UPPAAL SMC Tutorial: Schedulability of Herschel/Planck

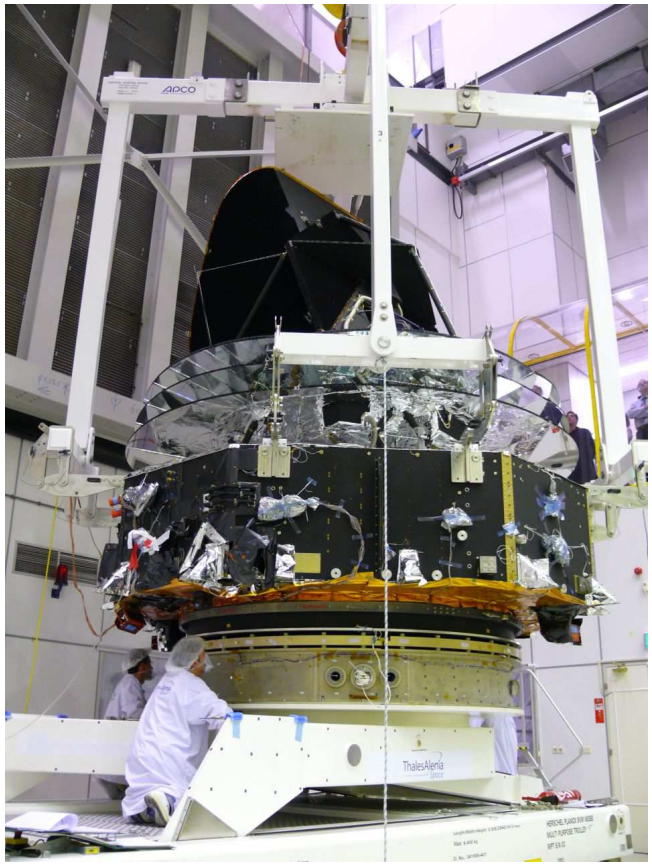
Alexandre David, Kim G. Larsen, Axel Legay,
Marius Mikučionis

Department of Computer Science
Aalborg University

November 19, 2012

Outline

Herschel-Planck Scientific Mission at ESA



- Attitude and Orbit Control System software.
- Terma A/S: Steen Ulrik Palm, Jan Storbank Pedersen, Poul Hougaard.

Satellite Architecture

ASW	Application software performs attitude and orbit control, handles tele-commands, fault detection isolation and recovery.
BSW	Basic software is responsible for low level communication and scheduling periodic events.
RTEMS	Real-time operating system, fixed priority preemptive scheduler.
Hardware	Single processor, a few communication buses, sensors and actuators.

Problem Statement

- Single CPU, fixed priority preemptive scheduler.
- Mixture of 32 tasks: periodic, sporadic with dependencies.
- Mixed resource sharing (make priorities dynamic):
  - BSW tasks use priority *inheritance* protocol.
  - ASW tasks use priority *ceiling* protocol.

At Terma A/S:

- 1 out of 4 configurations *could not be proved schedulable* using schedulability analysis by Alan Burns.
- Neither simulation nor execution show any problems.

At Aalborg:

- The techniques are conservative at assuming worst case.
- *Hypothesis: model more details and achieve more accurate analysis using symbolic reachability and simulations.*

Approach: combination of Symbolic and Statistical

Symbolic analysis:

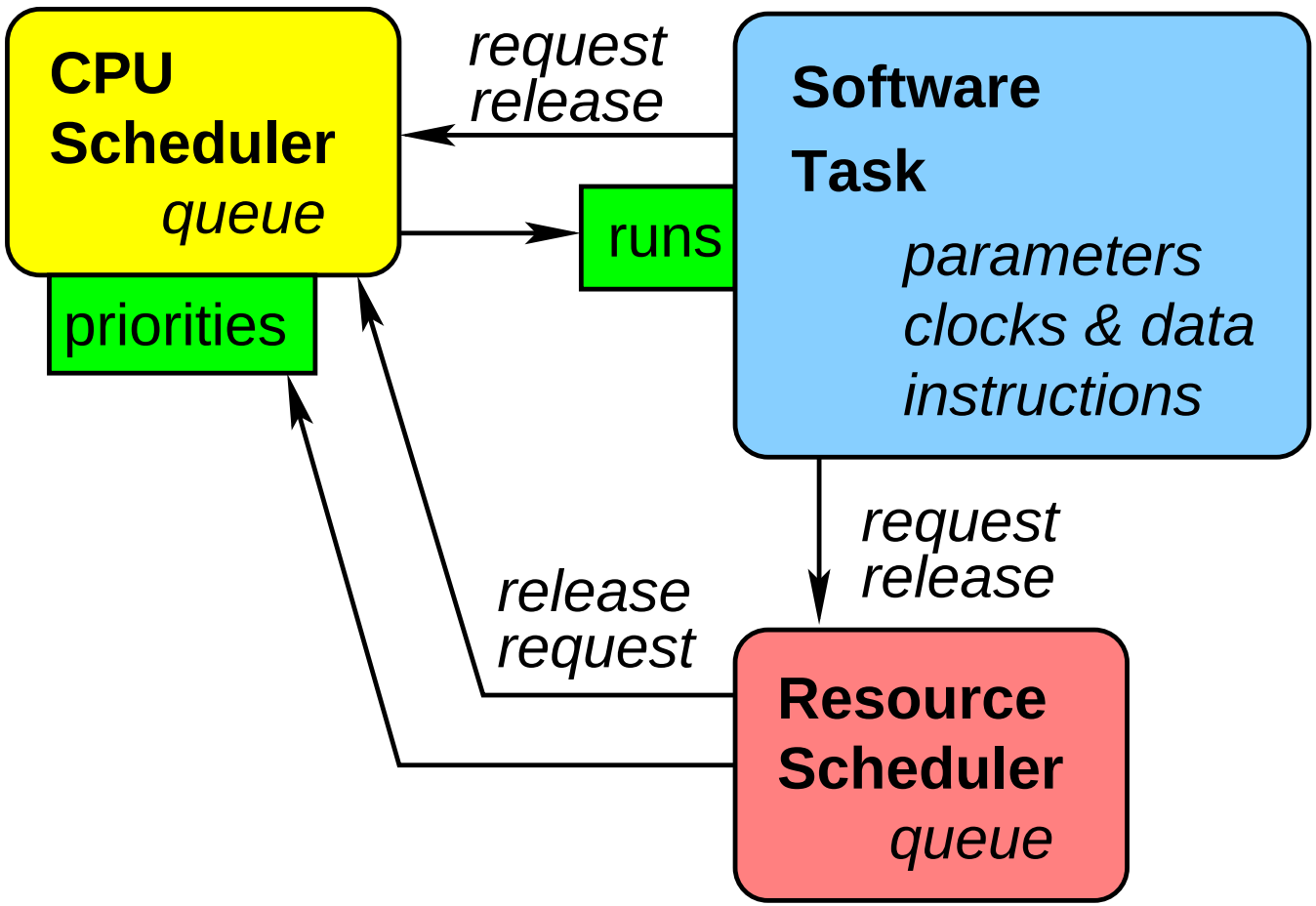
- Preemptive scheduler requires *stop-watches*.
- Exact reachability of stop-watch automata is *undecidable*.
- UPPAAL provides *over-approximation* for stop-watches.
- $\Rightarrow$ symbolic analysis may give spurious errors, but still suitable for *proving safety/schedulability*.

Statistical analysis:

- can show *presence of errors* but not absence.
- $\Rightarrow$ suitable for *disproving schedulability*.

$f = \mathrm{BCET}/\mathrm{WCET}$:	0-71%	72-86%	87-89%	90-100%
Symbolic MC:	maybe	maybe	n/a	**Safe**
Statistical MC:	**Unsafe**	maybe	maybe	maybe

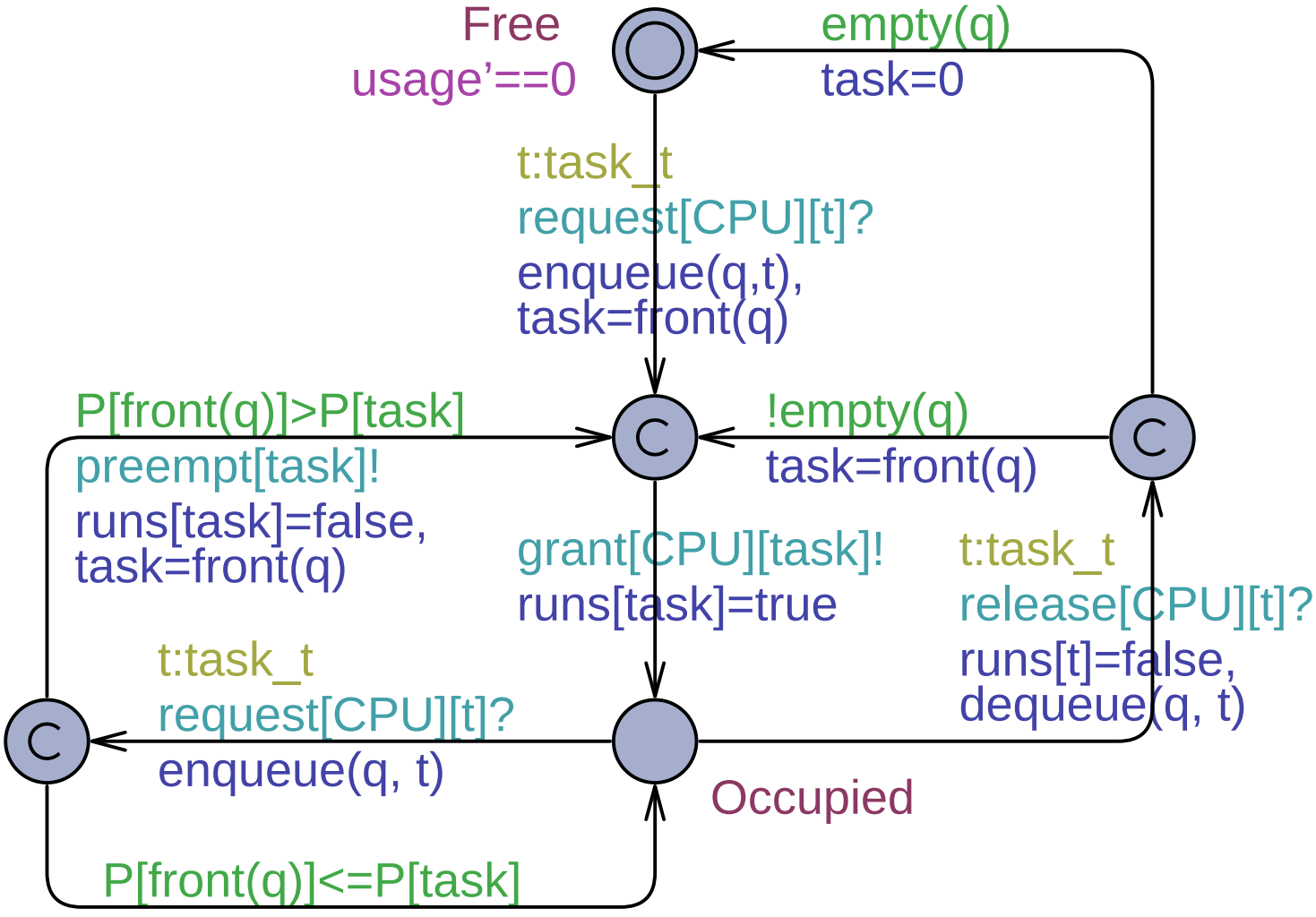
Model Overview



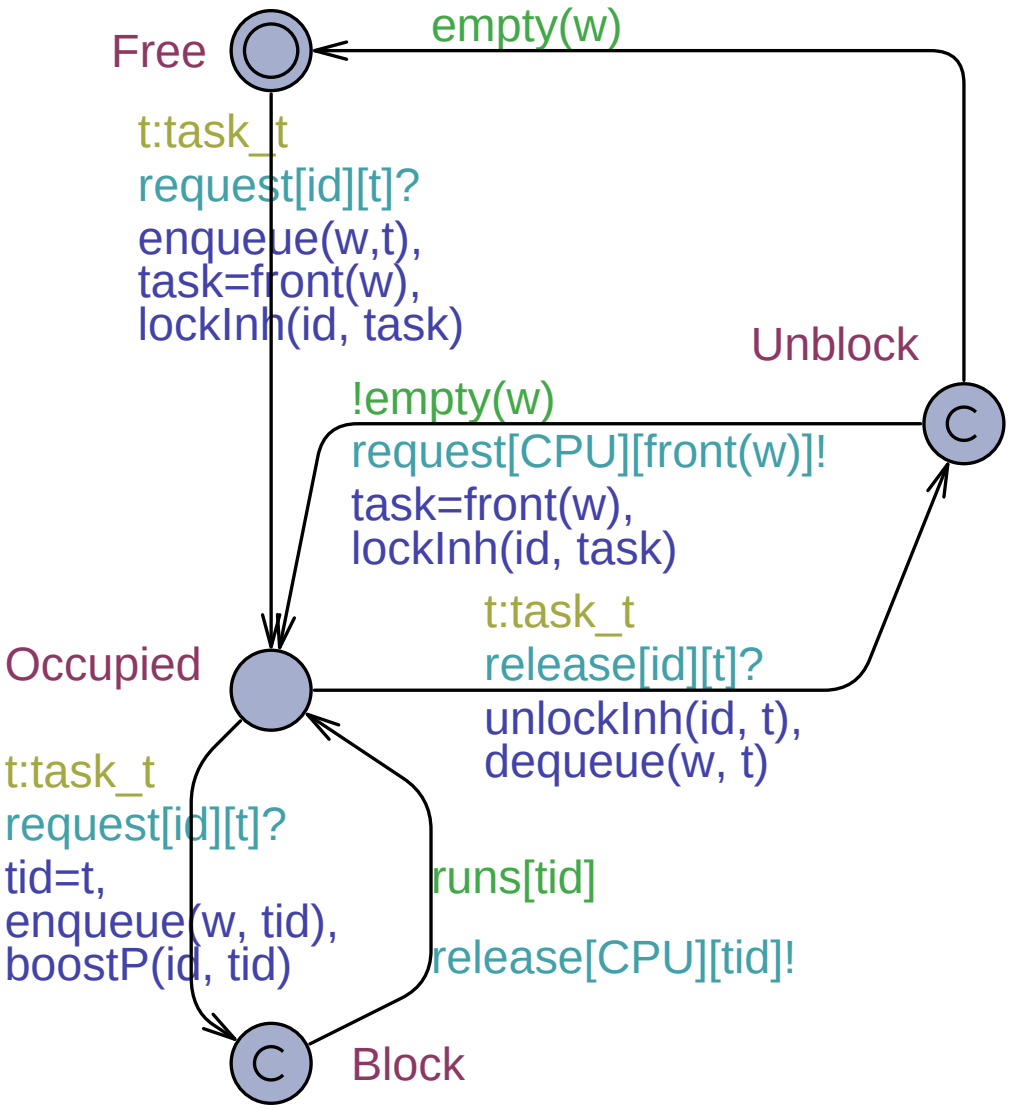
Some Declarations

```
1   const int NRTASK = 3; // number of tasks
2   typedef int [1, NRTASK] task_t; // type for task identifier
3   int P[task_t] = { 3, 2, 1 }; // task priorities as ints
4   task_t task; // currently executed task
5
6   const int NRRES = 1; // number of resources
7   typedef int [1, NRRES] res_t; // type for resource id
8   tasks_t owner[res_t]; // current resource owner
9
10  clock usage; // CPU usage
11
12  typedef tasks_t list_t [NRTASK]; // list of tasks
13  typedef struct {
14      list_t list ; // list of tasks
15      int [0, NRTASK] len; // length of the list
16  } queue_t; // queue structure
```

Fixed Priority Preemptive CPU Scheduler



Resource Scheduler using Priority Inheritance



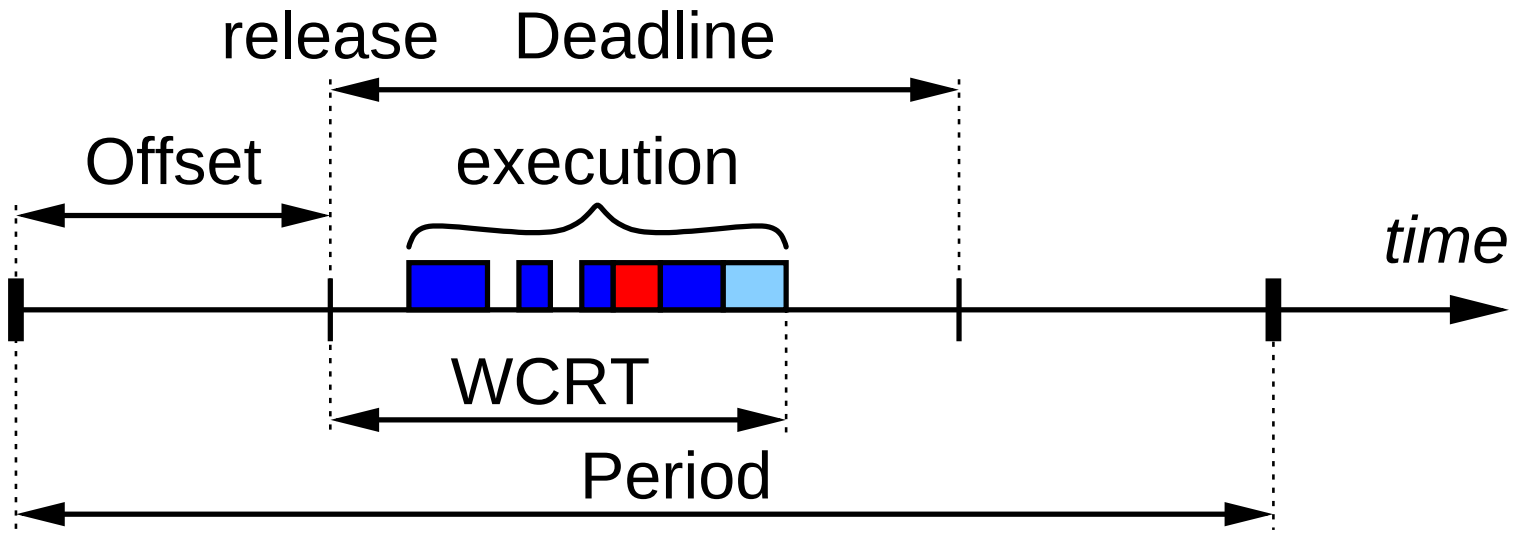
Priority Inheritance Protocol

```
1  /** Lock/acquire: */
2  void lockInh(resid_t res, taskid_t task) {
3      owner[res] = task; // mark as occupied by the task
4  }
5  /** Unlock/release: */
6  void unlockInh(resid_t res, taskid_t task) {
7      owner[res] = 0; // mark the resource as released
8      P[task] = def_prio(task); // return to default priority
9  }
10 /** Boost the priority of resource owner: */
11 void boostPrio(resid_t res, taskid_t task) {
12     if (P[owner[res]] <= def_prio(task)) {
13         P[owner[res]] = def_prio(task)+1;
14         sort(taskqueue);
15     }
16 }
```

Priority Ceiling Protocol

```
1  /** Lock/acquire: */
2  void lockCeil(resid_t res, taskid_t task) {
3      owner[res] = task; // mark resource occupied by the task
4      P[task] = ceiling[res]; // assume priority of resource
5  }
6  /** Unlock/release: */
7  void unlockCeil(resid_t res, taskid_t task){
8      owner[res] = 0; // mark the resource as released
9      P[task] = def_prio(task); // return to default priority
10 }
```

Software Task Parameters



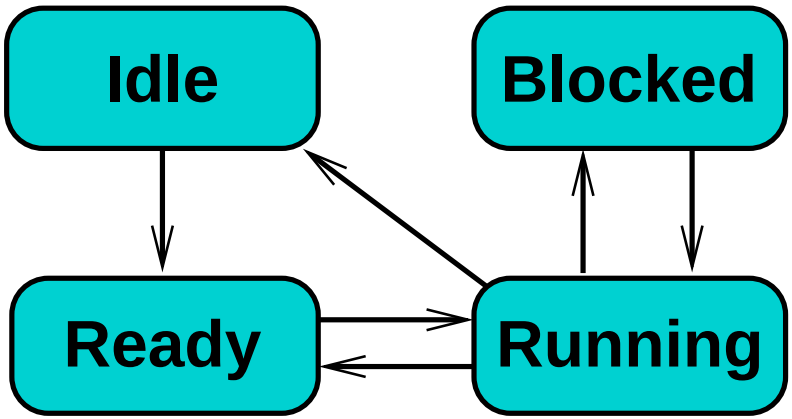

```
1  //          parameter          Task1 Task2 Task3
2  const int Period[task_t]   = {  100,  100, 100 };
3  const int Offset[task_t]   = {   20,    0,  10 };
4  const int WCET[task_t]     = {   15,   25,  40 };
5  const int BCET[task_t]     = {   12,   20,  32 };
6  const int Deadline[task_t] = {   20,   40,  70 };
```

Template for Task using 1 Resource

Parameters:

```
1  int  Period[task_t]    = {  100,  100, 100 };
2  int  Offset[task_t]    = {   20,    0,  10 };
3  int  WCET[task_t]      = {   15,   25,  40 };
4  int  BCET[task_t]      = {   12,   20,  32 };
5  int  Deadline[task_t]  = {   20,   40,  70 };
6  res_t  R[task_t]       = {    1,    1,   1 };
7  bool runs[task_t] = { 0, 0, 0 };
8  bool error = false; // global variable
```

OS task states:

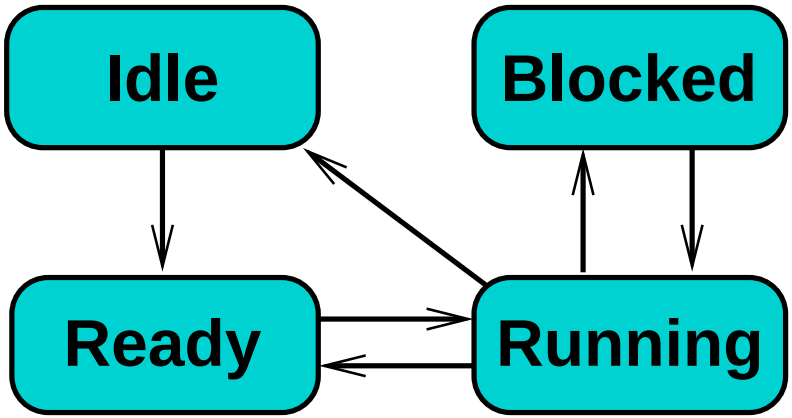


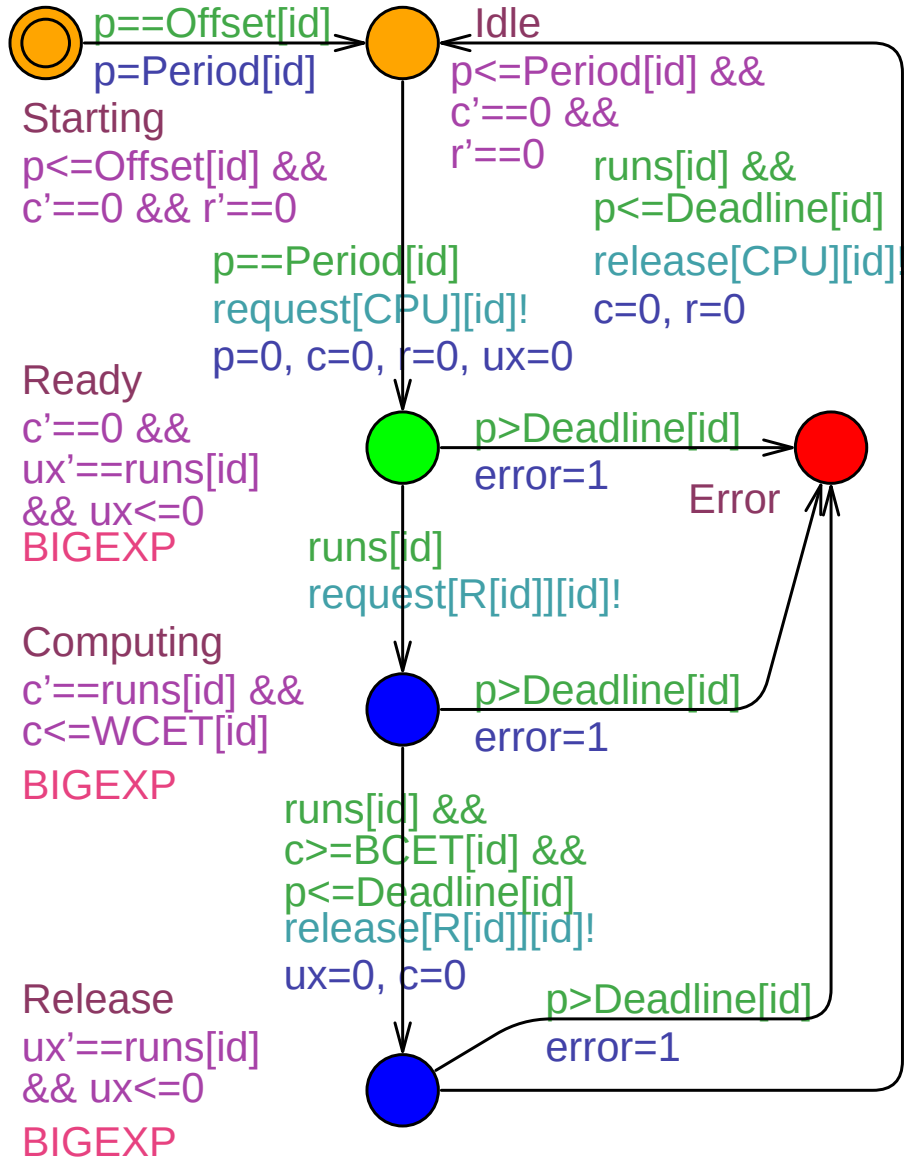
Template for Task using 1 Resource

Parameters:

```
1  int  Period[task_t]    = {  100,  100, 100 };
2  int  Offset[task_t]    = {   20,    0,  10 };
3  int  WCET[task_t]      = {   15,   25,  40 };
4  int  BCET[task_t]      = {   12,   20,  32 };
5  int  Deadline[task_t]  = {   20,   40,  70 };
6  res_t  R[task_t]       = {    1,    1,   1 };
7  bool runs[task_t] = { 0, 0, 0 };
8  bool error = false; // global variable
```

OS task states:





Satellite Software Task Operations

Primary Functions

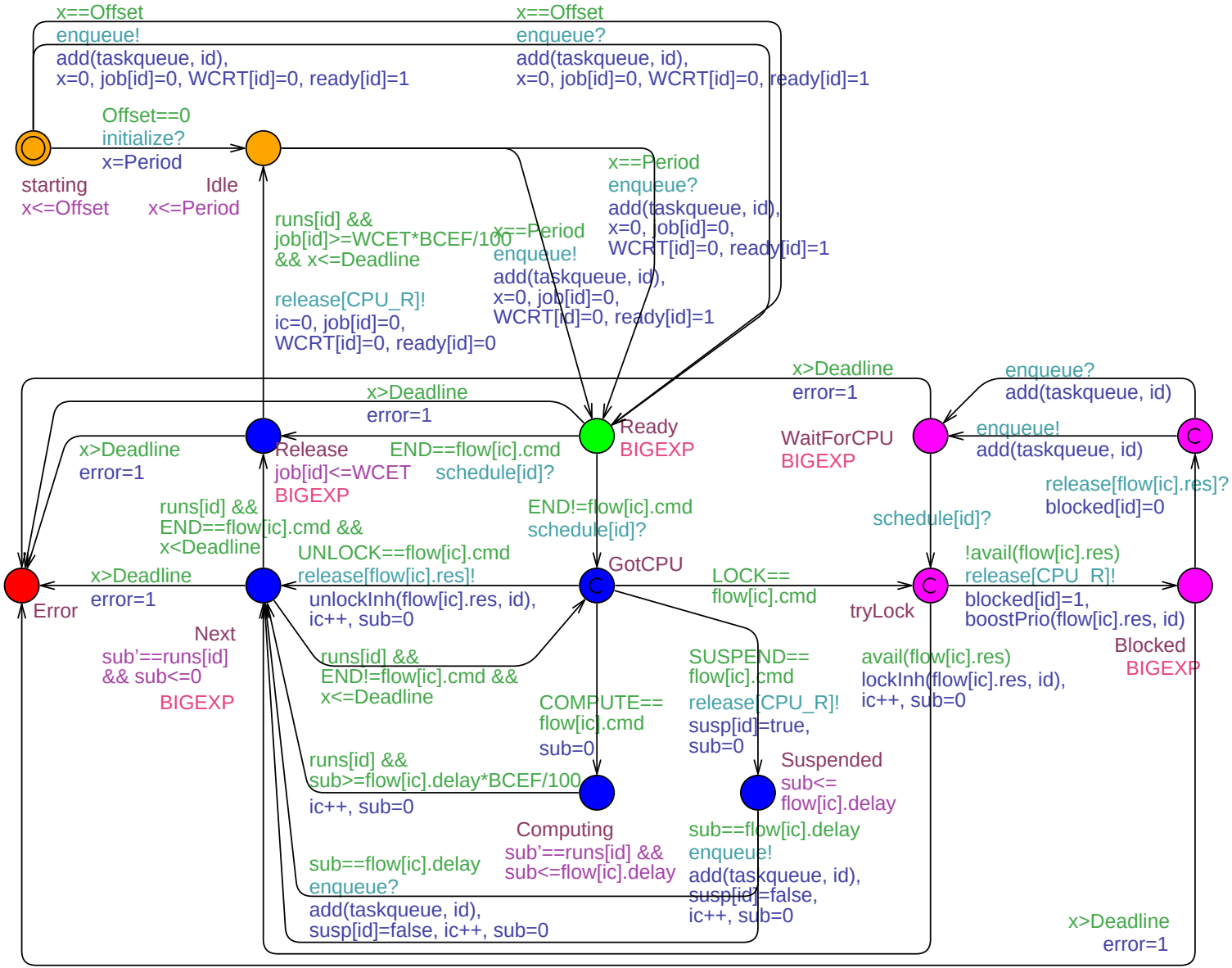
- Data processing	**20577**/2521
	Icb_R(LNS: 2, LCS: **1200**, LC: **1600**, MaxLC: 800)
- Guidance	**3440**/0
- Attitude determination	**3751**/1777
	Sgm_R(LNS: 5, LCS: **121**, LC: **1218**, MaxLC: 236)
- PerformExtraChecks	**42**/0
- SCM controller	**3479**/2096
	PmReq_R(LNS: 4, LCS: **1650**, LC: **3300**, MaxLC: 3300)
- Command RWL	**2752**/85

```
1  typedef int [0,4] funtype_t;   // function type
2  const funtype_t END = 0, COMPUTE = 1, // possible functions
3      LOCK = 2, UNLOCK = 3, SUSPEND = 4;
4  typedef struct {
5        funtype_t cmd; // call function command
6        resid_t res;    // resource argument
7        time_t delay;   // time argument
8  } fun_t;
9  typedef fun_t ASWFlow_t[28]; // list of functions
```

Refined Task Operations

```
1  const ASWFlow_t PF_f = { // Primary Functions:
2      { LOCK,    Icb_R, 0 },              // 0) ------ Data processing
3      { COMPUTE, CPU_R, 1600-1200 }, // 1) computing with Icb_R
4      { SUSPEND, CPU_R, 1200 },    // 2)  suspended with Icb_R
5      { UNLOCK, Icb_R, 0 },              // 3)
6      { COMPUTE, CPU_R, 20577-(1600-1200) }, // 4) computing w/o Icb_R
7      { COMPUTE, CPU_R, 3440 },    // 5) ------ Guidance
8      { LOCK,    Sgm_R, 0 },             // 6) ------ Attitude determination
9      { COMPUTE, CPU_R, 1218-121 }, // 7) computing with Sgm_R
10     { SUSPEND, CPU_R, 121 },     // 8)  suspended with Sgm_R
11     { UNLOCK, Sgm_R, 0 },              // 9)
12     { COMPUTE, CPU_R, 3751-(1218-121) }, //10) computing w/o Sgm_R
13     { COMPUTE, CPU_R, 42 },      //11) ------ Perform extra checks
14     { LOCK,    PmReq_R,0 },            //12) ------ SCM controller
15     { COMPUTE, CPU_R, 3300-1650 }, //13) computing with PmReq_R
16     { SUSPEND, CPU_R, 1650 },    //14)  suspended with PmReq_R
17     { UNLOCK, PmReq_R, 0 },      //15)
18     { COMPUTE, CPU_R, 3479-(3300-1650) },//16) comp. w/o PmReq_R
19     { COMPUTE, CPU_R, 2752 },    //17) ------ Command RWL
20     { END,     CPU_R, 0 }                //18) finished
21 };
```

Template for Satellite Software Task



forall(i: taskid_t)
job[i]'==runs[i]
&&
forall(i: taskid_t)
WCRT[i]'==ready[i]

Symbolic Verification

- Schedulability: **A[] not error**
- Parameterized model: $BCET = f \cdot WCET, f \in [0, 1]$

limit cycle	$f = 100\%$ states	mem	time	$f = 95\%$ states	mem	time
1	0.001	51.2	1.47	0.5	83.0	15:03
2	0.003	53.7	2.45	0.8	96.8	27:00
4	0.005	54.5	4.62	1.5	97.2	48:02
8	0.010	54.7	8.48	2.8	97.8	1:28:45
16	0.020	55.3	16.11	5.4	112.0	2:45:52
∞	0.196	58.8	2:39.64	52.7	553.9	27:05:07

limit cycle	$f = 90\%$ states	mem	time	$f = 86\%$ states	mem	time
1	1.5	124.1	1:22:43	3.3	186.9	6:39:47
2	2.4	139.7	2:09:15	5.3	198.7	9:14:59
4	4.4	138.3	3:48:40	9.2	274.6	14:12:57
8	9.1	156.5	8:38:42	18.2	364.6	28:35:32
16	17.8	176.0	16:42:05	35.4	520.4	44:06:57
∞	181.9	1682.2	147:23:25	**pos.unsafe**		99:07:56

Worst Case Response Times

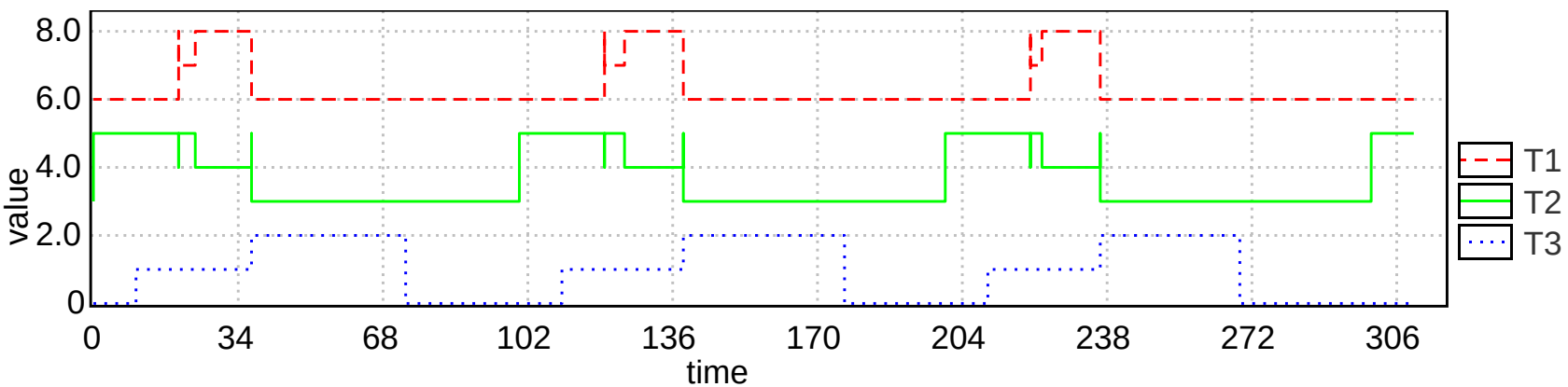
Estimate WCRT: **sup: WCRT[1],WCRT[2],...,WCRT[33]**

ID	Task	Specification			WCRT			
		Period	WCET	Deadline	Terma	$f = 100\%$	$f = 95\%$	$f = 90\%$
1	RTEMS_RTC	10.000	0.013	1.000	0.050	0.013	0.013	0.013
2	AswSync_SyncPulseIsr	250.000	0.070	1.000	0.120	0.083	0.083	0.083
3	Hk_SamplerIsr	125.000	0.070	1.000	0.120	0.070	0.070	0.070
4	SwCyc_CycStartIsr	250.000	0.200	1.000	0.320	0.103	0.103	0.103
5	SwCyc_CycEndIsr	250.000	0.100	1.000	0.220	0.113	0.113	0.113
6	Rt1553_Isr	15.625	0.070	1.000	0.290	0.173	0.173	0.173
7	Bc1553_Isr	20.000	0.070	1.000	0.360	0.243	0.243	0.243
8	Spw_Isr	39.000	0.070	2.000	0.430	0.313	0.313	0.313
9	Obdh_Isr	250.000	0.070	2.000	0.500	0.383	0.383	0.383
10	RtSdb_P_1	15.625	0.150	15.625	4.330	0.533	0.533	0.533
11	RtSdb_P_2	125.000	0.400	15.625	4.870	0.933	0.933	0.933
12	RtSdb_P_3	250.000	0.170	15.625	5.110	1.103	1.103	1.103
13	(no task, this ID is reserved for priority ceiling)							
14	**FdirEvents**	250.000	5.000	230.220	7.180	5.553	5.553	5.553
15	**NominalEvents_1**	250.000	0.720	230.220	7.900	6.273	6.273	6.273
16	**MainCycle**	250.000	0.400	230.220	8.370	6.273	6.273	6.273
17	HkSampler_P_2	125.000	0.500	62.500	11.960	5.380	7.350	8.153
18	HkSampler_P_1	250.000	6.000	62.500	18.460	11.615	13.653	14.153
19	Acb_P	250.000	6.000	50.000	24.680	6.473	6.473	6.473
20	IoCyc_P	250.000	3.000	50.000	27.820	9.473	9.473	9.473
21	**PrimaryF**	250.000	34.050	**59.600**	**65.47**	**54.115**	**56.382**	**58.586**
22	**RCSControlF**	250.000	4.070	239.600	76.040	53.994	56.943	58.095
23	Obt_P	1000.000	1.100	100.000	74.720	2.503	2.513	2.523
24	Hk_P	250.000	2.750	250.000	6.800	4.953	4.963	4.973
25	StsMon_P	250.000	3.300	125.000	85.050	17.863	27.935	28.086
26	TmGen_P	250.000	4.860	250.000	77.650	9.813	9.823	9.833
27	Sgm_P	250.000	4.020	250.000	18.680	14.796	14.880	14.973

Simulation (small model)

Simulate and inspect a random run (e.g. $f = 80\%$):

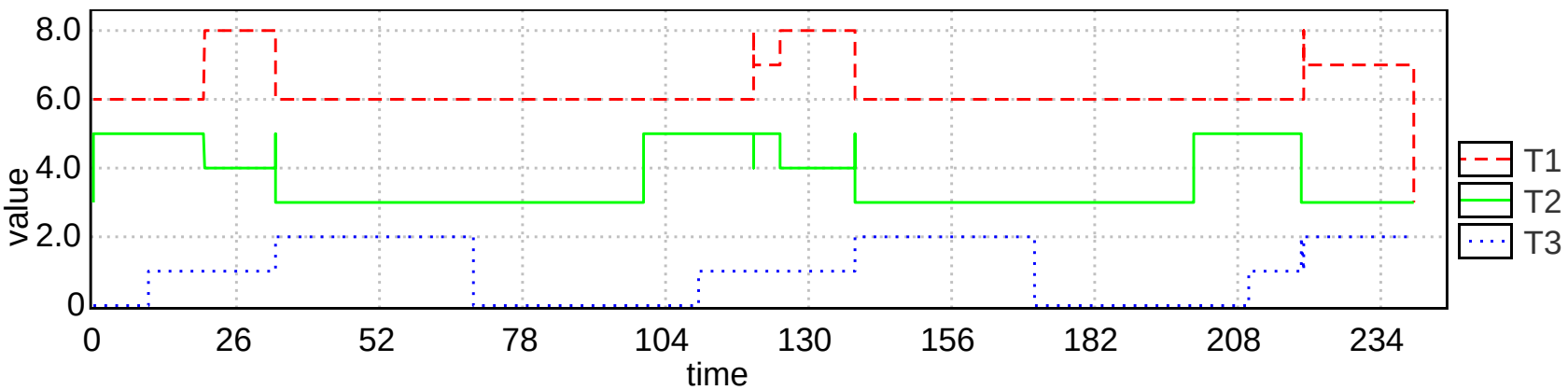
```
simulate 1 [<=300] {
   (T(1).Ready+T(1).Computing+T(1).Release+runs[1]-2*T(1)
   (T(2).Ready+T(2).Computing+T(2).Release+runs[2]-2*T(1)
   (T(3).Ready+T(3).Computing+T(3).Release+runs[3]-2*T(1)
}
```



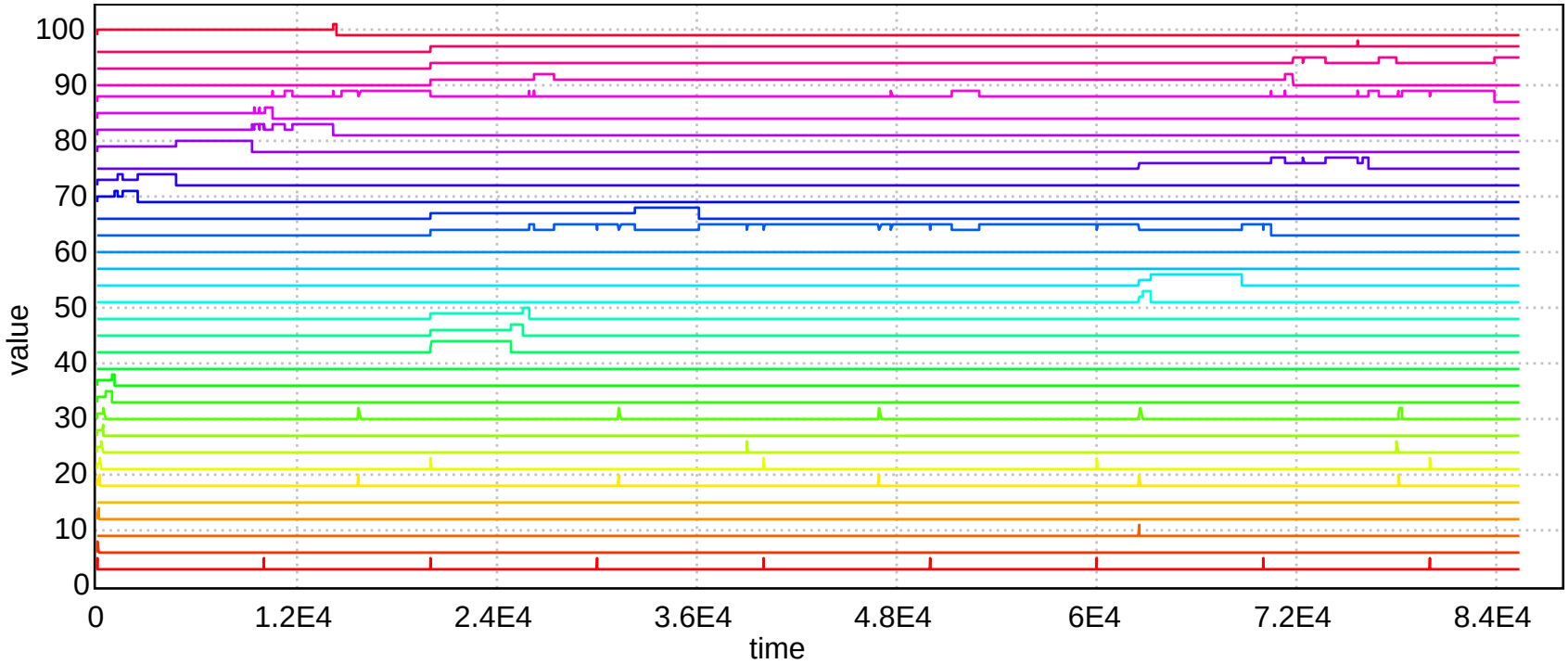
Simulation (small model)

Find and inspect failing run (e.g. $f = 79\%$):

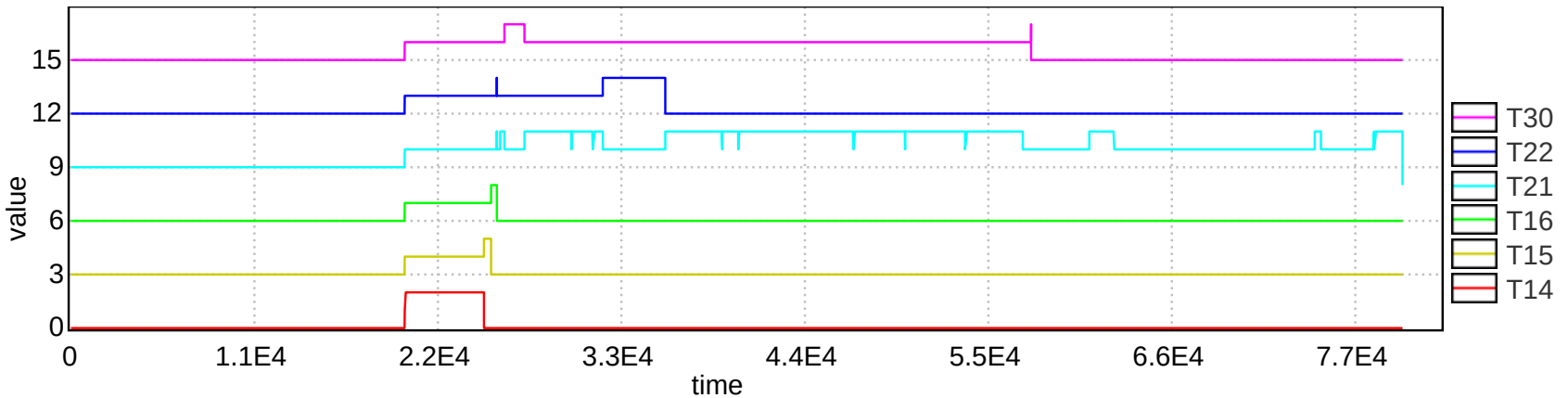
```
simulate 10000 [<=300] {
   (T(1).Ready+T(1).Computing+T(1).Release+runs[1]-2*T(1)
   (T(2).Ready+T(2).Computing+T(2).Release+runs[2]-2*T(1)
   (T(3).Ready+T(3).Computing+T(3).Release+runs[3]-2*T(1)
} : 1 : error
```



Herschel simulation run with $f = 90\%$:



Herschel deadline violation with $f = 50\%$:



SMC of Herschel Model

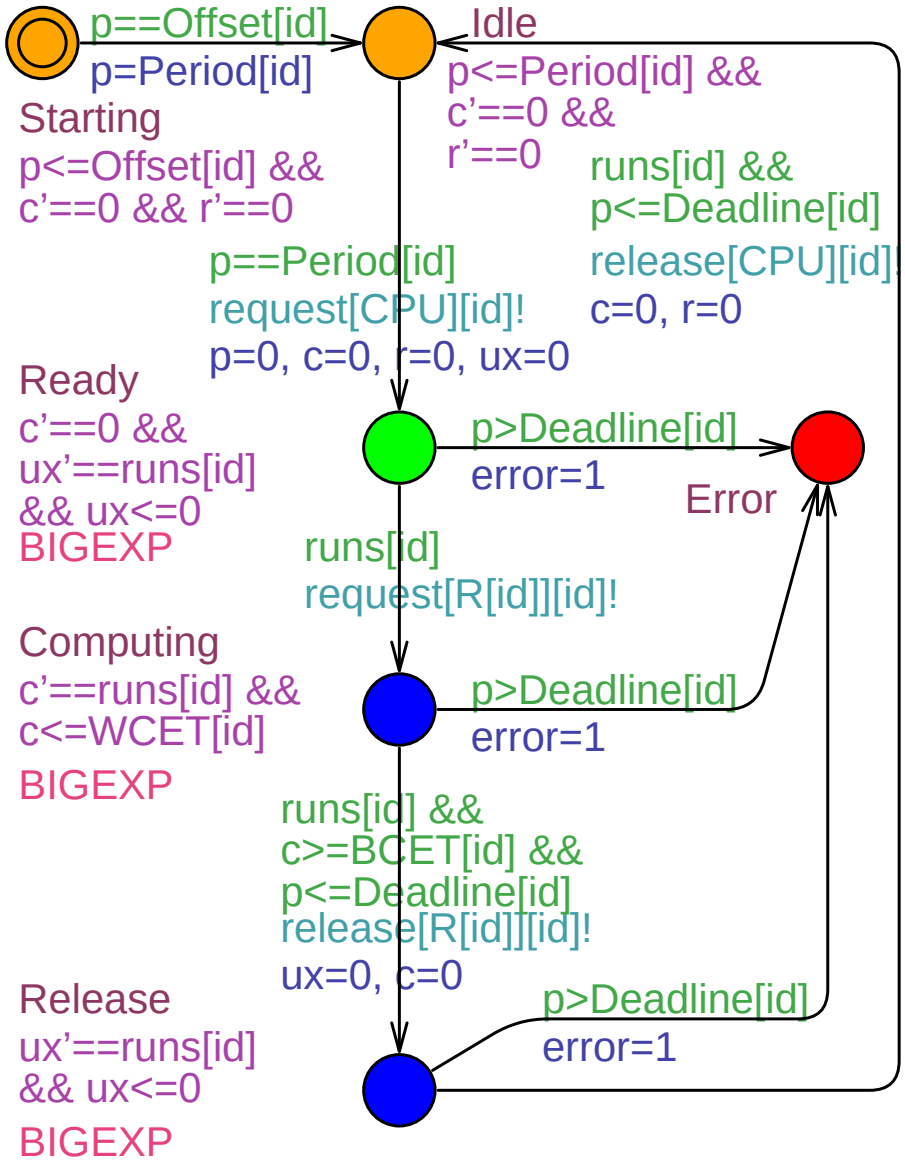
Estimate: **Pr[<=LIMIT*250000](<> error)**

Limit cycles	f %	SMC parameters α	ε	Total traces, #	Error traces #	Probability	Earliest Error cycle	offset	Verification time
1	0	0.0100	0.005	105967	1928	0.018194	0	79600.0	1:58:06
1	50	0.0100	0.005	105967	753	0.007106	0	79600.0	2:00:52
1	60	0.0100	0.005	105967	13	0.000123	0	79778.3	2:01:18
1	62	0.0005	0.002	1036757	34	0.000033	0	79616.4	19:52:22
160	63	0.0100	0.05	1060	177	0.166981	0	81531.6	2:47:03
160	64	0.0100	0.05	1060	118	0.111321	1	79803.0	2:55:13
160	65	0.0500	0.05	738	57	0.077236	3	79648.0	2:06:55
160	66	0.0100	0.05	1060	60	0.056604	2	82504.0	2:62:44
160	67	0.0100	0.05	1060	26	0.024528	1	79789.0	2:64:20
160	68	0.0100	0.05	1060	3	0.002830	67	81000.0	2:67:08
640	69	0.0100	0.05	1060	8	0.007547	114	80000.0	12:23:00
640	70	0.0100	0.05	1060	3	0.002830	6	88070.0	12:30:49
1280	71	0.0100	0.05	1060	2	0.001887	458	80000.0	25:19:35

Just find:

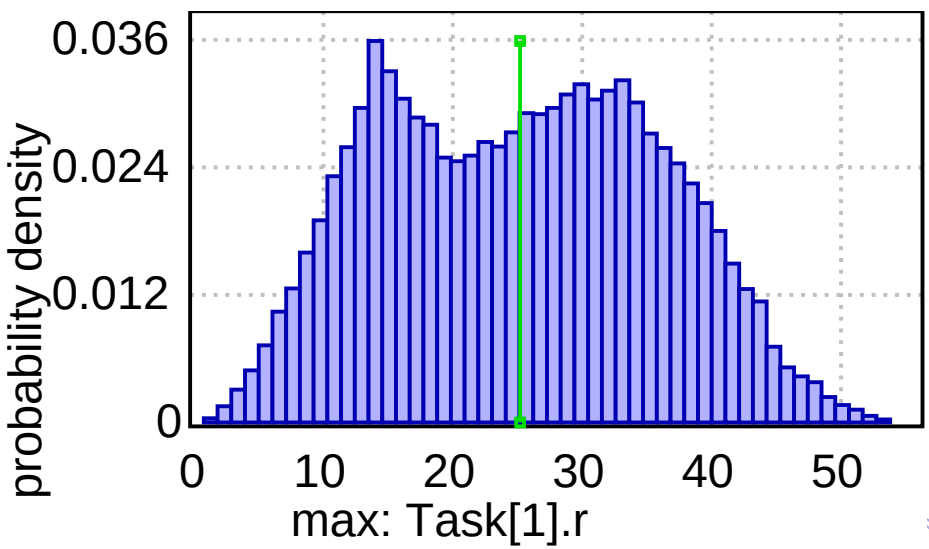
simulate 10000 [<=LIMIT*250000] { error } :1:error

Estimating Response Times using SMC



E[<=200; 50000] (max: r)

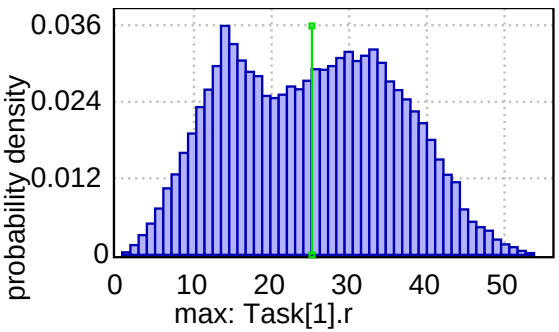
Pr[r<=10000](<> time==200)



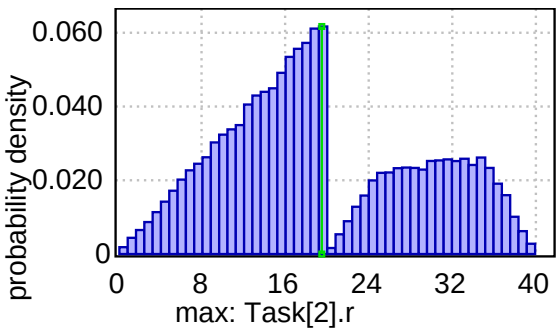
SMC: Response Times (small model, f=0%)

With $f = 0\%$ (BCET=0), T1 violates deadline at 20, thus we relax deadline parameters just to inspect the distribution of WCRT.

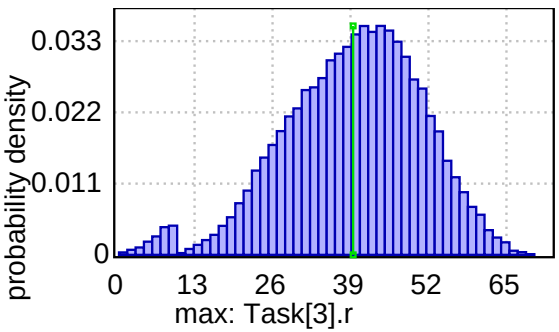
```
E[<=200; 50000] (max: T(1).r)
E[<=200; 50000] (max: T(2).r)
E[<=200; 50000] (max: T(3).r)
```



(a) Task `T1`.



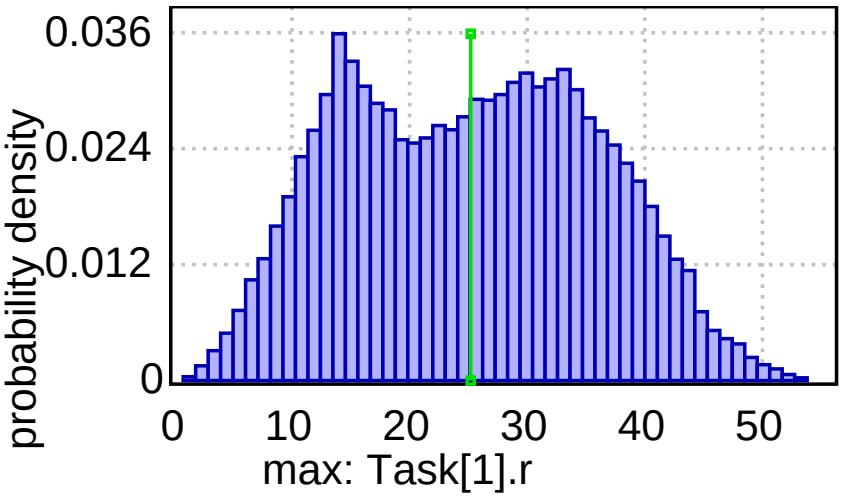
(b) Task `T2`.



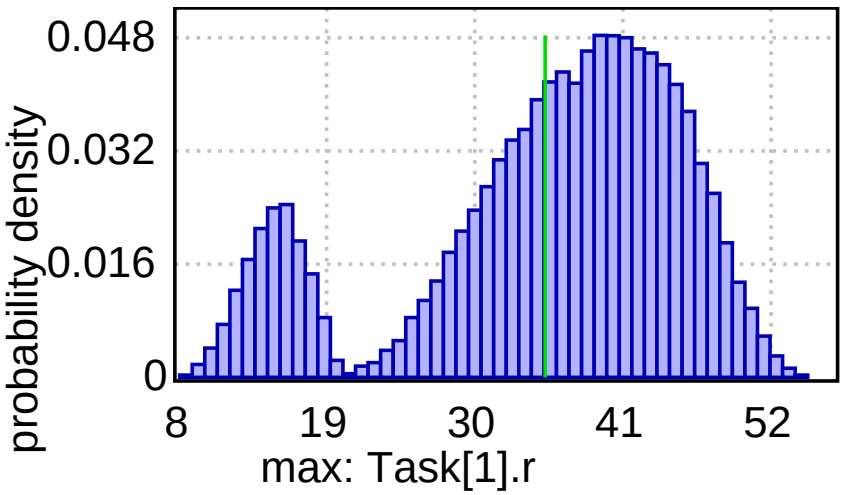
(c) Task `T3`.

SMC: Response Times of T1

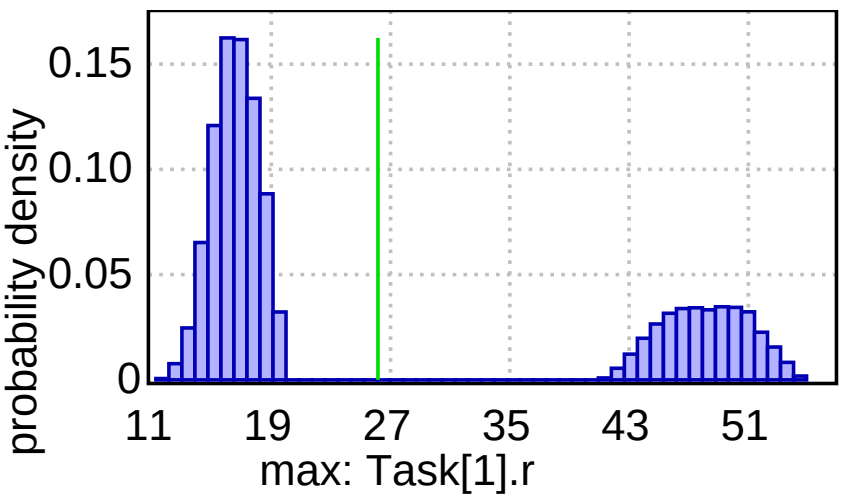
$f = 0\%$ (not safe).



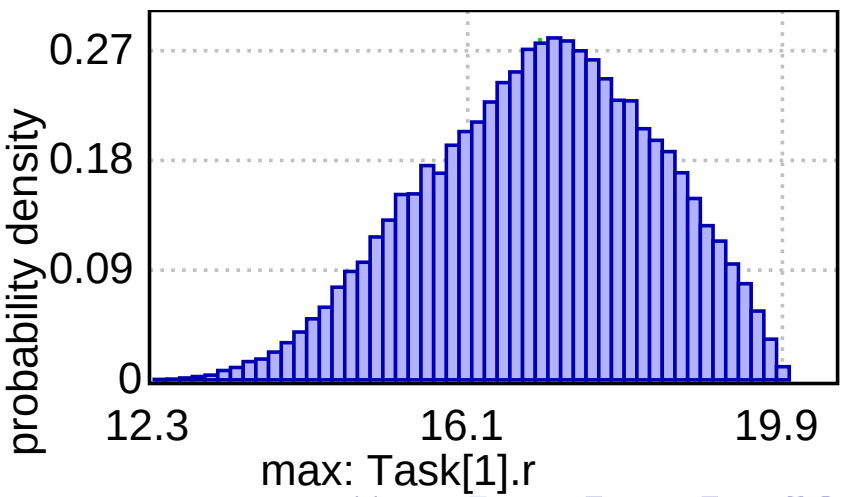
$f = 50\%$ (not safe).



$f = 79\%$ (not safe).

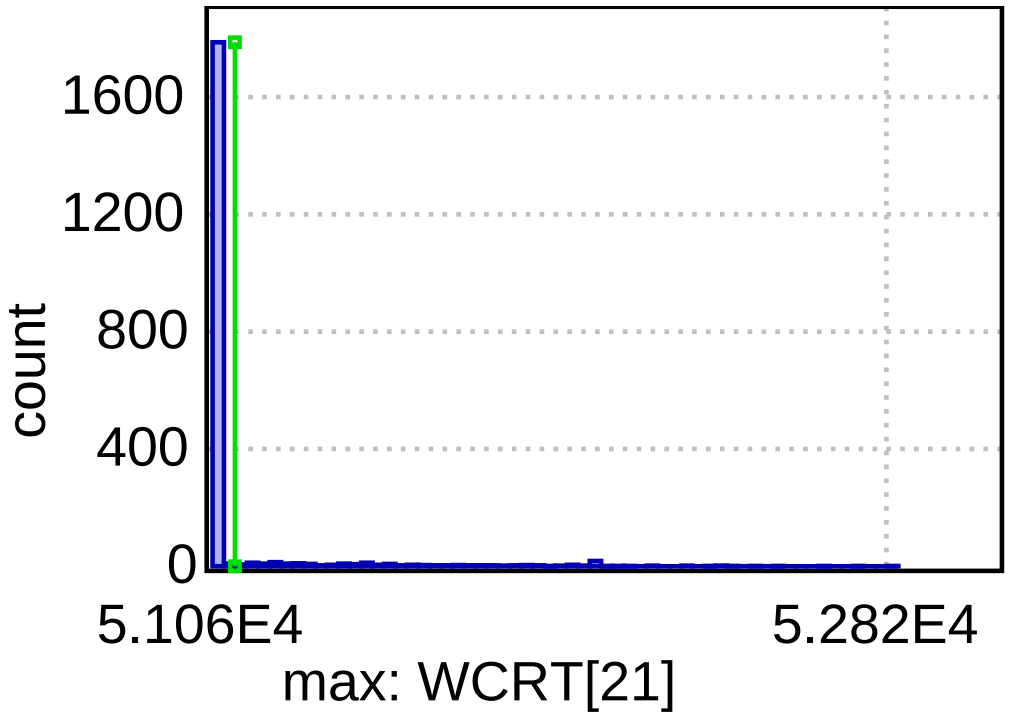


$f = 80\%$ (seem OK).



Estimating WCRT for Herschel

```
E[<=LIMIT*250000; 2000](max:  WCRT[21])
```



Plot for $f = \mathrm{BCET}/\mathrm{WCET} = 90\%$

Summary of Techniques Used

- Modeling:
  - Timed automata with clocks to express time constraints.
  - Stop-watches to track task progress.
  - Functions to implement resource sharing protocols.
  - Data structures to specify sequences of task operations.
- Symbolic model checking:
  - Exhaustive exploration of entire model state space.
  - Verification memory saving via sweep-line & CDS.
  - WCRT estimation using supremum query.
- Statistical model checking:
  - Trace visualization via simulate query.
  - Bounded simulations for disproving schedulability
  - WCRT estimation via probability density over clock values.

UPPAAL SMC is integrated in Uppaal, visit us at uppaal.org

Thank You
for your attention