Model Checking Task Sets with Preemption Thresholds

Mitchell L. Neilsen, { neilsen@k-state.edu }
Dept. of Computing and Information Sciences
Kansas State University
234 Nichols Hall
Manhattan, KS 66506

PDPTA 2011 - July 18-21, 2011

Periodic Task Model

- Each task τ_i is characterized by a 5-tuple of natural numbers denoted:

$$(C_i, T_i, D_i, \pi_i, \gamma_i)$$

where

  - C_i is the run-time of task τ_i,
  - T_i is the period of task τ_i,
  - D_i is its relative deadline,
  - π_i is its static priority, and
  - γ_i is its preemption threshold.

Example Periodic Task Set

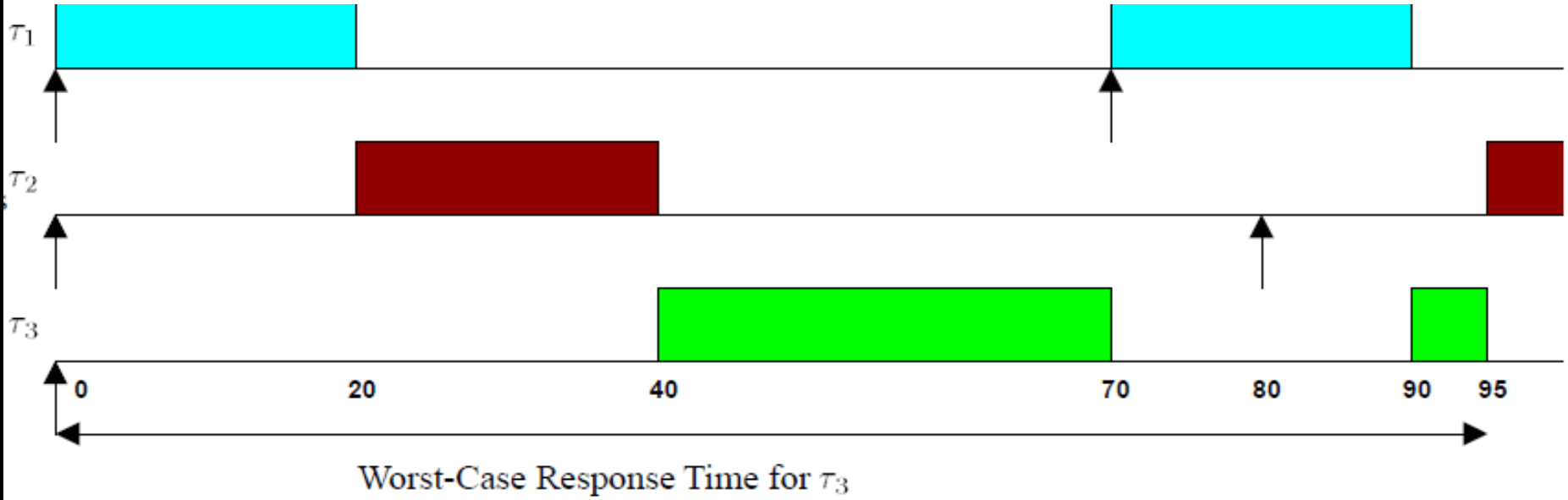
- From Wang and Saksena's original paper on preemption thresholds

Table 1. Periodic task set

i	C_i	T_i	D_i	π_i	γ_i
0	20	70	50	3	3
1	20	80	80	2	3
2	35	200	100	1	2

Task	π_i	WCRT Preemptive $\gamma_i = \pi_i$	WCRT Non-Preemptive $\gamma_i = 3$	γ_i	WCRT Preemption-Threshold
τ_1	3	20	**55**	3	40
τ_2	2	40	75	3	75
τ_3	1	**115**	75	2	95

Table 2. Response Times for Tasks under Different Schedulers.

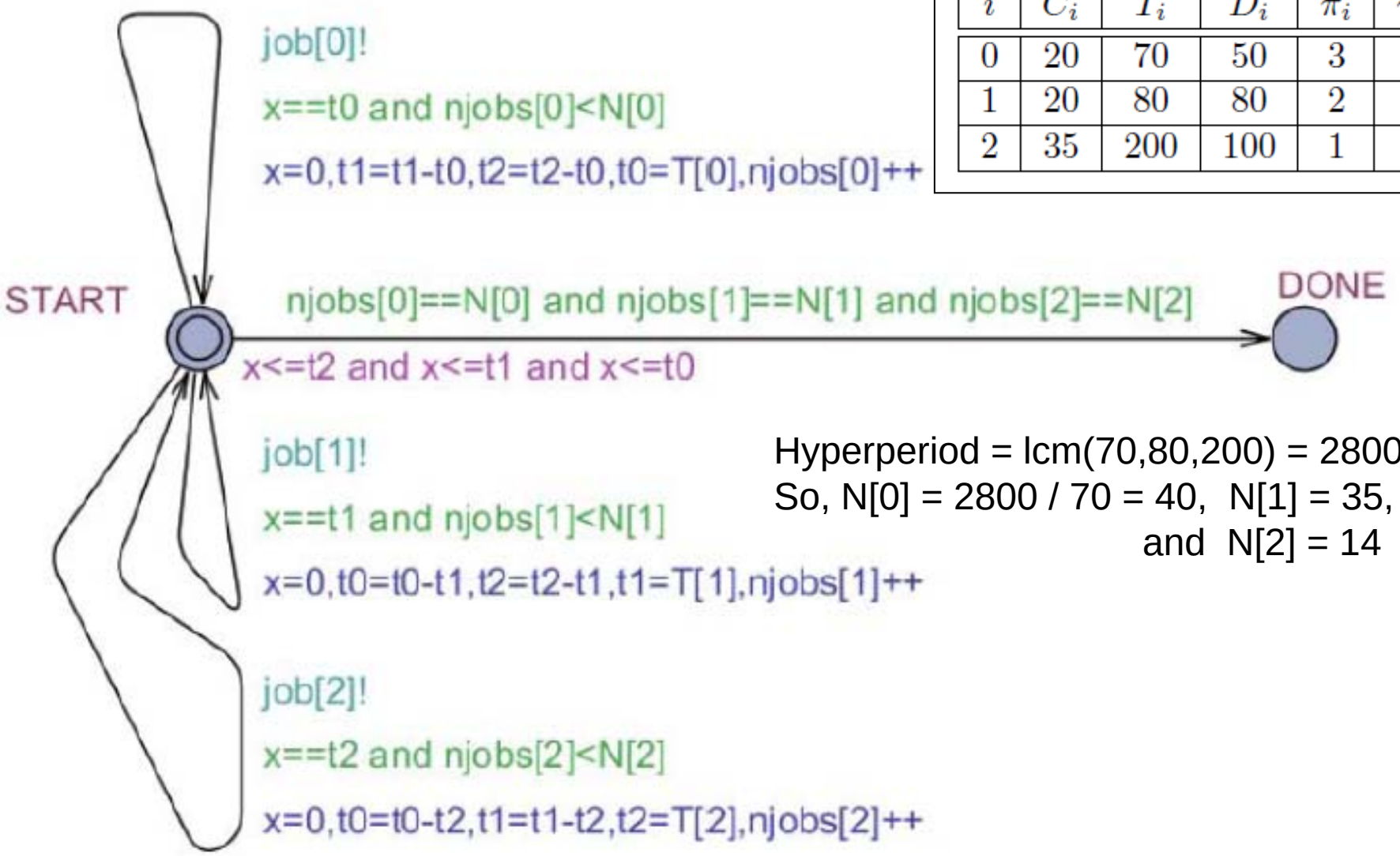


Symbolic Schedulability Analysis

- Construct a formal model consisting of two automata:
  - PERIODIC_TASKS - to generate jobs
  - SCHEDULER - to model scheduling algorithm
- Perform symbolic analysis on SCHEDULER automaton to see if an ERROR state is reachable
- UPPAAL model checker is used to perform analysis

Table 1. Periodic task set

i	C_i	T_i	D_i	π_i	γ_i
0	20	70	50	3	3
1	20	80	80	2	3
2	35	200	100	1	2

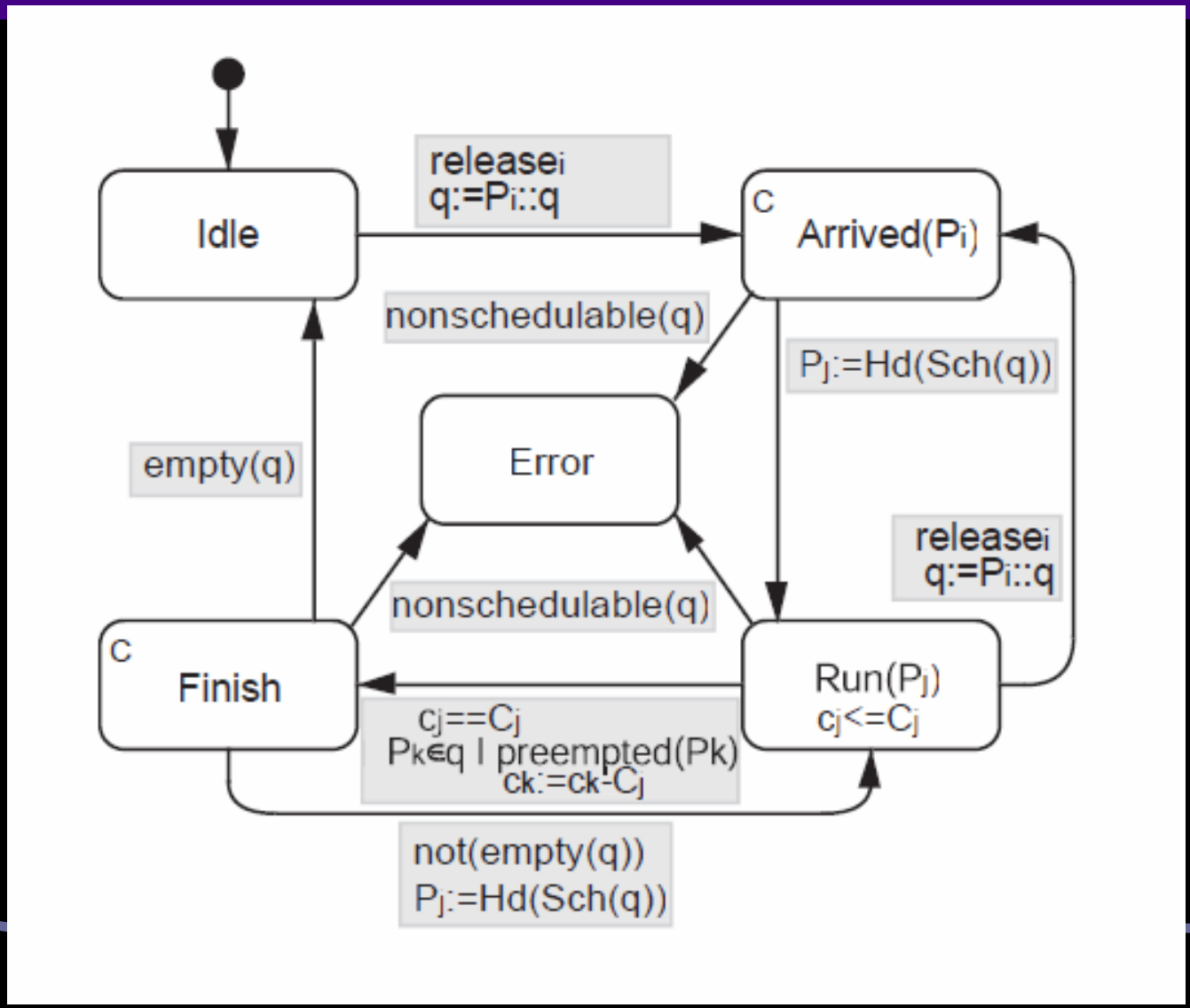


Hyperperiod = lcm(70,80,200) = 2800
So, N[0] = 2800 / 70 = 40, N[1] = 35,
and N[2] = 14

Figure 1. PERIODIC_TASKS automaton

General O(n)-Clocks Scheduler

- E. Fersman, et. al



New 2-Clocks Scheduler

- Correctly detects when an arbitrary task set is feasible.
- Only requires two clocks for ***n*** tasks:
  - One clock to check if the currently scheduled task has missed its deadline, **d**, and
  - One clock to check that the running task has completed it’s execution, **c**.
- For simplicity, we consider Wang and Saksena's example with just ***n=3*** tasks, but it is trivial to generalize to any ***n***.

SCHEDULER Automaton

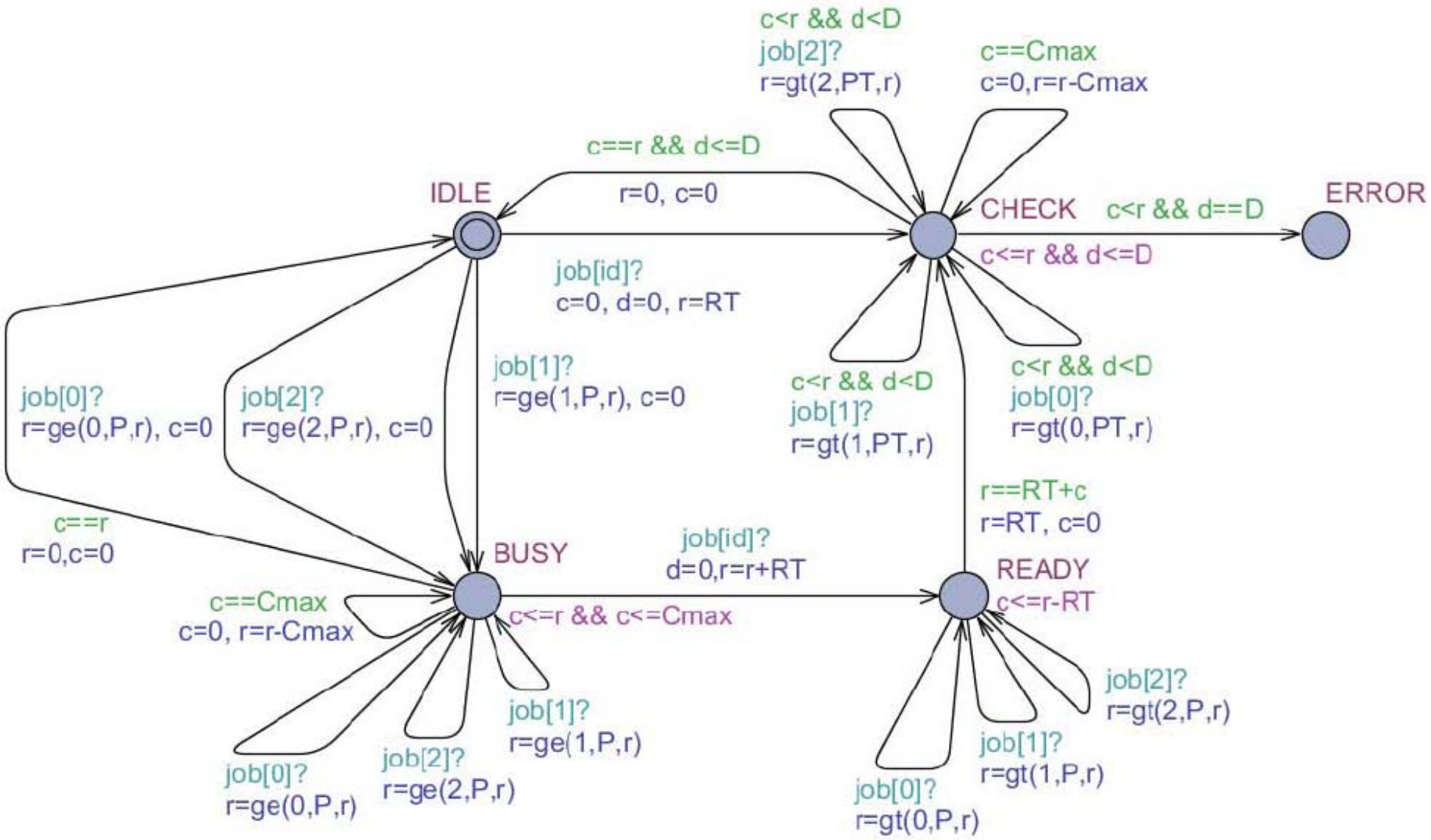


Figure 2. SCHEDULER automaton

System Declarations

```
S0 = SCHEDULER(PR[0],PRT[0],C[0],D[0],0);
S1 = SCHEDULER(PR[1],PRT[1],C[1],D[1],1);
S2 = SCHEDULER(PR[2],PRT[2],C[2],D[2],2);

system PERIODIC_TASKS,S0,S1,S2;
```

… and SCHEDULER prototype:

```
SCHEDULER( const int PR,  const int PRT,  const int RT,
                    const int D,  const int id );
```

SCHEDULER Automaton

```
clock c, d;
int r;
int Cmax = 1;

int gt(int i, int x, int r)
{
  if (PR[i]>x)
    return (r+C[i]);
  else
    return (r);
}
```

```
int ge(int i, int x, int r)
{
  if (PRT[i]>=x)
    return (r+C[i]);
  else
    return (r);
}
```

SCHEDULER States

- $Idle_i$ - the ready job queue is empty,
- $Busy_i$ - jobs with priority greater than or equal to task i have arrived for execution,
- $Ready_i$ - the job in task i to be checked for feasibility has been released for execution, but has not started executing, and
- $Check_i$ - the job in task i to be checked for feasibility has started executing, and
- $Error_i$ - the checked job in task i missed its deadline.

Preemptive Task Set

```
const int PR[3] = { 3, 2, 1 };
const int PRT[3] = { 3, 2, 1 };
```

	Task	π_i	WCRT Preemptive $\gamma_i = \pi_i$	WCRT Non-Preemptive $\gamma_i = 3$	γ_i	WCRT Preemption-Threshold
S0	τ_1	3	20	**55**	3	40
S1	τ_2	2	40	75	3	75
S2	τ_3	1	**115**	75	2	95

Table 2. Response Times for Tasks under Different Schedulers.

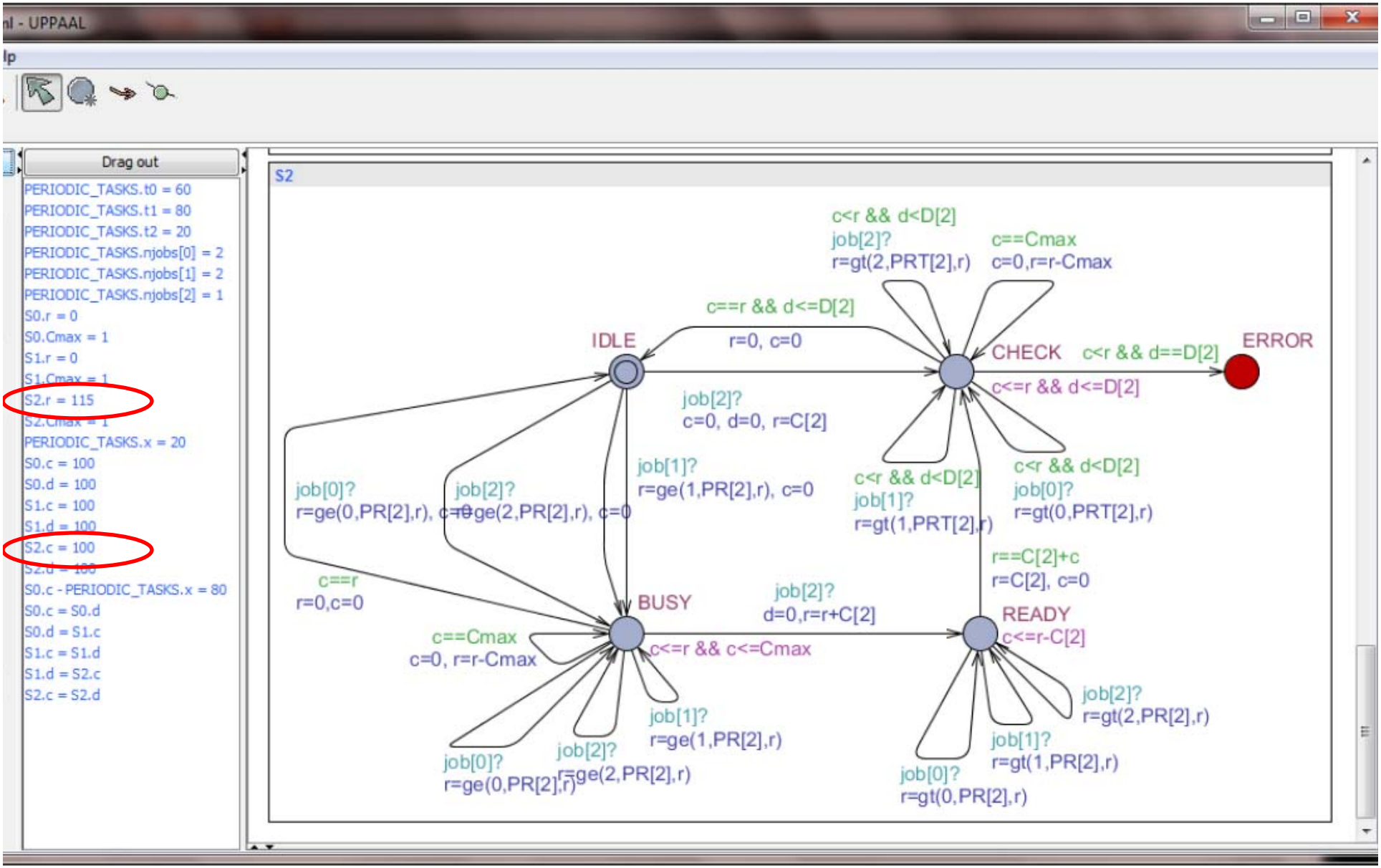


Figure 3. Missed deadline

Task Set is not Feasible

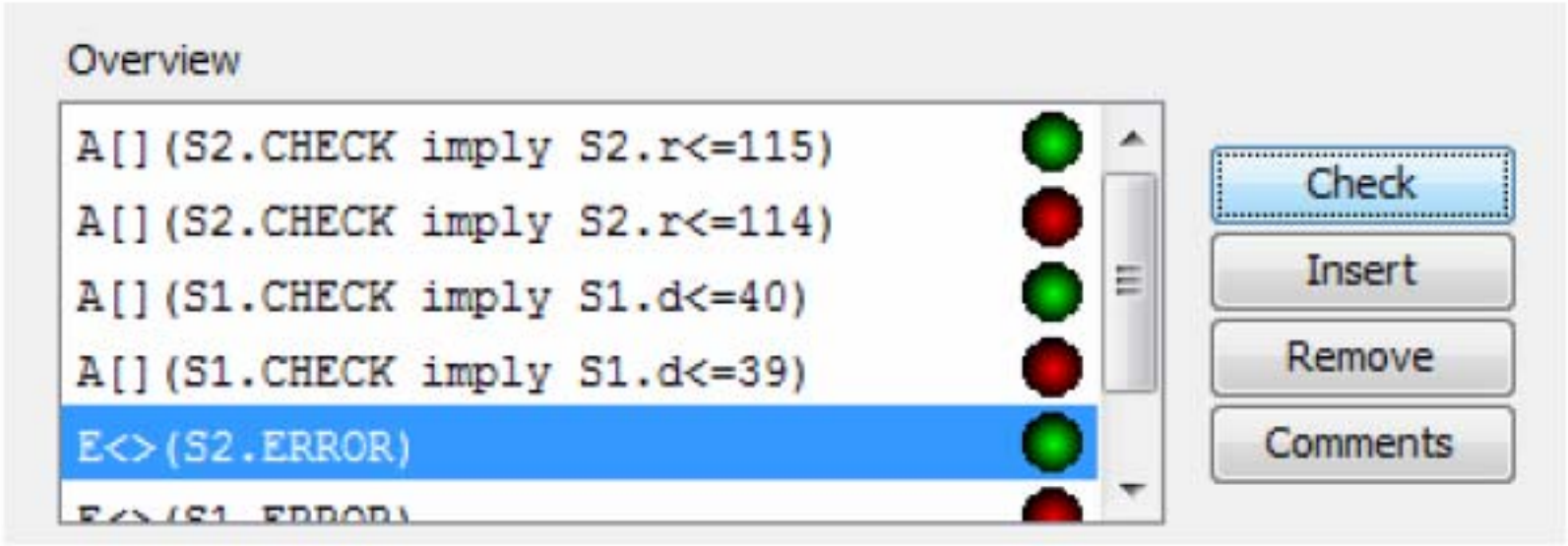


Figure 4. Preemptive tasks output

S0 = scheduler automaton for task τ_1

S1 = scheduler automaton for task τ_2

S2 = scheduler automaton for task τ_3

Non-Preemptive Task Set

const int PR[3] = { 3, 2, 1 };

const int PRT[3] = { 3, 3, 3 };

Task	π_i	WCRT Preemptive $\gamma_i = \pi_i$	WCRT Non-Preemptive $\gamma_i = 3$	γ_i	WCRT Preemption-Threshold
τ_1	3	20	**55**	3	40
τ_2	2	40	75	3	75
τ_3	1	**115**	75	2	95

Table 2. Response Times for Tasks under Different Schedulers.

In this case, E<> S0.Error is satisfied

Preemption-Threshold Task Set

$$\mathrm{PRT}[3] = \{\ 3,\ 3,\ 2\ \}$$

Task	π_i	WCRT Preemptive $\gamma_i = \pi_i$	WCRT Non-Preemptive $\gamma_i = 3$	γ_i	WCRT Preemption-Threshold
τ_1	3	20	**55**	3	40
τ_2	2	40	75	3	75
τ_3	1	**115**	75	2	95

Table 2. Response Times for Tasks under Different Schedulers.

Verifier Output

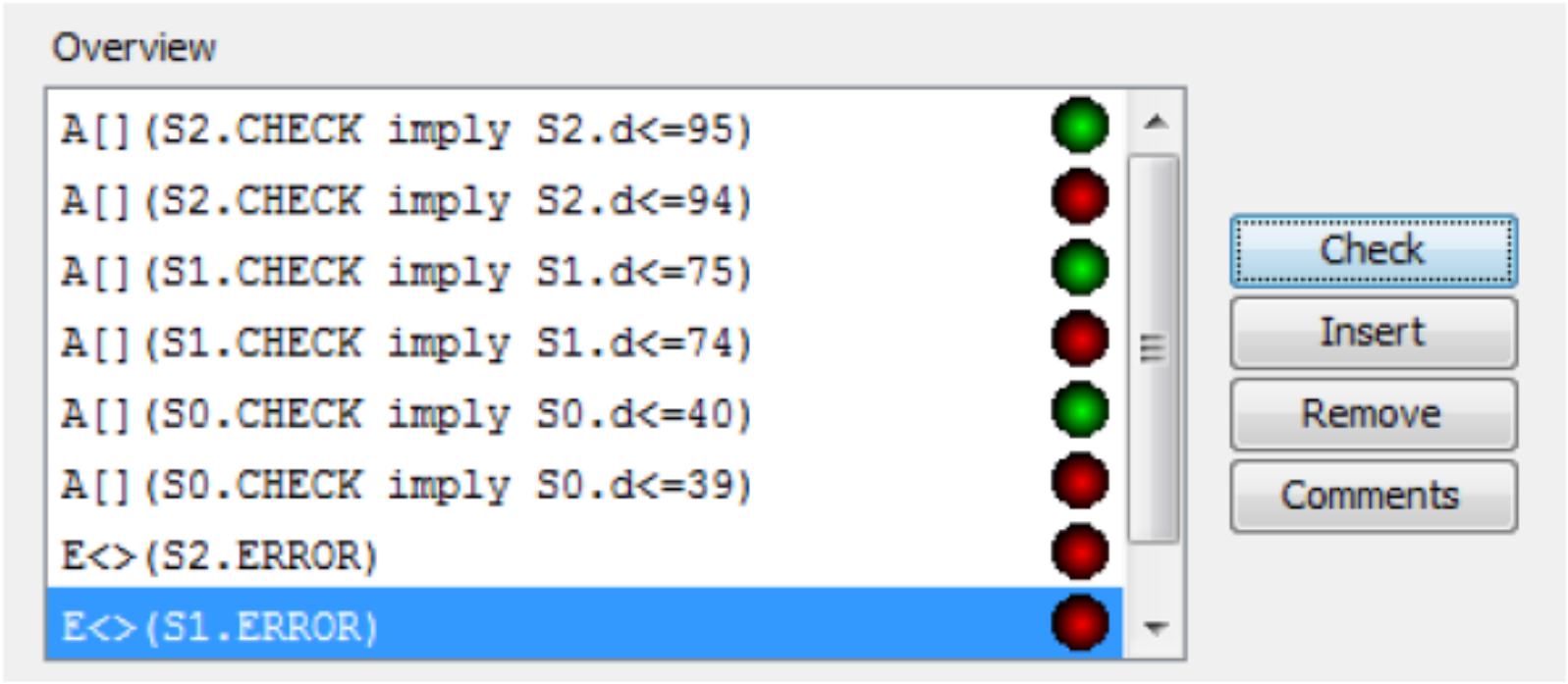


Figure 5. Verification output

SCHEDULER Automaton

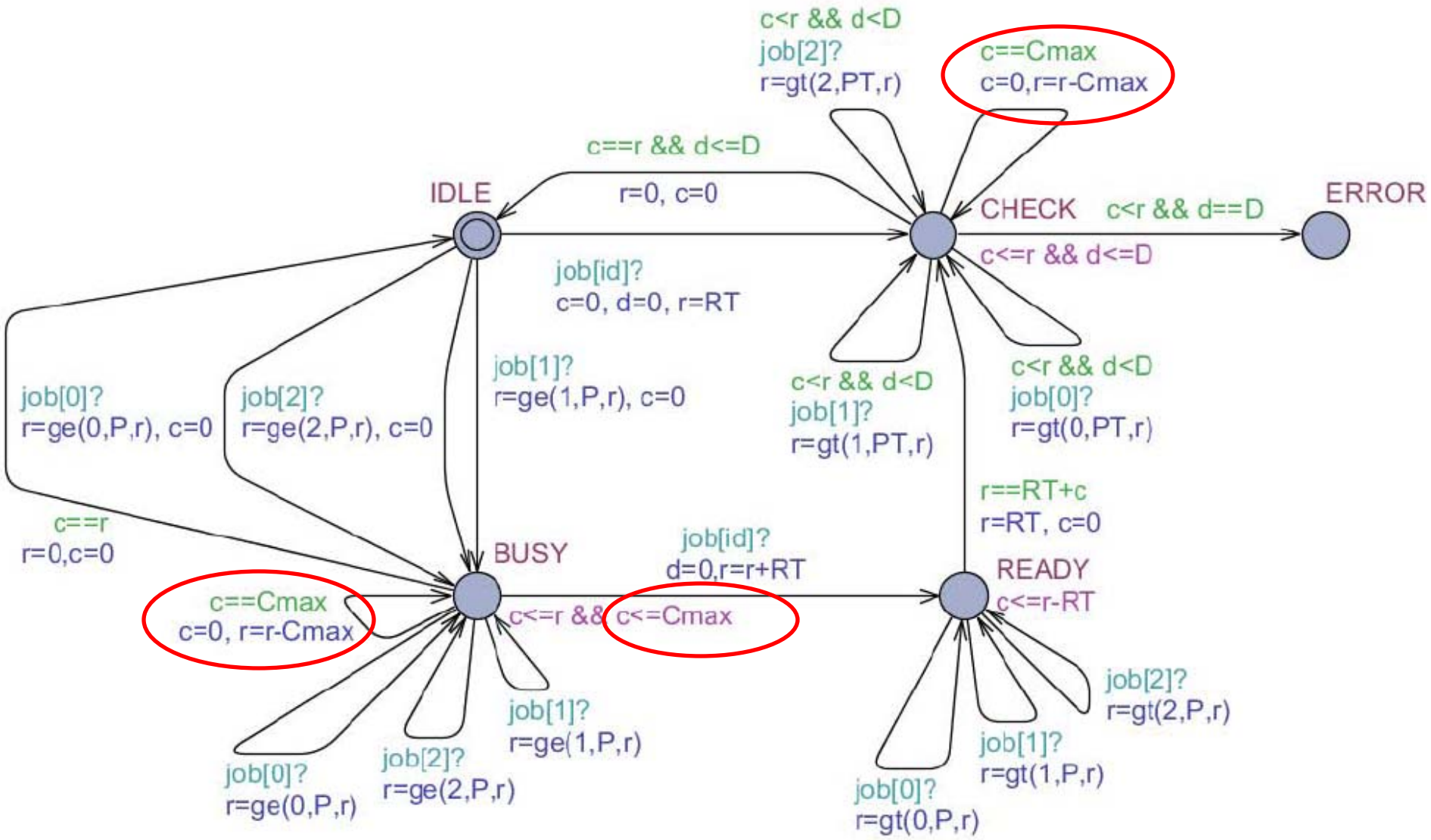


Figure 2. SCHEDULER automaton

Effect of Changing Cmax

Table 2. Bounds on number of states.

Cmax	States	Cmax	States
1	150,180	50	1,793
3	20,820	100	1,635
6	8,155	250	1,617
12	3,805	500	1,617

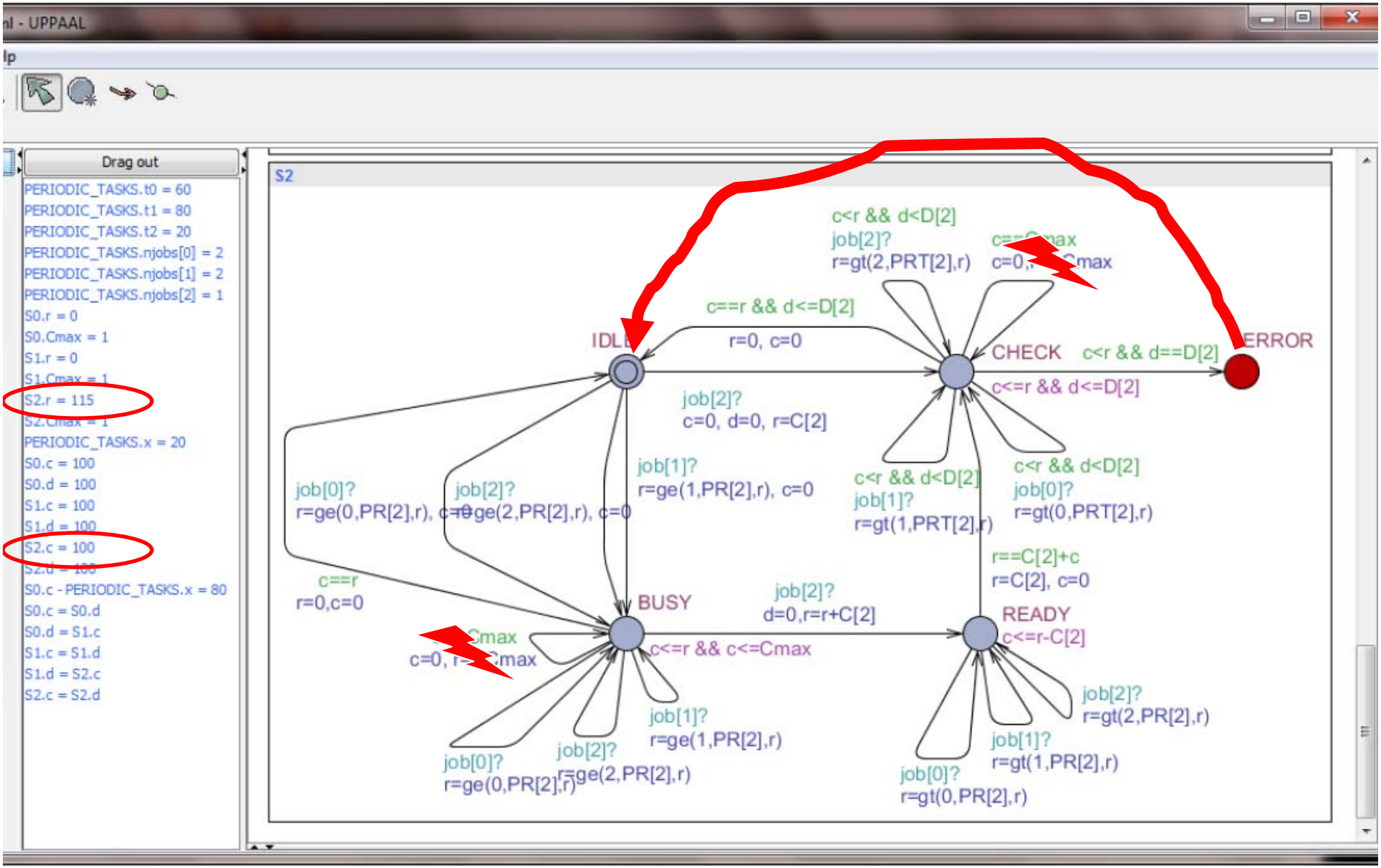
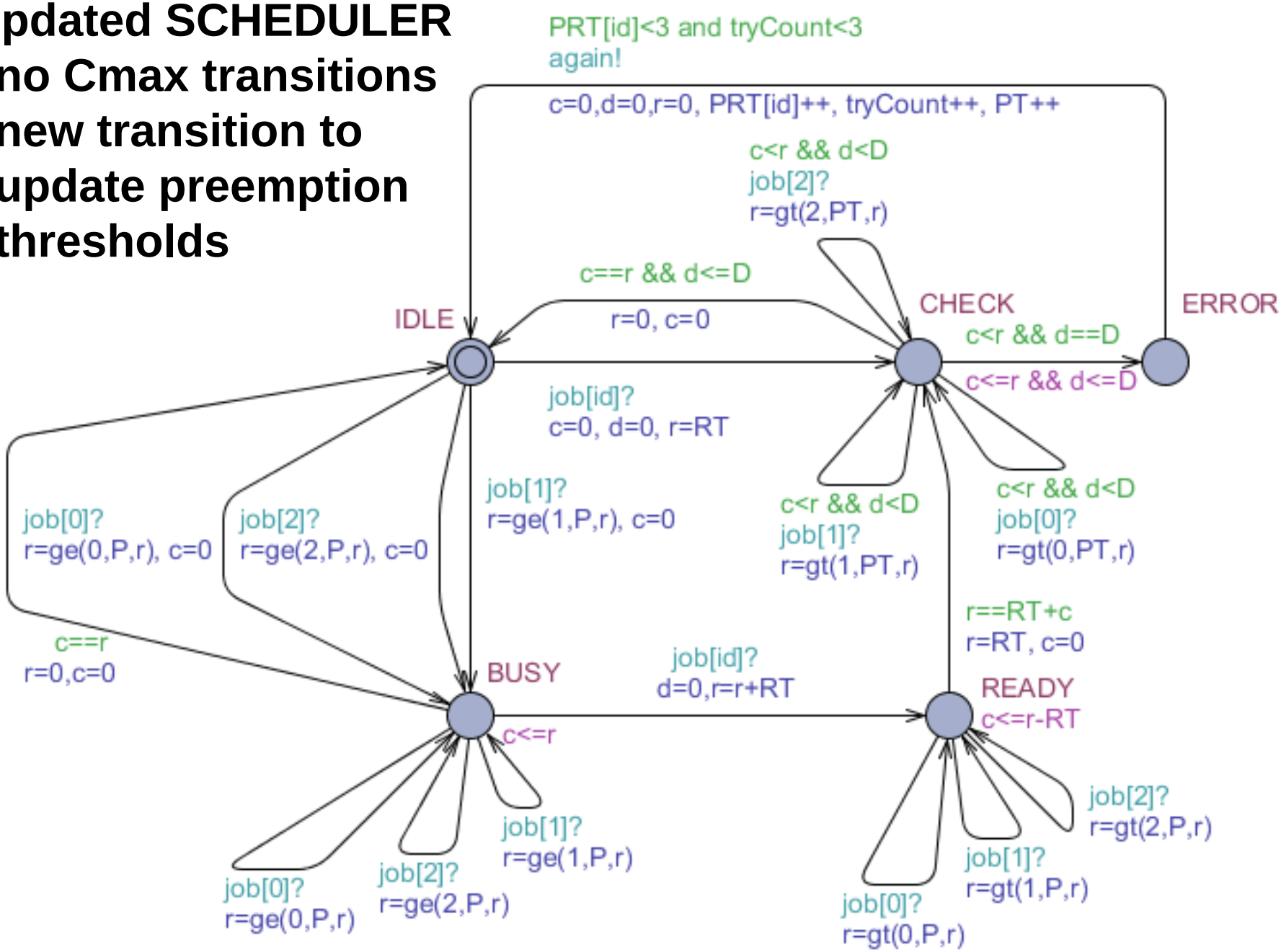


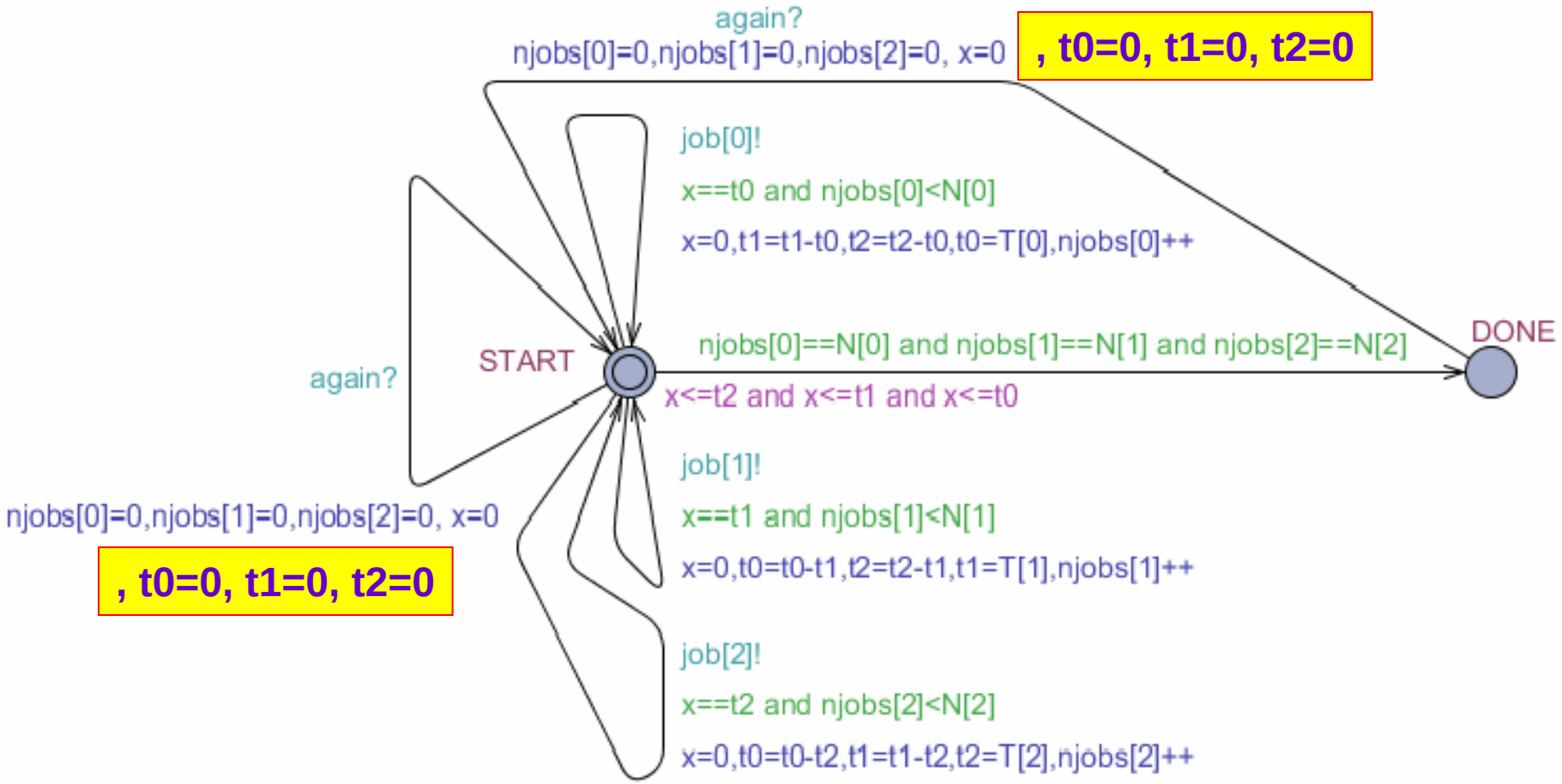
Figure 3. Missed deadline

Updated SCHEDULER

- no Cmax transitions
- new transition to update preemption thresholds

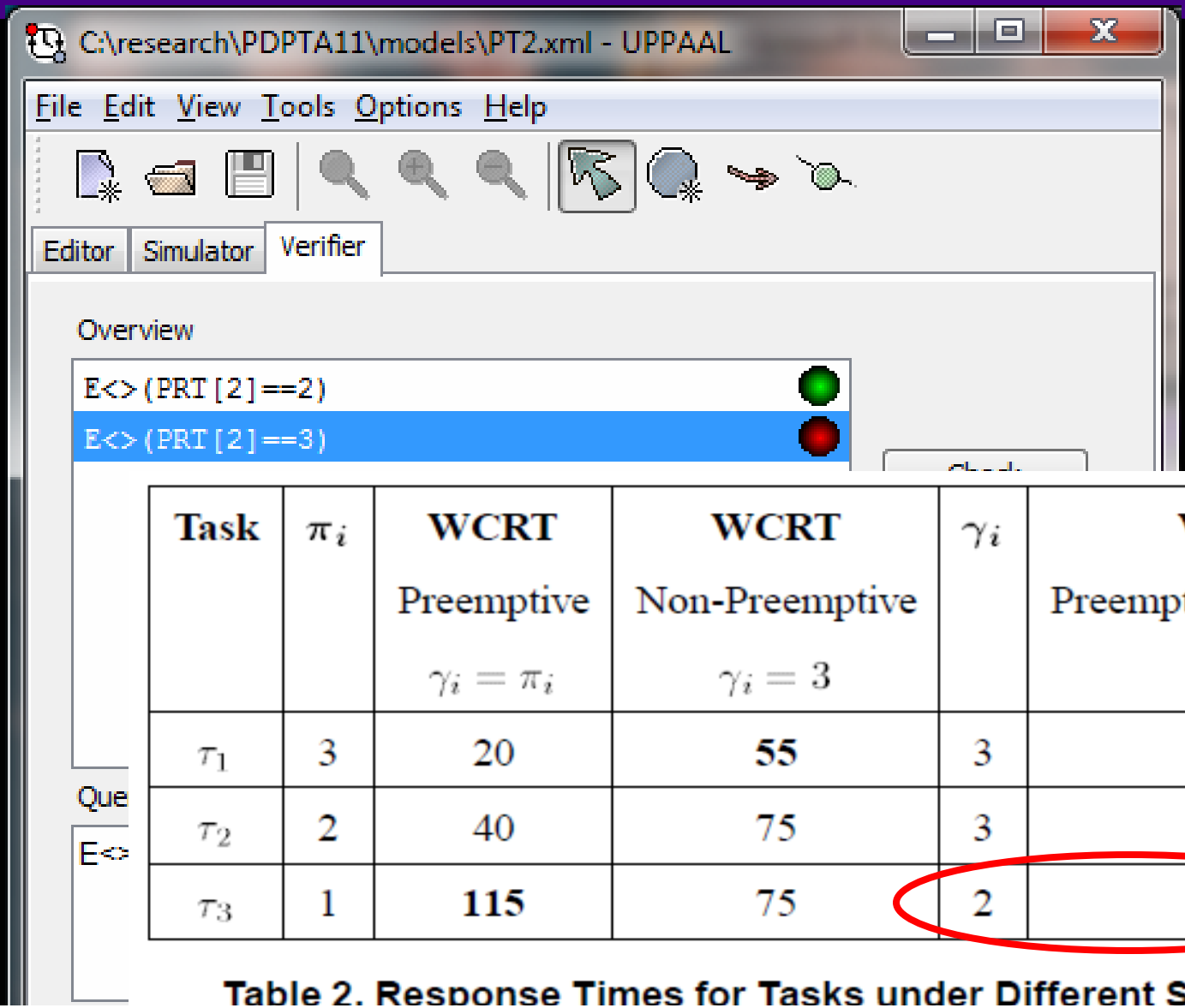


Updated PERIODIC_TASKS



*** oops, typo in paper**

Verification



Task	π_i	WCRT Preemptive $\gamma_i = \pi_i$	WCRT Non-Preemptive $\gamma_i = 3$	γ_i	WCRT Preemption-Threshold
τ_1	3	20	**55**	3	40
τ_2	2	40	75	3	75
τ_3	1	**115**	75	2	95

Table 2. Response Times for Tasks under Different Schedulers.

Summary

- The 2-Clocks Scheduler presented in this paper can be used to correctly test the feasibility of periodic task sets with preemption thresholds.
- The Updated Scheduler can be used to determine if a feasible set of preemption thresholds exist.
- All models and source code are available on-line at: www.cis.ksu.edu/~neilsen/pdpta11/