, idle, idle)
```



Enabled Transitions

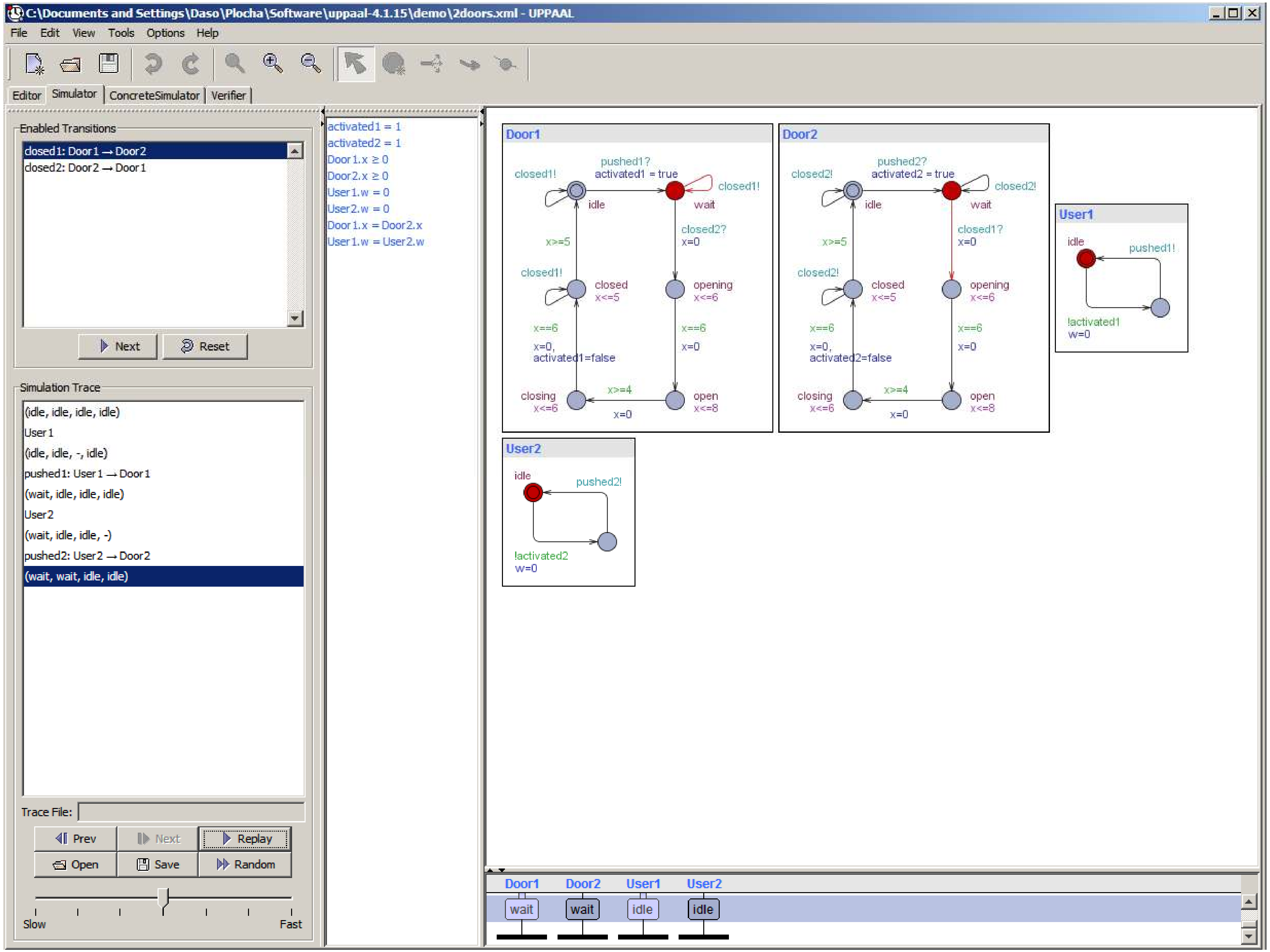
pushed2: User2 → Door2
closed2: Door2 → Door1

```
activated1 = 1
activated2 = 0
Door1.x ≥ 0
Door2.x ≥ 0
User1.w = 0
User2.w = 0
Door1.x = Door2.x
User1.w = User2.w
```

Simulation Trace

```
(idle, idle, idle, idle)
User1
(idle, idle, -, idle)
pushed1: User1 → Door1
(wait, idle, idle, idle)
User2
(wait, idle, idle, -)
```

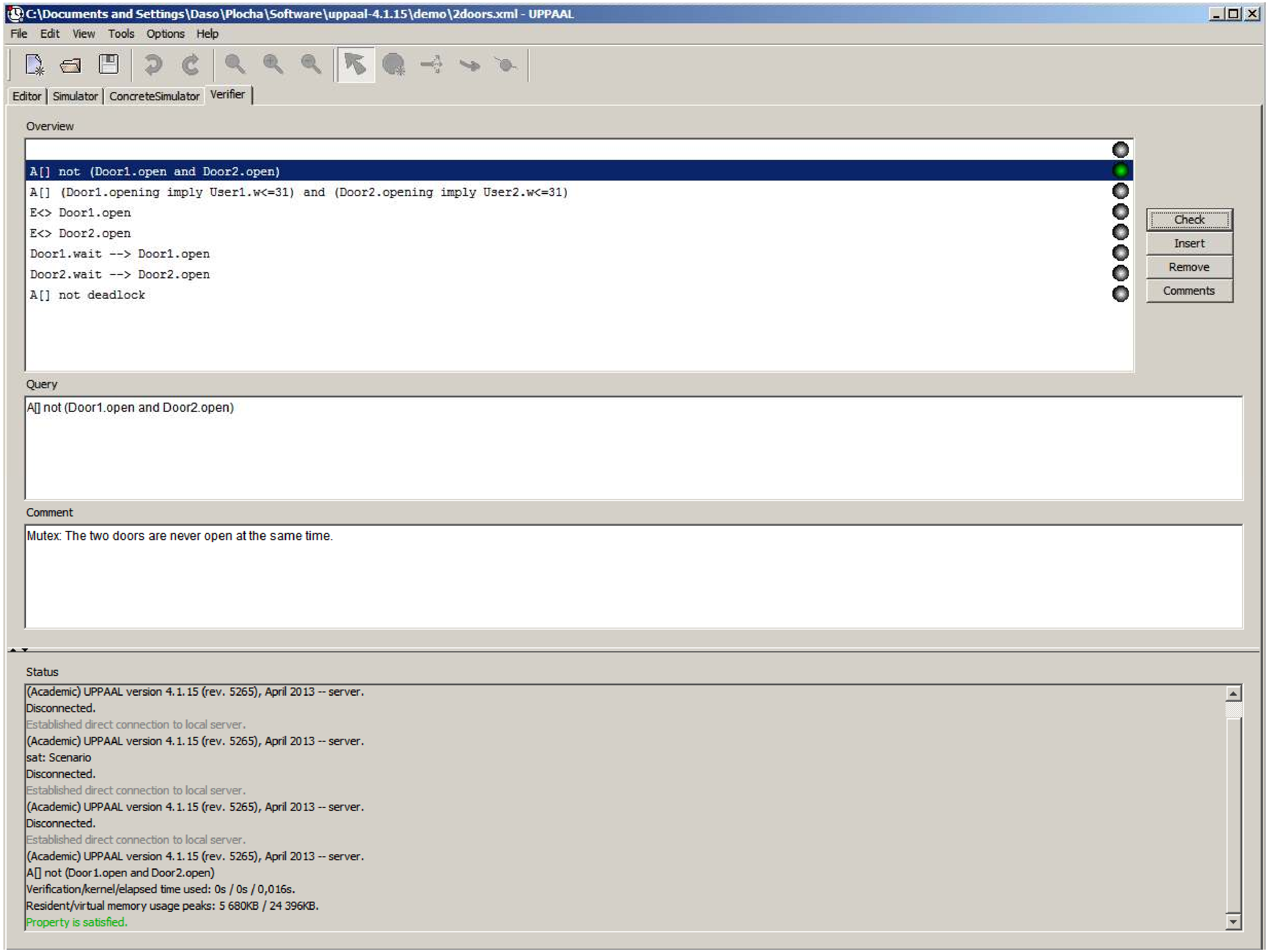
Trace File:



```
closed1: Door1 → Door2
closed2: Door2 → Door1
```

```
activated1 = 1
activated2 = 1
Door1.x ≥ 0
Door2.x ≥ 0
User1.w = 0
User2.w = 0
Door1.x = Door2.x
User1.w = User2.w
```

```
(idle, idle, idle, idle)
User1
(idle, idle, -, idle)
pushed1: User1 → Door1
(wait, idle, idle, idle)
User2
(wait, idle, idle, -)
pushed2: User2 → Door2
(wait, wait, idle, idle)
```

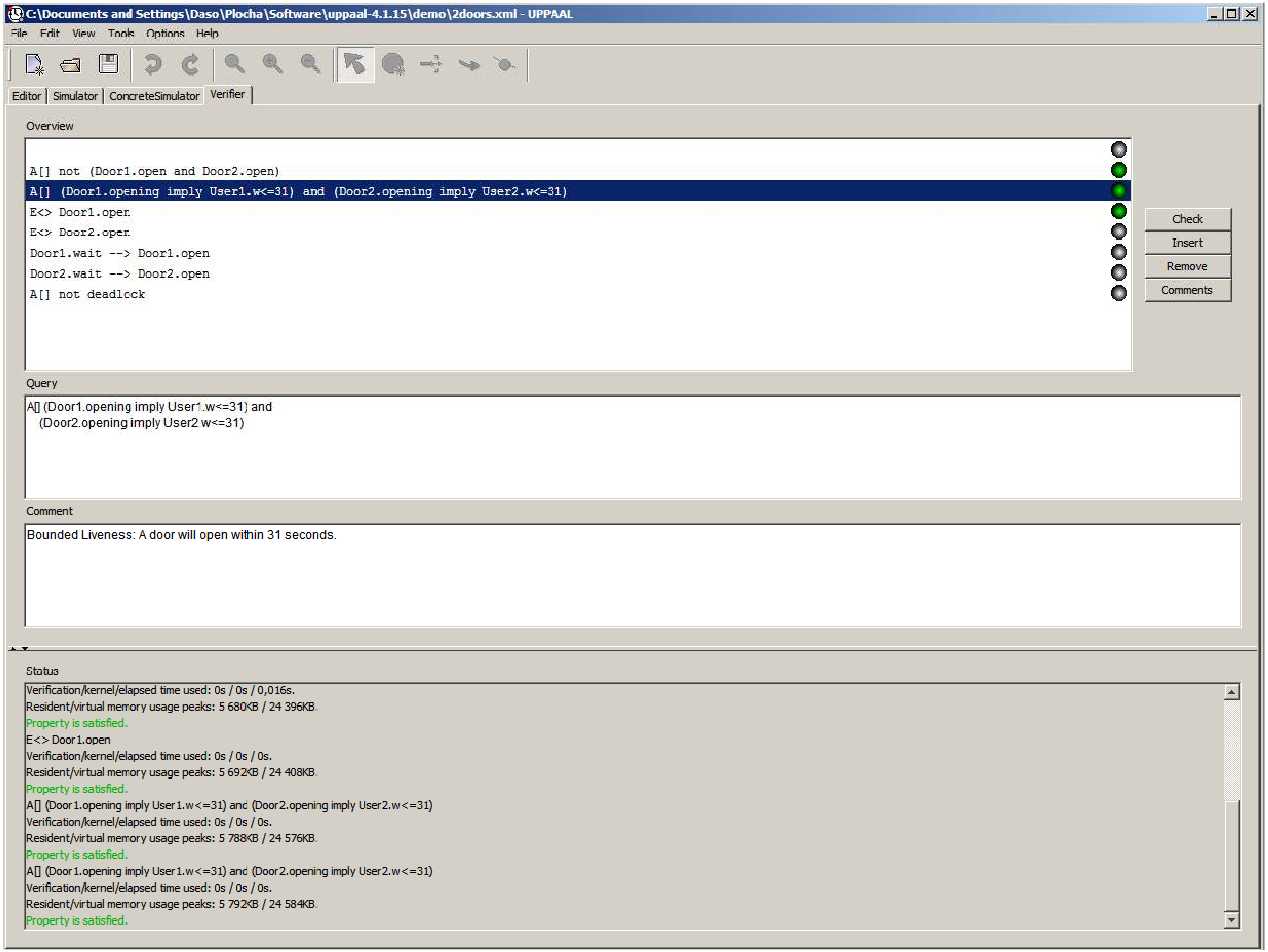

C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\2doors.xml - UPPAAL
File Edit View Tools Options Help
Editor
Simulator
ConcreteSimulator
Verifier
Overview
Check
Insert
Remove
Comments
Query
Comment
Status

```
A[] not (Door1.open and Door2.open)
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
E<> Door1.open
E<> Door2.open
Door1.wait --> Door1.open
Door2.wait --> Door2.open
A[] not deadlock
```

Query: A[] not (Door1.open and Door2.open)

Comment: Mutex: The two doors are never open at the same time.

```
(Academic) UPPAAL version 4.1.15 (rev. 5265), April 2013 -- server.
Disconnected.
Established direct connection to local server.
(Academic) UPPAAL version 4.1.15 (rev. 5265), April 2013 -- server.
sat: Scenario
Disconnected.
Established direct connection to local server.
(Academic) UPPAAL version 4.1.15 (rev. 5265), April 2013 -- server.
Disconnected.
Established direct connection to local server.
(Academic) UPPAAL version 4.1.15 (rev. 5265), April 2013 -- server.
A[] not (Door1.open and Door2.open)
Verification/kernel/elapsed time used: 0s / 0s / 0,016s.
Resident/virtual memory usage peaks: 5 680KB / 24 396KB.
Property is satisfied.
```


C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\2doors.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier
Overview
Check
Insert
Remove
Comments
Query
Comment
Status

```
A[] not (Door1.open and Door2.open)
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
E<> Door1.open
E<> Door2.open
Door1.wait --> Door1.open
Door2.wait --> Door2.open
A[] not deadlock
```

Query:

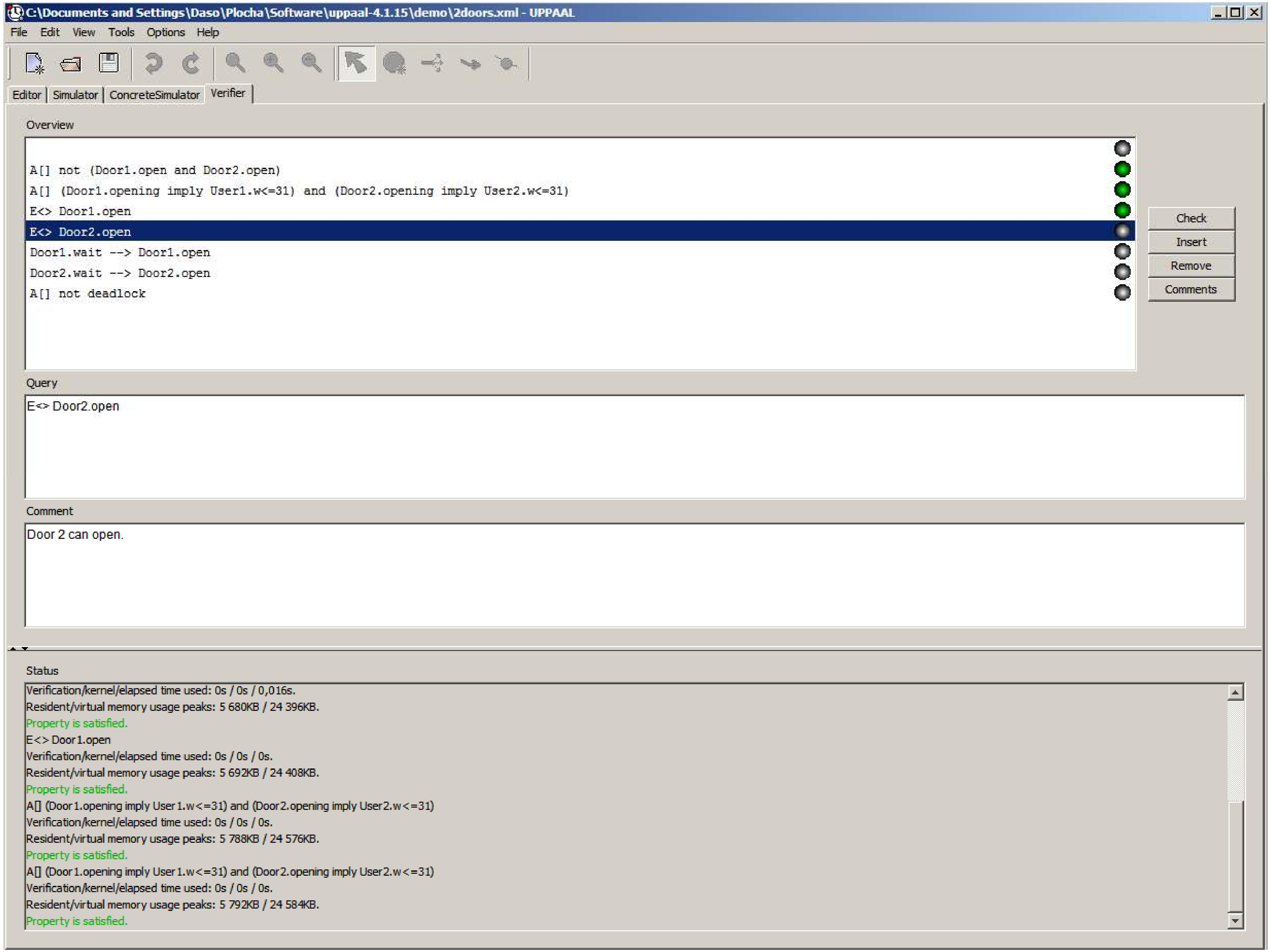
```
A[] (Door1.opening imply User1.w<=31) and
  (Door2.opening imply User2.w<=31)
```

Comment:

Bounded Liveness: A door will open within 31 seconds.

Status:

```
Verification/kernel/elapsed time used: 0s / 0s / 0,016s.
Resident/virtual memory usage peaks: 5 680KB / 24 396KB.
Property is satisfied.
E<> Door1.open
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 5 692KB / 24 408KB.
Property is satisfied.
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 5 788KB / 24 576KB.
Property is satisfied.
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 5 792KB / 24 584KB.
Property is satisfied.
```

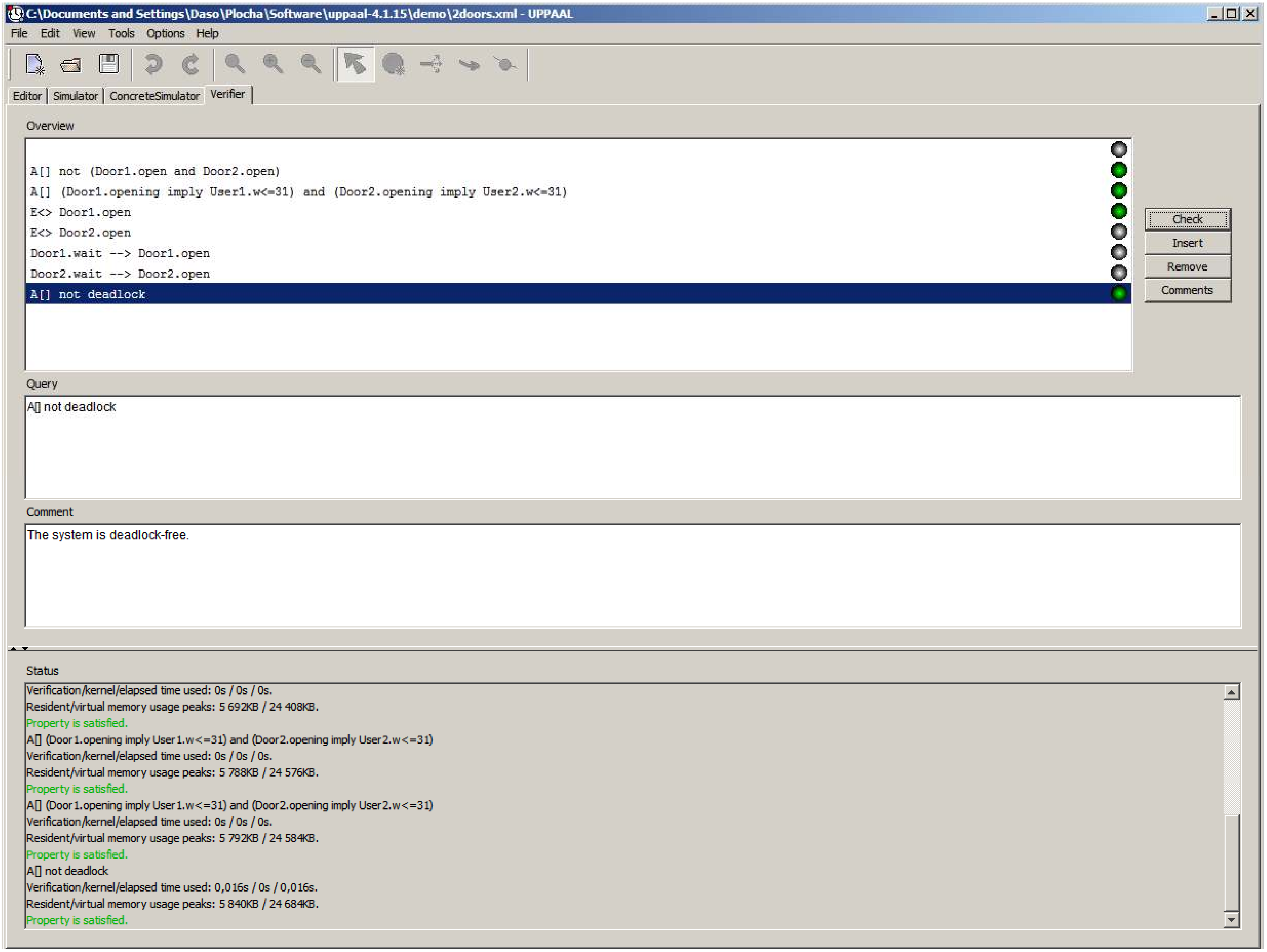


```
A[] not (Door1.open and Door2.open)
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
E<> Door1.open
E<> Door2.open
Door1.wait --> Door1.open
Door2.wait --> Door2.open
A[] not deadlock
```

Query: E<> Door2.open

Comment: Door 2 can open.

```
Verification/kernel/elapsed time used: 0s / 0s / 0,016s.
Resident/virtual memory usage peaks: 5 680KB / 24 396KB.
Property is satisfied.
E<> Door1.open
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 5 692KB / 24 408KB.
Property is satisfied.
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 5 788KB / 24 576KB.
Property is satisfied.
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 5 792KB / 24 584KB.
Property is satisfied.
```

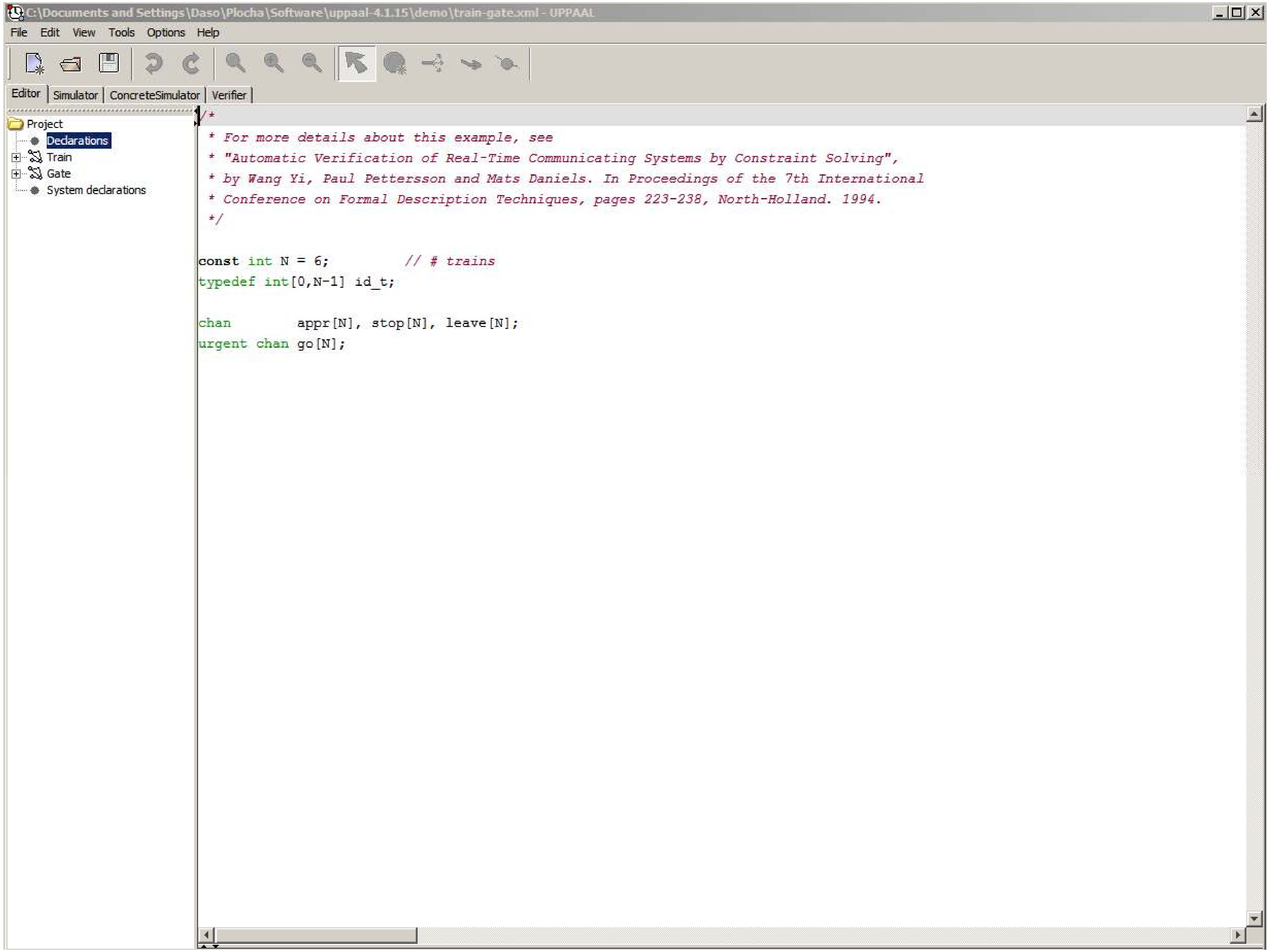

C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\2doors.xml - UPPAAL
File Edit View Tools Options Help
Editor
Simulator
ConcreteSimulator
Verifier
Overview
Check
Insert
Remove
Comments
Query
Comment
Status

```
A[] not (Door1.open and Door2.open)
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
E<> Door1.open
E<> Door2.open
Door1.wait --> Door1.open
Door2.wait --> Door2.open
A[] not deadlock
```

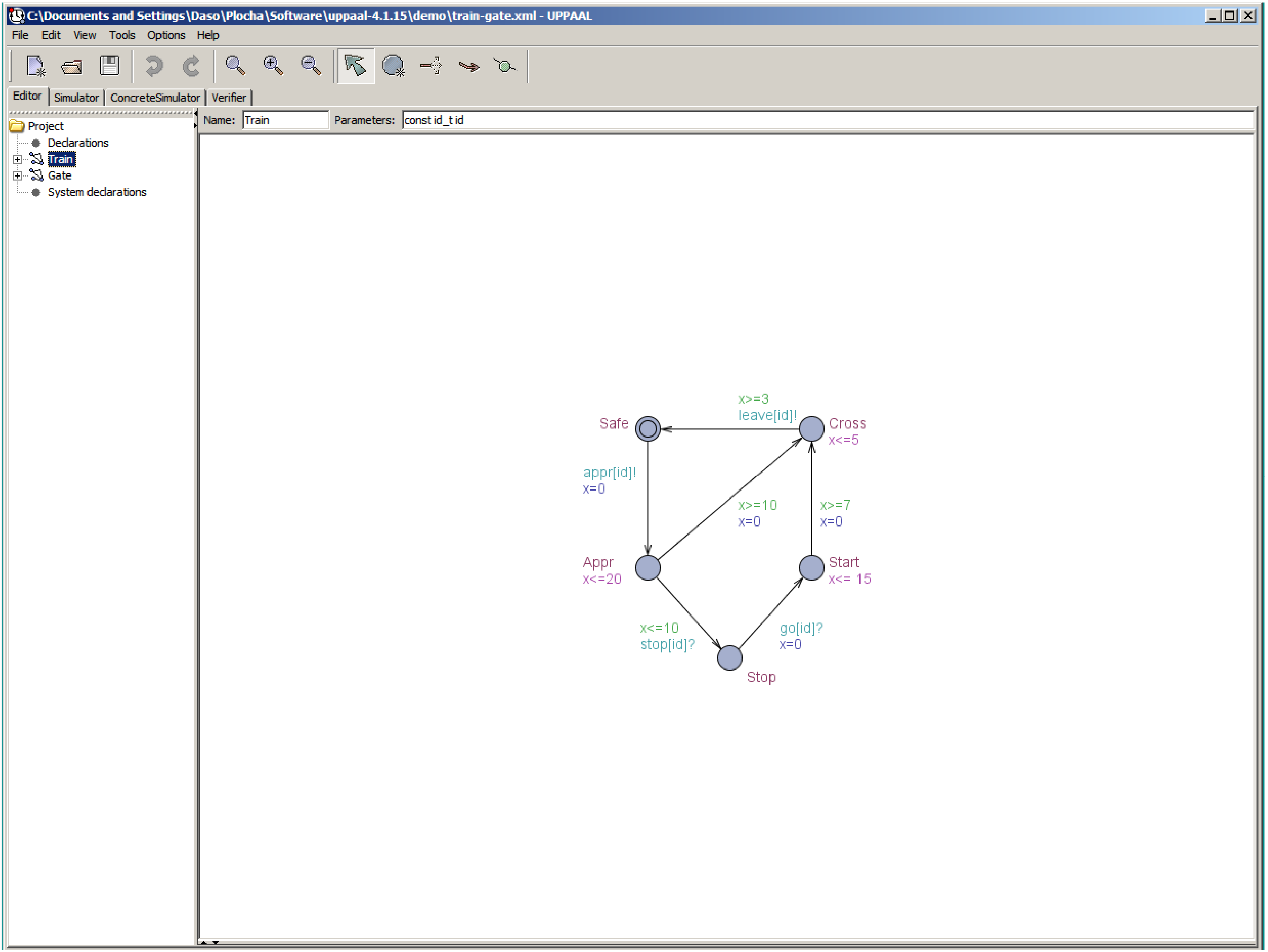
Query: A[] not deadlock

Comment: The system is deadlock-free.

```
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 5 692KB / 24 408KB.
Property is satisfied.
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 5 788KB / 24 576KB.
Property is satisfied.
A[] (Door1.opening imply User1.w<=31) and (Door2.opening imply User2.w<=31)
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 5 792KB / 24 584KB.
Property is satisfied.
A[] not deadlock
Verification/kernel/elapsed time used: 0,016s / 0s / 0,016s.
Resident/virtual memory usage peaks: 5 840KB / 24 684KB.
Property is satisfied.
```

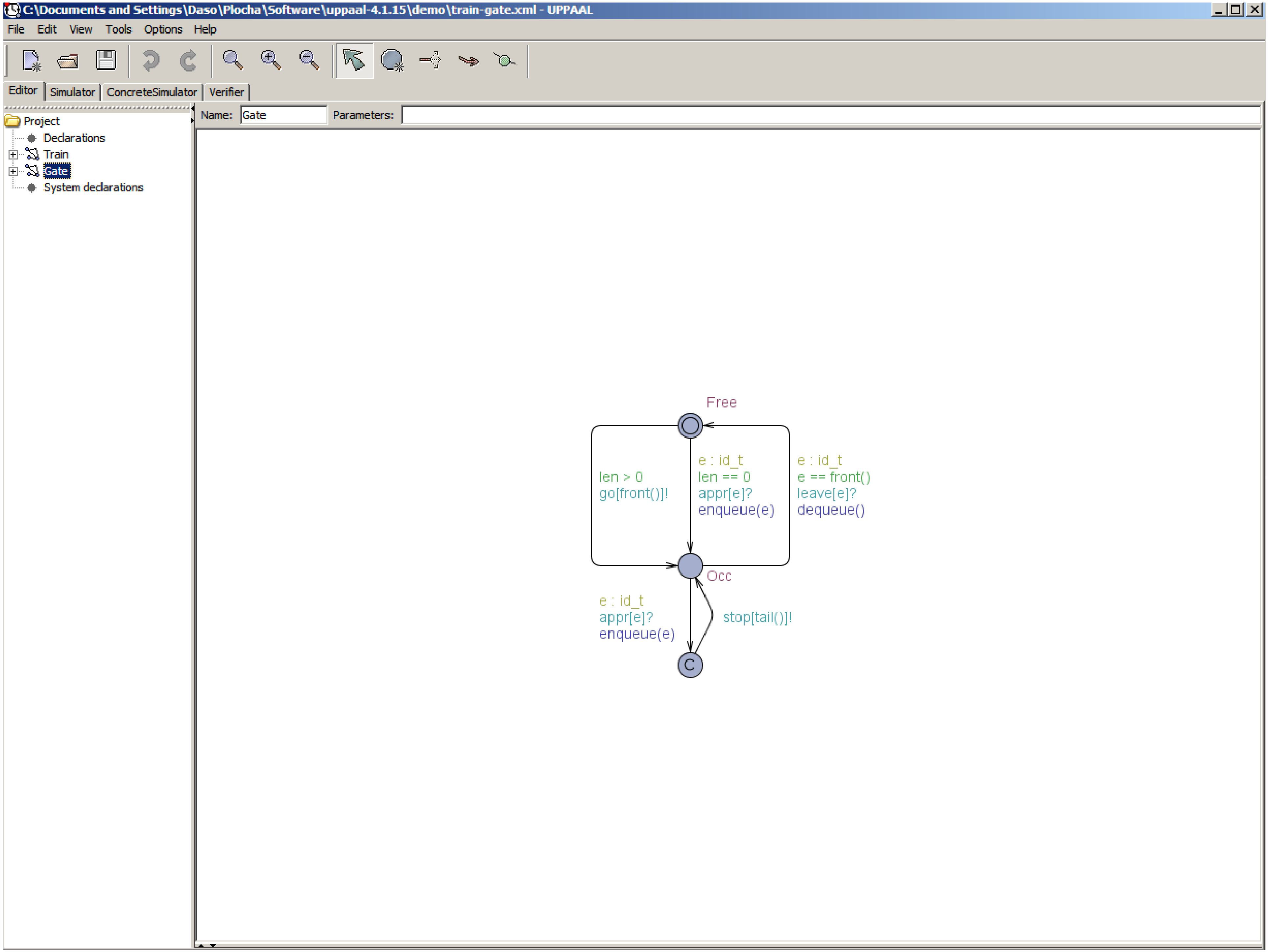

C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\train-gate.xml - UPPAAL
File Edit View Tools Options Help
Editor
Simulator
ConcreteSimulator
Verifier
Project
Declarations
Train
Gate
System declarations

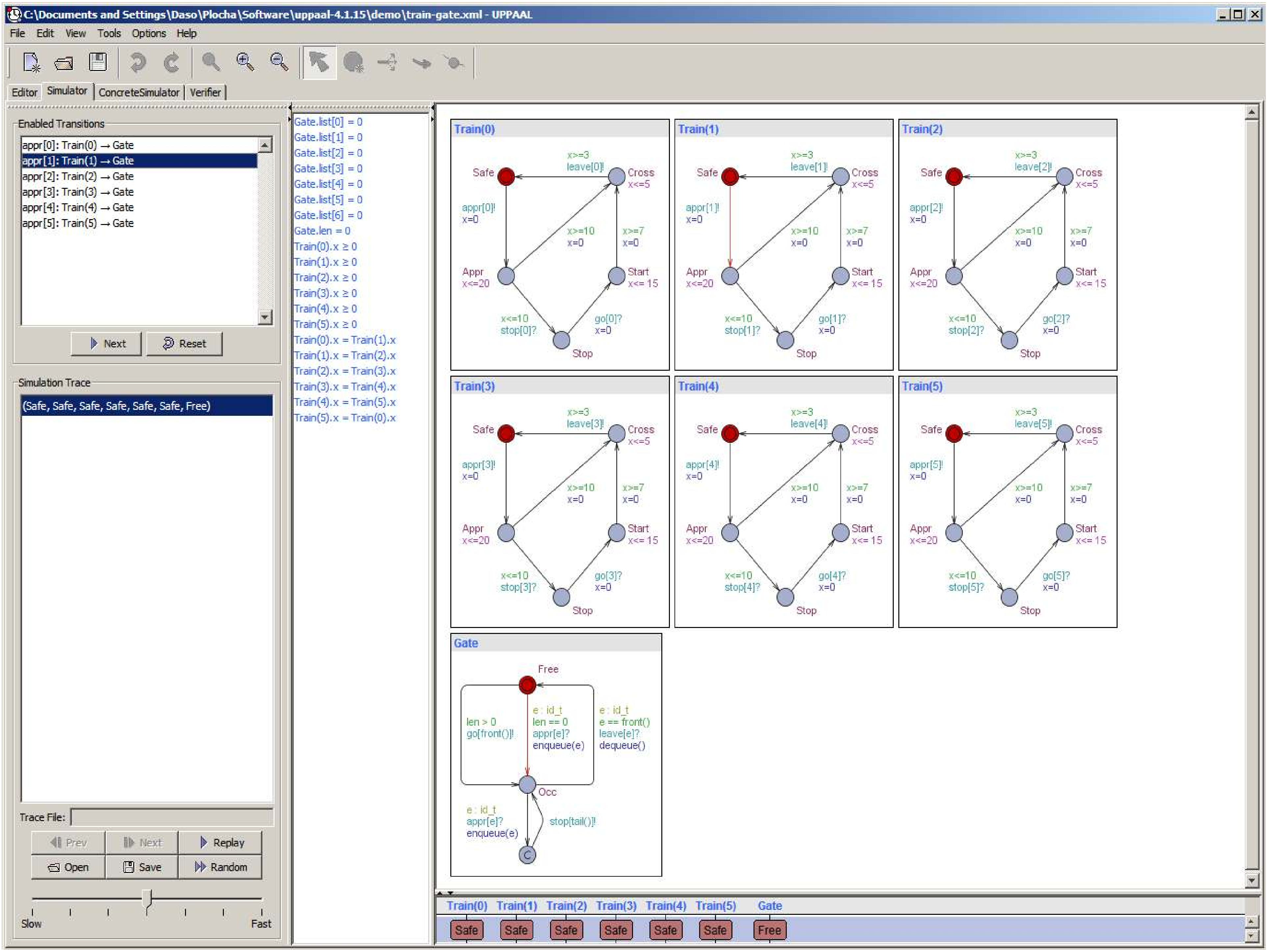
```
/*
 * For more details about this example, see
 * "Automatic Verification of Real-Time Communicating Systems by Constraint Solving",
 * by Wang Yi, Paul Pettersson and Mats Daniels. In Proceedings of the 7th International
 * Conference on Formal Description Techniques, pages 223-238, North-Holland. 1994.
 */

const int N = 6;         // # trains
typedef int[0,N-1] id_t;

chan        appr[N], stop[N], leave[N];
urgent chan go[N];
```


C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\train-gate.xml - UPPAAL

Name: Train

Parameters: const id_t id


C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\train-gate.xml - UPPAAL
File
Edit
View
Tools
Options
Help
Editor
Simulator
ConcreteSimulator
Verifier
Project
Declarations
Train
Gate
System declarations
Name: Gate
Parameters:
Free
len > 0
go[front()]!
e : id_t
len == 0
appr[e]?
enqueue(e)
e : id_t
e == front()
leave[e]?
dequeue()
Occ
e : id_t
appr[e]?
enqueue(e)
stop[tail()]!
C



C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\train-gate.xml - UPPAAL

Enabled Transitions

appr[0]: Train(0) → Gate
appr[1]: Train(1) → Gate
appr[2]: Train(2) → Gate
appr[3]: Train(3) → Gate
appr[4]: Train(4) → Gate
appr[5]: Train(5) → Gate

Simulation Trace

(Safe, Safe, Safe, Safe, Safe, Safe, Free)

```
Gate.list[0] = 0
Gate.list[1] = 0
Gate.list[2] = 0
Gate.list[3] = 0
Gate.list[4] = 0
Gate.list[5] = 0
Gate.list[6] = 0
Gate.len = 0
Train(0).x ≥ 0
Train(1).x ≥ 0
Train(2).x ≥ 0
Train(3).x ≥ 0
Train(4).x ≥ 0
Train(5).x ≥ 0
Train(0).x = Train(1).x
Train(1).x = Train(2).x
Train(2).x = Train(3).x
Train(3).x = Train(4).x
Train(4).x = Train(5).x
Train(5).x = Train(0).x
```

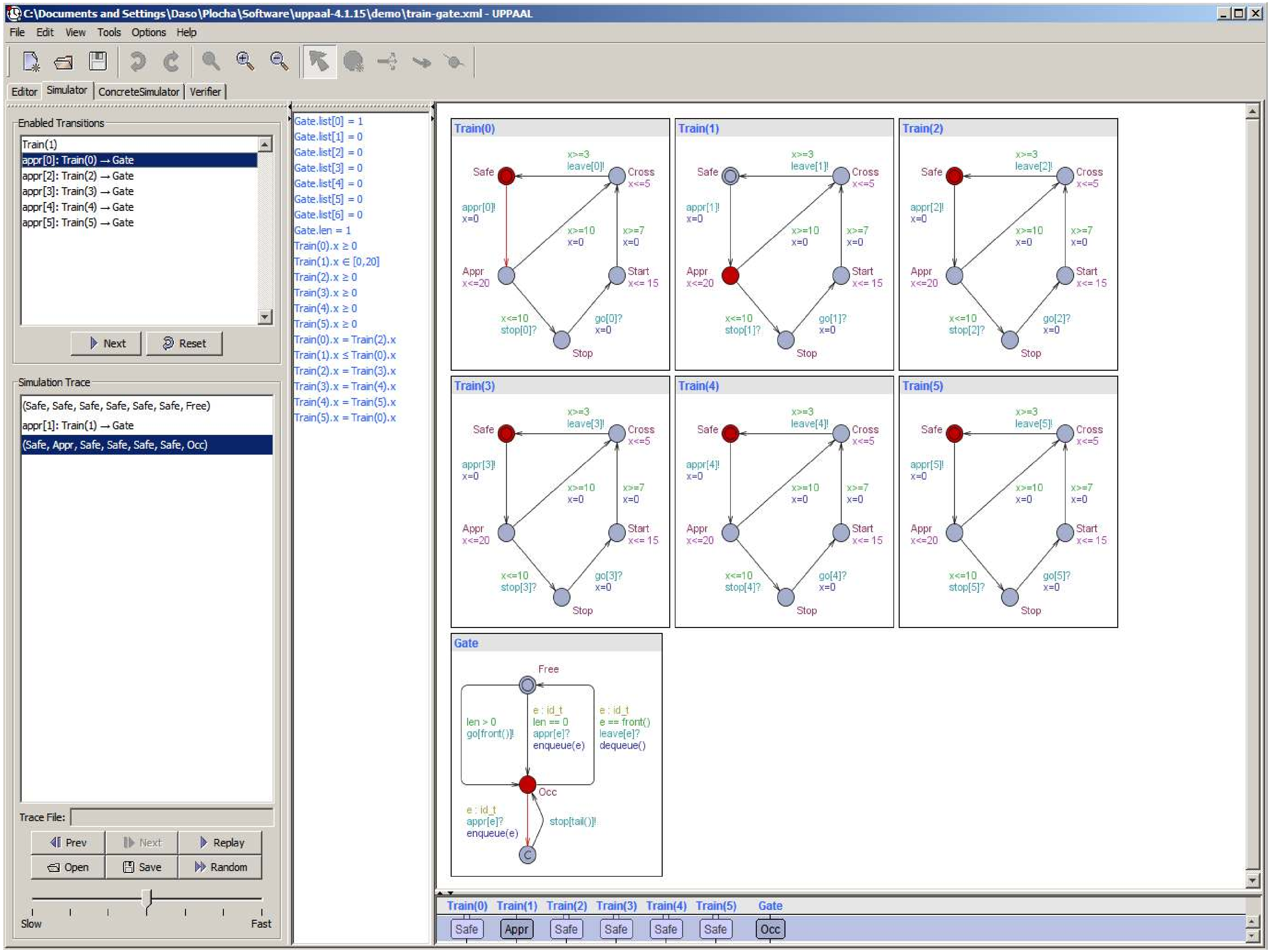
Train(i), i = 0…5 (each automaton drawn with its own index):

```
Safe -> Appr: appr[i]!  x=0
Appr x<=20 -> Cross: x>=10  x=0
Appr -> Stop: x<=10  stop[i]?
Stop -> Start: go[i]?  x=0
Start x<= 15 -> Cross: x>=7  x=0
Cross x<=5 -> Safe: x>=3  leave[i]!
```

Gate:

```
Free -> Occ: e : id_t  len == 0  appr[e]?  enqueue(e)
Free -> Occ: len > 0  go[front()]!
Occ -> Free: e : id_t  e == front()  leave[e]?  dequeue()
Occ -> C: e : id_t  appr[e]?  enqueue(e)
C -> Occ: stop[tail()]!
```

Train(0)	Train(1)	Train(2)	Train(3)	Train(4)	Train(5)	Gate
Safe	Safe	Safe	Safe	Safe	Safe	Free



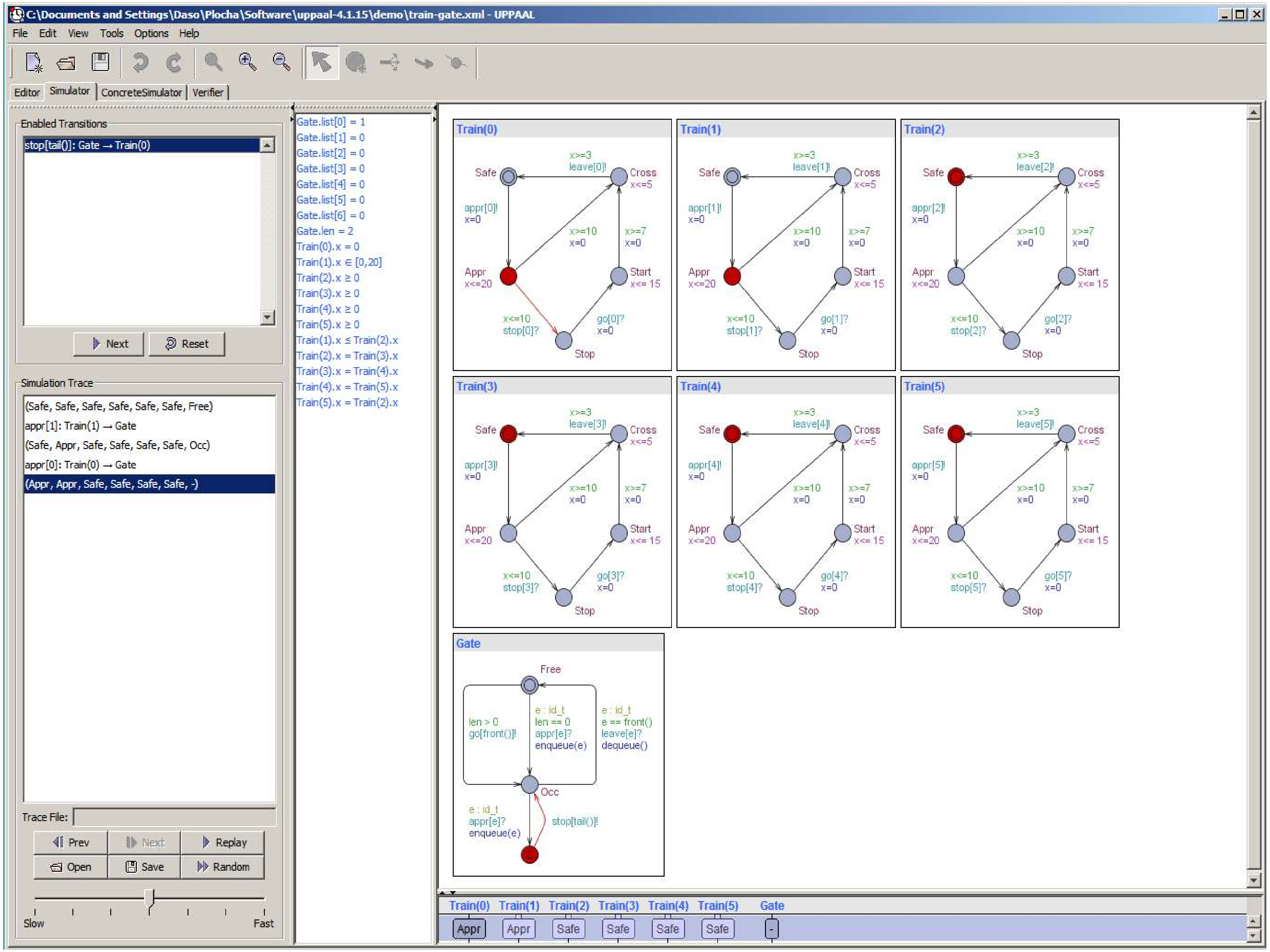
Enabled Transitions

```
Train(1)
appr[0]: Train(0) → Gate
appr[2]: Train(2) → Gate
appr[3]: Train(3) → Gate
appr[4]: Train(4) → Gate
appr[5]: Train(5) → Gate
```

Simulation Trace

```
(Safe, Safe, Safe, Safe, Safe, Safe, Free)
appr[1]: Train(1) → Gate
(Safe, Appr, Safe, Safe, Safe, Safe, Occ)
```

```
Gate.list[0] = 1
Gate.list[1] = 0
Gate.list[2] = 0
Gate.list[3] = 0
Gate.list[4] = 0
Gate.list[5] = 0
Gate.list[6] = 0
Gate.len = 1
Train(0).x ≥ 0
Train(1).x ∈ [0,20]
Train(2).x ≥ 0
Train(3).x ≥ 0
Train(4).x ≥ 0
Train(5).x ≥ 0
Train(0).x = Train(2).x
Train(1).x ≤ Train(0).x
Train(2).x = Train(3).x
Train(3).x = Train(4).x
Train(4).x = Train(5).x
Train(5).x = Train(0).x
```


C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\train-gate.xml - UPPAAL

Enabled Transitions

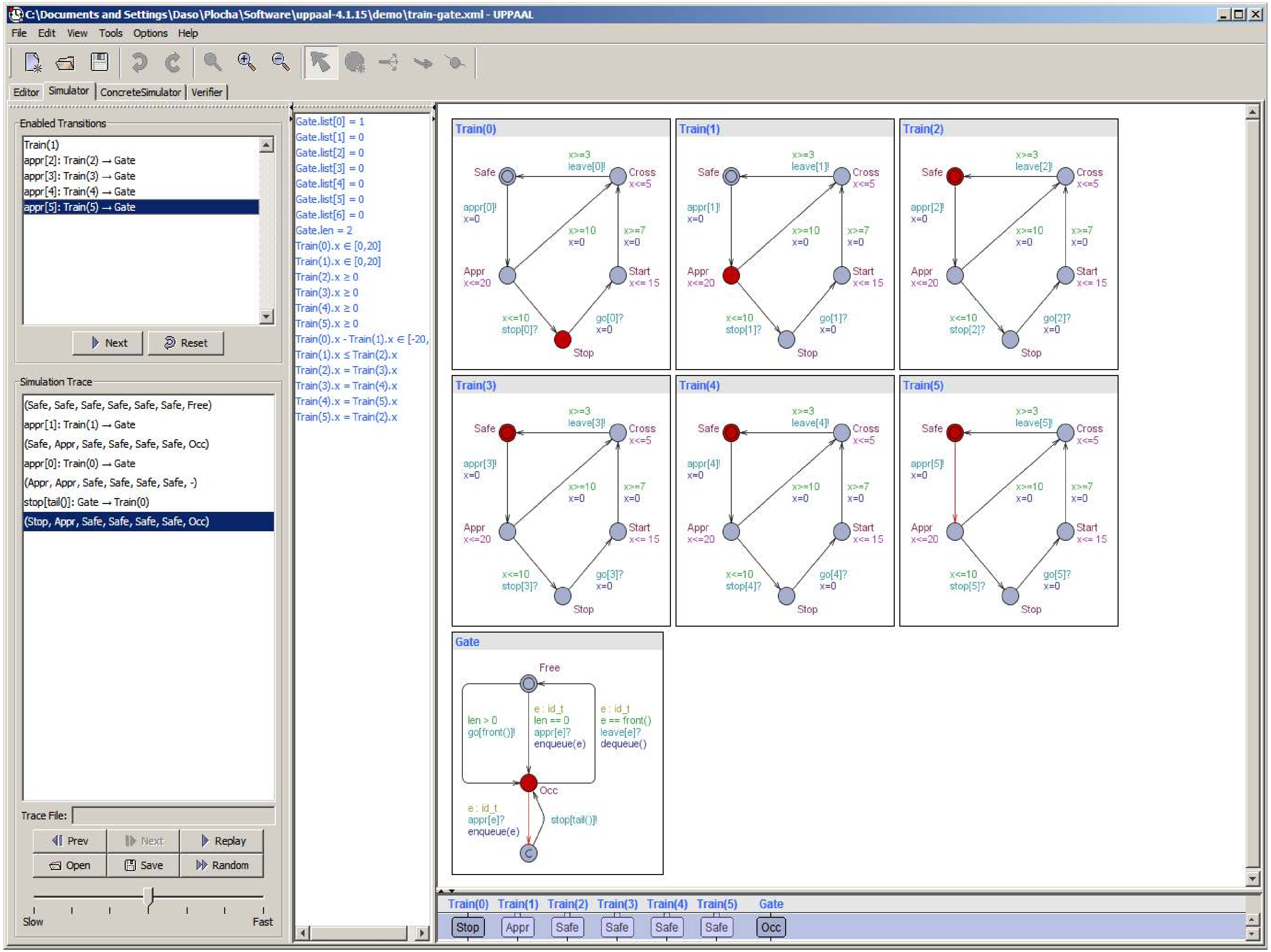
stop[tail()]: Gate → Train(0)

Simulation Trace

(Safe, Safe, Safe, Safe, Safe, Safe, Free)
appr[1]: Train(1) → Gate
(Safe, Appr, Safe, Safe, Safe, Safe, Occ)
appr[0]: Train(0) → Gate
(Appr, Appr, Safe, Safe, Safe, Safe, -)

Trace File:

```
Gate.list[0] = 1
Gate.list[1] = 0
Gate.list[2] = 0
Gate.list[3] = 0
Gate.list[4] = 0
Gate.list[5] = 0
Gate.list[6] = 0
Gate.len = 2
Train(0).x = 0
Train(1).x ∈ [0,20]
Train(2).x ≥ 0
Train(3).x ≥ 0
Train(4).x ≥ 0
Train(5).x ≥ 0
Train(1).x ≤ Train(2).x
Train(2).x = Train(3).x
Train(3).x = Train(4).x
Train(4).x = Train(5).x
Train(5).x = Train(2).x
```



C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\train-gate.xml - UPPAAL

Enabled Transitions

```
Train(1)
appr[2]: Train(2) → Gate
appr[3]: Train(3) → Gate
appr[4]: Train(4) → Gate
appr[5]: Train(5) → Gate
```

```
Gate.list[0] = 1
Gate.list[1] = 0
Gate.list[2] = 0
Gate.list[3] = 0
Gate.list[4] = 0
Gate.list[5] = 0
Gate.list[6] = 0
Gate.len = 2
Train(0).x ∈ [0,20]
Train(1).x ∈ [0,20]
Train(2).x ≥ 0
Train(3).x ≥ 0
Train(4).x ≥ 0
Train(5).x ≥ 0
Train(0).x - Train(1).x ∈ [-20,
Train(1).x ≤ Train(2).x
Train(2).x = Train(3).x
Train(3).x = Train(4).x
Train(4).x = Train(5).x
Train(5).x = Train(2).x
```

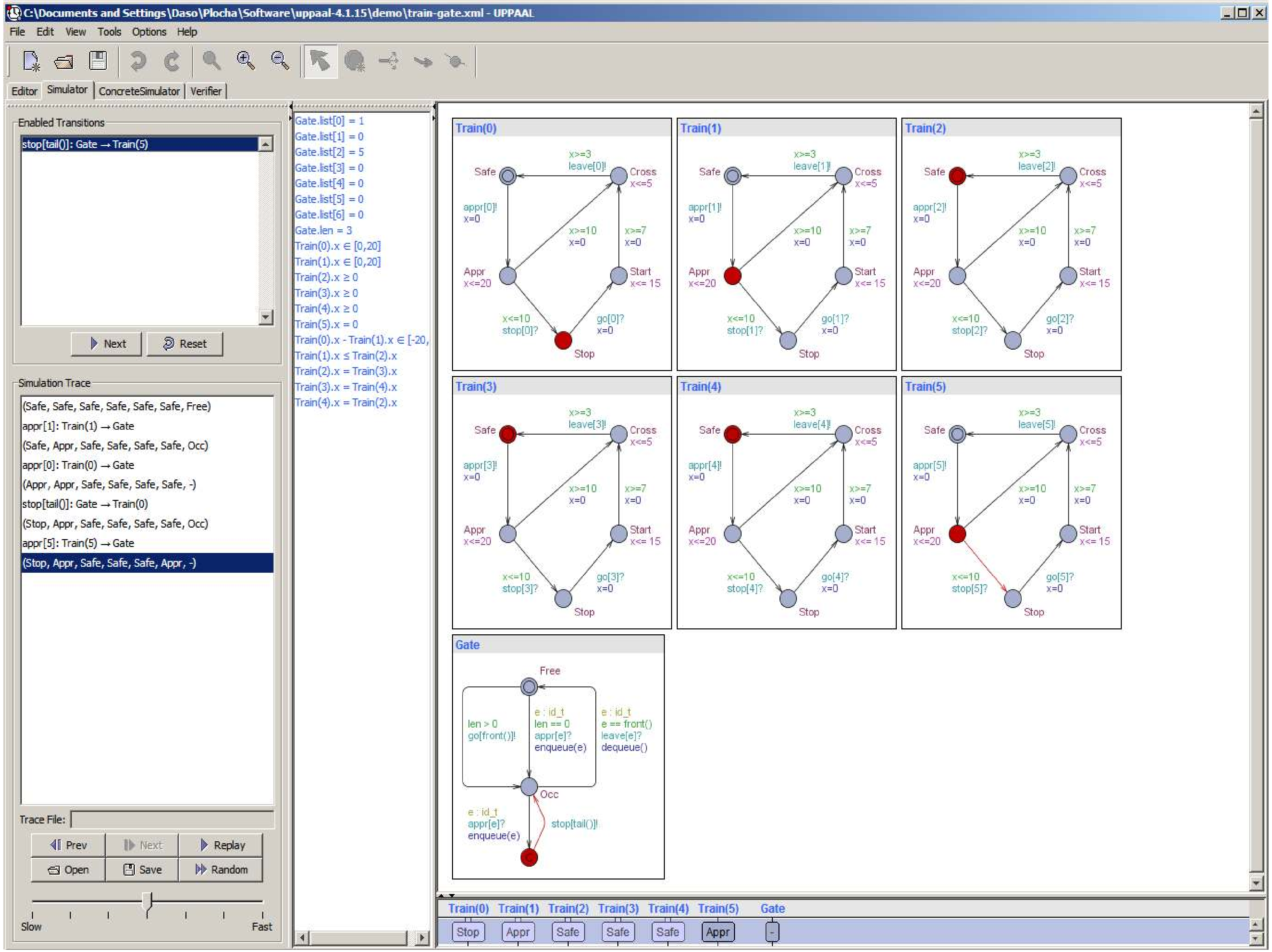
Simulation Trace

```
(Safe, Safe, Safe, Safe, Safe, Safe, Free)
appr[1]: Train(1) → Gate
(Safe, Appr, Safe, Safe, Safe, Safe, Occ)
appr[0]: Train(0) → Gate
(Appr, Appr, Safe, Safe, Safe, Safe, -)
stop[tail()]: Gate → Train(0)
(Stop, Appr, Safe, Safe, Safe, Safe, Occ)
```

Trace File:

Train(0) Train(1) Train(2) Train(3) Train(4) Train(5) Gate

Stop Appr Safe Safe Safe Safe Occ



Enabled Transitions:

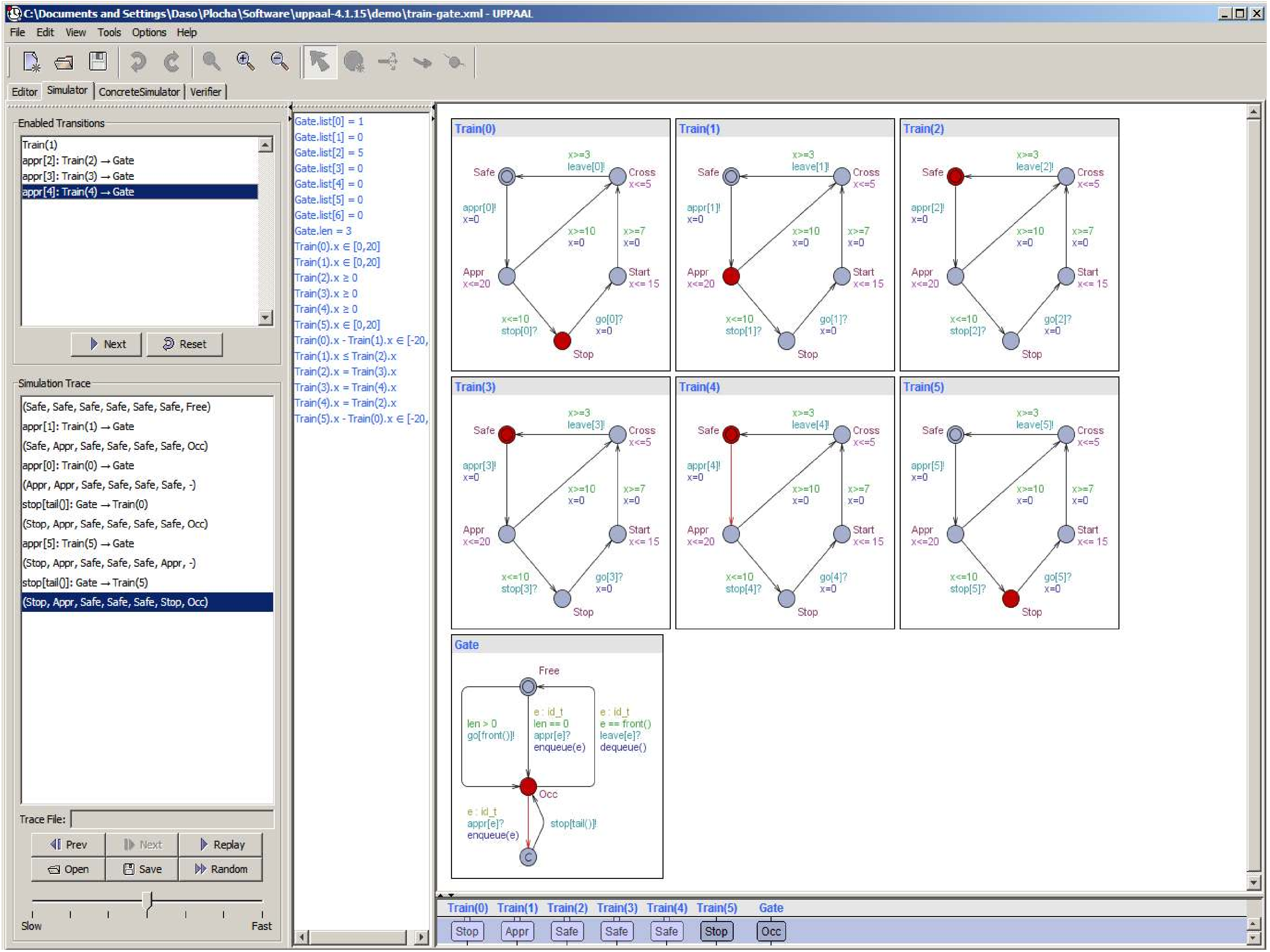
```
stop[tail()]: Gate → Train(5)
```

Simulation Trace:

```
(Safe, Safe, Safe, Safe, Safe, Safe, Free)
appr[1]: Train(1) → Gate
(Safe, Appr, Safe, Safe, Safe, Safe, Occ)
appr[0]: Train(0) → Gate
(Appr, Appr, Safe, Safe, Safe, Safe, -)
stop[tail()]: Gate → Train(0)
(Stop, Appr, Safe, Safe, Safe, Safe, Occ)
appr[5]: Train(5) → Gate
(Stop, Appr, Safe, Safe, Safe, Appr, -)
```

Variables:

```
Gate.list[0] = 1
Gate.list[1] = 0
Gate.list[2] = 5
Gate.list[3] = 0
Gate.list[4] = 0
Gate.list[5] = 0
Gate.list[6] = 0
Gate.len = 3
Train(0).x ∈ [0,20]
Train(1).x ∈ [0,20]
Train(2).x ≥ 0
Train(3).x ≥ 0
Train(4).x ≥ 0
Train(5).x = 0
Train(0).x - Train(1).x ∈ [-20,
Train(1).x ≤ Train(2).x
Train(2).x = Train(3).x
Train(3).x = Train(4).x
Train(4).x = Train(2).x
```



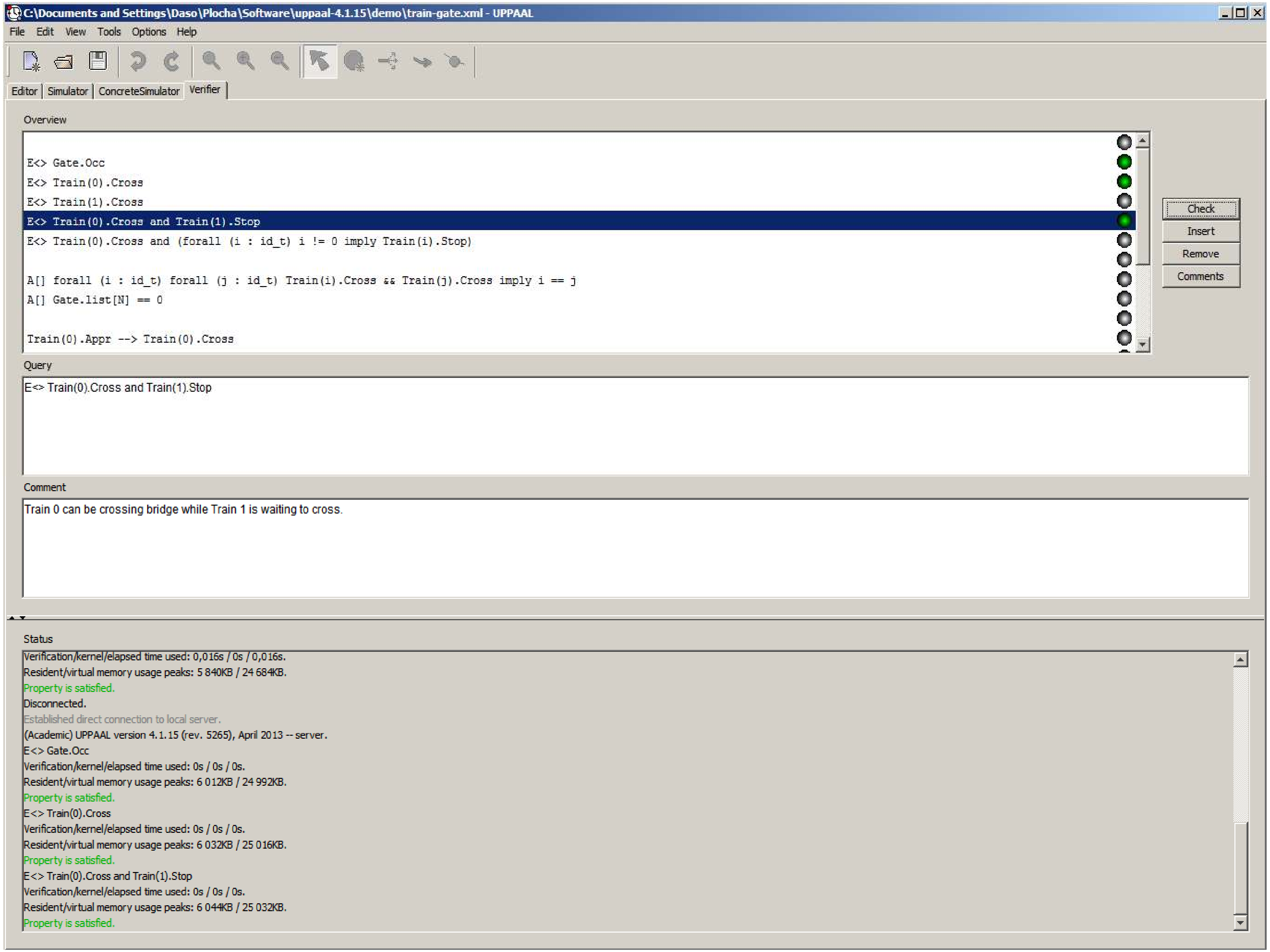
Enabled Transitions

```
Train(1)
appr[2]: Train(2) → Gate
appr[3]: Train(3) → Gate
appr[4]: Train(4) → Gate
```

Simulation Trace

```
(Safe, Safe, Safe, Safe, Safe, Safe, Free)
appr[1]: Train(1) → Gate
(Safe, Appr, Safe, Safe, Safe, Safe, Occ)
appr[0]: Train(0) → Gate
(Appr, Appr, Safe, Safe, Safe, Safe, -)
stop[tail()]: Gate → Train(0)
(Stop, Appr, Safe, Safe, Safe, Safe, Occ)
appr[5]: Train(5) → Gate
(Stop, Appr, Safe, Safe, Safe, Appr, -)
stop[tail()]: Gate → Train(5)
(Stop, Appr, Safe, Safe, Safe, Stop, Occ)
```

```
Gate.list[0] = 1
Gate.list[1] = 0
Gate.list[2] = 5
Gate.list[3] = 0
Gate.list[4] = 0
Gate.list[5] = 0
Gate.list[6] = 0
Gate.len = 3
Train(0).x ∈ [0,20]
Train(1).x ∈ [0,20]
Train(2).x ≥ 0
Train(3).x ≥ 0
Train(4).x ≥ 0
Train(5).x ∈ [0,20]
Train(0).x - Train(1).x ∈ [-20,
Train(1).x ≤ Train(2).x
Train(2).x = Train(3).x
Train(3).x = Train(4).x
Train(4).x = Train(2).x
Train(5).x - Train(0).x ∈ [-20,
```


C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\train-gate.xml - UPPAAL
File Edit View Tools Options Help
Editor
Simulator
ConcreteSimulator
Verifier
Check
Insert
Remove
Comments

Overview

```
E<> Gate.Occ
E<> Train(0).Cross
E<> Train(1).Cross
E<> Train(0).Cross and Train(1).Stop
E<> Train(0).Cross and (forall (i : id_t) i != 0 imply Train(i).Stop)

A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imply i == j
A[] Gate.list[N] == 0

Train(0).Appr --> Train(0).Cross
```

Query

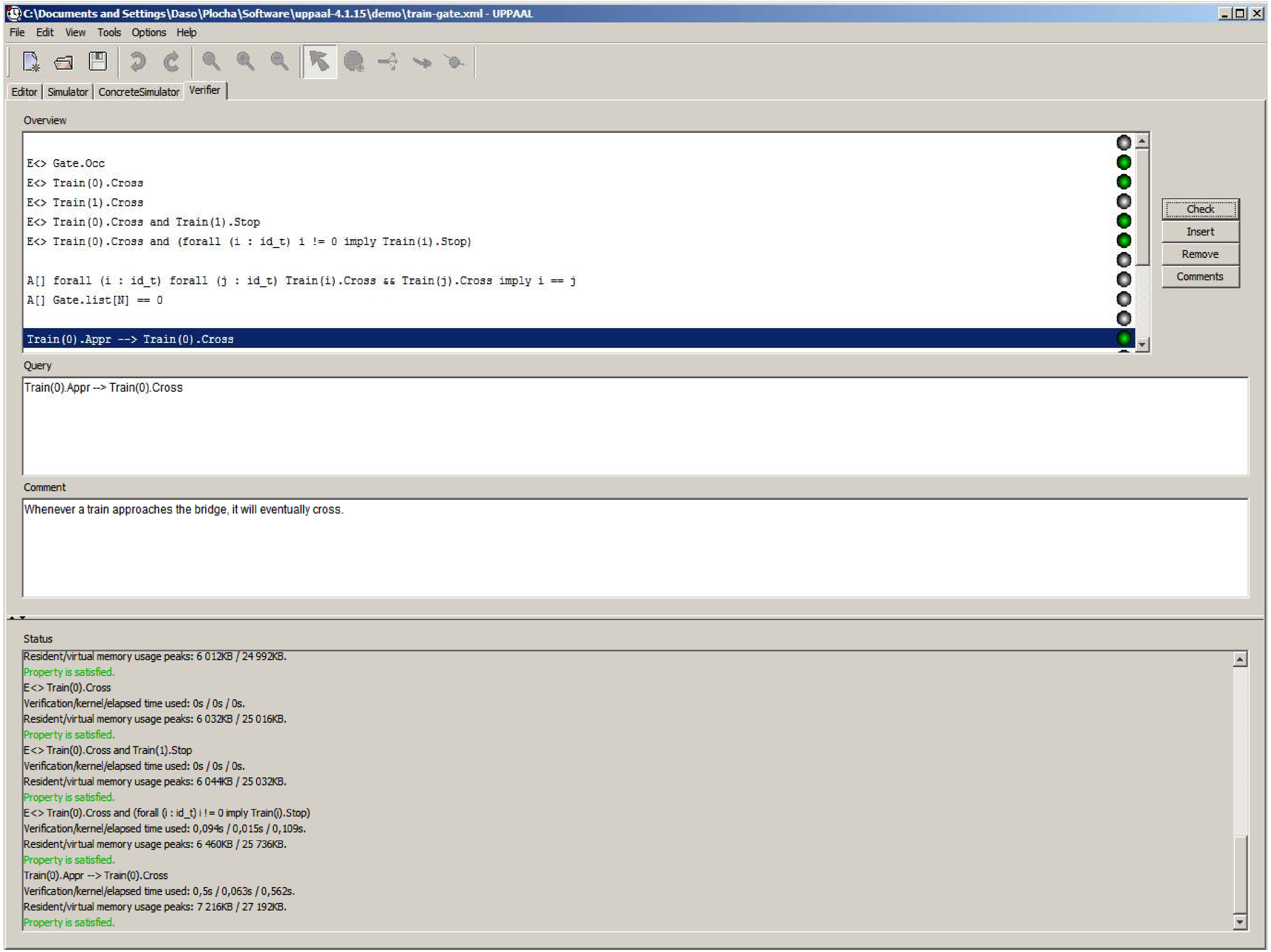
```
E<> Train(0).Cross and Train(1).Stop
```

Comment

Train 0 can be crossing bridge while Train 1 is waiting to cross.

Status

```
Verification/kernel/elapsed time used: 0,016s / 0s / 0,016s.
Resident/virtual memory usage peaks: 5 840KB / 24 684KB.
Property is satisfied.
Disconnected.
Established direct connection to local server.
(Academic) UPPAAL version 4.1.15 (rev. 5265), April 2013 -- server.
E<> Gate.Occ
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 6 012KB / 24 992KB.
Property is satisfied.
E<> Train(0).Cross
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 6 032KB / 25 016KB.
Property is satisfied.
E<> Train(0).Cross and Train(1).Stop
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 6 044KB / 25 032KB.
Property is satisfied.
```


C:\Documents and Settings\Daso\Plocha\Software\uppaal-4.1.15\demo\train-gate.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier
Overview
Check
Insert
Remove
Comments
Query
Comment
Status

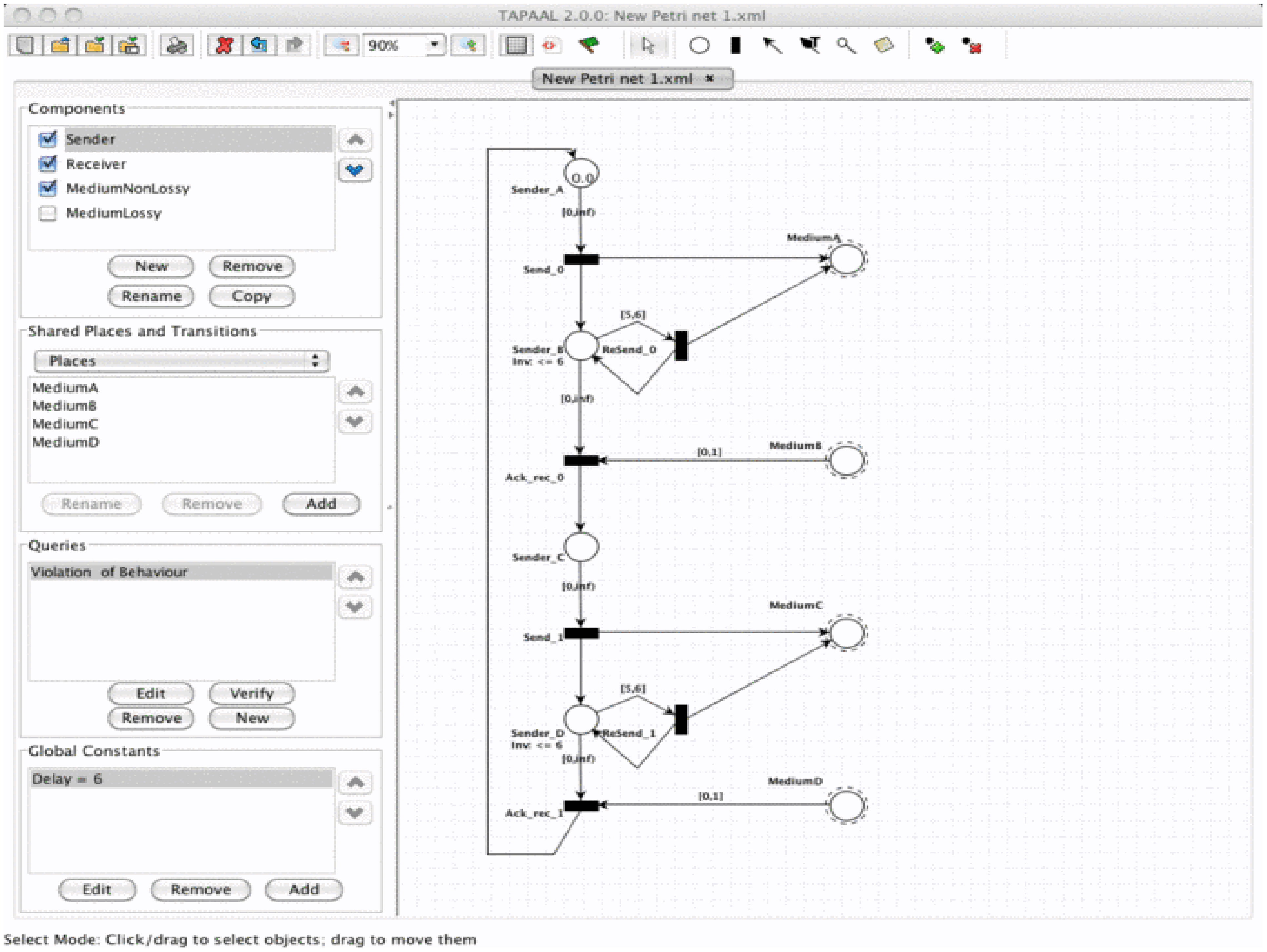
```
E<> Gate.Occ
E<> Train(0).Cross
E<> Train(1).Cross
E<> Train(0).Cross and Train(1).Stop
E<> Train(0).Cross and (forall (i : id_t) i != 0 imply Train(i).Stop)

A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imply i == j
A[] Gate.list[N] == 0

Train(0).Appr --> Train(0).Cross
```

Query: Train(0).Appr --> Train(0).Cross

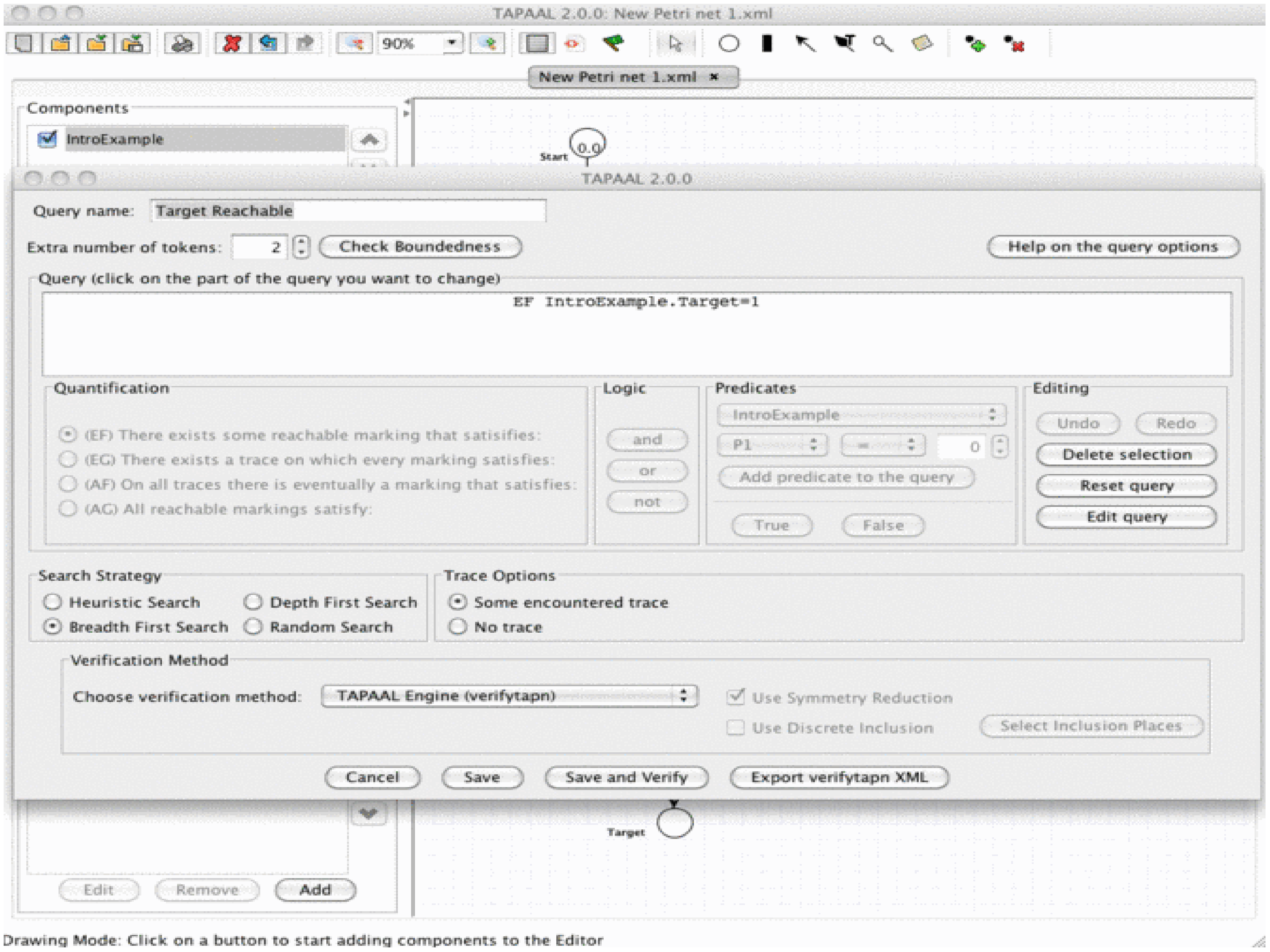
Comment: Whenever a train approaches the bridge, it will eventually cross.

```
Resident/virtual memory usage peaks: 6 012KB / 24 992KB.
Property is satisfied.
E<> Train(0).Cross
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 6 032KB / 25 016KB.
Property is satisfied.
E<> Train(0).Cross and Train(1).Stop
Verification/kernel/elapsed time used: 0s / 0s / 0s.
Resident/virtual memory usage peaks: 6 044KB / 25 032KB.
Property is satisfied.
E<> Train(0).Cross and (forall (i : id_t) i != 0 imply Train(i).Stop)
Verification/kernel/elapsed time used: 0,094s / 0,015s / 0,109s.
Resident/virtual memory usage peaks: 6 460KB / 25 736KB.
Property is satisfied.
Train(0).Appr --> Train(0).Cross
Verification/kernel/elapsed time used: 0,5s / 0,063s / 0,562s.
Resident/virtual memory usage peaks: 7 216KB / 27 192KB.
Property is satisfied.
```

TAPAAL



Select Mode: Click/drag to select objects; drag to move them


TAPAAL 2.0.0: New Petri net 1.xml

New Petri net 1.xml

Components

IntroExample

TAPAAL 2.0.0

Query name: Target Reachable

Extra number of tokens: 2

Query (click on the part of the query you want to change)

```
EF IntroExample.Target=1
```

Quantification

- [✓ (EF) There exists some reachable marking that satisifies:]
- (EG) There exists a trace on which every marking satisfies:
- (AF) On all traces there is eventually a marking that satisfies:
- (AG) All reachable markings satisfy:

Predicates: IntroExample, P1, =, 0

Search Strategy

- Heuristic Search
- Depth First Search
- [✓ Breadth First Search]
- Random Search

Trace Options

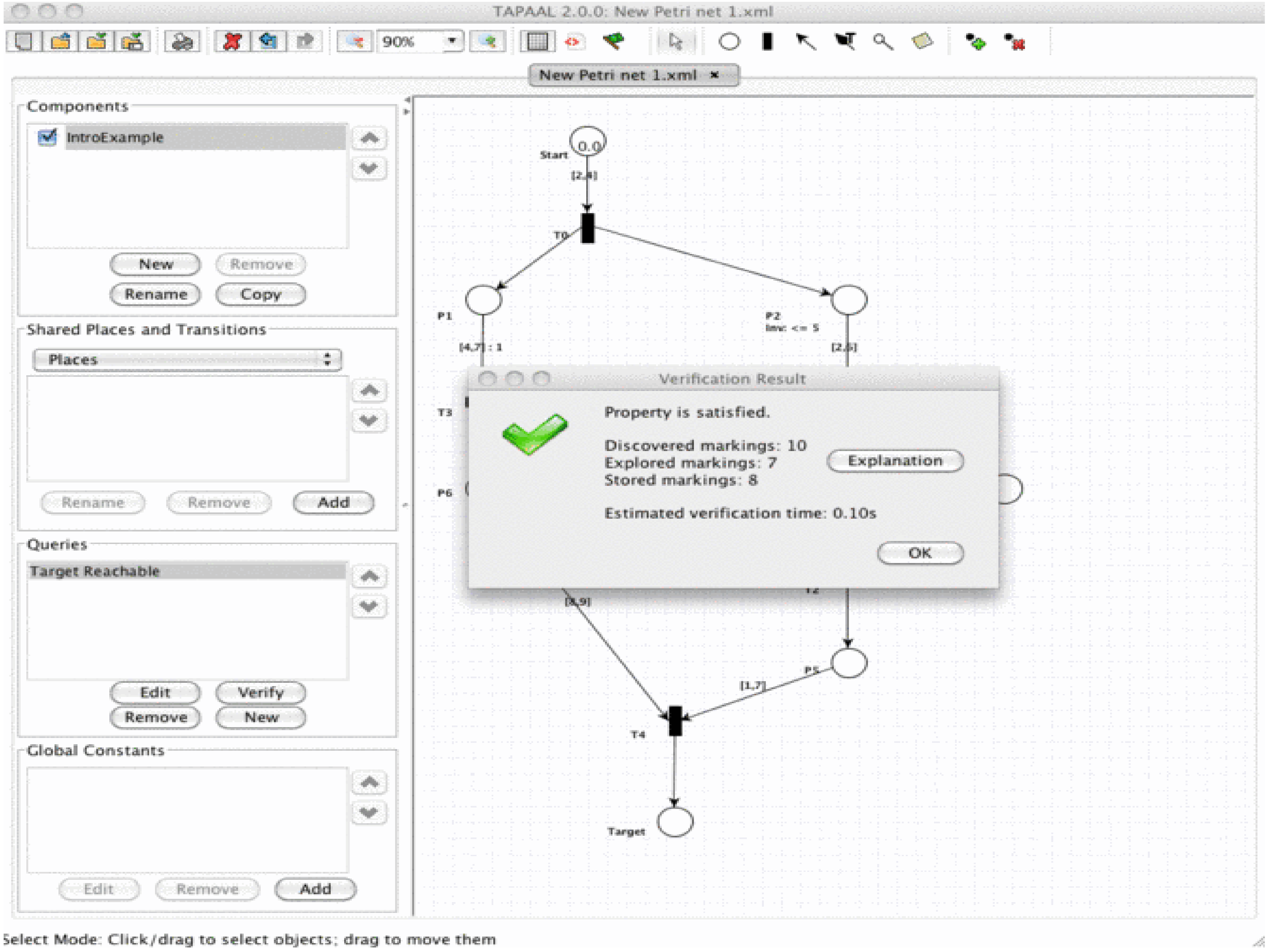
- [✓ Some encountered trace]
- No trace

Verification Method

Choose verification method: TAPAAL Engine (verifytapn)

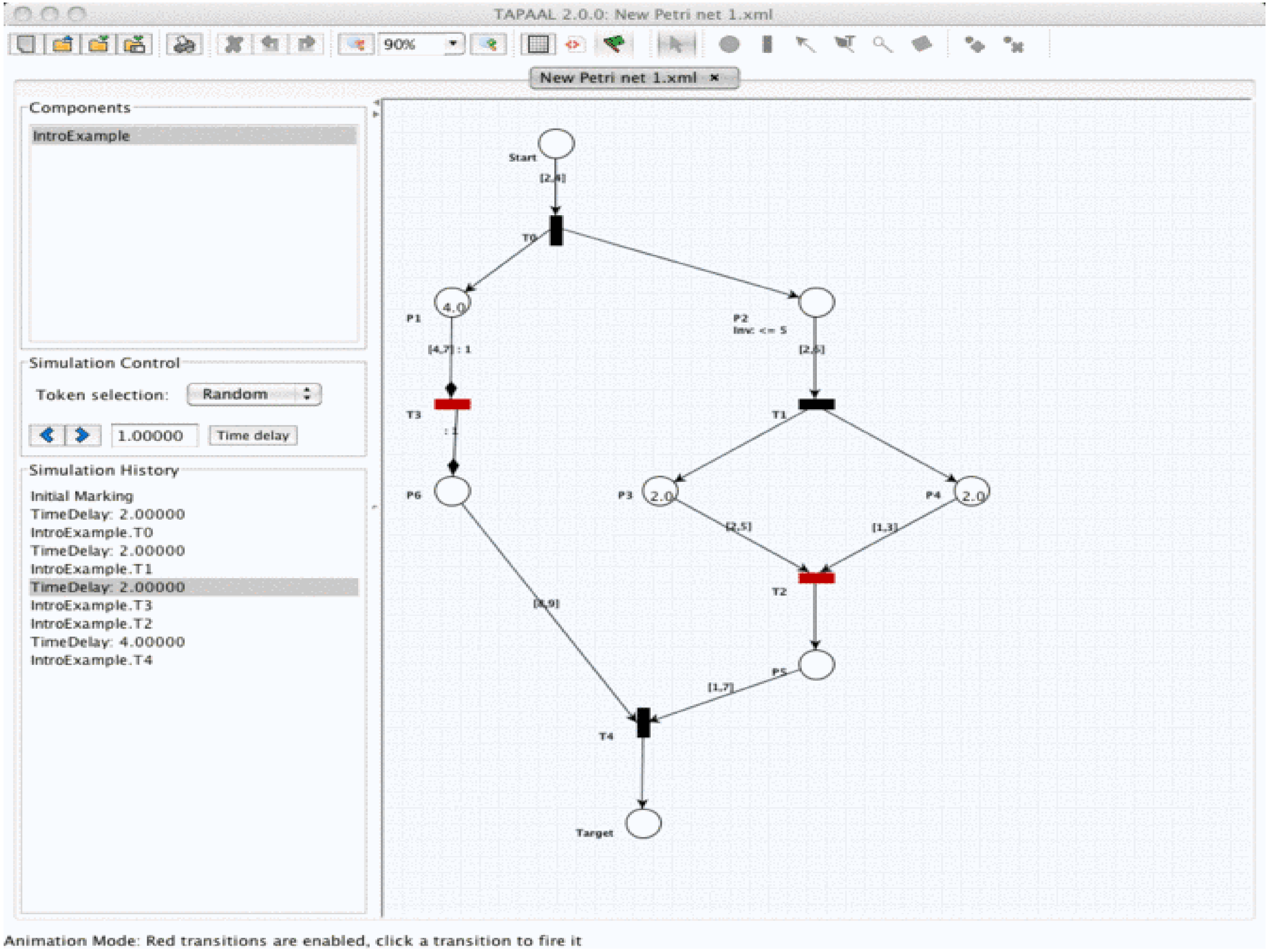
- [✓ Use Symmetry Reduction]
- Use Discrete Inclusion

Drawing Mode: Click on a button to start adding components to the Editor



```
Property is satisfied.

Discovered markings: 10
Explored markings: 7
Stored markings: 8

Estimated verification time: 0.10s
```

Select Mode: Click/drag to select objects; drag to move them



Components

IntroExample

Simulation Control

Token selection: Random

1.00000 Time delay

Simulation History

Initial Marking
TimeDelay: 2.00000
IntroExample.T0
TimeDelay: 2.00000
IntroExample.T1
TimeDelay: 2.00000
IntroExample.T3
IntroExample.T2
TimeDelay: 4.00000
IntroExample.T4

Animation Mode: Red transitions are enabled, click a transition to fire it