2010 Summer ROSAEC Center Workshop

실시간 플랫폼을 위한 정형모델

2010. 8. 24

김진현

고려대학교 정형기법 연구실

목차

- 소개
  - 왜 플랫폼에 대한 모델이 필요한가?
- 우리의 접근
  - 상태 기반 모델로...
  - 프로세스 대수 기반 모델로...
- 결론

(SW) 플랫폼

- 다양화
- 다변화
- 표준화

Android 2.2

iOS 4.0.2

OSEK VDX

ARINC Specification 653P1-2

653P1-2 Avionics Application Software Standard Interface, Part 1 - Required Services

애플리케이션

- 같은 기능이지만 여러 플랫폼에 포팅된다.
- 플랫폼의 제약사항을 만족시킬 필요가 있다.

우리의 접근

"따라서, 플랫폼의 행위 모델을
만들어 애플리케이션의 행위를
분석한다."

우리의 접근

- 상태 기반 모델 : 상태차트
- 프로세스 대수 모델 : CCS 기반의 Algebra of Communicating Shared Resources (ACSR)

플랫폼 제약사항

- 공간적 제약사항
- 시간적 제약사항

실시간 SW 시스템

"애플리케이션의 실시간 성질은 애플리케이션의 요구사항에서 근거하지만, 플랫폼의 제약사항에 종속된다."

SW 시스템 정확성

“제어 흐름의 정확성 역시 데이터 흐름의 정확성 만큼 중요하다.”

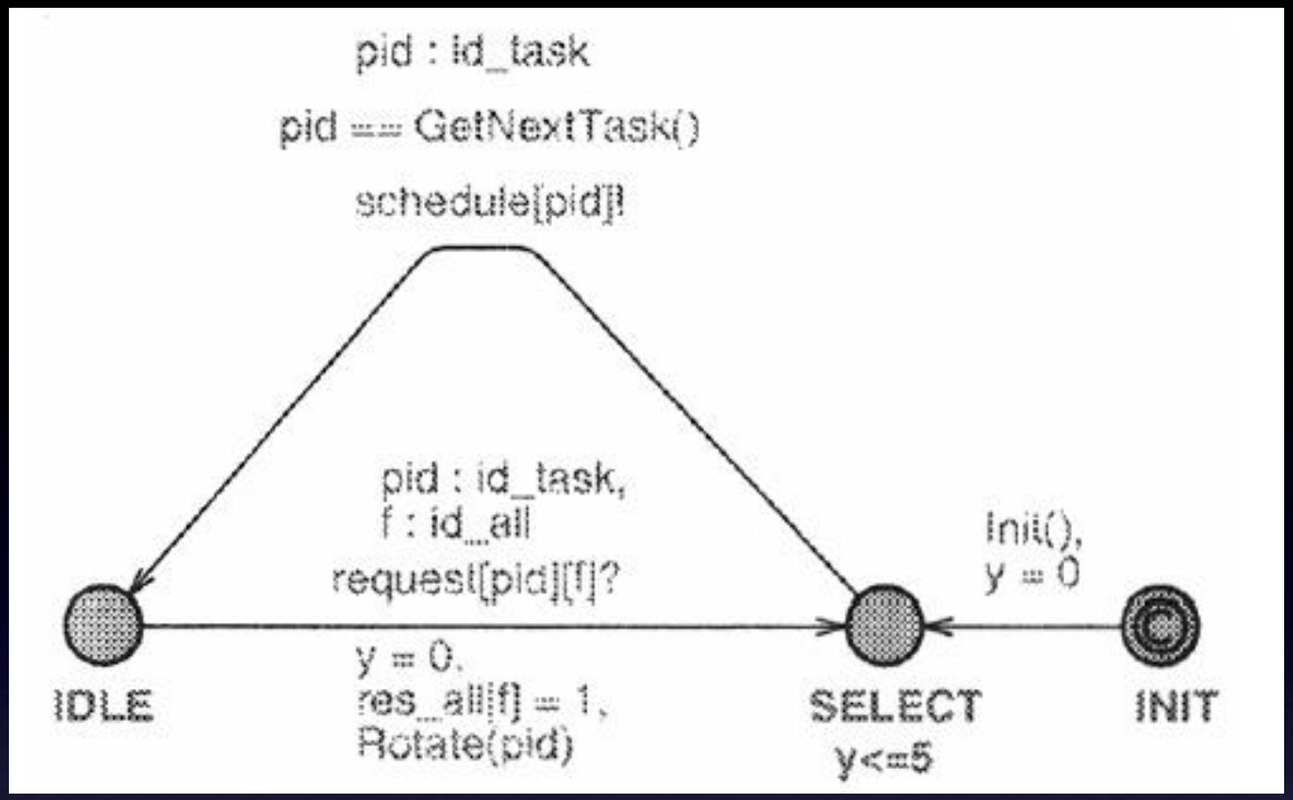
소프트웨어 제어흐름

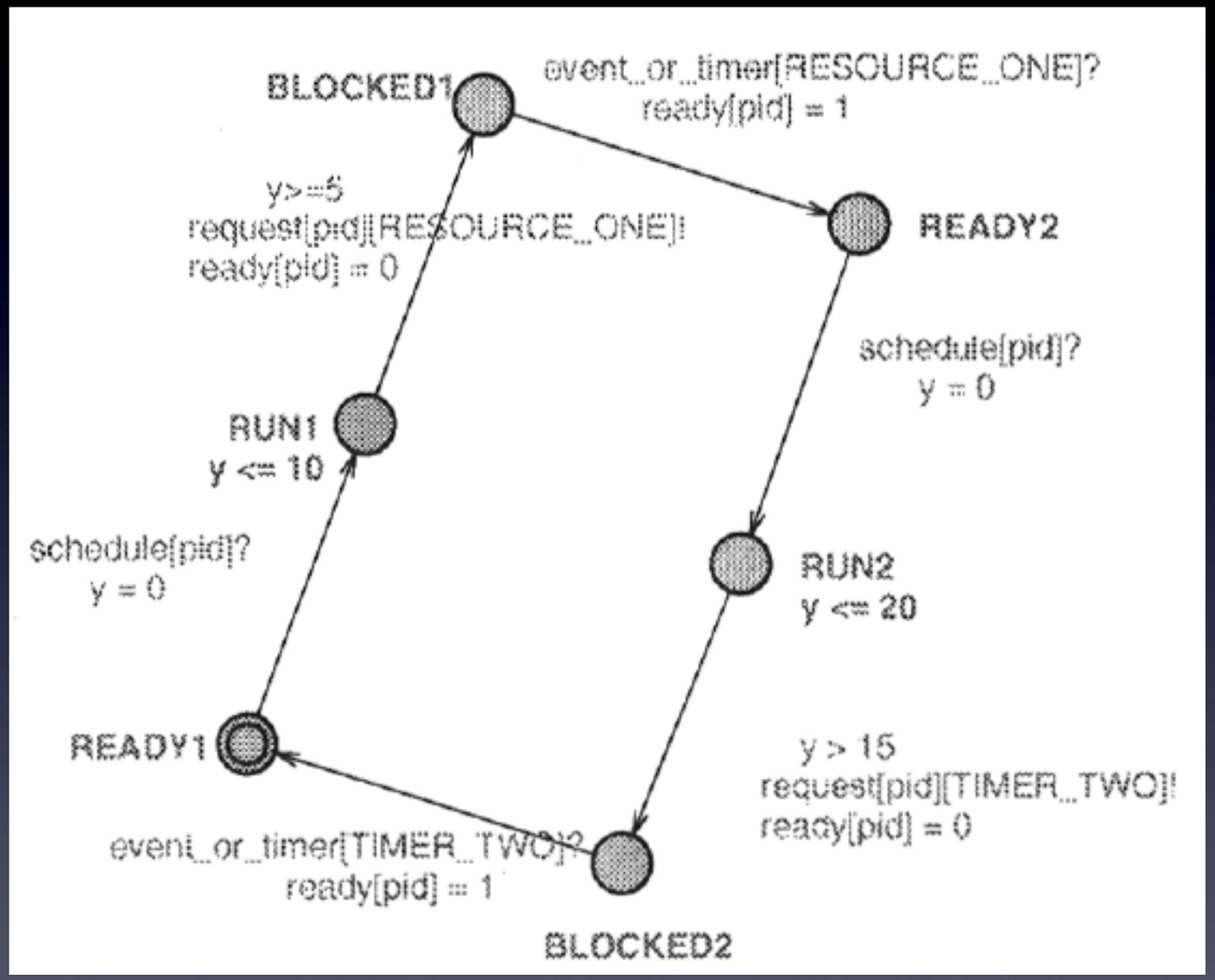
- 프로세스 제어
- 통신, 동기화
- 스케줄링

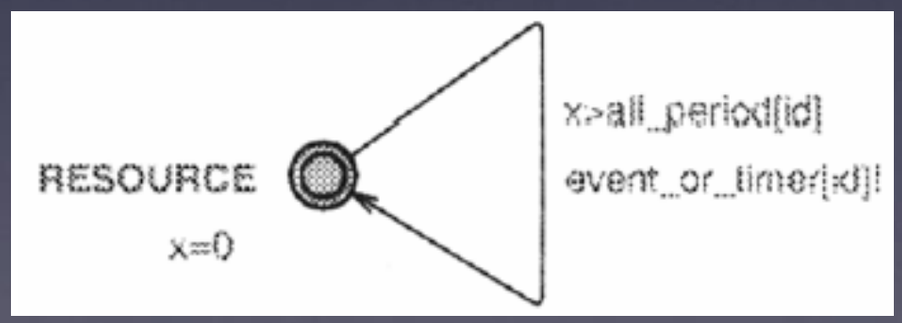
상태기반 언어를 통한 접근

(가장 깊은) 관련 최근 연구

- Tudor Zaharia and Piroska Haller, “Formal Verification and Implementation of Real Time Operating System Based Applications”, 4th International Conference on Intelligent Computer Communication and Processing, 2008. ICCP, 2008.


pid : id_task
pid == GetNextTask()
schedule[pid]!
pid : id_task,
f : id_all
request[pid][f]?
y = 0,
res_all[f] = 1,
Rotate(pid)
Init(),
y = 0
IDLE
SELECT
y<=5
INIT


BLOCKED1
event_or_timer[RESOURCE_ONE]?
ready[pid] = 1
READY2
y>=5
request[pid][RESOURCE_ONE]!
ready[pid] = 0
schedule[pid]?
y = 0
RUN1
y <= 10
schedule[pid]?
y = 0
RUN2
y <= 20
READY1
y > 15
request[pid][TIMER_TWO]!
ready[pid] = 0
event_or_timer[TIMER_TWO]?
ready[pid] = 1
BLOCKED2


RESOURCE
x=0
x>all_period[id]
event_or_timer[id]!

플랫폼에 대한 관점

“플랫폼, 특히 운영체제는 자원 관리자이다.”

자원 지향 모델

$Step1)\ M_{Platform} \models Const_{Platform}$

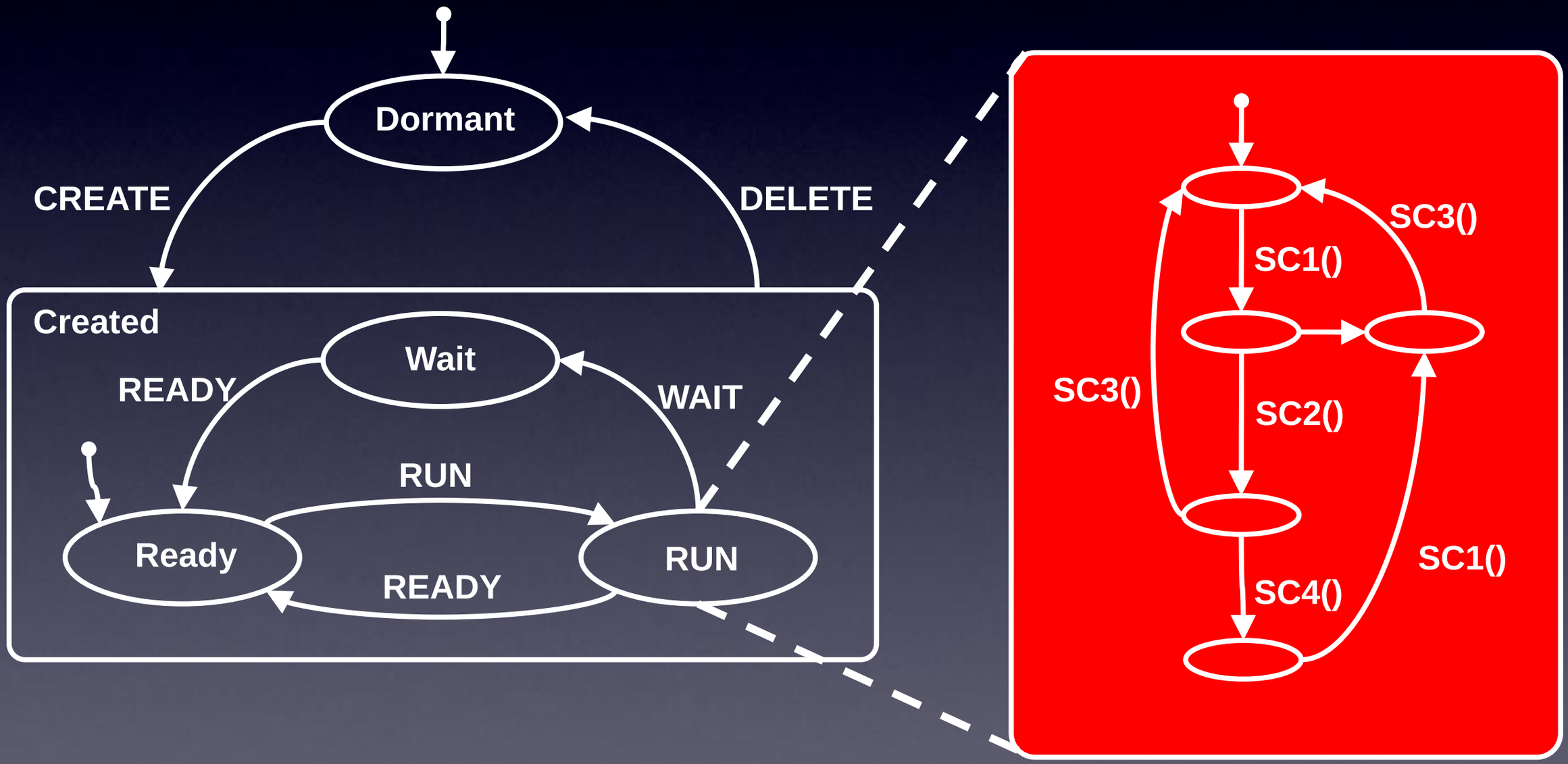
$Step2)\ M_{App} \models Req_{Functionality}$

$Step3)\ M_{App} \parallel M_{Platform} \models Req_{Functionality}$

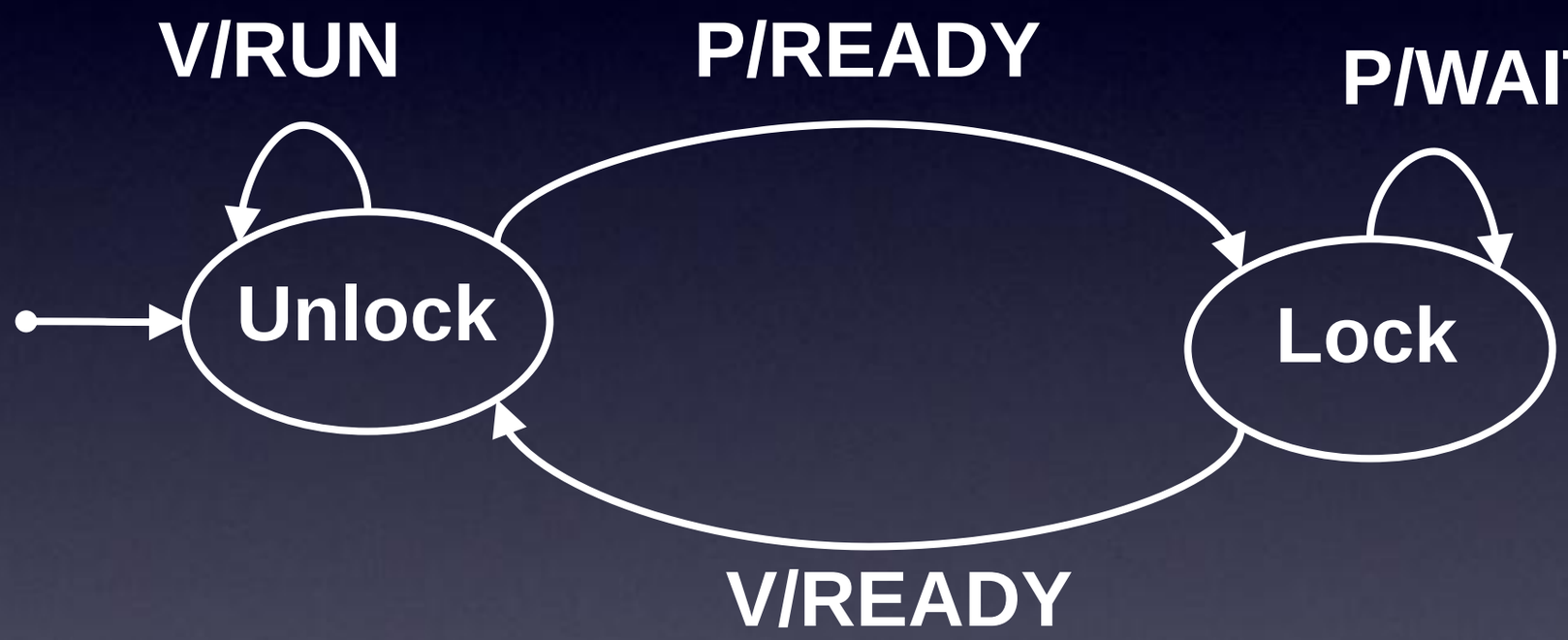
프로세스 행위 (M_{App})

"플랫폼과 다른 프로세스와 상호작용
상에서
프로세스 관점에서의 취해지는 반응"

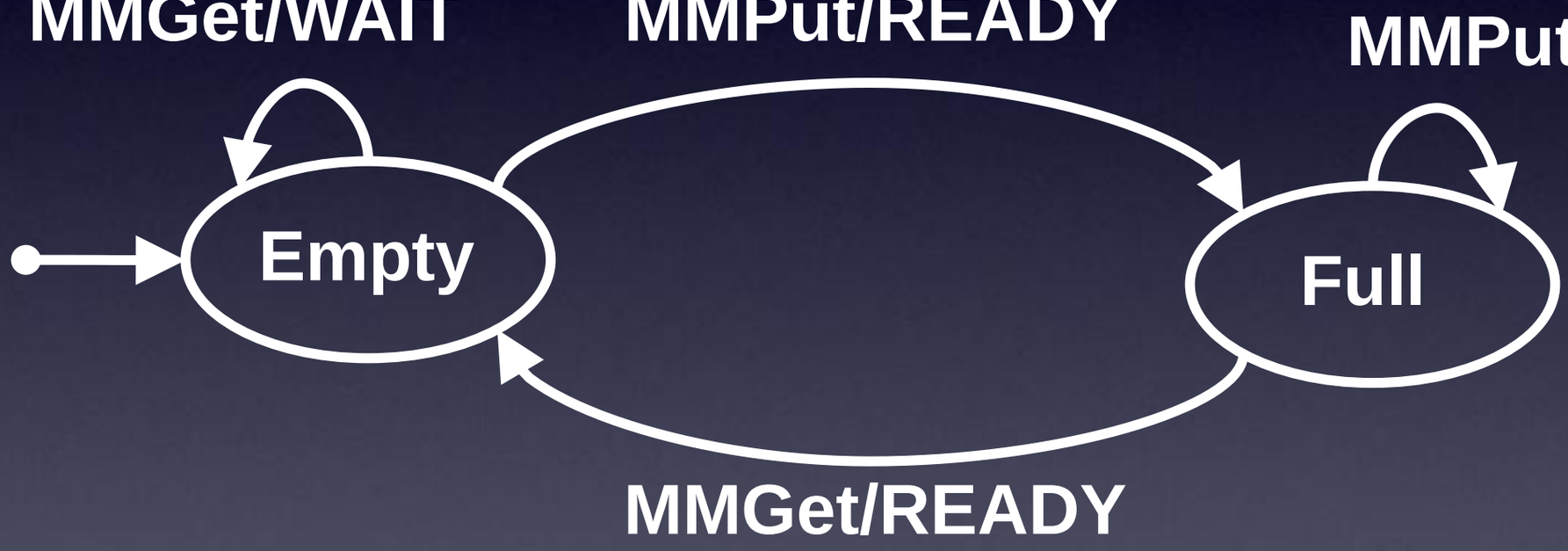
프로세스 행위 모델


Dormant
CREATE
DELETE
Created
Wait
READY
WAIT
RUN
Ready
RUN
READY
SC3()
SC1()
SC3()
SC2()
SC1()
SC4()

플랫폼 모델(세마포어)



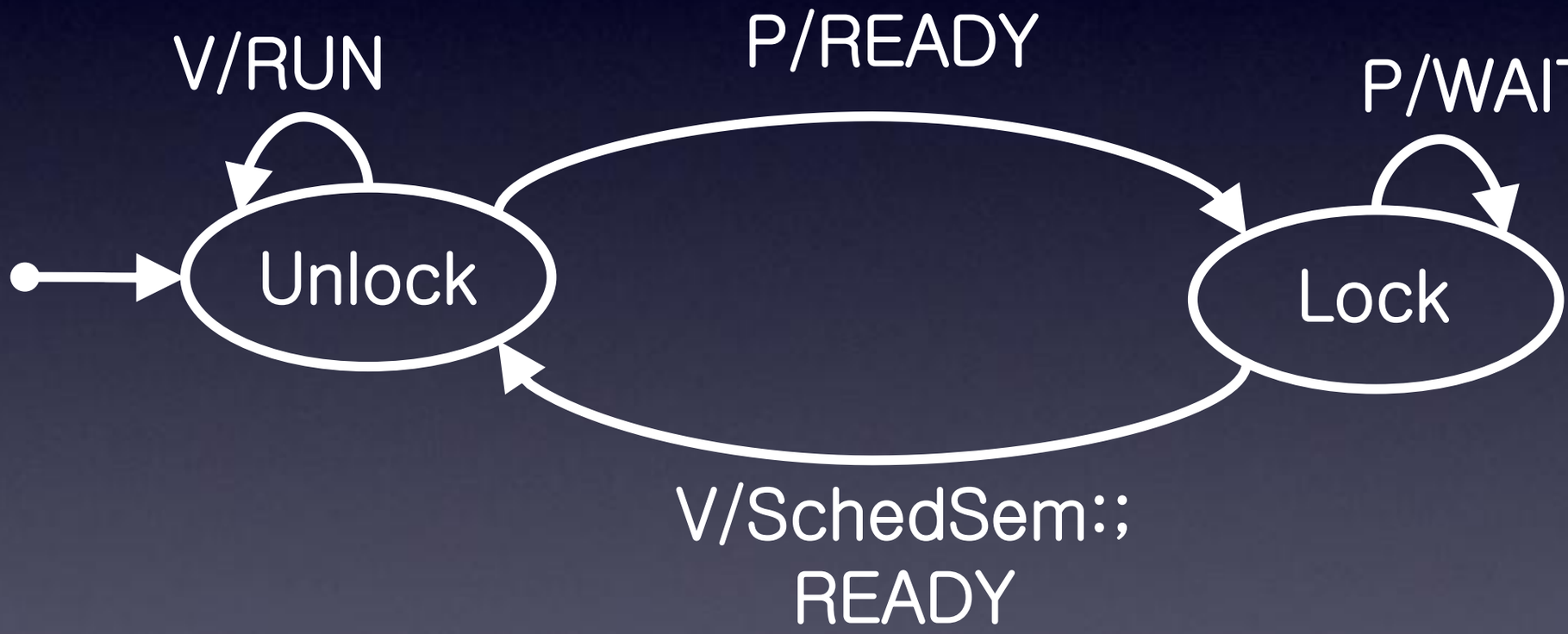
메시지 큐



하지만…

- 여러 개의 프로세스 일 때의 상황을 고려하지 않았다.
- 따라서, 다음 두 가지 스케줄링을 고려한다.
  - CPU 이외의 자원에 대한 스케줄링
  - 프로세스 스케줄링

세마포어



메시지 메일박스

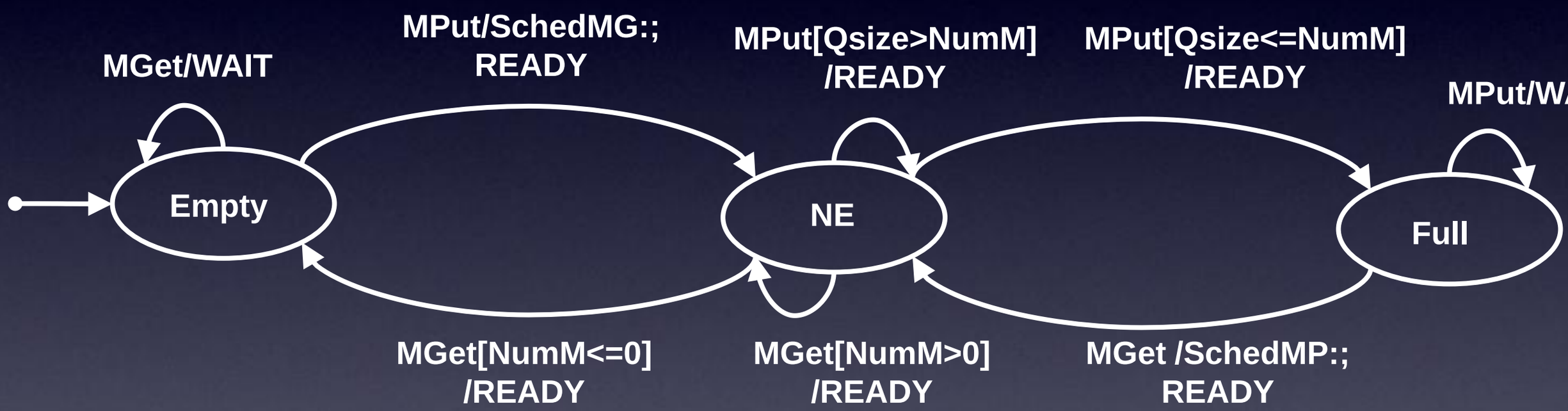
MMPut/SchedMMGet:;
READY

MMGet/WAIT

MMPut/WAIT

MMGet/SchedMMPut:;
READY

메시지 메일박스


MGet/WAIT
MPut/SchedMG:;
READY
MPut[Qsize>NumM]
/READY
MPut[Qsize<=NumM]
/READY
MPut/WAIT
Empty
NE
Full
MGet[NumM<=0]
/READY
MGet[NumM>0]
/READY
MGet /SchedMP:;
READY

문제점

- 모든 자원에 대한 스케줄링을 일일이 모델링해야 한다.

프로세스 대수 기반
언어를 통한 접근

ACSR

- Algebra of Communicating Shared Resources
- CCS (*Calculus of Communicating Systems*) 기반
- 실시간 시스템 분석

문법

$$
\begin{array}{rcl}
P & ::= & \mathrm{NIL} \mid A : P \mid (a, n).P \mid P + P \mid P \| P \mid \\
 & & P \, \triangle_t^a \, (Q, R, S) \mid [P]_I \mid P \backslash F \mid P \backslash\backslash H \mid rec \; X.P \mid X
\end{array}
$$

의미론

Unprioritized Transition relation

Act $A : P \xrightarrow{A} P$ ActI1 $(\tau, ve) . P \xrightarrow{(\tau,[ve])} P$

ActI2 $(l, ve)?(\underline{x}) . P \xrightarrow{(l,[ve])?(\underline{n})} P[\underline{n}/\underline{x}] \quad (\underline{n} \in \mathbb{Z}^*)$

ActI3 $(l, ve_1)!(\underline{ve_2}) . P \xrightarrow{(l,[ve_1])!(\!|\underline{ve_2}|\!)} P$

Cond $\dfrac{P \xrightarrow{\alpha} P', \ [be] = true}{(be \rightarrow P) \xrightarrow{\alpha} P'}$

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

ParC $\dfrac{P \xrightarrow{(l,m)!(\underline{k})} P', \quad Q \xrightarrow{(l,n)?(\underline{k})} Q'}{P \| Q \xrightarrow{(\tau,m+n)} P' \| Q'}$

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

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

HideT $\dfrac{P \xrightarrow{A} P'}{P \setminus\setminus I \xrightarrow{A'} P' \setminus\setminus I} \quad (A' = \{(r, p) \in A \mid r \notin I\})$

HideI $\dfrac{P \xrightarrow{e} P'}{P \setminus\setminus I \xrightarrow{e} P' \setminus\setminus I}$ **Rec** $\dfrac{P[\underline{k}/\underline{x}] \xrightarrow{\alpha} P'}{C(\underline{k}) \xrightarrow{\alpha} P'} \quad (C(\underline{x}) \stackrel{\text{def}}{=} P)$

Preemption relation

(Preemption Relation) *For two actions, α, β, we say that β preempts α $(\alpha \prec \beta)$, if one of the following cases hold:*

(1) Both α and β are events in $\mathcal{D}_E$, where $\alpha = (a, p)$, $\beta = (a, p')$, and $p < p'$

(2) Both α and β are actions in $\mathcal{D}_R$, where

$$
\begin{array}{l}
(\rho(\beta) \subseteq \rho(\alpha)) \wedge \\
(\forall (r, p) \in \alpha \;.\; (((r, p') \in \beta \Longrightarrow p \leq p') \;\wedge\; ((r, p') \notin \beta \Longrightarrow p = 0))) \wedge \\
(\exists (r, p') \in \beta \; \exists (r, p) \in \alpha \;.\; p < p')
\end{array}
$$

(3) $\alpha \in \mathcal{D}_R$ and $\beta \in \mathcal{D}_E$, with $\beta = (\tau, p)$ and $p > 0$. □

Prioritized Transition Relation

The labelled transition system "$\rightarrow_\pi$" is defined as follows: $P \xrightarrow{\alpha}_\pi P'$ *if and only if*

a) $P \xrightarrow{\alpha} P'$ *is an unprioritized transition, and*

b) There is no unprioritized transition $P \xrightarrow{\beta} P''$ *such that* $\alpha \prec \beta$. □

예

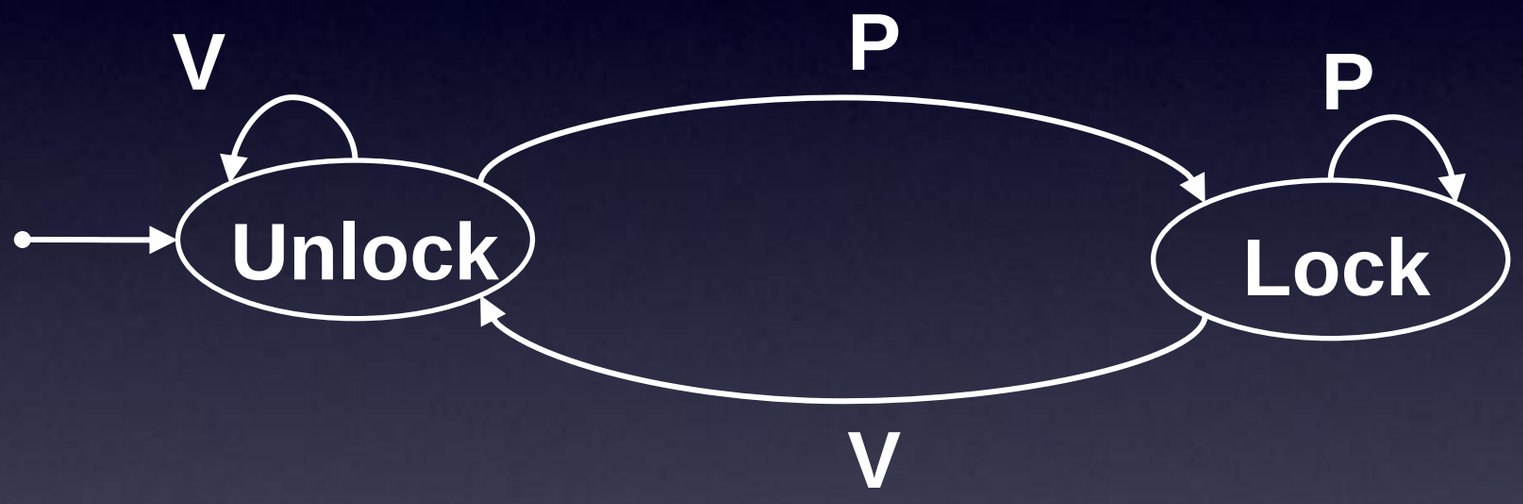
- $\{(r_1, 2), (r_2, 5)\} \prec \{(r_1, 7), (r_2, 5)\}$
  $\{(r_1, 2), (r_2, 5)\} \not\prec \{(r_1, 7), (r_2, 3)\}$

- $\{(r_1, 2), (r_2, 0)\} \prec \{(r_1, 7)\}$
  $\{(r_1, 2), (r_2, 1)\} \not\prec \{(r_1, 7)\}$
  $\{(r_1, 2), (r_2, 5)\} \prec (\tau, 2)$

- $(l, 1) \prec (l, 1), \quad (l_1, 1) \not\prec (l_2, 2) \quad (\tau, 1) \prec (\tau, 2)$

세마포어 모델

Semaphore



Sem1Unlock = {}:Sem1Unlock + (V,1). Sem2Lock ;
Sem2Lock = {}:Sem2Lock + (P,1). Sem1Unlock ;

멀티프로세스

System = Semaphore || Proc1 || Proe2;

Sem1Unlock = {}:Sem1Unlock + (P,1). Sem2Lock ;
Sem2Lock = {}:Sem2Lock + (V,1). Sem1Unlock ;

Proc1 = ….
Proc1Run1 = {}: Proc1Run1 + (‘P,1).Proc1Ready2;
Proc1Ready2 = {}: Proc1Ready2 + (RunProc1,1).Proc1Run3;
Proc1Run3 = {}:Proc1Run + (‘V,1).Proc1Ready2;
Proc1Ready2 = ….

Proc2 = …
Proc2Run1 = {}:Proc2Run1 + (‘P,2).Proc2Ready2;
Proc2Ready2 = {}:Proc2Run2 + (RunProc1,2). Proc2Run3;
Proc2Run3 = {}:Proc2Run3 + (‘V,2).Proc2Ready2;
Proc2Ready2 = ….

System = ProcSched || Semaphore || Proc1 || Proc2 ;

ProcSched = {}:ProcSched + (‘RunProc1,1).ProcRunning + (‘RunProc2,1).ProcRunning ;
ProcRunning = {}: ProcRunning + (‘WaitProc1,1).ProcSched + (‘WaitProc2,1).ProcSched ;

Sem1Unlock = {}:Sem1Unlock + (P,1). Sem2Lock ;
Sem2Lock = {}:Sem2Lock + (V,1). Sem1Unlock ;

Proc1 = {}:Proc1 + (‘WaitProc1,1).Proc1Wait1 + (‘P,1).Proc1Ready2 ;
Proc1Wait1 = {}:Proc1Wait1 + (‘P,1).Proc1Ready2;
Proc1Ready2 = {}:Proc1Ready2 + (RunProc1,1).Proc1Run3;
Proc1Run3 = {}:Proc1Run3 + (‘V,1).Proc1Ready2;
Proc1Ready2 = …

Proc2 = {}:Proc2 + (‘WaitProc2,2).Proc1Wait1 + (‘P,2).Proc2Ready2;
Proc2Wait1 = {}:Proc2Wait1 + (‘P,2).Proc1Ready2;
Proc2Ready2 = {}:Proc2Ready2 + (RunProc1,2).Proc2Run3;
Proc2Run3 = {}:Proc2Run + (‘V,2).Proc1Ready2;
Proc2Ready2 = …

분석

- VERSA로 시스템의 상태를 만들어내고, 교착상태를 찾아낸다.
- 또한 스케줄링 가능성 역시 분석한다.

장점

- 자원을 위한 스케줄링을 모델링할 필요 없다.
- 검증기도 지원한다.

단점

- 현재, 고정 우선 순위 스케줄링을 모델링한다.
- 실제 시스템 모델을 정확히 요약할 방법이 필요하다.
- “프로세스+프로그램”을 요약할 방법이 필요하다.

지금까지…

- ARINC 653 API (APEX)를 ACSR로 모델링하였다.
- “프로세스 + 프로그램”을 모델링할 모델 (TRoS : Timed Resource-oriented Statecharts) 개발하였다.
- 두 모델, “프로세스 + 프로그램” 모델과 플랫폼 모델을 통합하는 의미론을 정의하였다

앞으로…

- 두 결합된 모델을 위한 검증 방법을 개발 중이다.

고맙습니다.

?