UML & FM 2010 in Shanghai

GENERATING COMPOSITE BEHAVIOR OF EMBEDDED SOFTWARE COMPONENTS BASED ON UML BEHAVIORAL MODEL AND PROCESS ALGEBRA

2010.11.16

Jinhyun Kim
Korea University

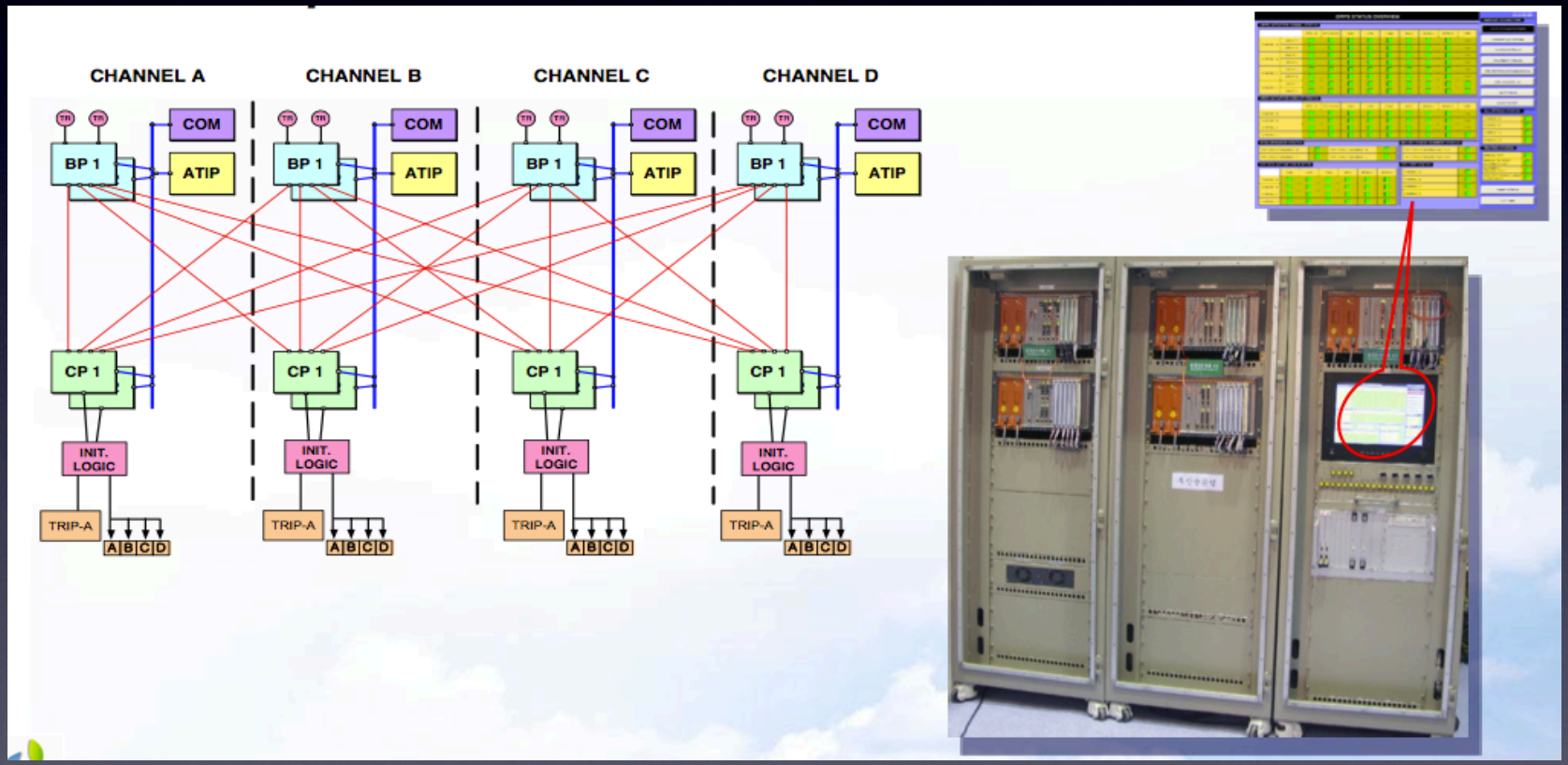
Contents

- Introduction
- Background
  - TRoS and ACSR
- TRoS || ACSR for Embedded Software Components
- Specifying Real-time Embedded Applications in TRoS || ACSR
- Conclusions

Contents

- Introduction
- Background
  - TRo1S and ACSR
- TRoS || ACSR for Embedded Software Components
- Specifying Real-time Embedded Applications in TRoS || ACSR
- Conclusions

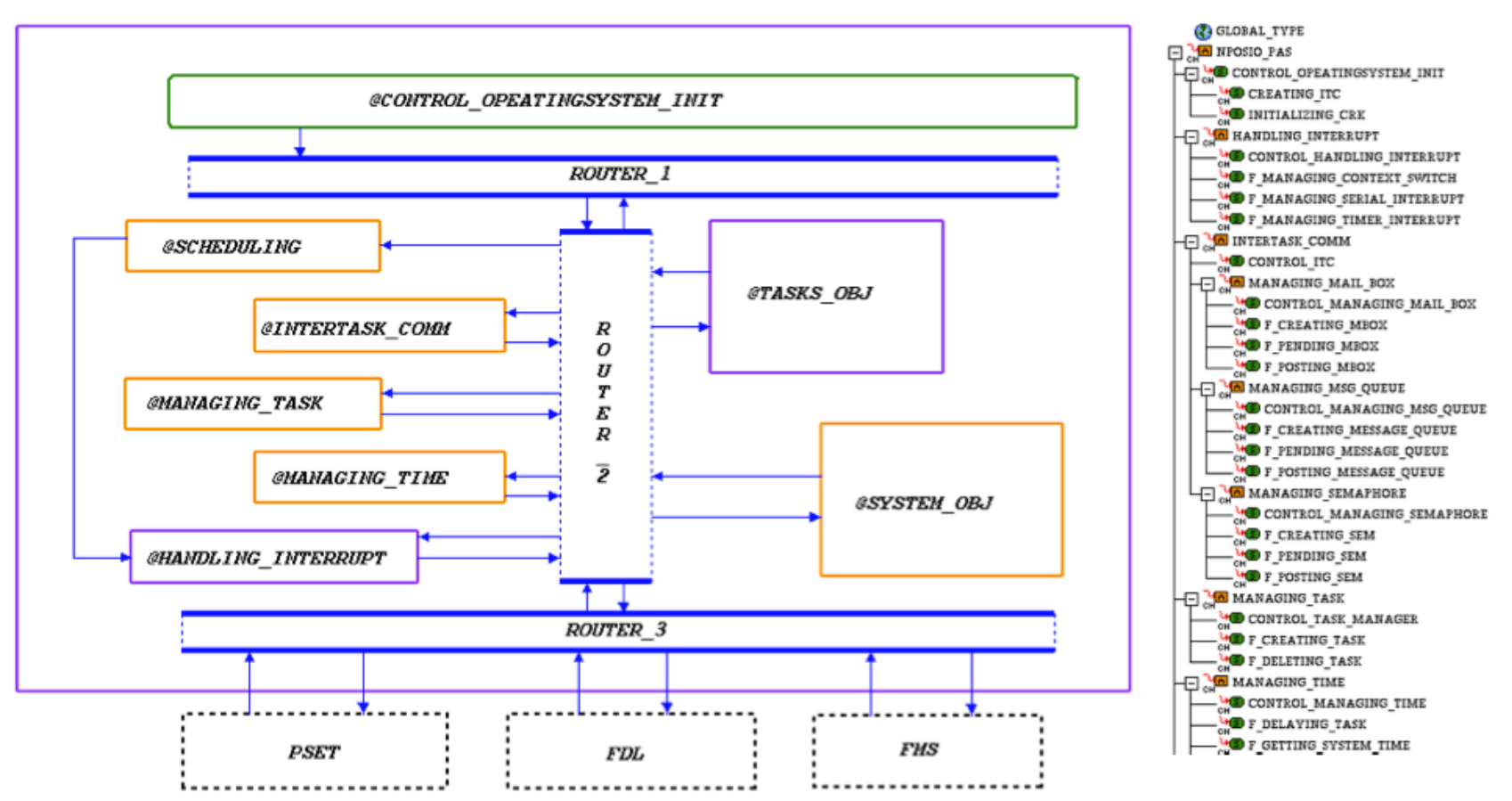
PLC for NPP


CHANNEL A
CHANNEL B
CHANNEL C
CHANNEL D
COM
BP 1
ATIP
CP 1
INIT. LOGIC
TRIP-A
A B C D

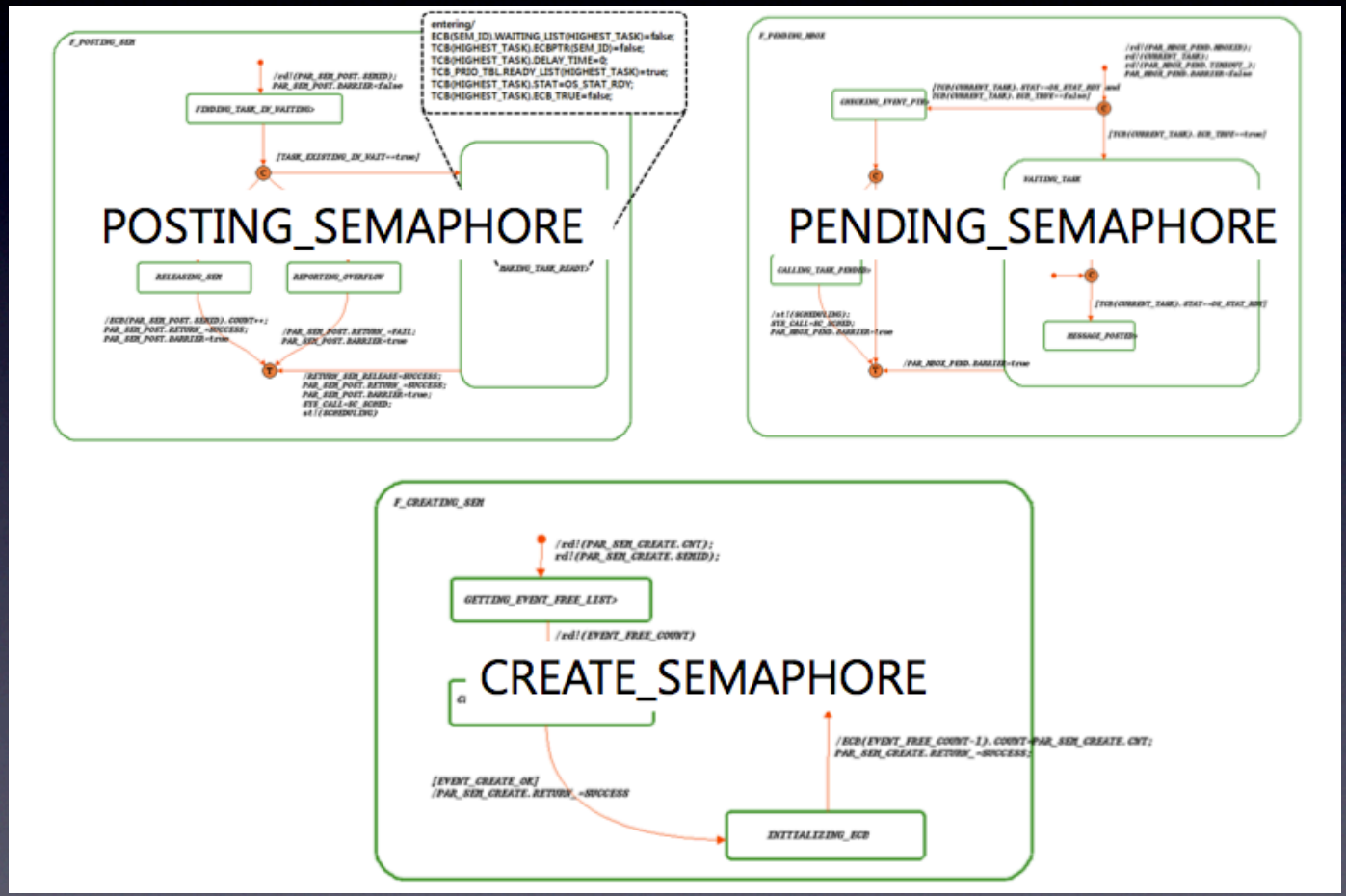
STATEMATE Statecharts

- It is widely used to model not only HW but also embedded SW,
- There are plenty of analyzing tools, such as model checking, simulation, and test

Model of pCOS



Semaphore Model in SC



Semaphore in ACSR

```
SemMng_B_SEM_1
        = {}:SemMng_B_SEM_1
        + (OSSemCreate_B_SEM_1,1).B_SEM_1_Pend ;

B_SEM_1_Pend
        = {}:B_SEM_1_Pend
        + (OSSemPend_SEM_1,1).B_SEM_1_Post
        + (OSSemDelete_B_SEM_1,1).SemMng_B_SEM_1;

B_SEM_1_Post
        = {}:B_SEM_1_Post
        + (OSSemPost_SEM_1,1).('OSSched,1).B_SEM_1_Pend
```

Contents

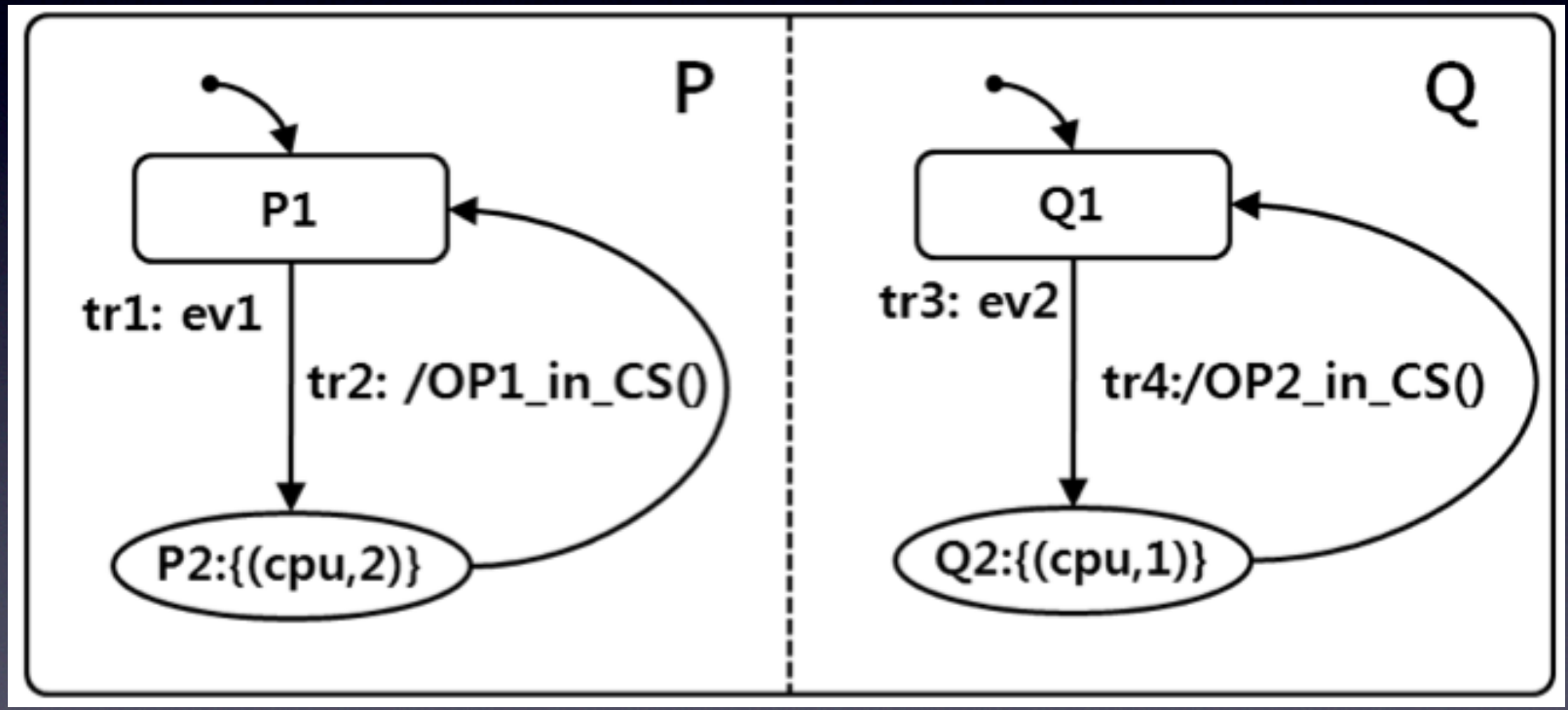
TRoS

- TRoS = Statecharts + Timed Actions
- The timed action specifies timed behaviors constrained by the prioritized use of resources.
- It can be transformed directly from ACSR in easy way by rules we have defined.

Timed Action

- $A ::= \{S\}^n \mid \langle S \rangle^n$
- $S ::= \varepsilon \mid (r, ve), S$

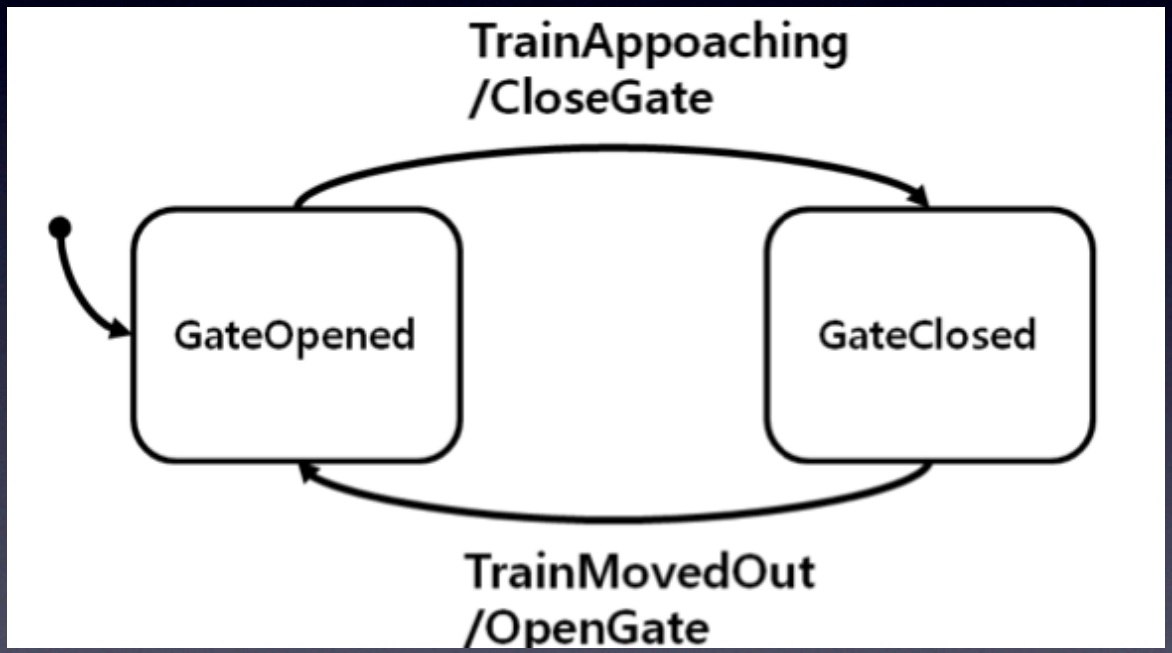
Example


P
P1
tr1: ev1
tr2: /OP1_in_CS()
P2:{(cpu,2)}
Q
Q1
tr3: ev2
tr4:/OP2_in_CS()
Q2:{(cpu,1)}

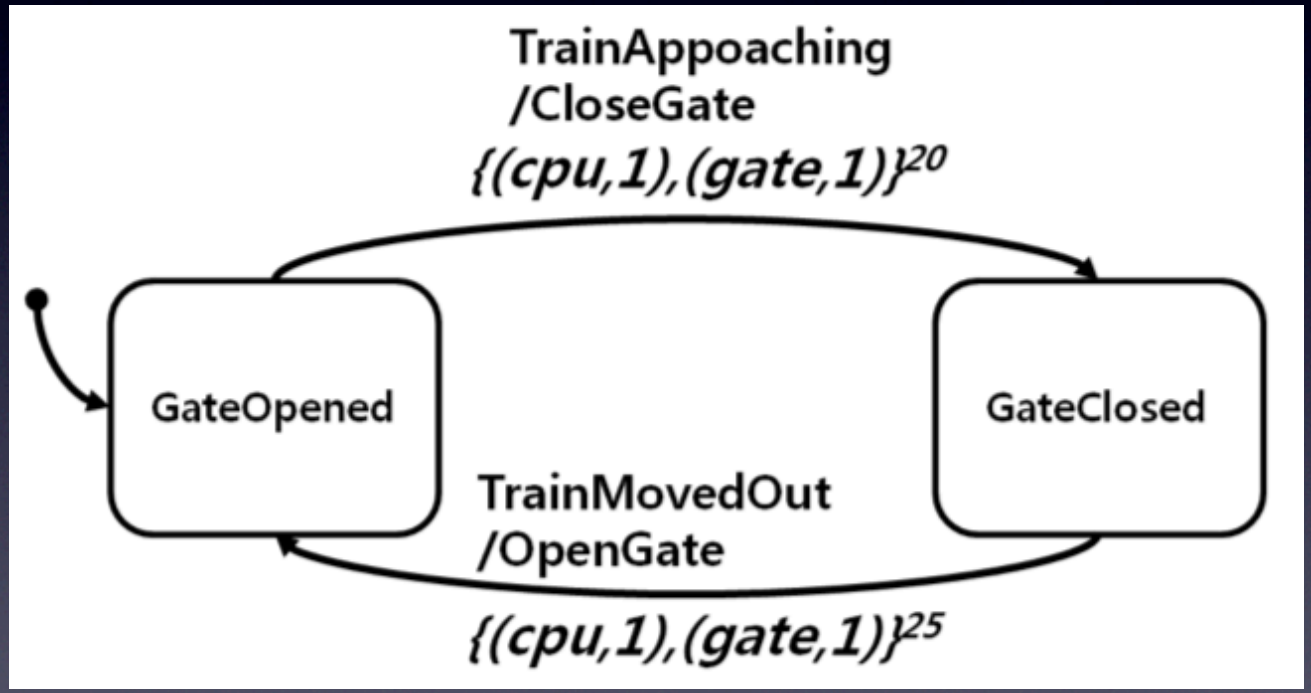
Algorithm 1 TRoSexec

```
1: procedure TRoSexec(M)
2:     Initialize the current configuration by default nodes;
3:     Initialize input set I;
4:     Initialize clock ;
5:     while true do
6:         Read external input events into I;
7:         while I is not empty  do
8:             Execute instantaneous events ;
9:         end while
10:        Execute timed actions ;
11:        Increase clock ;
12:    end while
13: end procedure
```

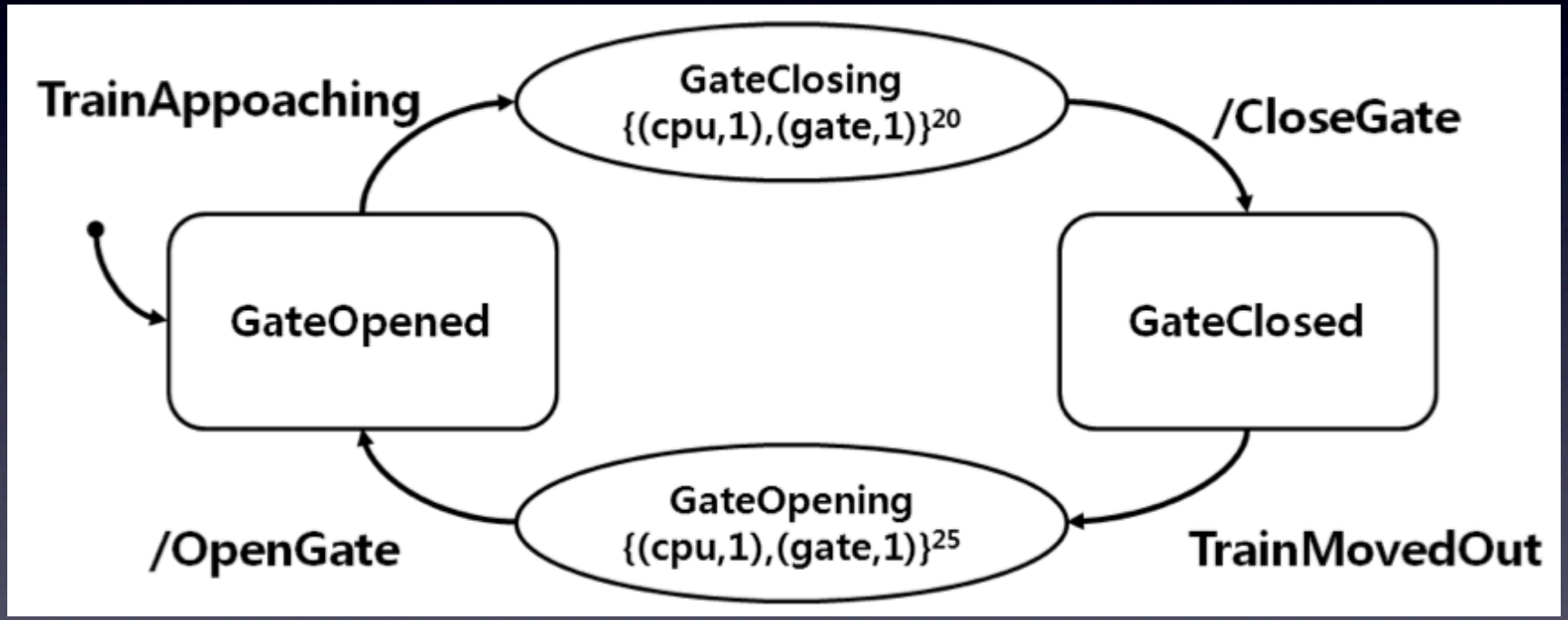
Transformation of SC into TROS



Annotation

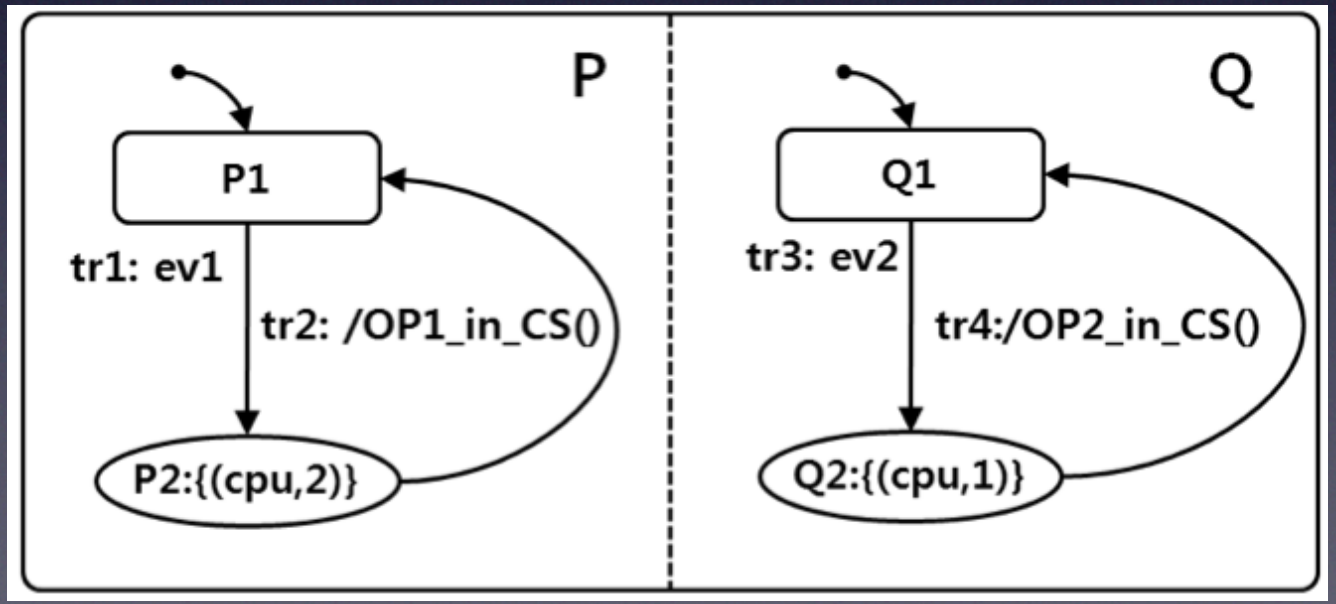

TrainAppoaching
/CloseGate
{(cpu,1),(gate,1)}20
GateOpened
GateClosed
TrainMovedOut
/OpenGate
{(cpu,1),(gate,1)}25

Transformation



Algorithm 1 TRoSexec

```
1: procedure TRoSExec(M)
2:     Initialize the current configuration by default nodes;
3:     Initialize input set I;
4:     Initialize clock ;
5:     while true do
6:         Read external input events into I;
7:         while I is not empty do
8:             Execute instantaneous events ;
9:         end while
10:        Execute timed actions ;
11:        Increase clock ;
12:    end while
13: end procedure
```



ACSR

- A formal specification for real-time systems
- The actions of ACSR consists of two kinds of actions: timed actions and event actions.

Sematnics

ActT: $\dfrac{-}{A : P \xrightarrow{A} P}$ **ActI:** $\dfrac{-}{(a,n).P \xrightarrow{(a,n)} P}$

Choice: $\dfrac{P \xrightarrow{\alpha} P_i, i = 1,2}{P_1 + P_2 \xrightarrow{\alpha} P_i}$ **Par:** $\dfrac{P \xrightarrow{\alpha} P_i, i = 1,2}{P_1 || P_2 \xrightarrow{\alpha} P_i}$

ParT: $\dfrac{P \xrightarrow{A_1} P', Q \xrightarrow{A_2} Q'}{P||Q \xrightarrow{A_1 \cup A_2} P'||Q'}$ $(\rho(A_1) \cap \rho(A_2)) = \emptyset$

ParC: $\dfrac{P \xrightarrow{(l,m)} P', Q \xrightarrow{(inv(l),n)} Q'}{P||Q \xrightarrow{(\tau,m+n)} P'||Q'}$

CloseT: $\dfrac{P \xrightarrow{A_1} P'}{[P]_I \xrightarrow{A_1 \cup A_2} [P']_I}$ $(A_2 = \{(r,0) | r \in I - \rho(A_1)\})$

CloseI: $\dfrac{P \xrightarrow{e} P'}{[P]_I \xrightarrow{e} [P']_I}$ **ResT:** $\dfrac{P \xrightarrow{A} P'}{P\backslash F \xrightarrow{A} P'\backslash F}$

ResI: $\dfrac{P \xrightarrow{e} P'}{P\backslash F \xrightarrow{e} P'\backslash F}$ $(\gamma(e) \notin F)$

HideT: $\dfrac{P \xrightarrow{A} P'}{P\backslash\backslash H \xrightarrow{A} P'\backslash\backslash H}$ $(A' = \{(r,n) \in A | r \notin H\})$

HideI: $\dfrac{P \xrightarrow{e} P'}{P\backslash\backslash H \xrightarrow{e} P'\backslash\backslash H}$

The non-prioritized relation of ACSR

Preemption Relations

there are three cases for $\alpha \prec \beta$:

1) α and β are events with the same label and β has a higher priority;

2) α and β are actions and β uses a subset of resources with the following two conditions: all resources in β have at least the same priority as in β and at least one at a higher priority than in α, and every resources in α that is not in β must have a zero priority;

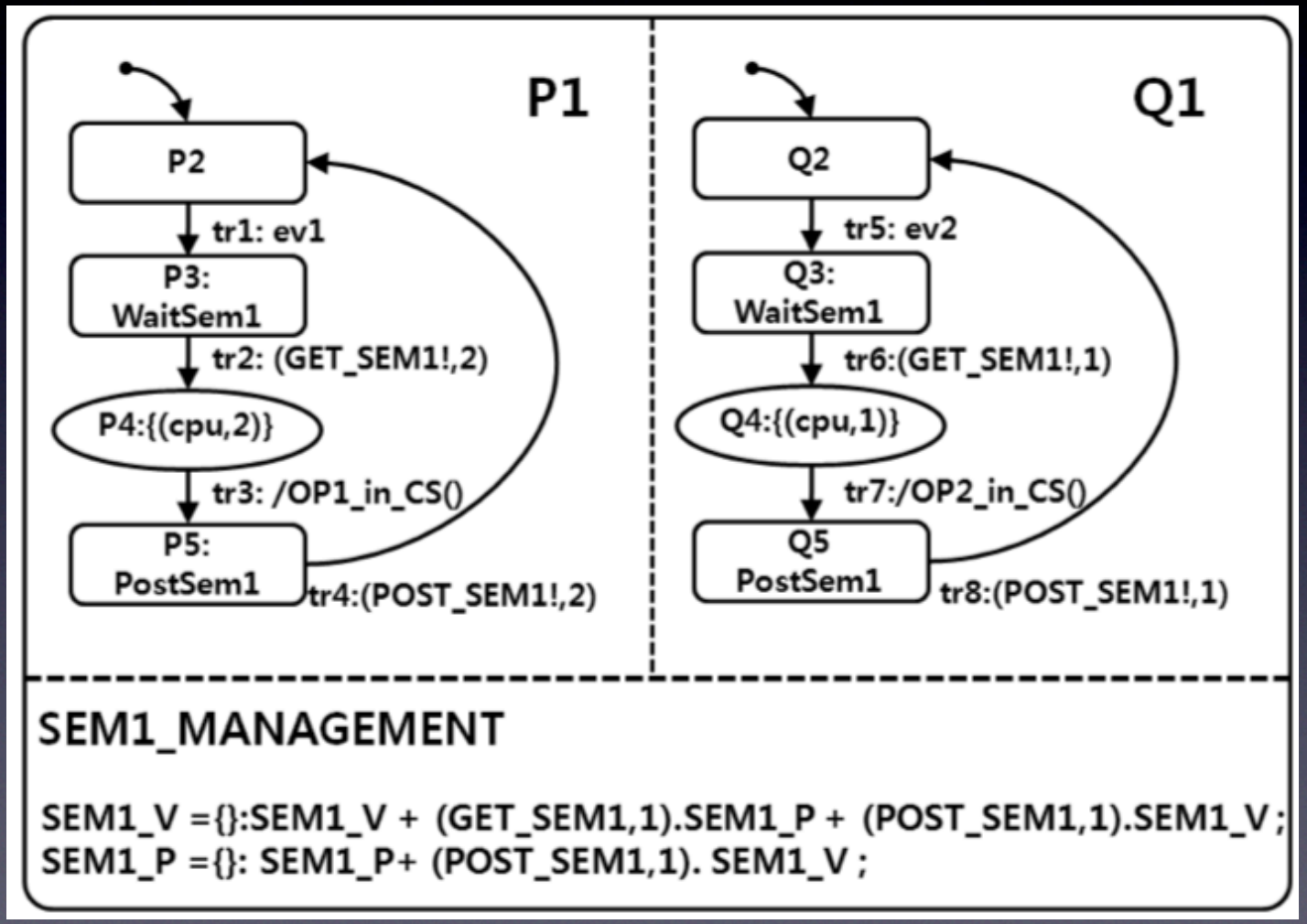
3) β is a τ event with a non-zero priority while α is a timed action;

Contents

TRoS || ACSR

- it represents a composition of two behavioral systems in TRoS and ACSR respectively.

An Simple Example



Semantics

- The behavior of TRoS || ACSR is defined in the same way defining ACSR,
- First, the non-priority relations of their composite behavior are defined,
- Then, the non-prioritized behaviors are prioritized based on preemption relations

Synchronization Event for TRoS

$$E ::= \varepsilon \mid (e?, ve).E \mid (e!, ve).E$$

TRoS in TRoS || ACSR

- A TRoS is defined as a tuple (*N,T,E,G,A*), where
  - *N* is a set of nodes,
  - *T* is a set of transitions,
  - *E* is a set of primitive event expressions,
  - *G* is a set of guard expressions,
  - *A* is a set of action expressions.

TRoS in TRoS || ACSR

- $I_B = E \cup A$: Broadcasting event communicating within TRoS
- I_S : Synchronization event between TRoS and ACSR

A Behavior of TRoS

- A behavior of a system in TRoS is defined as a set of possible of runs[HN96]. A run is a series of statuses. A status is defined by the tuple $s^T = (C_B, C_T, clk, I)$, where
  - C_B denotes a basic configuration,
  - C_T denotes a timed configuration,
  - clk denotes the current time,
  - $I = I_B \cup I_S$.

A Run of TRoS

- A run of the system is a sequence $c^T = s^T_0\ s^T_1\ ...s^T_n$, and we let S^T be the set of statuses.

ACSR in TRoS || ACSR

- A system of ACSR is represented in labeled transition system $T^A = (S^A, Act, \rightarrow_\pi, s^A_0)$ [LPS07], where
  - S^A is a set of states,
  - *Act* includes event and timed actions,
  - $\rightarrow_\pi$: prioritized transitions,
  - s^A_0 : Initial parallel processes (A configuration of processes)

TRoS || ACSR

- A system of TRoS ⫴ ACSR is defined in a labeled transition system $T = (S, A_{sync}, \rightarrow_\sigma, s_0)$, where
  - $S = \{(s^T, s^A) \mid s^T \in S^T \text{ and } s^A \in S^A\}$,
  - $A_{sync} = I_S \cup \{e \mid inv(e) \in Act\} \cup \{\varepsilon\}$,
  - $\rightarrow_\sigma$: prioritized transition relation of TRoS || ACSR
  - s_0 : Initial configuration of TRoS || ACSR consisting of default state nodes and initial processes.

TRoS || ACSR

- The event ε implies that no action takes place in ACSR when a step in TRoS is executed.
- A behavior of TRoS ‖ ACSR is defined by a sequence of $c = s_0 \beta_1 s_1 ... \beta_n s_n$, such that $si \in S$, $\beta i \in Act \cup \{\varepsilon\}$, and $(si, \beta i+1, si+1) \in \rightarrow \sigma$, for all $0 \leq i < n$.

Non-prioritized Relation of TRoS || ACSR

$s_i^T = (\mathcal{C}_B, \mathcal{C}_T, clk, I)$, $s_i^A \in S^A$

TauAct: $$\frac{s_i^A \xrightarrow{(\tau,m)}_\pi s_{i+1}^A}{(s_i^T, s_i^A) \xrightarrow{(\tau,m)} (s_i^T, s_{i+1}^A)}$$

ACSREventAct: $$\frac{s_i^A \xrightarrow{e}_\pi s_{i+1}^A}{(s_i^T, s_i^A) \xrightarrow{e} (s_i^T, s_{i+1}^A)}$$, where $inv(e) \notin I_S$

TRoSEventAct: $$\frac{s_i^T \rightarrow_\gamma s_{i+1}^T, s_{i+1}^T = (\mathcal{C}'_B, \mathcal{C}_T, clk, I')}{(s_i^T, s_i^A) \xrightarrow{\epsilon} (s_{i+1}^T, s_i^A)}$$

, where $I_S = \emptyset$

EventComm:

$$\frac{s_i^T \rightarrow_\gamma s_{i+1}^T, s_i^A \xrightarrow{(a!,m)}_\pi s_{i+1}^A, s_{i+1}^T = (\mathcal{C}'_B, \mathcal{C}_T, clk, I')}{(s_i^T, s_i^A) \xrightarrow{(\tau,n+m)} (s_{i+1}^T, s_{i+1}^A)}$$

, where $(a?, n) \in I_S$ and $(a?, n) \notin I'$

TimedAct:

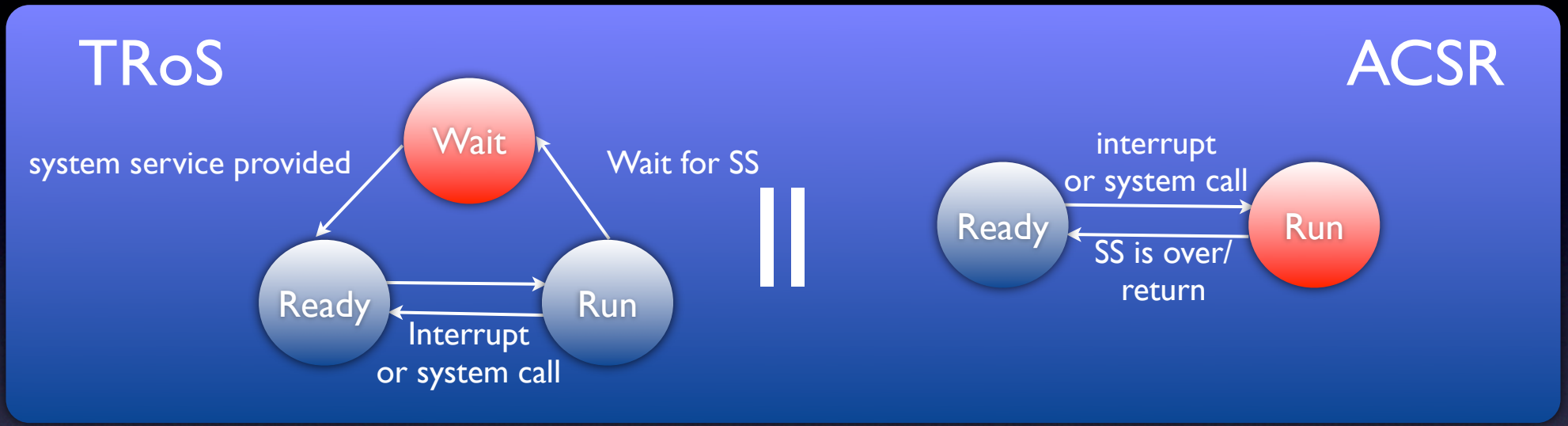
$$\frac{s_i^T \xrightarrow{A_1}_\gamma s_{i+1}^T, s_i^A \xrightarrow{A_2}_\pi s_{i+1}^A, s_{i+1}^T = (\mathcal{C}_B, \mathcal{C}'_T, clk+1, I')}{(s_i^T, s_i^A) \xrightarrow{A_1 \cup A_2} (s_{i+1}^T, s_{i+1}^A)}$$

, where $\rho(A_1) \cap \rho(A_1) = \emptyset$

A Status of TRoS || ACSR

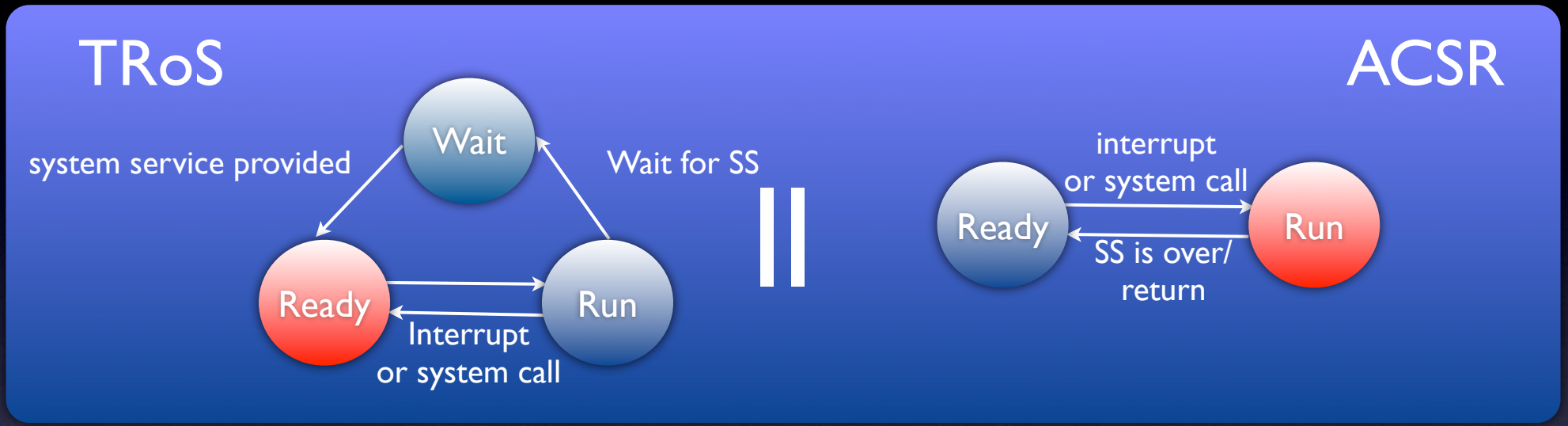
$$s_i^T = (\mathcal{C}_B, \mathcal{C}_T, clk, I),\ s_i^A \in S^A$$

Rule : TauAct



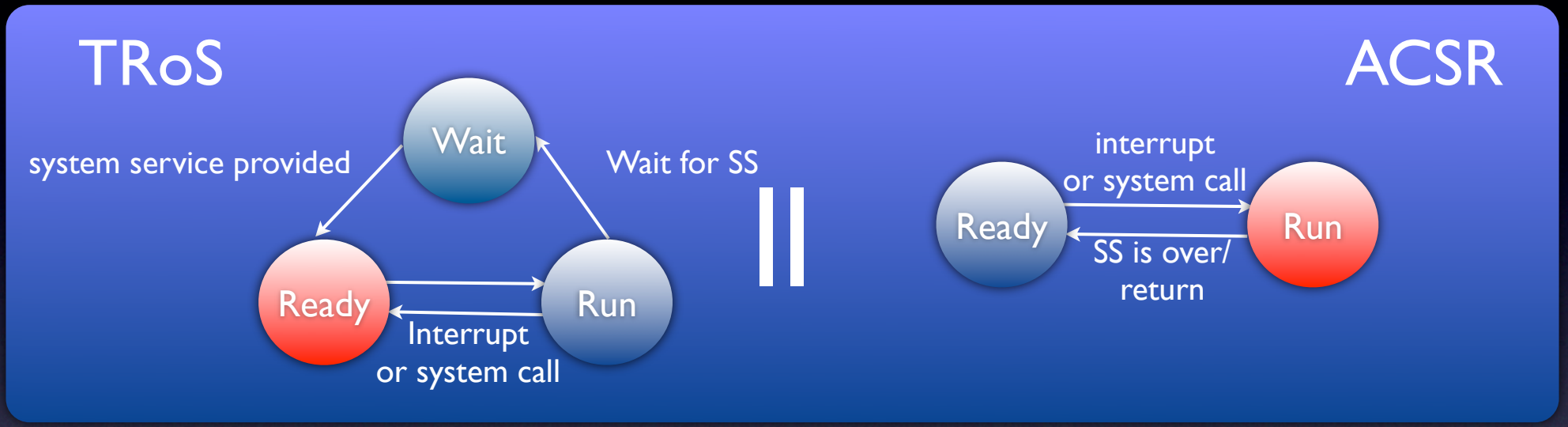
$$\frac{s_i^A \xrightarrow{(\tau,m)}_\pi s_{i+1}^A}{(s_i^T, s_i^A) \xrightarrow{(\tau,m)} (s_i^T, s_{i+1}^A)}$$

Rule : TauAct



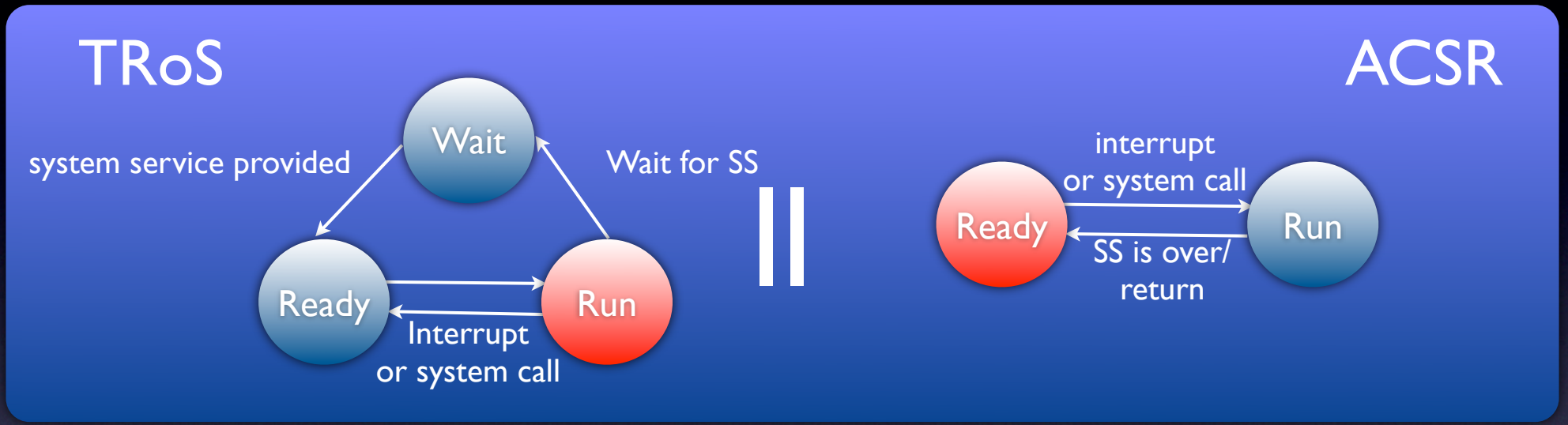
$$\frac{s_i^A \xrightarrow{(\tau,m)}_\pi s_{i+1}^A}{(s_i^T, s_i^A) \xrightarrow{(\tau,m)} (s_i^T, s_{i+1}^A)}$$

Rule : ACSREventAct



$$\frac{s_i^A \xrightarrow{e}_\pi s_{i+1}^A}{(s_i^T, s_i^A) \xrightarrow{e} (s_i^T, s_{i+1}^A)}, \text{ where } inv(e) \notin I_S$$

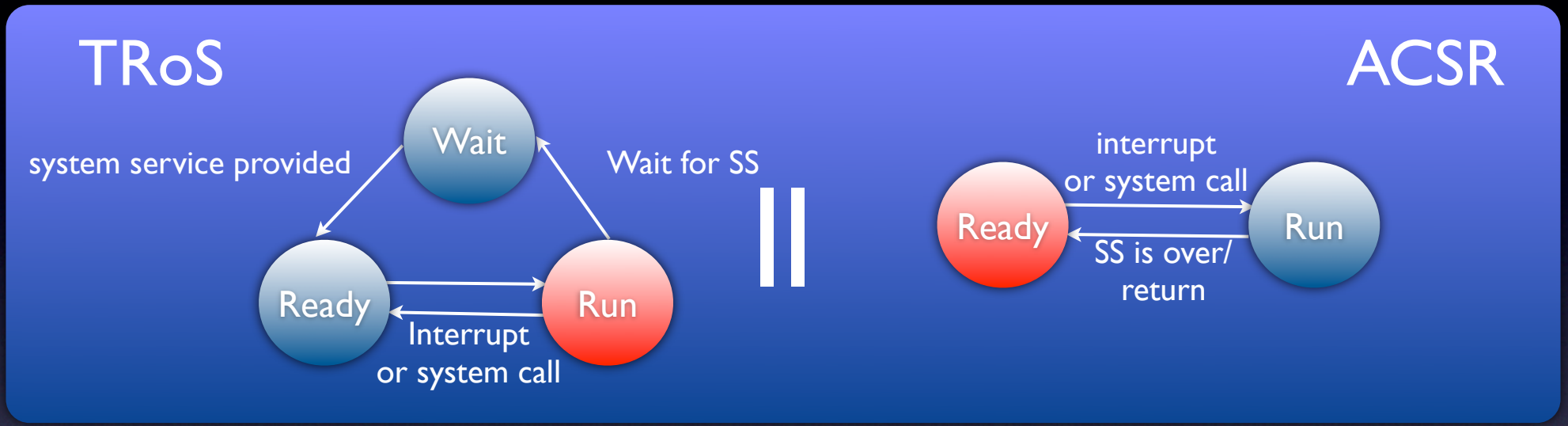
Rule : TRoSEventAct



$$\frac{s_i^T \rightarrow_\gamma s_{i+1}^T, s_{i+1}^T = (\mathcal{C}'_B, \mathcal{C}_T, clk, I')}{(s_i^T, s_i^A) \xrightarrow{\epsilon} (s_{i+1}^T, s_i^A)}$$

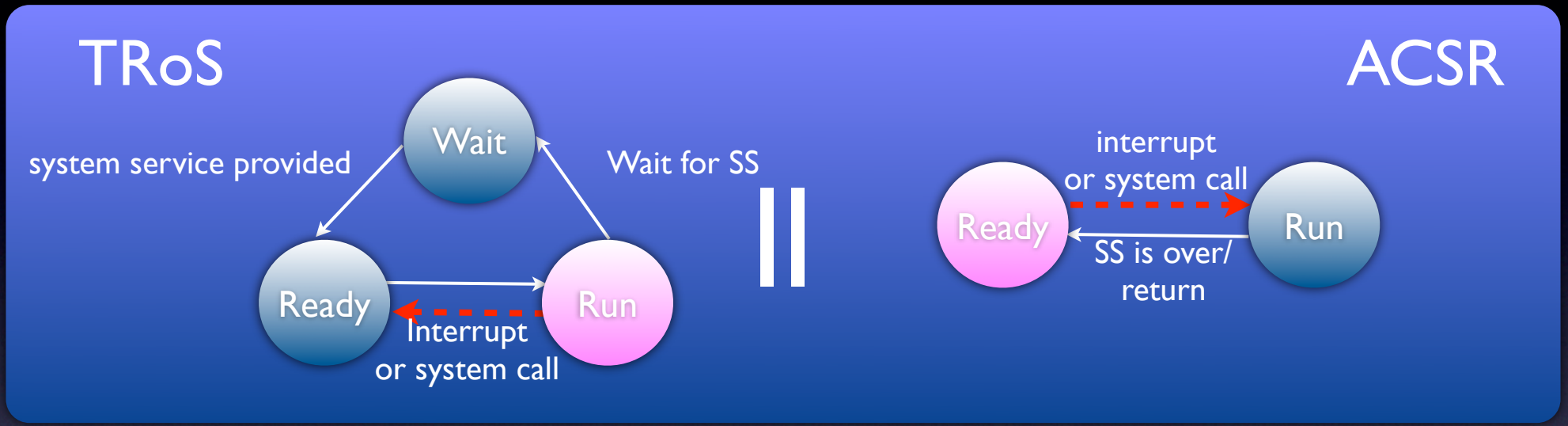
, where $I_S = \emptyset$

Rule : TimedAction



$$\frac{s_i^T \xrightarrow{A_1}_{\gamma} s_{i+1}^T, s_i^A \xrightarrow{A_2}_{\pi} s_{i+1}^A, s_{i+1}^T = (\mathcal{C}_B, \mathcal{C}'_T, clk+1, I')}{(s_i^T, s_i^A) \xrightarrow{A_1 \cup A_2} (s_{i+1}^T, s_{i+1}^A) \quad , \text{ where } \rho(A_1) \cap \rho(A_1) = \emptyset}$$

Rule : EventComm



$$\frac{s_i^T \rightarrow_\gamma s_{i+1}^T, s_i^A \xrightarrow{(a!,m)}_\pi s_{i+1}^A, s_{i+1}^T = (C'_B, C_T, clk, I')}{(s_i^T, s_i^A) \xrightarrow{(\tau, n+m)} (s_{i+1}^T, s_{i+1}^A)}$$

$$, \text{ where } (a?, n) \in I_S \text{ and } (a?, n) \notin I'$$

Preemption Relation

- ***Definition (Preemption Relation for TRoS || ACSR) Given two actions α and β, we say that β preempts α, denoted by (α ≺ β), if one of the following cases hold:***
  1. Both α and β are events in D_E, where $\alpha = (a,p), \beta = (a,p')$, and $p<p'$
  2. Both α and β are actions in D_R, where

     $(\rho(\beta) \subseteq \rho(\alpha)) \wedge$

     $(\forall (r,p) \in \alpha(((r,p') \in \beta \Rightarrow p \leq p') \wedge ((r,p') \not\in \beta \Rightarrow p = 0))) \wedge$

     $(\exists (r,p') \in \beta \exists (r,p) \in \alpha . p < p')$
  3. $\alpha \in D_R$ *and* $\beta \in D_E$, *with* $\beta=(\tau,p)$ *and* $p>0$
  4. **$\alpha=\varepsilon$ *and* $\beta \in D_E$, *with* $\beta=(\tau,p)$ *and* $p>0$**
  5. **$\alpha \in D_R$ *and* $\beta=\varepsilon$.**

Prioritized Transition System

Definition) The labeled transition "$\rightarrow\boldsymbol{\sigma}$" is defined as follows: $s \xrightarrow{\alpha}\sigma\ s'$ if and only if ***1) $s \rightarrow s'$ is an unprioritized transition*** and ***2) there is no unprioritized transition system $s \xrightarrow{\beta} s''$ such that $\alpha < \beta$.***

Contents

Specifying RTOS Services

- Scheduling
- Synchronization
- Communication
- Time management

Semaphore in ACSR

```
SEM_Pend = {}:SEM_Pend
         + (OSSemPend,1).SEM_Post + (OSSemPost,1).SEM_Pend;
SEM_Post = {}:SEM_Post + (OSSemPost,1).SEM_Pend ;
```

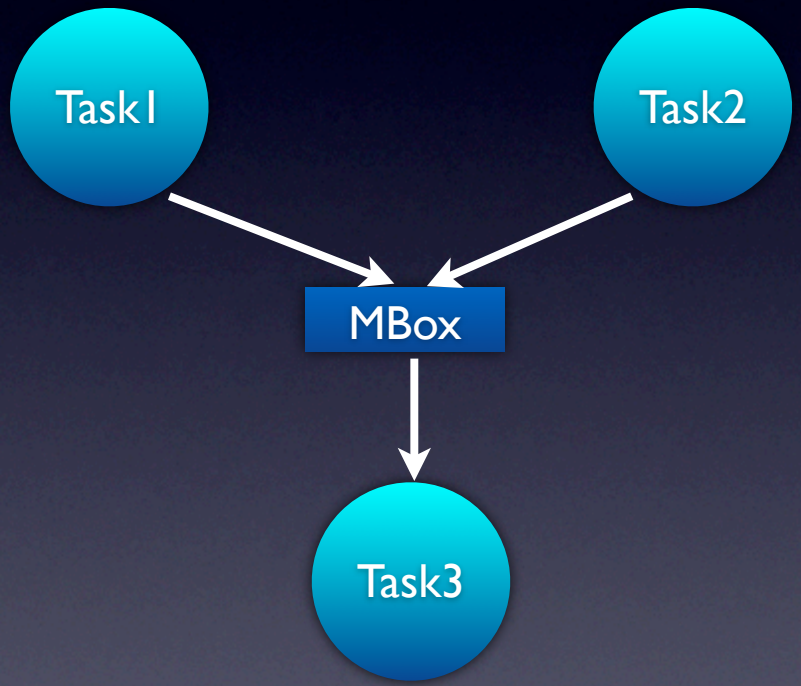
Message Queue

```
MngMQ1   = {}:MqMng_MQ_1 + (CreateMQ1,1).MQPend ;
MQPend   = {}:MQPend + (PostMQ1,1).MQPost1
         + (DeleteMQ1,1).MngMQ1;
MQPost1 = {}:MQPost1 + (PostMQ1,1).MQPost2
         + (PendMQ1,1).MQPend + (DeleteMQ1,1).MngMQ1;
MQPost2 = {}:MQPost2 + (PostMQ1,1).MQPost3
         + (PendMQ1,1).MQPost1 + (DeleteMQ1,1).MngMQ1;
MQPost3 = {}:MQPost3 + (PendMQ1,1).MQPost2
         + (OSMqDelete_MQ_1,1).MngMQ1;
```

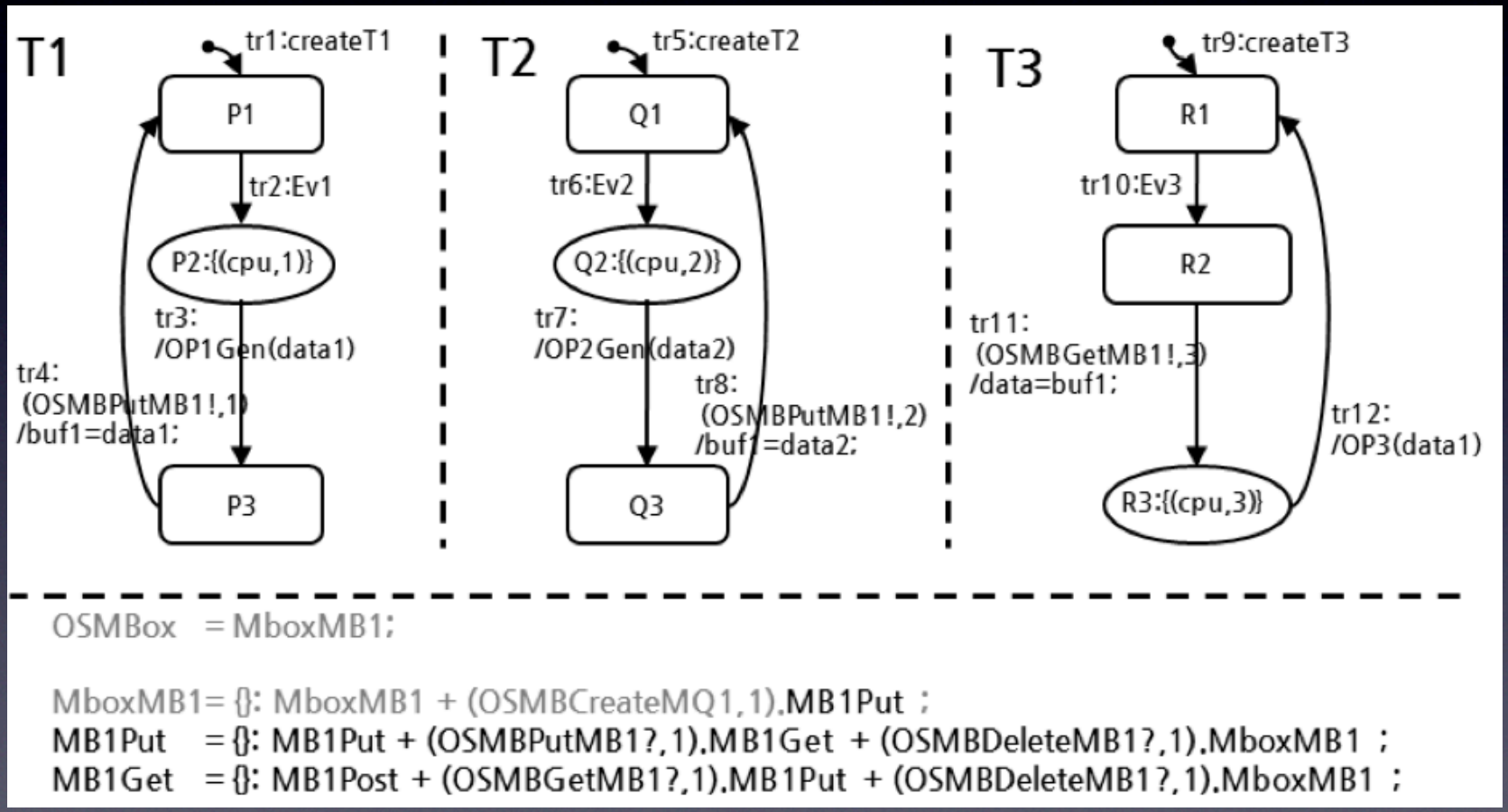
Time Manager

```
TimerSEC10 = {}:TimerSEC
           + (OSTimeDlySEC10Req,1).TimerSEC10Ack;
TimerSEC10Ack = {}^10:TimerSECAckExe;
TimerSEC10AckExe = {}:TimerSECAckExe
                 + (OSTimeDlySEC10Ack,1).TimerSEC10;
```

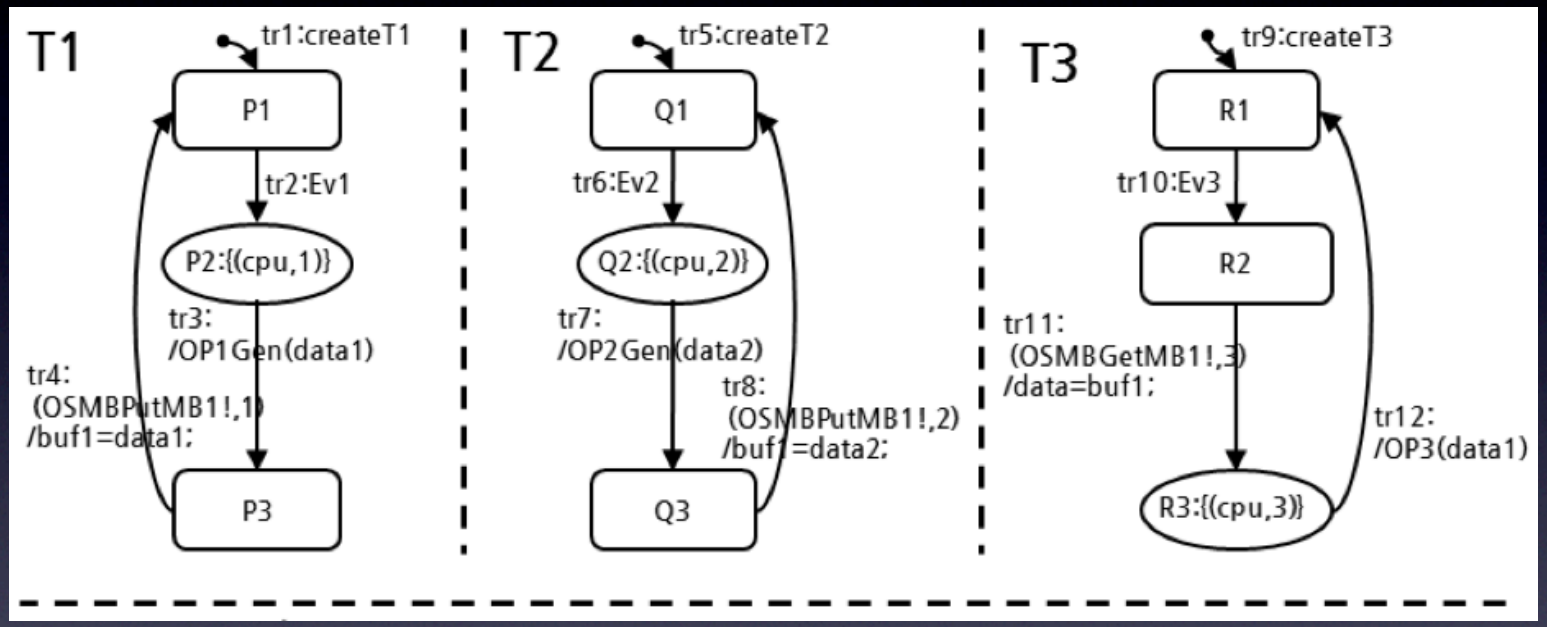
Example of TRoS || ACSR

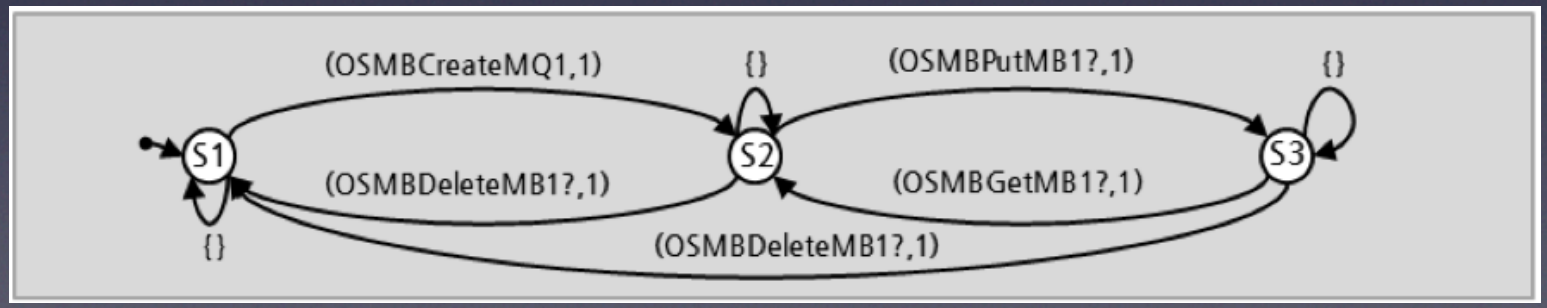


A TRoS || ACSR Model

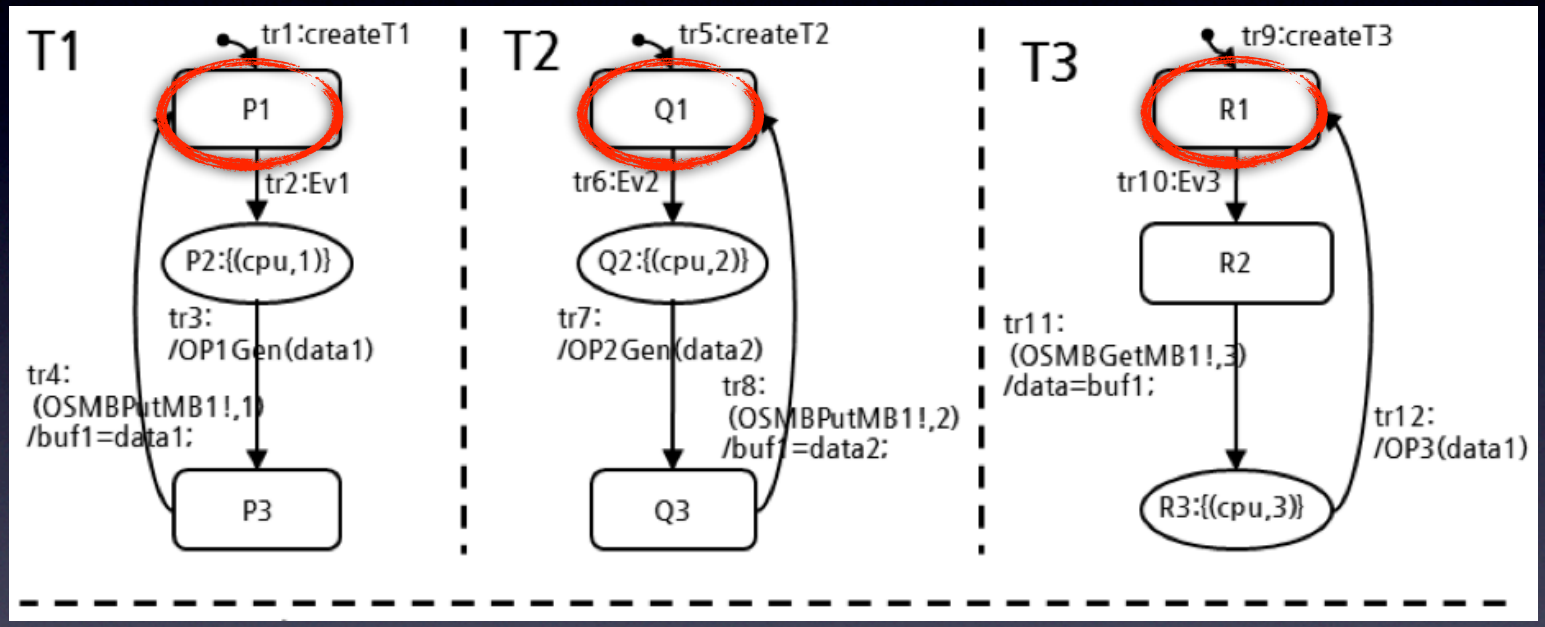


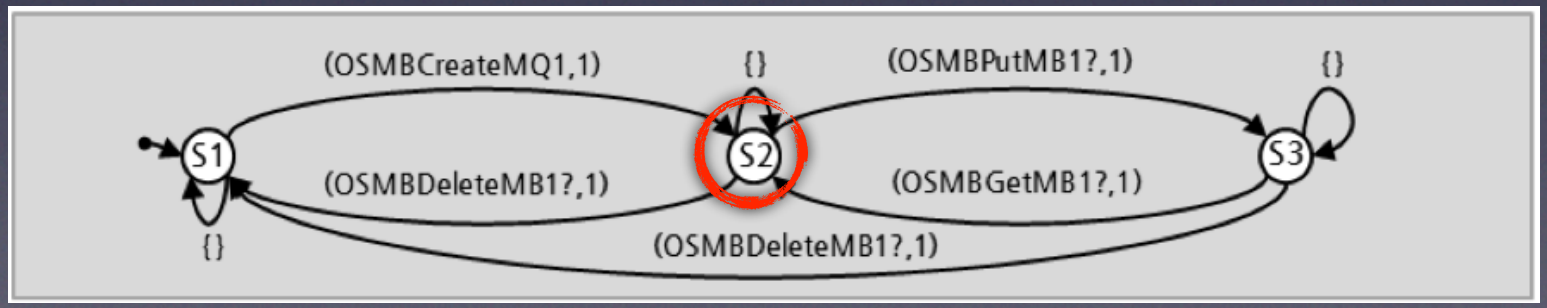
A TRoS || ACSR Model





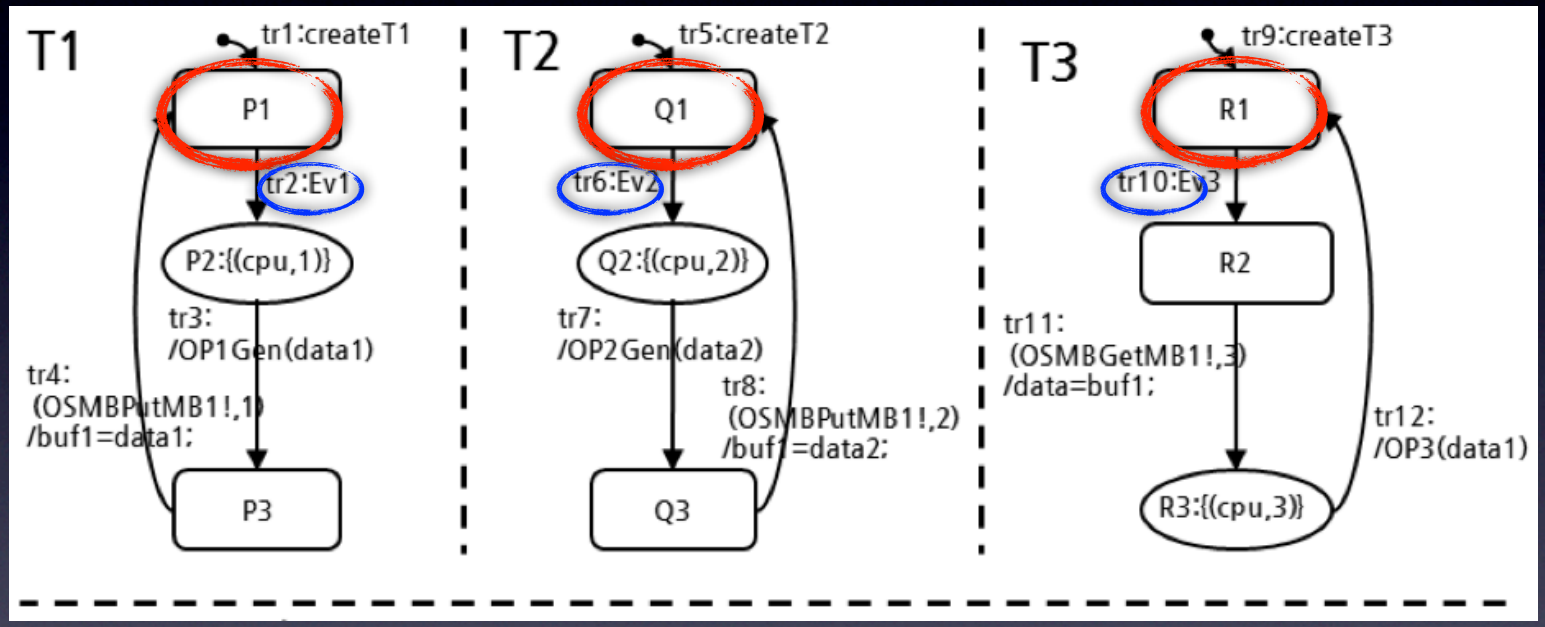
A TRoS || ACSR Model

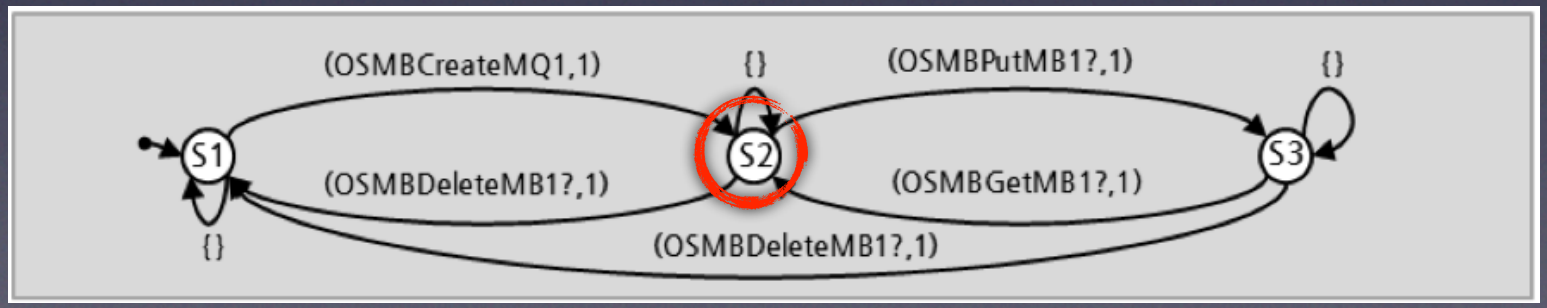




Current Time: 0

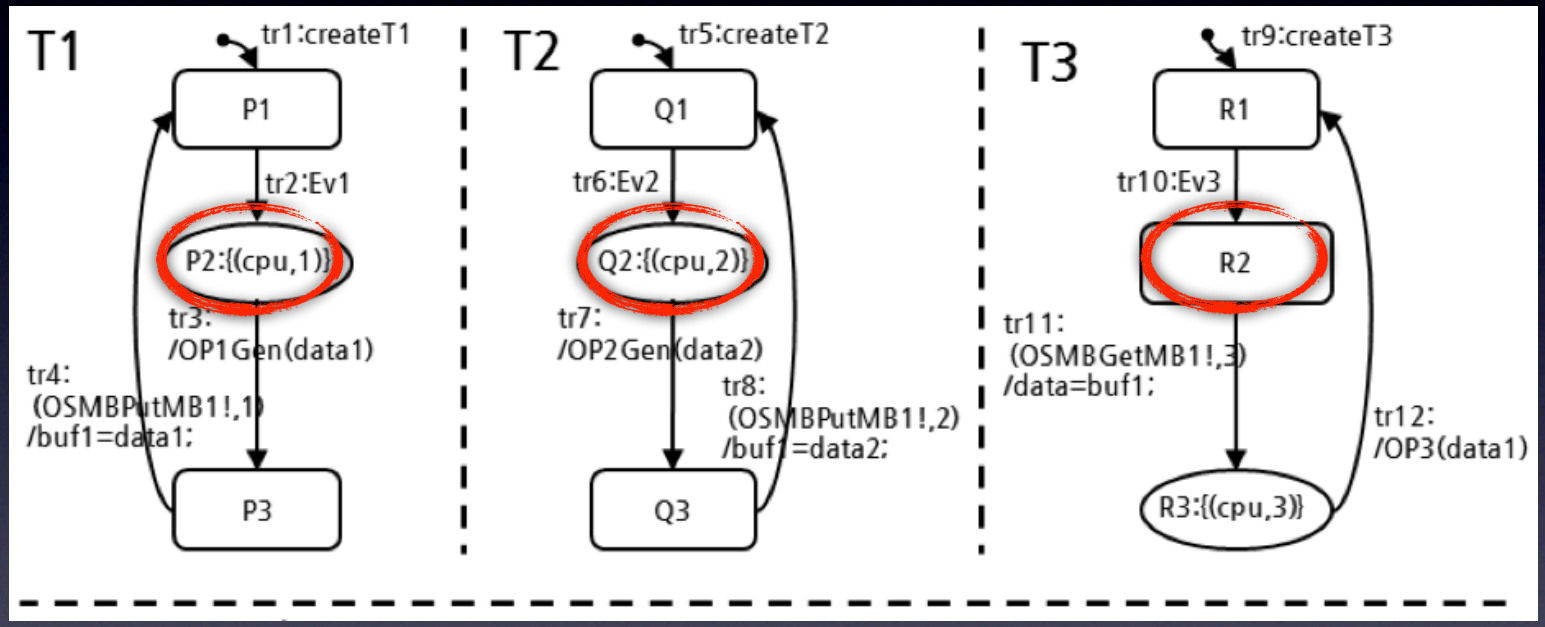
A TRoS || ACSR Model

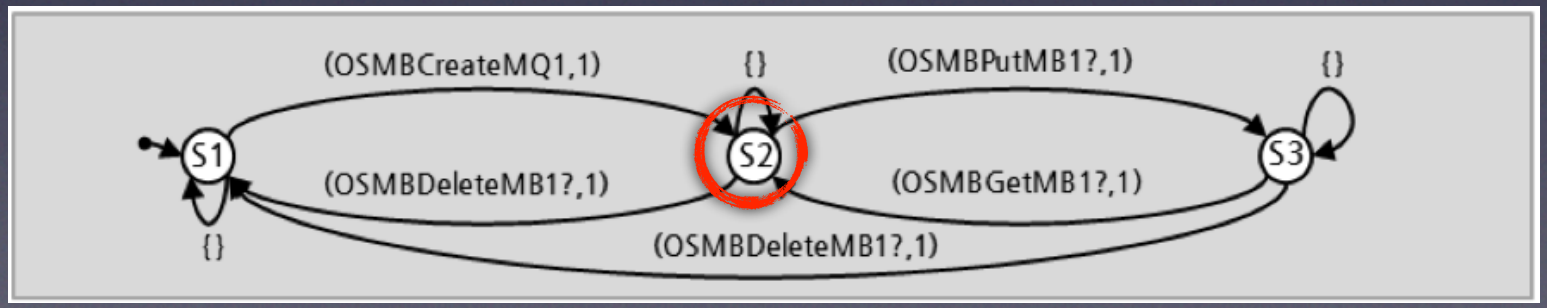




Current Time: 0

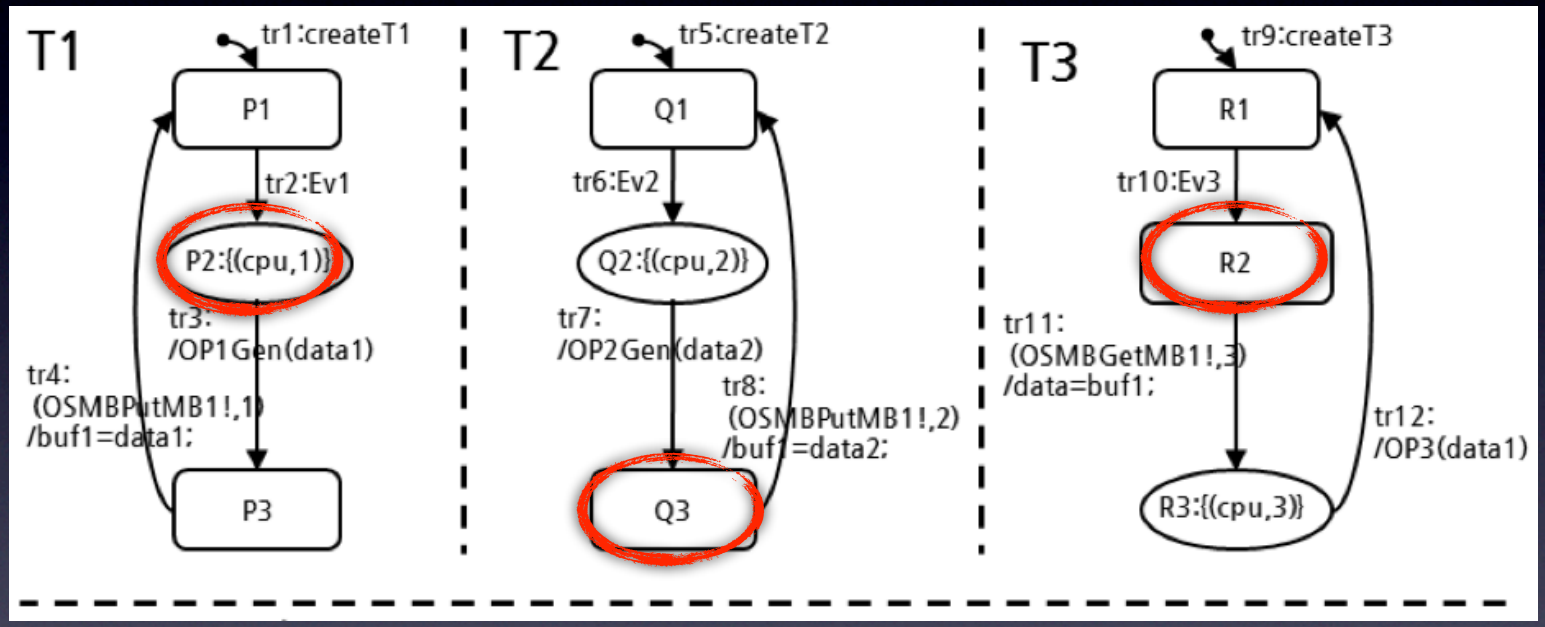
A TRoS || ACSR Model

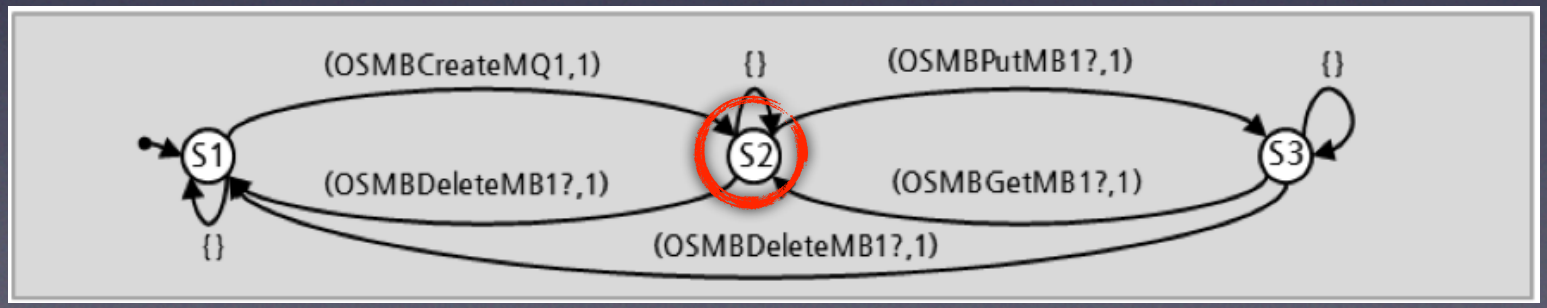




Current Time: 0

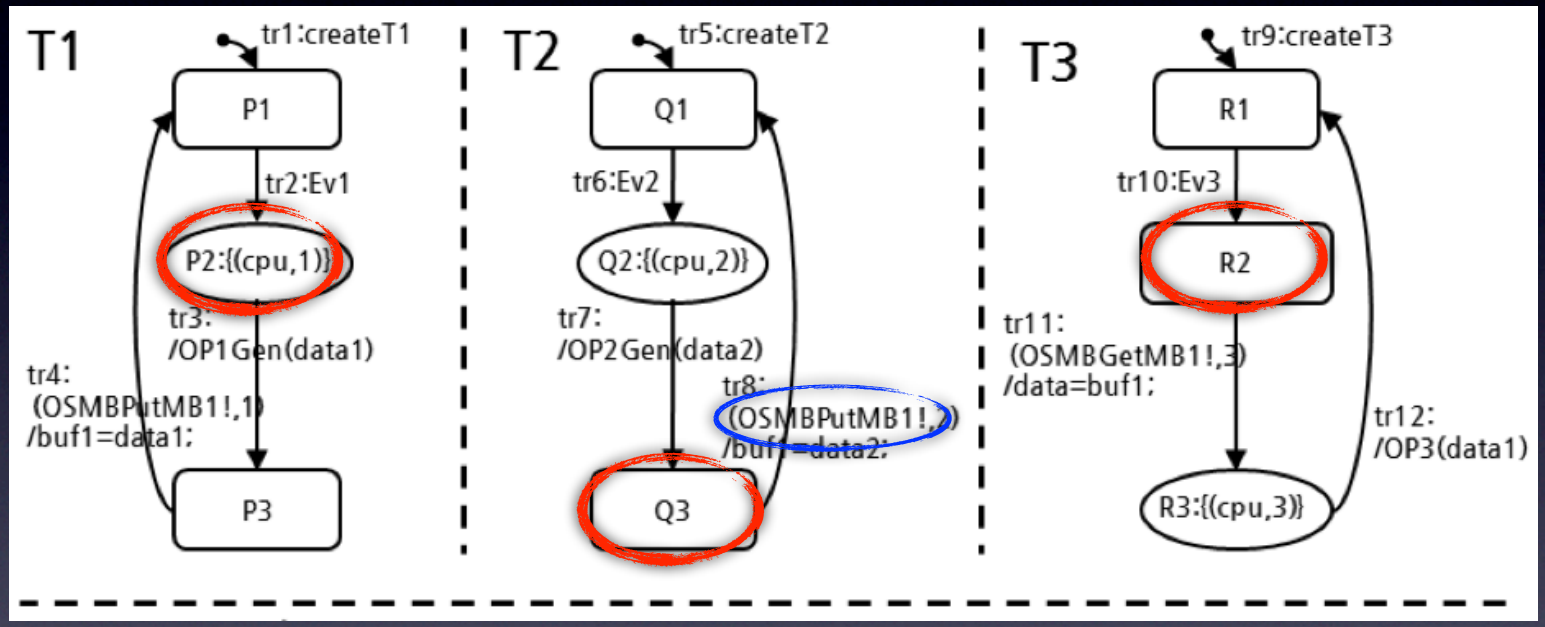
A TRoS || ACSR Model

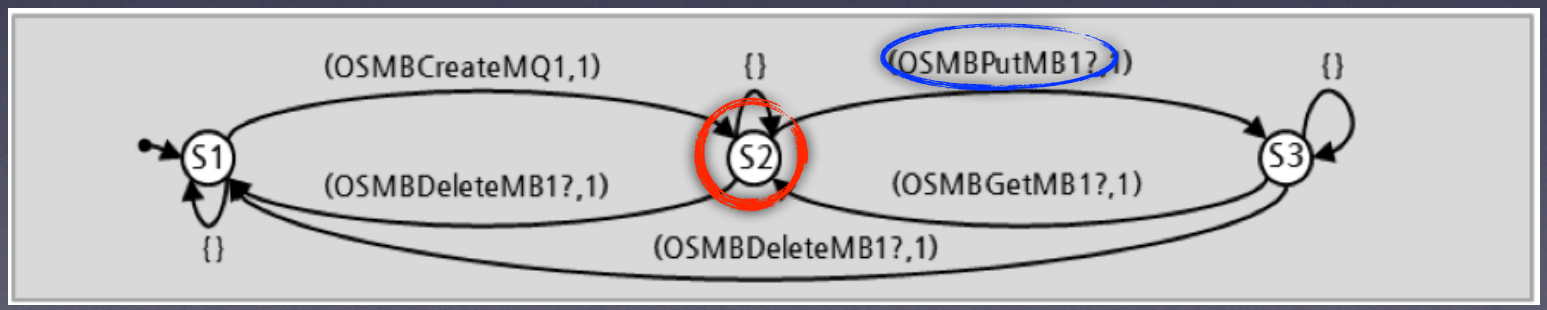




Current Time: 0

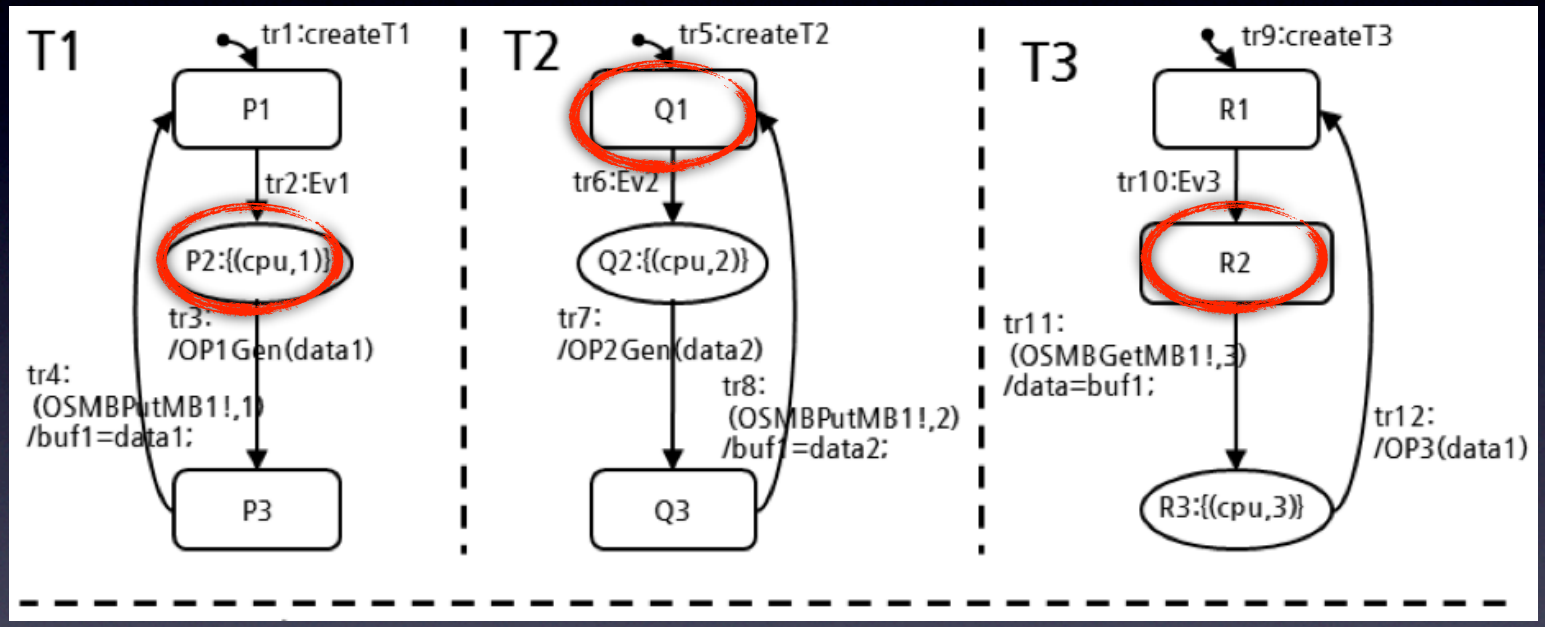
A TRoS || ACSR Model

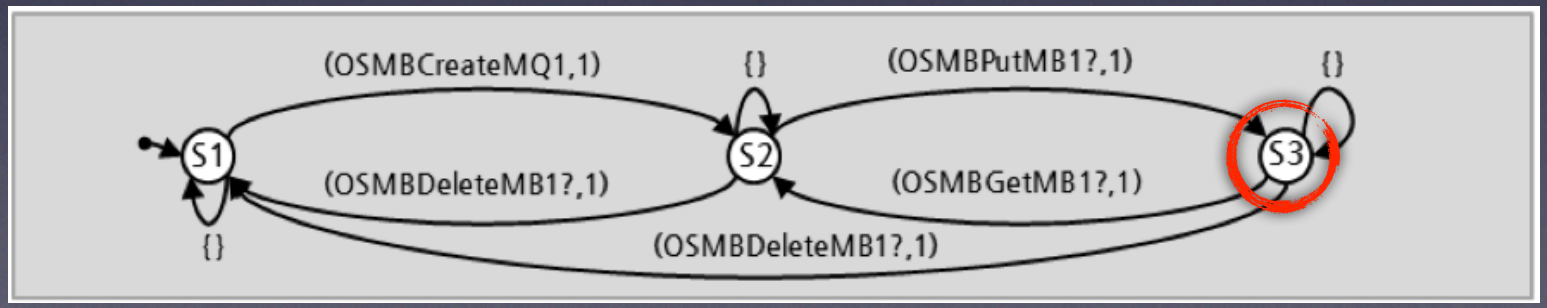




Current Time: 0

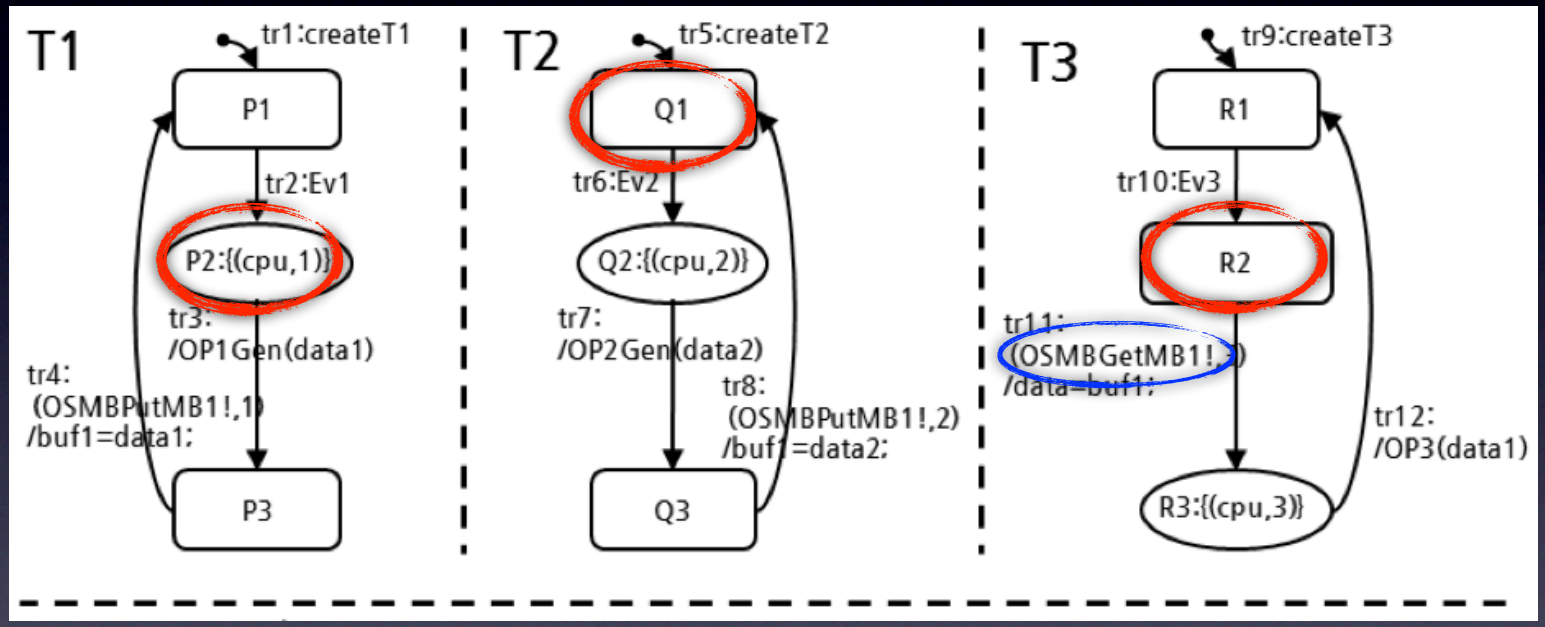
A TRoS || ACSR Model

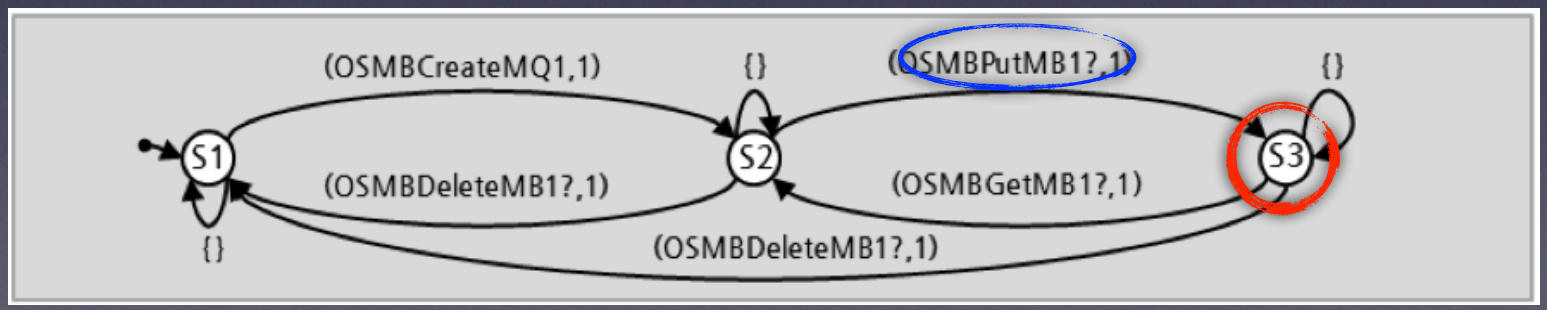




Current Time: 0

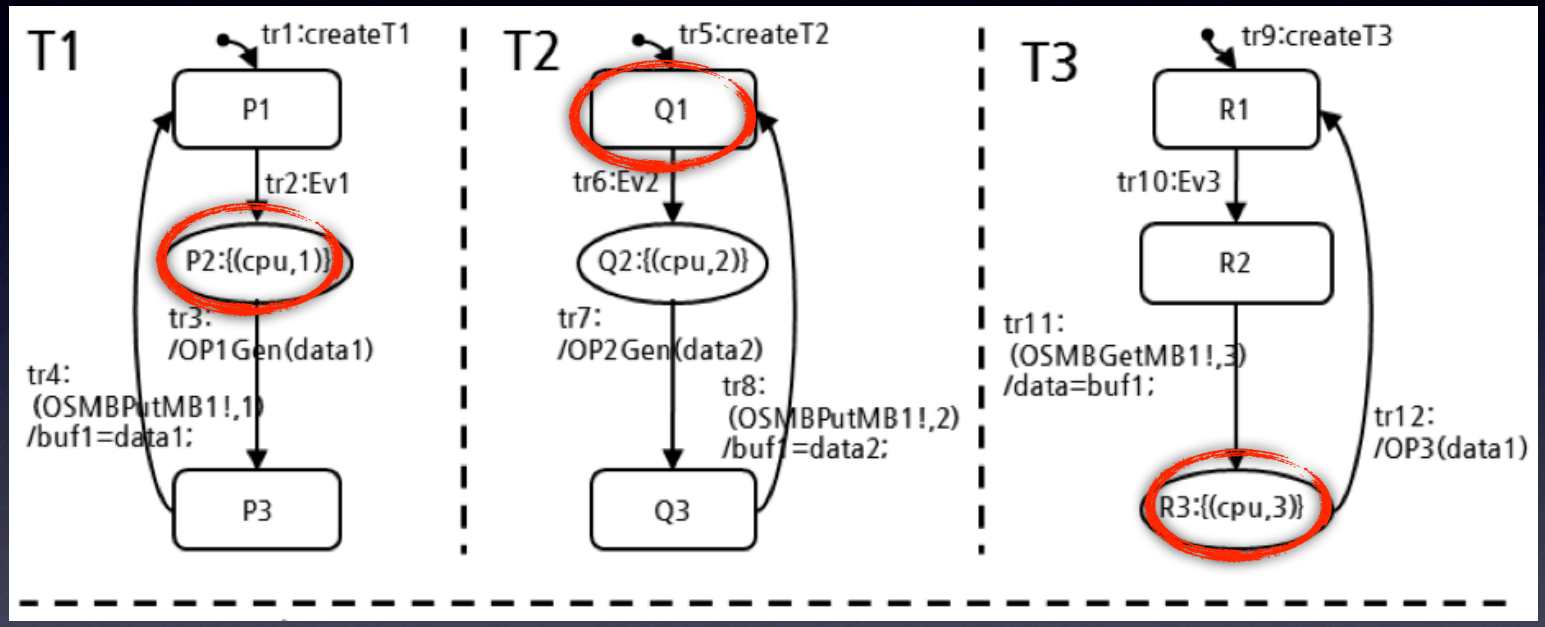
A TRoS || ACSR Model

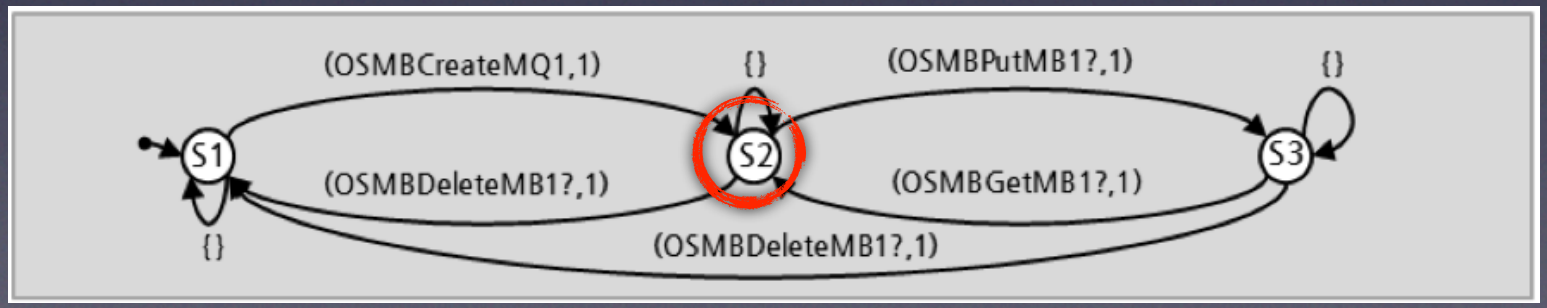




Current Time: 0

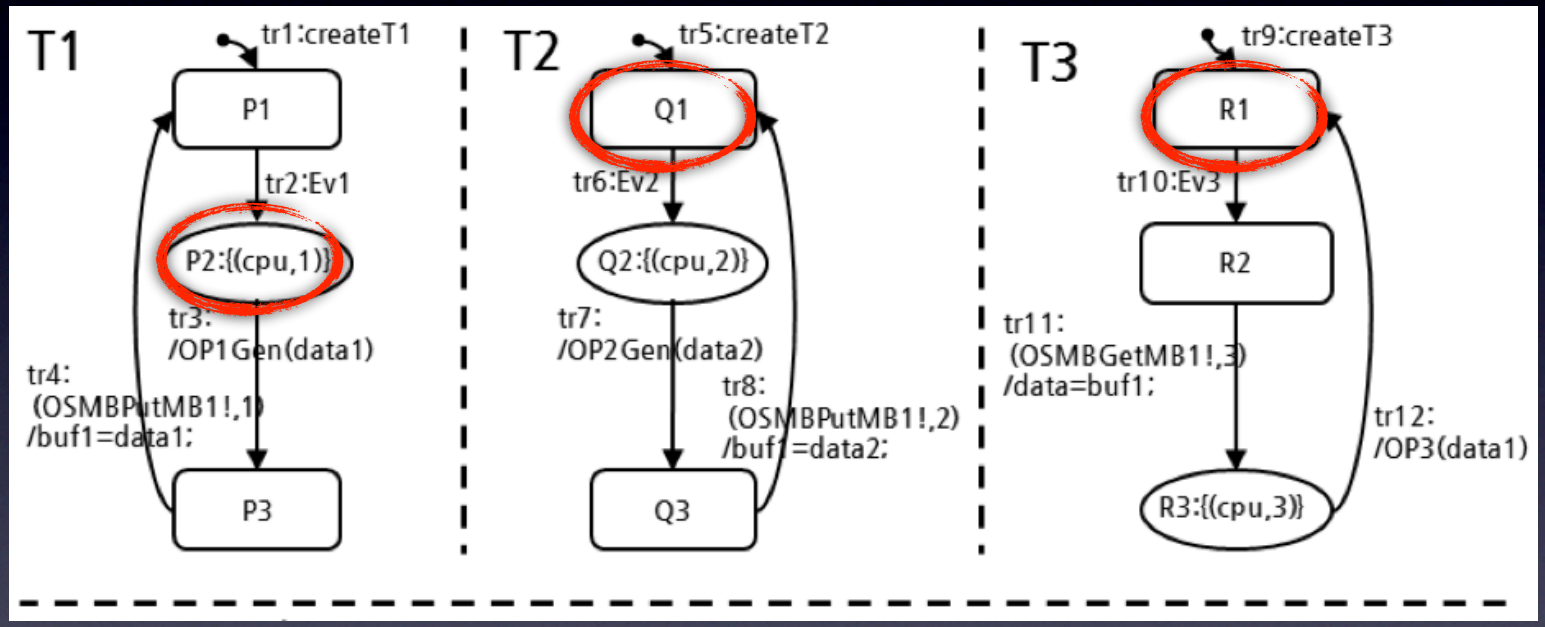
A TRoS || ACSR Model

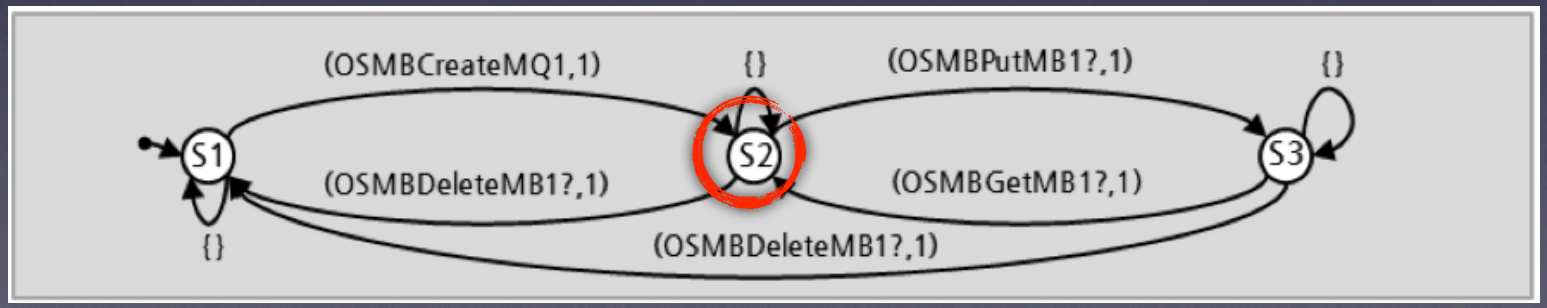




Current Time: 0

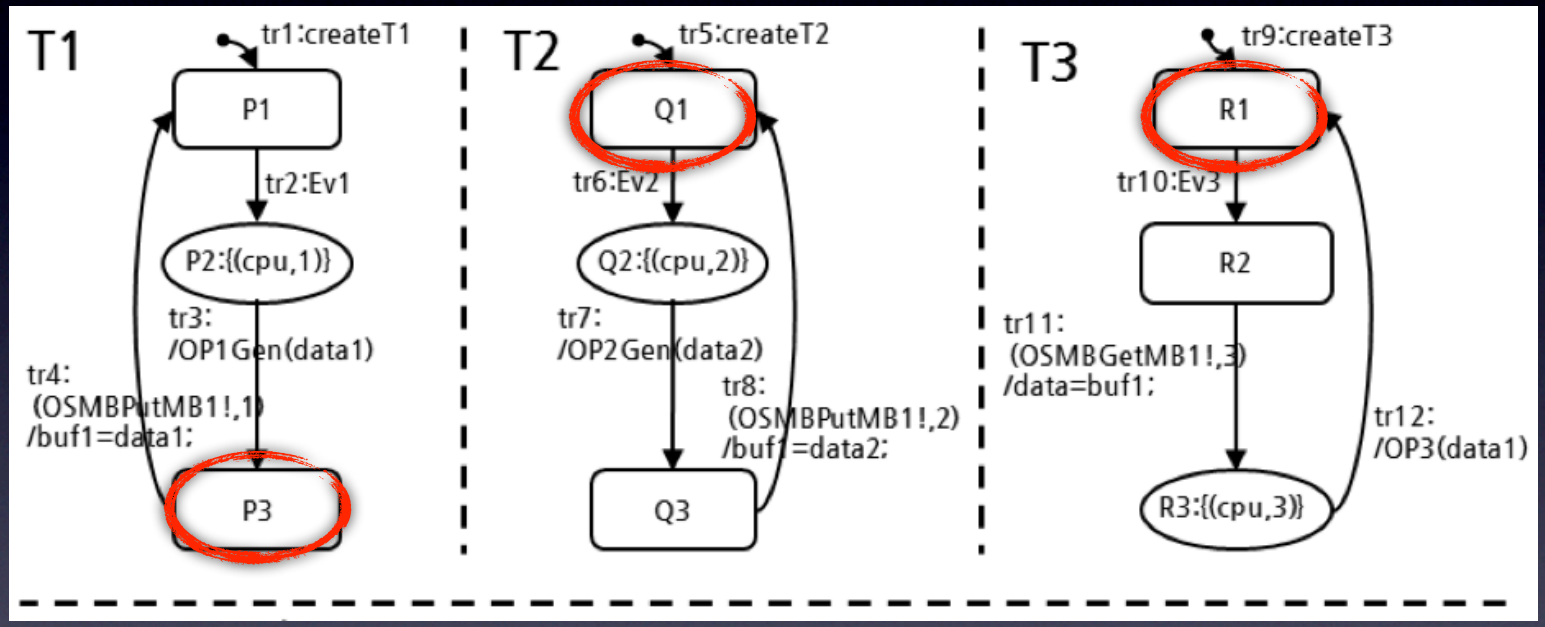
A TRoS || ACSR Model

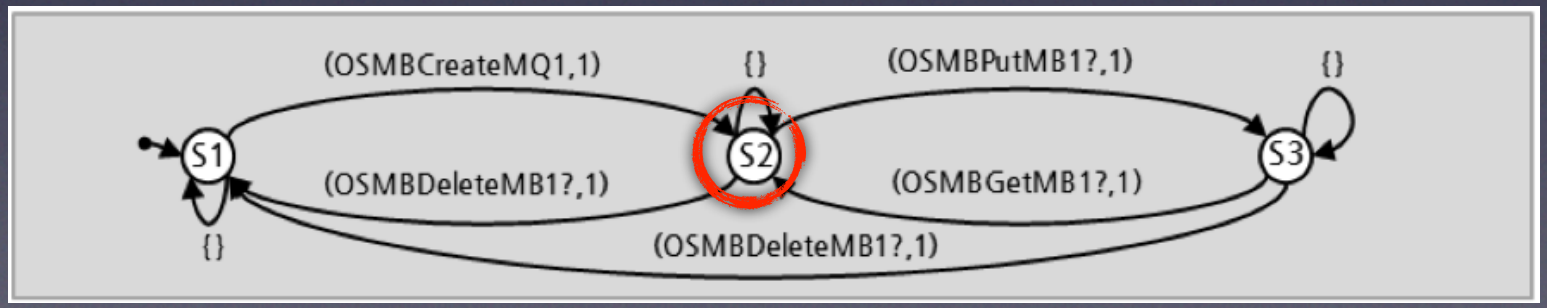




Current Time: 0

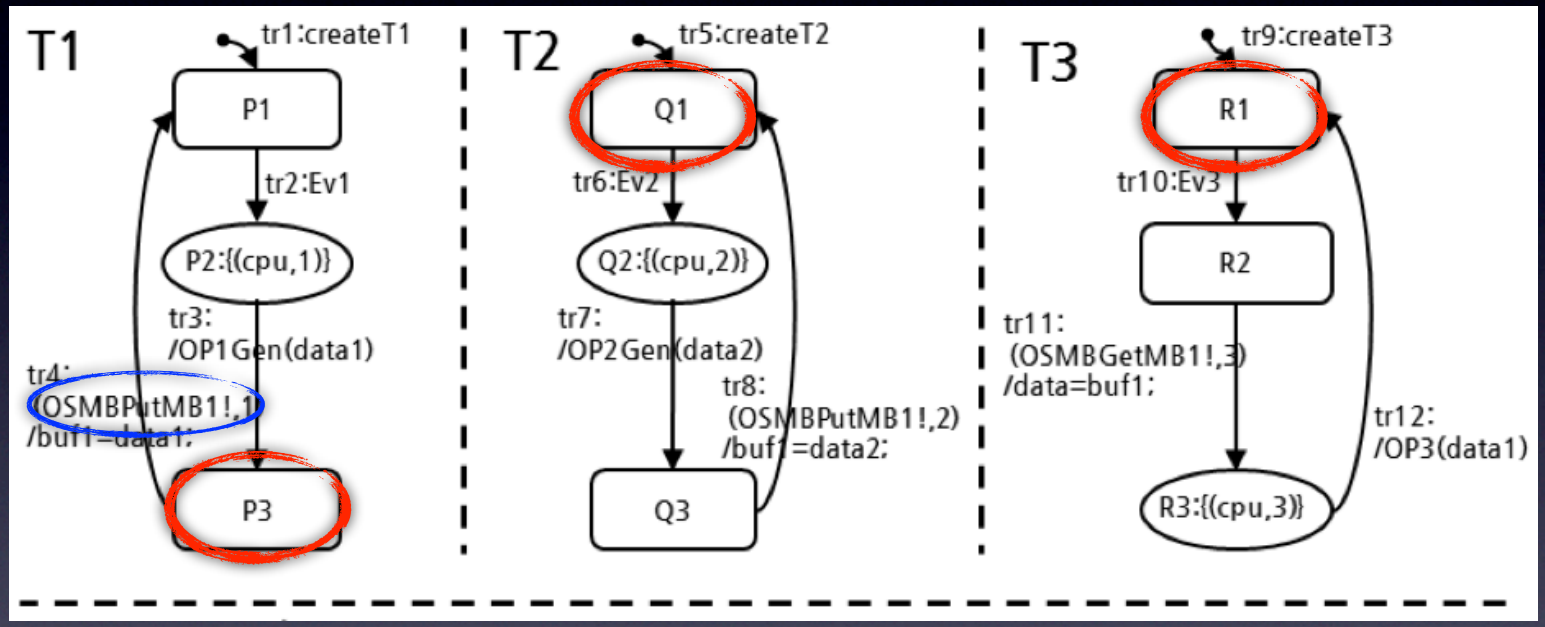
A TRoS || ACSR Model

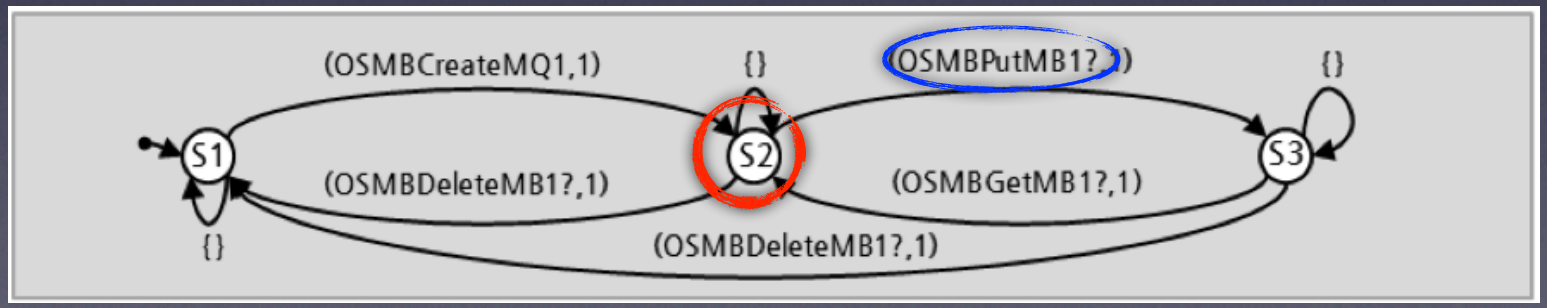




Current Time: 0

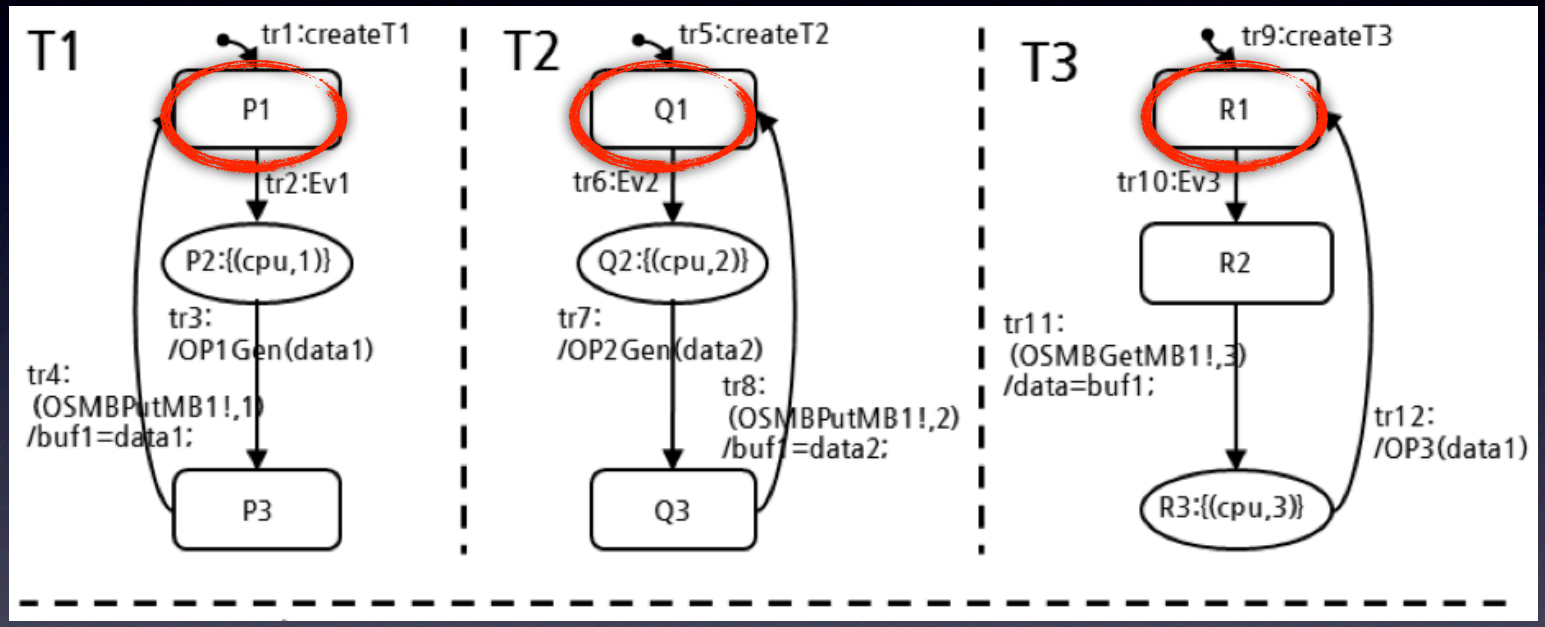
A TRoS || ACSR Model

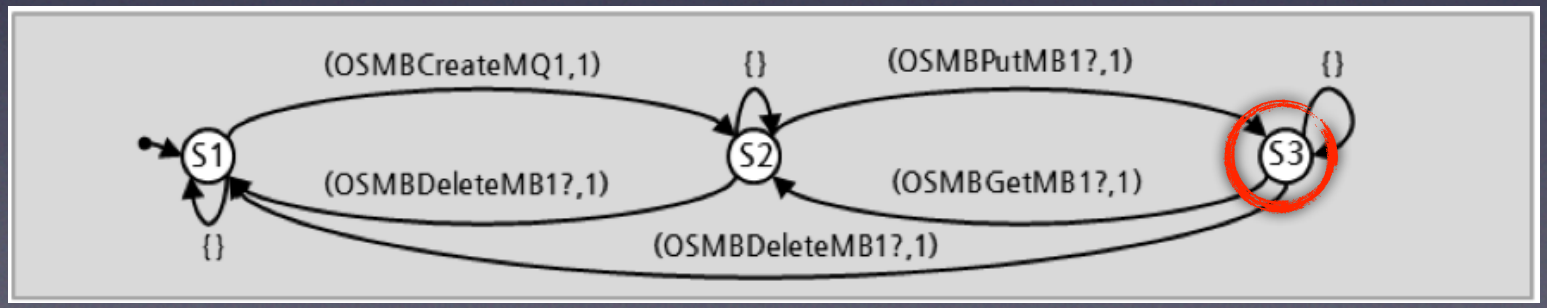




Current Time: 0

A TRoS || ACSR Model





Current Time: 0

Contents

- Introduction
- Background
  - TRo1S and ACSR
- TRoS || ACSR for Embedded Software Components
- Specifying Real-time Embedded Applications in TRoS || ACSR
- Conclusions

Conclusions

- We define here
  - the semantics of a composition of TRoS || ACSR representing application software and RTOS respectively

Conclusions

- TRoS
  - extends Statecharts in terms of time and resource constraints by annotation methods.
  - presents a way to gain a timed and resource-constrained behavioral model from Statecharts in easy way.

Conclusions

- ACSR
  - useful to capture RTOS in easy way
  - provides explicit notion of timely prioritized use of resource
  - able to verified with FM verification tools

Conclusions

- TRoS || ACSR
  - defines a composition of two different systems representing application software and platform software, i.e., RTOS.

Conclusions

- This work contributes to
  - independently designing application software and platform software with their appropriate formal specification languages,
  - analyzing their composite behaviors based on our behavioral semantics of TRoS || ACSR.

References

[GG02] Abdoulaye Gamati ´e and Thierry Gautier. Syn- chronous Modeling of Modular Avionics Archi- tectures using the SIGNAL Language. Research Report RR-4678, INRIA, 2002.

[KKLC10a] Jin Hyun Kim, Inhye Kang, Insup Lee, and Jin- Young Choi. Timed and resource-oriented state- charts for embedded software. Accepted to IEEE Transactions on Industrial Informatics, 2010.

[HN96] David Harel and Amnon Naamad. The STATEMATE semantics of statecharts. ACM Trans. Softw. Eng. Methodol., 5(4):293–333, 1996.

[LBGG94] Insup Lee, P. Br'emond-Gr'egoire, and R. Ger- ber. A process algebraic approach to the specifi- cation and analysis of resource-bound real-time systems. Proceedings of the IEEE Special Is- sue on Real-Time Systems, pages 158–171, Jan 1994.

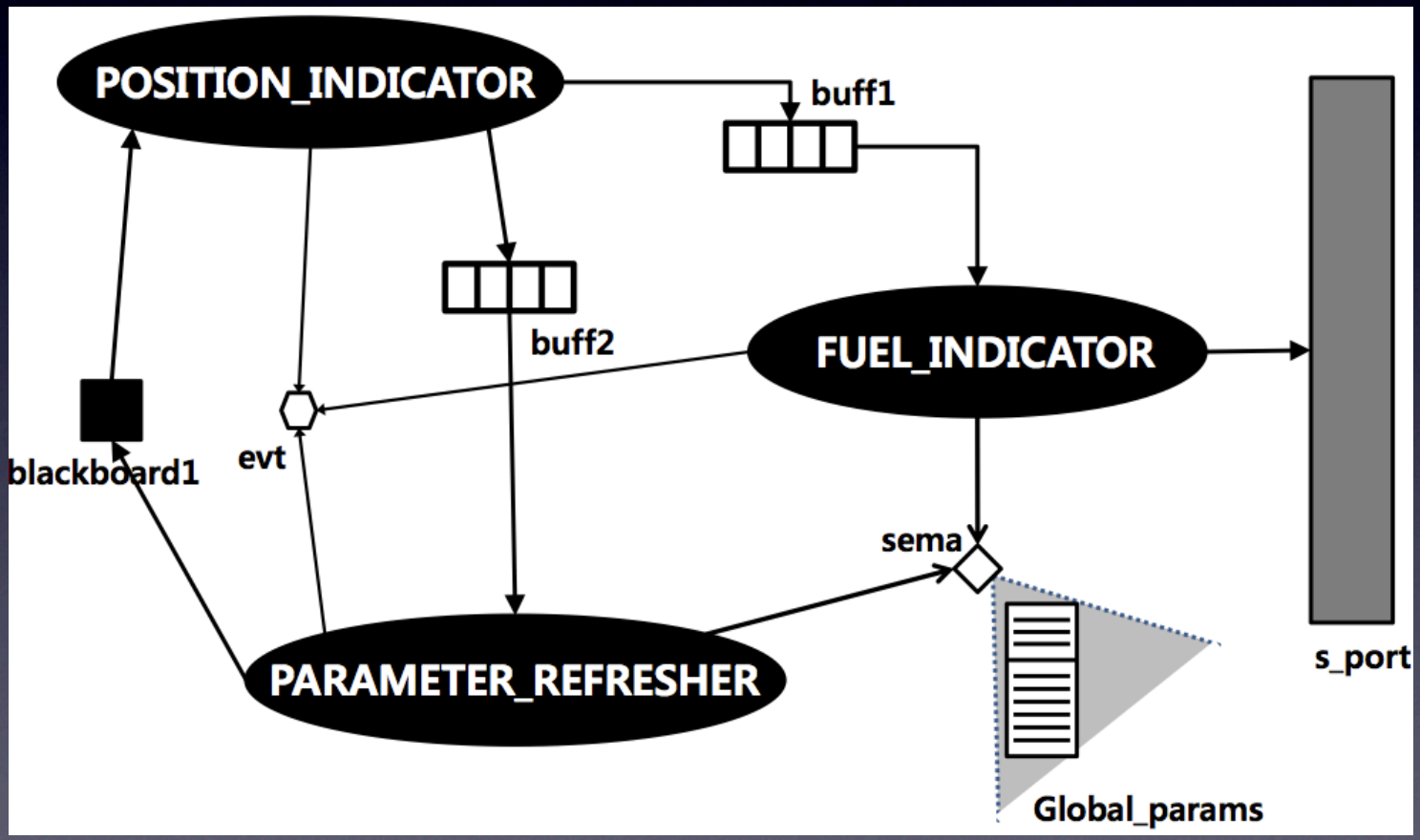
[LPS07] Insup Lee, Anna Philippou, and Oleg Sokolosky. Resources in process algebra. Journal of Logic and Algebraic Programming, 72(1):98 – 122, 2007. Algebraic Process Calculi: The First Twenty Five Years and Beyond. II.

[SLC06] O. Sokolsky, I. Lee, and D. Clarke. Schedulabil- ity analysis of AADL models. Parallel and Dis- tributed Processing Symposium, International, 0:164, 2006.

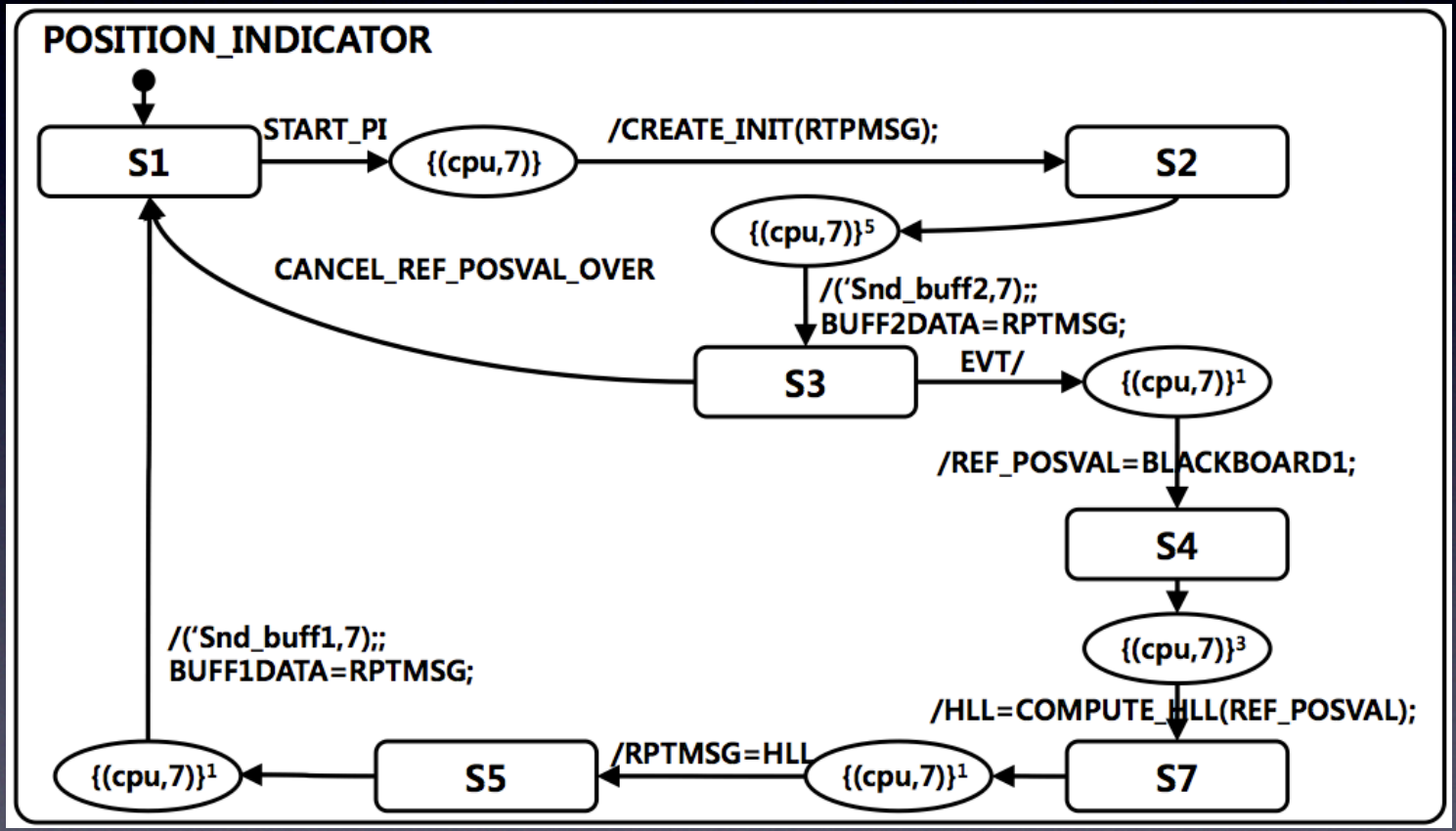
Thanks

Case Study

Avionics Systems based on ARINC 653



Position Indicator in TRoS



ARINC 653 Semaphore in ACSR

```
BUFFER_SERVICE  = SERVICE_BUFF1  ;

SERVICE_BUFF1   = SERVICE_BUFF_buff1 || STAT_BUFF_buff1 ;
STAT_BUFF_buff1 = {}:STAT_BUFF_buff1
                + (Get_buff1_id,1)
                  .(('get_buff1_id_NO_ERROR,1)
                    .STAT_BUFF_buff1
                  + (buff1_name_not_identified,1)
                    .('get_buff1_id_INVALID_CONFIG,1)
                    .STAT_BUFF_buff1)
                + (Get_buff1_status,1)
                  .(('get_buff1_status_NO_ERROR,1)
                    .STAT_BUFF_buff1
                  + (buff1_ID_not_identified,1)
                    .('get_buff1_status_INVALID_CONFIG,1)
                    .STAT_BUFF_buff1) ;

SERVICE_BUFF_buff1   = CREATEBUFF_buff1  ;

CREATEBUFF_buff1 = {}:CREATEBUFF_buff1  + (Crt_buff1,1).CrtBUFFRtnCdbuff1;
CrtBUFFRtnCdbuff1= ('crtbuff1_NO_ERROR,1).BUFFinSVC_buff1
                 + (no_enough_buff_space,1)
```

Verification

```
C:\Documents and Settings\Administrator\My Documents\My Works\Chi_Paper Work\ROD
\TROS Series 2\TRoS..ACSR\APEX ACSR model>vxp ARINC653basedTasks.acsr
:-) SYS!
(x) (1)  --(tau,(7+1))@ExSTART_PI-->  (x)
[(x)] :-0 show stats
State machine contains 153 reachable states (0 deadlocked), 153 edges.
Edges represent 120 timed transitions,
                27 internal actions, and
                6 external untimed actions.
LTS is non-Zeno.
6 non-deadlocked states are capable of stopping the clock.
Time to compute LTS: 0 seconds user time; 0 seconds system time.
[(x)] :-0
```