Resource-bound process algebras for Schedulability and Performance Analysis of Real-Time and Embedded Systems

Insup Lee[1], Oleg Sokolsky[1], Anna Philippou[2]

[1] RTG (Real-Time Systems Group)
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA

[2] Department of Computer Science
University of Cyprus
Nicosia, CY

Outline

- Real-Time and Embedded systems
- Resource-bound computation
- Resource-bound formalisms
  - ACSR (Algebra of communicating shared resources)
  - Schedulability Analysis Problem
  - PACSR (Probabilistic ACSR)
  - Schedulability analysis for soft real-time systems
  - Design framework for embedded systems
  - P^2ACSR (Probabilistic ACSR with power consumption)
  - Scheduling synthesis and parametric schedulability analysis
  - ACSR-VP (ACSR with Value-Passing)
- Conclusions

Real-time, Embedded Systems

- Difficulties
  - Increasing complexity
  - Decentralized
  - Safety critical
  - End-to-end timing constraints
  - Resource constrained
    - Non-functional: power, size, etc.
- Development of trustworthy (i.e., reliable, robust, safe, secure, etc.) embedded software

Properties of embedded systems

- Adherence to safety-critical properties
- Meeting timing constraints
- Satisfaction of resource constraints
- Confinement of resource accesses
- Supporting fault tolerance
- Domain specific requirements
  - Mobility
  - Software configuration

Real-time Behaviors

- Correctness and reliability of real-time systems depends on
  - Functional correctness
  - Temporal correctness
    - End-to-end temporal constraints
- Factors that affect temporal behavior are
  - Synchronization and communication
  - Resource limitations and availability/failures
  - Scheduling algorithms
  - Interaction with physical world
- An integrated framework to bridge the gap between concurrency theory and real-time scheduling

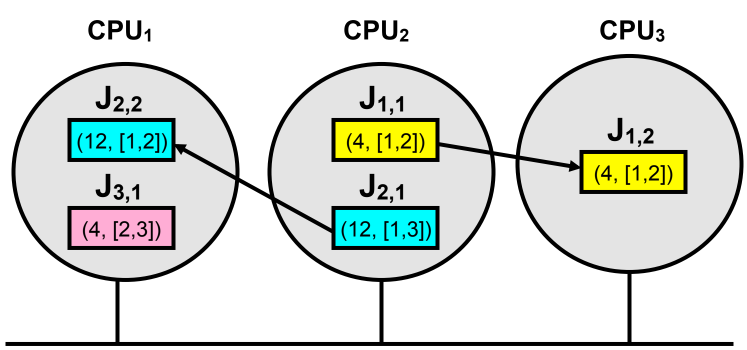
Scheduling Problems

- Priority Assignment Problem
- Schedulability Analysis Problem
  - Compositional analysis
  - Hierarchical system
- Soft timing/performance analysis (Probabilistic Performance Analysis)
- End-to-end Design Problem
  - Parametric Analysis
  - End-to-end constraints, intermediate timing constraints
  - Execution Synchronization Problem
  - Start-time Assignment Problem with Inter-job Temporal Constraints
- Fault tolerance: dealing with failures, overloads

Scheduling Factors

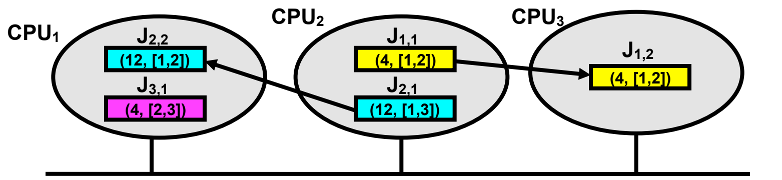
- Static priority vs dynamic priority
  - Cyclic executive, RM (Rate Monotonic)
  - EDF (Earliest Deadline First)
- Priority inversion problem
- Independent tasks vs. dependent tasks
- Single processor vs. multiple processors
- Communication delays
- Uncertainty in execution times
- Resource use tradeoffs
- End-to-end timing requirements

Example: Simple Scheduling Problem

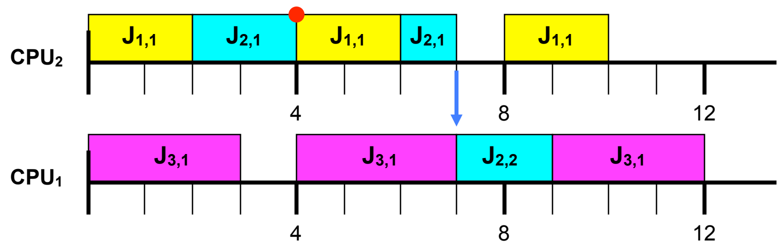


- (period, [e^-, e^+]), where e^- and e^+ are the lower and upper bound of execution time, respectively.
- Goal is to find the priority of each job so that jobs are schedulable
- Considering only worst case leads to scheduling anomaly

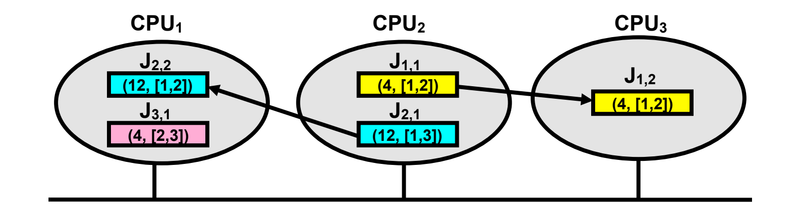
Example (2)



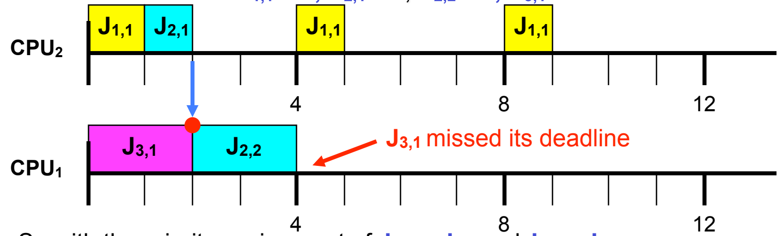
Let $J_{1,1} > J_{2,1}$ and $J_{2,2} > J_{3,1}$
Consider worst case execution time for all jobs, i.e.,
Execution time $E_{1,1} = 2$, $E_{2,1} = 3$, $E_{2,2} = 2$, $E_{3,1} = 3$



Example (3)



With same priorities, $J_{1,1} > J_{2,1}$ and $J_{2,2} > J_{3,1}$
Let execution time $E_{1,1} = 1$, $E_{2,1} = 1$, $E_{2,2} = 2$, $E_{3,1} = 3$



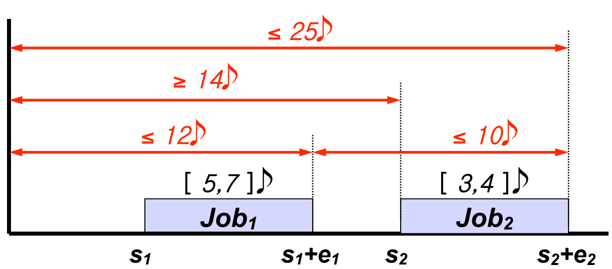
So with the priority assignment of $J_{1,1} > J_{2,1}$ and $J_{2,2} > J_{3,1}$,
jobs cannot be scheduled and scheduling problems are in general NP-hard

End-to-end Design Problem

- Given a task set with end-to-end constraints on inputs and outputs
  - Freshness from input X to output Y (F(Y|X)) constraints: bound time from input X to output Y
  - Correlation between input X1 and X2 (C(Y|X1,X2)) constraints: max time-skew between inputs to output
  - Separation between output Y (u(Y) and l(Y)) constraints: separation between consecutive values on a single output Y
- Derive scheduling for every task
  - Periods, offsets, deadlines
  - priorities
- Meet the end-to-end requirements
- Subject to
  - Resource limitations, e.g., memory, power, weight, bandwidth

Example: Start-time Problem

Start-time Assignment Problem with Inter-job Temporal Constraints



Goal is to statically determine the range of start times for each job so that jobs are schedulable and all *inter-job temporal constraints* are satisfied.

Example: power-aware RT scheduling

- Dynamic Voltage Scaling allows tradeoffs between performance and power consumption
- Problem is how to minimize power consumption while meeting timing constraints.
- Example: three tasks with probabilistic execution time distribution

Task	Worst-case execution time	Period
1	3	8
2	3	10
3	2	14

Our approach and objectives

- Design formalisms for real-time and embedded systems
  - Resource-bound real-time process algebras
  - Executable specifications
  - Logic for specifying properties
- Design analysis techniques
  - Automated verification techniques
  - Parameterized end-to-end schedulability analysis
- Toolset implementation

Resource-bound computation

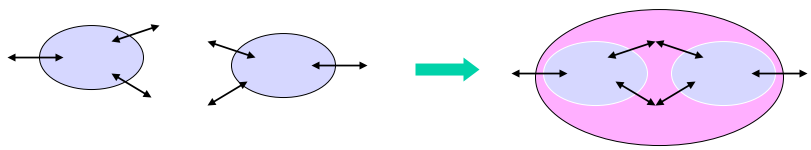
- Computational systems are always constrained in their behaviors
- Resources capture physical constraints
- Resources should be supported as a first-class notion in modeling and analysis
- Resource-bound computation is a general framework of wide applicability

Resources

- Resources capture constraints on executions
- Resources can be
  - Serially reusable:
    - processors, memory, communication channels
  - Consumable
    - power
- Resource capacities
  - Single-capacity resources
  - Multiple-capacity resources
  - Time-sliced, etc.

Process Algebras

- Process algebras are abstract and compositional methodologies for concurrent-system specification and analysis.
- "*Design methodology which systematically allows to build complex systems from smaller ones*" [Milner]



Process Algebras

- A process algebra consists of
  - a set of operators and syntactic rules for constructing processes
  - a semantic mapping which assigns meaning or interpretation to every process
  - a notion of equivalence or partial order between processes
  - a set of algebraic laws that allow syntactic manipulation of processes.
- Ancestors
  - CCS, CSP, ACP,…
  - focus on communication and concurrency

Advantages of Process Algebra

A large system can be broken into simpler subsystems and then proved correct in a modular fashion.

1. A hiding or restriction operator allows one to abstract away unnecessary details.
2. Equality for the process algebra is also a congruence relation; and thus, allows the substitution of one component with another equal component in large systems.

ACSR

ACSR

- ACSR (Algebra of Communicating Shared Resource)
  - A real-time process algebra which features discrete time, resources, and priorities
  - Timeouts, interrupts, and exception handling
  - Two types of actions:
    - Instantaneous events
    - Timed actions

Events

- Events represent non-time consuming activities
  - events are instantaneous: `crash`

  - point-to-point synchronization

Events

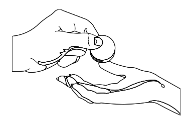
- Events

  - have priorities: **(job,10^{10})**

  - have input and output capabilities

  or **(e, p_1)** **($\bar{e}$, p_2)**

  (e ?, p_1) **(e!, p_2)**



Actions

- Actions represent activities that
  - take time
  - require access to resources
  - each resource usage has priority of access

    $$A = \{(r_1, p_1), (r_2, p_2)\}$$

  - each resource can be used at most once
  - resources of action A: $\rho(A)$
  - idling action: $\emptyset$
- Examples:

  {(cpu,2)}, {(cpu_1,3),(cpu_2,4)},
  {(semaphore,5)}

Syntax for ACSR processes

- Process terms
- Process names

$C \overset{def}{=} P$

$$
\begin{aligned}
P \; ::= \; & NIL \\
| \; & A : P \\
| \; & (a, n).P \\
| \; & P + P \\
| \; & P \,\|\, P \\
| \; & P \Delta_t^a (Q, R, S) \\
| \; & [P]_I \\
| \; & P \backslash F \\
| \; & b \rightarrow P \\
| \; & C
\end{aligned}
$$

Constant and Nil

$C \overset{def}{=} P$

C is a constant that represents the process algebra expression P

P = NIL

P does nothing

Prefix Operators

P = A:Q — P performs timed action A and then behaves as Q

P = (a,n).Q — P performs event (a,n) and then behaves as Q

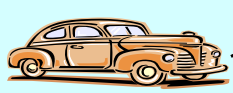
EXAMPLE

$$Operator \stackrel{def}{=} (ring,1).(pickup,1).Talk$$
$$Talk = \{(phone,2)\} : (hangup,1).Operator$$

Choice

P = Q+R — P can choose nondeterministically to behave like Q or R

EXAMPLE



$$CAR \stackrel{def}{=} (goleft,1).CAR' + (goright,1).CAR''$$

Parallel Composition

$P = Q \,||\, R$

P is composed by Q and R that may synchronize on events and must synchronize on timed actions

EXAMPLE

$$Operator \stackrel{def}{=} (ring?,1).\{(phone,2)\} : (hangup?,1).Operator$$

$$Caller \stackrel{def}{=} (ring!,2).\{(phone',3)\} : (hangup!,1).Caller$$

$$Converse \stackrel{def}{=} Operator \,||\, Caller$$

Scope

$P \stackrel{def}{=} Q \;\; \Delta_t^a \; (R,S,T)$

Q may execute for at most t time units. If message a is produced, control is delegated to R, else control is delegated to S. At any time T may interrupt.

EXAMPLE

$$Runner \stackrel{def}{=} Run \;\; \Delta_{10}^{finish} \; (GoForCoffee, GoToWork, BeepedToWork)$$

$$Run \stackrel{def}{=} \{(run,1)\} : Run + finish!.NIL$$

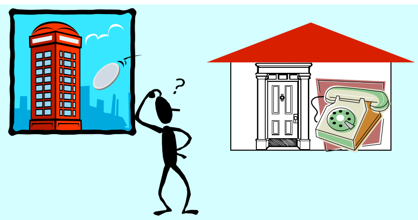
Hiding/Restriction

$P = [Q]_I$ — P behaves just as Q but resources in I are no longer visible to the environment

$P = Q \backslash F$ — P behaves just as Q but labels in F are no longer visible to the environment

EXAMPLE

Caller || PayPhone || $[\text{Home}]_{\text{phone}}$



ACSR semantics

- Gives an unambiguous meaning to language expressions.
- Semantics is operational, given by a set of semantic rules.

ACSR specification → Semantic rules → Labeled transition system

- Example of a labeled transition system:

$$P_0 \xrightarrow{\varnothing} P_1 \xrightarrow{N\,C} P_2 \xrightarrow{\{gate,\, train\}} P_3 \xrightarrow{\{gate,\, train\}} P_4 \xrightarrow{IC} \ldots$$

ACSR semantics

- Two-level semantics:
  - A collection of inference rules gives the unprioritized transition relation

$$P \xrightarrow{\alpha} P'$$

  - A preemption relation on actions and events disables some of the transitions, giving a prioritized transition relation

$$P \xrightarrow{\alpha}_{\pi} P'$$

Unprioritized transition relation

- Prefix operators

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

- Choice

ChoiceL $\dfrac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$

- Parallel

ParIL $\dfrac{P \xrightarrow{(a,p)} P'}{P \| Q \xrightarrow{(a,p)} P' \| Q}$

Unprioritized transition relation (II)

- Resource-constrained execution

ParT $$\frac{P \xrightarrow{A_1} P' \quad Q \xrightarrow{A_2} Q'}{P \,||\, Q \xrightarrow{A_1 \cup A_2} P' \,||\, Q'} \quad \rho(A_1) \cap \rho(A_2) = \varnothing$$

- Priority-based communication

ParCom $$\frac{P \xrightarrow{(a?,p_1)} P' \quad Q \xrightarrow{(a!,p_2)} Q'}{P \,||\, Q \xrightarrow{(\tau, p_1+p_2)} P' \,||\, Q'}$$

- Resource closure

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

Examples

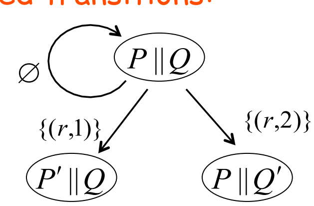
- Resource conflict

$P = \{(r,1)\} : P'$ $\quad Q = \{(r,2)\} : Q'$ $\quad P \,||\, Q \sim NIL$

- Processes must provide for preemption

$P = \{(r,1)\} : P' + \varnothing : P$ $\quad Q = \{(r,2)\} : Q' + \varnothing : Q$

- Unprioritized transitions:



Unprioritized transition relation (III)

ScopeCT $$\frac{P \xrightarrow{A} P'}{P\Delta^a_t(Q,R,S) \xrightarrow{A} P'\Delta^a_{t-1}(Q,R,S)} \quad (t > 0)$$

ScopeCI $$\frac{P \xrightarrow{e} P'}{P\Delta^a_t(Q,R,S) \xrightarrow{e} P'\Delta^a_t(Q,R,S)} \quad (l(e) \neq a, t > 0)$$

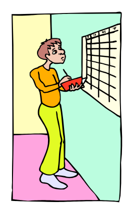
ScopeE $$\frac{P \xrightarrow{(a,n)} P'}{P\Delta^a_t(Q,R,S) \xrightarrow{(\tau,n)} Q} \quad (t > 0)$$

ScopeT $$\frac{R \xrightarrow{\alpha} R'}{P\Delta^a_t(Q,R,S) \xrightarrow{\alpha} R'} \quad (t = 0)$$

ScopeI $$\frac{S \xrightarrow{\alpha} S'}{P\Delta^a_t(Q,R,S) \xrightarrow{\alpha} S'} \quad (t > 0)$$

Example

- A Scheduler



$$Sched = \phi : Sched$$
$$+ (tc,1).\phi^\infty \Delta^y_{t_{\max}} (NIL, \overline{kill}.Sched, rc.Sched)$$

$$\begin{tikzcd}
Sched \arrow[loop below, "\varnothing"] \arrow[r, "(tc{,}1)"] & \phi^\infty : \Delta^y_{t_{\max}}(\ldots) \arrow[r, "\phi"] \arrow[d, "rc"'] & \phi^\infty : \Delta^y_{t_{\max}-1}(\ldots) \arrow[r, "\phi"] \arrow[d, "rc"'] & \ldots \arrow[r, "\phi"] & \phi^\infty : \Delta^y_0(\ldots) \arrow[d, "\overline{kill}"] \\
 & Sched & Sched & & Sched
\end{tikzcd}$$

Preemption relation

- To take priorities into account in the semantics we define the relation α *is preempted by* β : $\alpha \prec \beta$
- An action β preempts an action α iff
  - no lower priorities: $\forall r \in \rho(\alpha), \pi_r(\alpha) \leq \pi_r(\beta)$
  - some higher priorities: $\exists r \in \rho(\beta), \pi_r(\alpha) < \pi_r(\beta)$
  - it contains fewer resources $\rho(\beta) \subseteq \rho(\alpha)$

  e.g. $\{(r_1,3),(r_2,5)\} \prec \{(r_1,7),(r_2,5)\}$
- An event preempts another event iff
  - same label, higher priority e.g. $(a!,1) \prec (a!,3)$
- An event preempts an action iff
  - τ with non-zero priority preempts all actions e.g. $\{(r,4)\} \prec (\tau,1)$

Prioritized transition relation

- We define

  $$P \xrightarrow{\alpha}_{\pi} P'$$

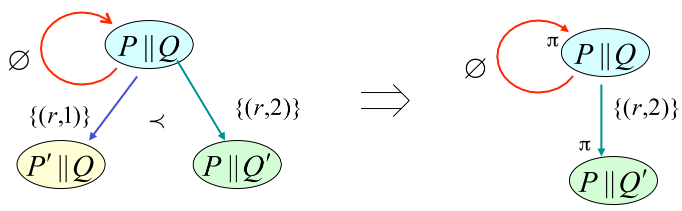
  when
  - there is an unprioritized transition

    $$P \xrightarrow{\alpha} P'$$
  - there is no $P \xrightarrow{\beta} P''$ such that $\alpha \prec \beta$
- Compositional

Example

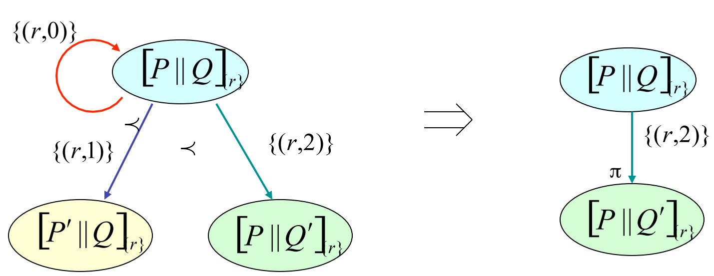
- Unprioritized and prioritized transitions:

$$P = \{(r,1)\} : P' + \varnothing : P \qquad Q = \{(r,2)\} : Q' + \varnothing : Q$$



Example (cont.)

- Resource closure enforces progress



Compositionality of preemption relation

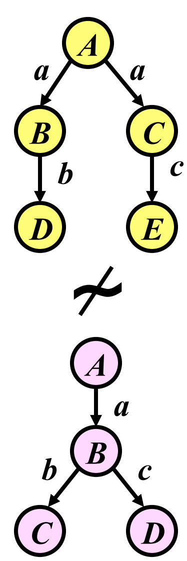
- Given

$$P_1 = (a,2).S_1 + (b,1).S_2$$
$$P_2 = (a,2).S_1$$
$$Q_1 = (\overline{a},3).T_1 + (\overline{b},5).T_2$$
$$Q_2 = (\overline{a},3).T_1 + (\overline{b},2).T_2$$
$$R_1 = (b,2).S_1 + (b,1).S_2$$
$$R_2 = (b,2).S_1$$

- Given P_1 and P_2, can they be treated as equivalent? That is, for all Q, $P_1 \,||\, Q = P_2 \,||\, Q$?
- How about R_1 and R_2?

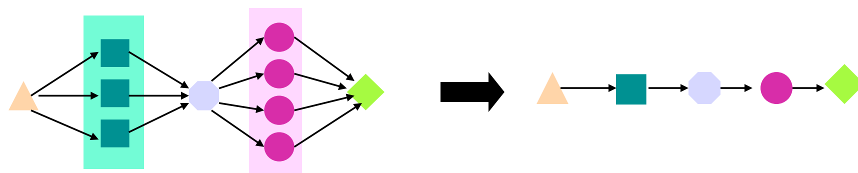
Bisimulation

- Observational equivalence is based on the idea that two equivalent systems exhibit the same behavior at their interfaces with the environment.
- This requirement was captured formally through the notion of bisimulation, a binary relation on the states of systems.
- Two states are bisimilar if for each single computational step of the one there exists an appropriate matching (multiple) step of the other, leading to bisimilar states.



Prioritized strong equivalence

- An equivalence relation is congruence when it is preserved by all the operators of the language.
- This implies that replacement of equivalent components in any complex system leads to equivalent behavior.



- Strong bisimulation $\sim_\pi$ over $P \xrightarrow{\alpha}_\pi P'$ is a congruence relation with respect to the ACSR operators.

Equational Laws

- Equational laws are a set of axioms on the syntactic level of the language that characterize the equivalence relation.
- They may be used for manipulating complex systems at the level of their syntactic (ACSR) description.
- There is a set of laws that is complete for finite state ACSR processes:

$$P + NIL = P \qquad P + P = P$$

$$P + Q = Q + P \qquad (P \parallel Q) \parallel R = P \parallel (Q \parallel R)$$

$$\ldots$$

Equational Laws

- ACSR-specific laws for scope and resource closure:

$$
\begin{array}{rcll}
A{:}P\Delta_t^a(Q,R,S) & = & A{:}\left(P\Delta_{t-1}^a(Q,R,S)\right)+S & \text{if } t>0 \\
e.P\Delta_t^a(Q,R,S) & = & e.\left(P\Delta_t^a(Q,R,S)\right)+S & \text{if } t>0 \wedge \overline{l(e)} \neq a \\
e.P\Delta_t^a(Q,R,S) & = & \left(\tau,\pi(e)\right).Q+S & \text{if } t>0 \wedge \overline{l(e)} = a \\
P\Delta_0^a(Q,R,S) & = & R & \\
\left[A_1:P\right]_I & = & (A_1 \cup A_2):P & A_2=\left\{(r,0) \mid r \in I-\rho(A_1)\right\} \\
\left[e.P\right]_I & = & e.\left[P\right]_I &
\end{array}
$$

Laws (1)

Choice(1)	$P + \mathrm{NIL} = P$
Choice(2)	$P + P = P$
Choice(3)	$P + Q = Q + P$
Choice(4)	$(P + Q) + R = P + (Q + R)$
Choice(5)	$\alpha P + \beta Q = \beta Q \quad \text{if } \alpha \prec \beta$
Par(1)	$P \parallel Q = Q \parallel P$
Par(2)	$(P \parallel Q) \parallel R = P \parallel (Q \parallel R)$
Par(3)	$(\sum_{i \in I} A_i : P_i + \sum_{j \in J} e_j.Q_j) \parallel (\sum_{k \in K} B_k : R_k + \sum_{l \in L} f_l : S_l)$

$$
= \left[
\begin{array}{l}
\displaystyle\sum_{\substack{i \in I, k \in K, \\ \rho(A_i) \cap \rho(B_k) = \emptyset}} (A_i : B_k) : (P_i \parallel R_k) \\
\displaystyle + \sum_{j \in J} e_j \cdot (Q_j \parallel (\sum_{k \in K} B_k : R_k + \sum_{l \in L} f_l \cdot S_l)) \\
\displaystyle + \sum_{l \in L} f_l \cdot ((\sum_{i \in I} A_i : P_i + \sum_{j \in J} e_j \cdot Q_j) \parallel S_l) \\
\displaystyle \sum_{\substack{j \in J, l \in L, \\ l(e_j) = \overline{l(f_l)}}} (\tau, \pi(e_j) + \pi(f_l)).(Q_j \parallel S_l)
\end{array}
\right.
$$

Laws (2)

Scope(1) $A:P\Delta_t^b(Q,R,S) = A:(P\Delta_{t-1}^b(Q,R,S)) + S$ if $t > 0$

Scope(2) $e.P\Delta_t^b(Q,R,S) = e.(P\Delta_{t-1}^b(Q,R,S)) + S$ if $t > 0 \wedge \overline{l(e)} \neq b$

Scope(3) $e.P\Delta_t^b(Q,R,S) = (\tau,\pi(e)).Q + S$ if $t > 0 \wedge \overline{l(e)} = b$

Scope(4) $P\Delta_0^b(Q,R,S) = R$

Scope(5) $(P_1 + P_2)\Delta_t^b(Q,R,S) = P_1\Delta_t^b(Q,R,S) + P_2\Delta_t^b(Q,R,S)$

Scope(6) $\mathrm{NIL}\Delta_t^b(Q,R,S) = S$ if $t > 0$

Res(1) $\mathrm{NIL} \setminus F = \mathrm{NIL}$

Res(2) $(P+Q)\setminus F = (P\setminus F) + (Q \setminus F)$

Res(3) $(A:P)\setminus F = A:(P\setminus F)$

Res(4) $((a,n).P)\setminus F = (a,n).(P\setminus F)$ if $a,\overline{a} \notin F$

Res(5) $((a,n).P)\setminus F = \mathrm{NIL}$ if $a,\overline{a} \in F$

Res(6) $P\setminus E \setminus F = P \setminus E \cup F$

Res(7) $P \setminus \emptyset = P$

Laws (3)

Close(1) $[\mathrm{NIL}]_I = \mathrm{NIL}$

Close(2) $[P+Q]_I = [P]_I + [Q]_I$

Close(3) $[A_1 : P]_I = (A_1 \cup A_2):[P]_I$ where $A_2 = \{(r,0) \mid r \in I - \rho(A_I)\}$

Close(4) $[e.P]_I = e.[P]_I$

Close(5) $[[P]_I]_J = [P]_{I\cup J}$

Close(6) $[P]_\emptyset = P$

Close(7) $[P\setminus E]_I = [P]_I \setminus E$

Rec(1) $rec\ X.P = P[rec\ X.P / X]$

Rec(2) If $P = Q[P/X]$ and X is guarded in Q then $P = rec\ X.Q$

Rec(3) $rec\ X.(P + \sum_{i\in I}[X\setminus E_i]_{U_i}) = rec\ X.(\sum_{J\subseteq I}[P\setminus E_J]_{U_J})$

where $E_J = \bigcup_{i\in J} E_i, U_J = \bigcup_{i\in J} U_i$, I is finite and X is guarded in P

Soundness of the laws

- **Theorem:**

  if $P=Q$ then $P \sim_\pi Q$

- Proof approach:
  - Construct the set of prioritized derivations for each P
  - Prove that if $P=Q$, then the sets of derivations are the same

Completeness of the laws

- **Theorem:**

  if P and Q are finite-state processes and $P \sim_\pi Q$ then P=Q

Schedulability Analysis

Schedulability Analysis

- Can all real-time tasks meet their deadlines?
- Factors include
  - Delay caused by synchronization between tasks
  - Delay caused by precedence between tasks
  - Delay caused by resource constraints
  - Scheduling disciplines and synchronization protocols

Outline

- ACSR-VP: ACSR with value-passing and dynamic priorities
- Specifying real-time systems using ACSR-VP
  - Specifying task models
  - Specifying scheduling disciplines
- Analyzing real-time systems using bisimulation
  - Specification correctness
  - Schedulability analysis
- Schedulability analysis using VERSA (ACSR Toolkit)

ACSR (Algebra of Communicating Shared Resources)

- A timed process algebra based on CCS with notions of time, resources and priorities
- Discrete time and dense time
- Static priorities
- Actions: Instantaneous Events + Timed Actions
  - Timed action: a set of (resource, priority) pairs
    $\{(cpu, 4),(data, 3)\}$, $\{(cpu_1, 2),(cpu_2, 3)\}$, $\emptyset$
  - Instantaneous event: (event, priority) pair
    $(signal, 2)$, $(chan, 2)$ $(\tau, 3)$
- Real-time operators for timeout, interrupt, exception
- Graphical specification language (GCSR)
- Toolkit (VERSA)
- No value passing communication, no variables for priorities

ACSR-VP (ACSR with Value Passing)

- Extends ACSR with variables and value passing communications
- Values can be specifies using expressions
  - Timed Actions:
    $\{(cpu, x), (data, y+1)\}$
  - Instantaneous events:
    $(signal!8, x)$ – output
    $(chan?y, 2)$ – input
- Dynamic priorities
- Exchange priority information without global variables

ACSR-VP Syntax

P	$::=$	NIL	process that does nothing
	$\vert$	$A:P$	timed action prefix
	$\vert$	$e.P$	instantaneous event prefix
	$\vert$	$be \rightarrow P$	conditional process (be : boolean expression)
	$\vert$	$P_1 + P_2$	choice
	$\vert$	$P_1 \parallel P_2$	parallel composition
	$\vert$	$[P]_I$	resource close
	$\vert$	$P \backslash F$	event restriction
	$\vert$	$P \backslash\backslash I$	resource hiding
	$\vert$	$C(\underline{x})$	process name defined to be a process $C(\underline{x}) = P$

ACSR-VP Example

Preemptable and Non-preemptable Jobs

- Both jobs execute *c* time units on *cpu* with priority π
- Non-preemptable job: once it acquires *cpu*, it executes to completion

$$\mathrm{Job}_1 \stackrel{def}{=} \varnothing : \mathrm{Job}_1 + \mathrm{Exec}_1(0)$$
$$\mathrm{Exec}_1(s) \stackrel{def}{=} (s < c) \rightarrow \{(cpu,\pi)\} : \mathrm{Exec}_1(s+1)$$

- Preemptable job: its execution can be preempted by actions on *cpu* of other jobs with higher priorities

$$\mathrm{Job}_2 \stackrel{def}{=} \varnothing : \mathrm{Job}_2 + \mathrm{Exec}_2(0)$$
$$\mathrm{Exec}_2(s) \stackrel{def}{=} (s < c) \rightarrow \{(cpu,\pi)\} : \mathrm{Exec}_2(s+1) + \varnothing : \mathrm{Exec}_2(s)$$

5/27/08 Korea University 59

Unprioritized Operational Semantics

Act	$\mathrm{A} : \mathrm{P} \xrightarrow{\mathrm{A}} P$
ActI1	$(l?\langle \underline{x} \rangle, ve).P \xrightarrow{(l?\langle \underline{n} \rangle, [ve])} P[\underline{n}/\underline{x}]$
ActI2	$(l!\langle \underline{ve_2} \rangle, ve_1).P \xrightarrow{(l!\langle [\underline{ve_2}] \rangle, [ve_1])} P$
ActI3	$(\tau, ve).P \xrightarrow{(\tau, [ve])} P$
ParT	$\dfrac{P \xrightarrow{A_1} P', Q \xrightarrow{A_2} Q'}{P \Vert Q \xrightarrow{A_1 \cup A_2} P' \Vert Q'} (\rho(A_1) \cap \rho(A_2) = \varnothing$
ParC2	$\dfrac{P \xrightarrow{(l!\langle \underline{k} \rangle, m)} P', Q \xrightarrow{(l?\langle \underline{k} \rangle, n)} Q'}{P \Vert Q \xrightarrow{(\tau, m+n)} P' \Vert Q'}$

5/27/08 Korea University 60

Unprioritized Operational Semantics

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

CloseI $\dfrac{P \xrightarrow{e} P'}{[P]_I \xrightarrow{e} [P']_I}$

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

HideI $\dfrac{P \xrightarrow{e} P'}{P \backslash\backslash I \xrightarrow{e} P \backslash\backslash I}$

Preemption

A preemption relation is defined for two any actions α and β, denoted $\alpha \prec \beta$, read β preempts α.

Examples:

- $\{(r_1,2),(r_2,5)\} \prec \{(r_1,7),(r_2,5)\}$
- $\{(r_1,2),(r_2,5)\} \not\prec \{(r_1,7),(r_2,3)\}$
- $\{(r_1,2),(r_2,0)\} \prec \{(r_1,7)\}$
- $\{(r_1,2),(r_2,1)\} \not\prec \{(r_1,7)\}$
- $(\mathrm{a},2) \prec (a,5)$
- $(\mathrm{a},1) \not\prec (b,2)$
- $(\tau,1) \prec (\tau,2)$
- $\{(r_1,2),(r_2,5)\} \prec (\tau,2)$

Prioritized Operational Semantics

The operational semantics of ACSR-VP, the *prioritized transition relation* $\xrightarrow{\alpha}_{\pi}$ is defined as follows:

$$P \xrightarrow{\alpha}_{\pi} P' \text{ iff } \ (1)\ P \xrightarrow{\alpha} P'$$
$$(2) \text{ there is no } P \xrightarrow{\beta} P'' \text{ such that } \alpha \prec \beta$$

Example: $P \overset{def}{=} \{(cpu,2)\}:P_1 + \{(cpu,3)\}:P_2$

- Unprioritized transition : $\begin{cases} P \xrightarrow{\{(cpu2)\}} P_1 \\ P \xrightarrow{\{(cpu3)\}} P_2 \end{cases}$
- Prioritized transition : $P \xrightarrow{\{(cpu3)\}}_{\pi} P_2$

Modeling a Real-Time System

- A real-time system consists of a set of tasks running in parallel under a specific scheduling discipline
- A task is a process composed of a sequence of jobs executed serially
- A task can be
  - Independent or dependent
  - Preemptable or non-preemptable
  - Periodic or aperiodic
- Possible timing constraints of a task are:

b	Starting time
c, d	Execution time and deadline
p	Period for periodic task
p_1, p_2	Minimum and maximum inter-arrival times for aperiodic task

Specification of a real-Time System

A real-time system is specified by the process **RTS**:

$$\mathrm{RTS} \stackrel{def}{=} \left[\mathrm{T}_1 \parallel \mathrm{T}_2 \parallel \cdots \parallel \mathrm{T}_n\right]_R$$

Tasks are specified by the processes $\mathbf{T}_i$:

$$\mathrm{T}_i \stackrel{def}{=} (\mathrm{Job}_i \parallel \mathrm{Activator}_i) \setminus \{start, end\}$$

- Process $\mathbf{Job}_i$: internal characteristics, e.g.,:
  - resource requirements
  - synchronization
- Process $\mathbf{Activator}_i$: external timing attributes, e.g.,
  - periodic or aperiodic
  - period and deadline
- Events *start, end* are synchronization events:
  - *start*: activate jobs
  - *end*: mark deadlines of jobs – deadlock if unsuccessful

Sample Activators

Activator 1. A periodic task with (b, d, p)

$$\begin{aligned} \mathrm{Activator} &\stackrel{def}{=} \varnothing^{\mathrm{b}} : \mathrm{Activator'} \\ \mathrm{Activator'} &\stackrel{def}{=} (start!,1).\varnothing^{\mathrm{d}} : (end!,2). \\ &\quad \varnothing^{\mathrm{p\text{-}d}} : \mathrm{Activator'} \end{aligned}$$

Activator 2. An aperiodic task with (b, d, p_1, p_2)

$$\begin{aligned} \mathrm{Activator} &\stackrel{def}{=} \varnothing^{\mathrm{b}} : \mathrm{Activator'} \\ \mathrm{Activator'} &\stackrel{def}{=} (start!,1).\varnothing^{\mathrm{d}} : (end!,2). \\ &\quad \varnothing^{\mathrm{p}_1\text{-d}\ldots\mathrm{p}_2-d} : \mathrm{Activator'} \end{aligned}$$

where

$$\varnothing^{n} \stackrel{def}{=} \varnothing : \cdots : \varnothing \qquad \text{(idling for } n \text{ time units)}$$

$$\varnothing^{m..n} \stackrel{def}{=} \varnothing^{m} + \varnothing^{m+1} + \cdots + \varnothing^{n}$$

Sample Jobs

Job 1

- preemptable, independent jobs
  running on *cpu*
  priority π and execution time c:

$$
\begin{aligned}
\text{Job} &\overset{def}{=} \varnothing : \text{Job} + (start?,1).\text{Exec}(0,0) \\
\text{Exec}(s,t) &\overset{def}{=} (s < c) \rightarrow (\{cpu,\pi\}\} : \text{Exec}(s+1,t+1) \\
&\qquad + \varnothing : \text{Exec}(s,t+1)) \\
&\quad + (s = c) \rightarrow \text{Wait} \\
\text{Wait} &\overset{def}{=} \varnothing : \text{Wait} + (end?,1).\text{Job}
\end{aligned}
$$

- *s* for accumulated execution time
- *t* for the elapsed time
- **Job** can response to *end* event only when its current execution is finished

Sample Jobs

Job 2

- nonpreemptable, independent jobs
  on multiprocessors $cpu_1, \ldots, cpu_k$
  with priorities $\pi_1, \ldots, \pi_k$ and execution time c:

$$
\begin{aligned}
\text{Job} &\overset{def}{=} \varnothing : \text{Job} + (start?,1)\,.\text{Exec} \\
\text{Exec} &\overset{def}{=} \sum_{1 \le i \le k} (\{(cpu_i,\pi_i)\}^c : \text{Wait}) \\
\text{Wait} &\overset{def}{=} \varnothing : \text{Wait} + (end?,1)\,.\text{Job}
\end{aligned}
$$

- A job can be executed on any of the processors
- Once a processor is assigned to a job, the job executes on that processor until completion

Sample Jobs

Job 3

- dependent jobs on processor *cpu* with priority π and execution time c
  a single preemptable critical section of length cs on resource *data* (with priority π') after at c' time units execution:

$$
\begin{array}{lcl}
\text{Job} & \stackrel{def}{=} & \varnothing : \text{Job} + (start?,1).\text{Exec}(0,0) \\
\text{Exec} & \stackrel{def}{=} & (\text{s} < \text{c} \wedge \text{s} \neq \text{c}') \rightarrow (\{(\text{cpu},\pi)\} : \text{Exec}(s+1,t+1) \\
 & & \qquad\qquad + \varnothing : \text{Exec}(s,t+1)) \\
 & & +(s = c') \qquad \rightarrow ((p!,0).\text{CS}(s,t) \\
 & & \qquad\qquad + \varnothing : \text{Exec}(s,t+1)) \\
 & & +(s = c) \qquad \rightarrow \text{Wait} \\
\text{Wait} & \stackrel{def}{=} & \varnothing : \text{Wait} + (end?,1).\text{Job} \\
\hline
\text{CS}(s,t) & \stackrel{def}{=} & (s < c'+cs) \rightarrow (\{(cpu,\pi)\} : \text{CS}(s+1,t+1) \\
 & & \qquad\qquad + \varnothing : \text{CS}(s,t+1)) \\
 & & +(s = c'+cs) \rightarrow (v!,0).\text{Exec}(s,t) \\
\hline
\text{P} & \stackrel{def}{=} & (p?,0).\text{V} + \varnothing : \text{P} \\
\text{V} & \stackrel{def}{=} & (v?,0).\text{P} + \varnothing : \text{V}
\end{array}
$$

- P and V operations are modeled by the processes P and V with events (p?,0) and (v?,0)
- When *s* equals c', Exec waits for (p?,0) to enter the critical section CS(s,t)

Scheduling Disciplines

Earliest Deadline First

- Tasks $\text{T}_i \stackrel{def}{=} \text{Job}\,1 + \text{Activator}\,1$
- Priority $\pi_i = \text{d}_{max} - (\text{d}_i - t)$

where $\text{d}_{max} \stackrel{def}{=} (1 + \max\{\text{d}_1, \cdots, \text{d}_n\})$

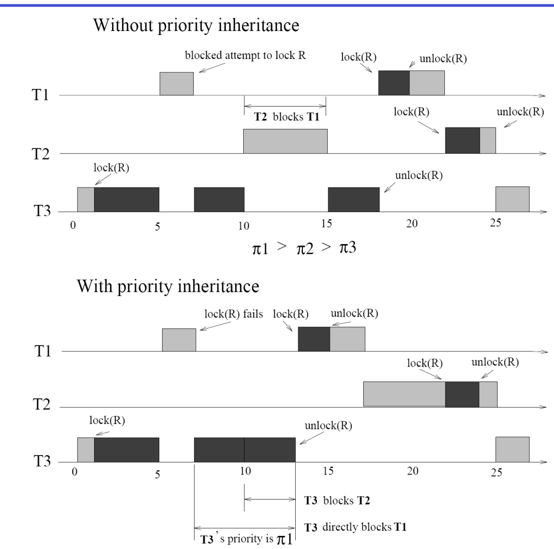
$$
\begin{array}{lcl}
\text{EDFSys} & \stackrel{def}{=} & [\text{T}_1 \parallel \text{T}_2 \parallel \cdots \parallel \text{T}_n]_{\{cpu\}} \\
\text{T}_i & \stackrel{def}{=} & (\text{Job}_i \parallel \text{Activator}_i) \backslash \{start, end\} \\
\hline
\text{Job}_i & \stackrel{def}{=} & \varnothing : \text{Job}_i + (start?,1).\text{Exec}_i(0,0) \\
\text{Exec}_i(s,t) & \stackrel{def}{=} & (s < c_i) \rightarrow \{(cpu, \text{d}_{max} - (\text{d}_i - t))\} \\
 & & \qquad\qquad : \text{Exec}_i(s,t+1) \\
 & & \qquad + \varnothing : \text{Exec}_i(s,t+1) \\
 & & +(s = c) \rightarrow \text{Wait} \\
\text{Wait}_i & \stackrel{def}{=} & \varnothing : \text{Wait}_i + (end?,1).\text{Job}_i \\
\hline
\text{Activator}_i & \stackrel{def}{=} & (start!,1).\varnothing^{\text{d}_i} : (end!,2).\varnothing^{\text{p-d}} : \text{Activator}_i
\end{array}
$$

Other Time-Driven Scheduling Disciplines

Deadline Monotonic	$\pi_i \overset{def}{=} d_{max} - d_i$
Shortest Remaining Time First	$\pi_i \overset{def}{=} c_{max} - (c_i - s)$
Least Laxity First	$\pi_i \overset{def}{=} d_{max} - (d_i - t) - (c_i - s)$

where $c_{max} \overset{def}{=} (1 + \max\{c_1, \cdots, c_n\})$

The Priority Inversion Problem



Task parameters

Resources :	*cpu*	processor			
Constants :	ready time :	$r_1 = 5$	$r_2 = 10$	$r_3 = 0$	
	comp. time :	$c_1 = 6$	$c_2 = 8$	$c_3 = 13$	
	deadline :	$d_1 = 30$	$d_2 = 30$	$d_3 = 30$	
	start time of CS :	$cs_1 = 3$	$cs_2 = 5$	$cs_3 = 1$	
	length of CS :	$c'_1 = 2$	$c'_2 = 2$	$c'_3 = 10$	
	priority :	$\pi_1 = 3$	$\pi_2 = 2$	$\pi_3 = 1$	
	max priority :	$\pi_{max} = 4$			

Priority Inheritance Protocol

$$\mathrm{T}_i \overset{def}{=} \text{Job 3} + \text{Activator 1} + \text{Priority - Passing Events}$$

$$\begin{array}{lcl}
\text{PIPSys} & \overset{def}{=} & \left[\left(\mathrm{T}_1 \| \mathrm{T}_2 \| \mathrm{T}_3 \| \mathrm{P} \right) \backslash \{req, chan, p, v\} \right]_{\{cpu\}} \\
\hline
\mathrm{T}_i & \overset{def}{=} & \left(\mathrm{Job}_i \| \mathrm{Activator}_i \right) \backslash \{start, end\} \\
\hline
\mathrm{Job}_i & \overset{def}{=} & \varnothing : \mathrm{Job}_i + (start?,1).\mathrm{Exec}_i(0,0) \\
\mathrm{Exec}_i(s) & \overset{def}{=} & (s < \mathrm{c}_i \wedge s \neq \mathrm{cs}_i) \rightarrow (\{cpu, \pi_i)\} : \mathrm{Exec}_i(s+1) \\
 & & \qquad + \varnothing : \mathrm{Exec}_i(s)) \\
 & & + (s = \mathrm{cs}_i) \rightarrow ((req!\pi_i, \pi_i).\mathrm{Req}_i(s) + \varnothing : \mathrm{Exec}_i(s)) \\
 & & + (s = \mathrm{c}_i) \rightarrow \text{Wait} \\
\mathrm{Wait}_i & \overset{def}{=} & \varnothing : \mathrm{Wait}_i + (end?,1).\mathrm{Job}_i \\
\hline
\mathrm{Req}_i(s) & \overset{def}{=} & (p!\pi_i, \pi_i).\mathrm{CS}_i(s, \pi_i) + \varnothing : \mathrm{Req}_i(s) \\
\mathrm{CS}_i(s, \pi) & \overset{def}{=} & \quad (s < \mathrm{c'}_i + \mathrm{cs}_i) \rightarrow (\{cpu, \pi)\} : \mathrm{CS}_i(s+1, \pi) \\
 & & \qquad + (chan?new,1).\mathrm{CS}_i(s, new) \\
 & & \qquad + \varnothing : \mathrm{CS}_i(s, \pi) \\
 & & + (s = \mathrm{c'}_i + \mathrm{cs}_i) \rightarrow (v?,1).\mathrm{Exec}_i(s) \\
\hline
\mathrm{Activator}_i & \overset{def}{=} & \varnothing^{\mathrm{r}_i} : (start!,1).\varnothing^{\mathrm{d}_i} : (end!,2).\varnothing^{\infty} \\
\hline
\mathrm{P} & \overset{def}{=} & (p?x,1).\mathrm{V}(x) + (req?x, \pi_{max}).(p?x,1).\mathrm{V}(x) + \varnothing : \mathrm{P} \\
\mathrm{V}(max) & \overset{def}{=} & (v!,1).\mathrm{P} + \varnothing : \mathrm{V}(max) \\
 & & \quad + (req?x,1).((x > max) \rightarrow (chan!x,1).\mathrm{V}(x) \\
 & & \qquad + (x \leq max) \rightarrow \mathrm{V}(max))
\end{array}$$

Parameters of T_i
- pty_i Priority
- c_i Execution time of a job
- $\mathrm{c'}_i$ Time for enetring critical section
- cs_i Execution time in critical section

Traces of tasks

Time	process T_1	process T_2	process T_3	process P
0	$\{\}$	$\{\}$	$start?,\{(cpu,1)\}$	P
1	$\{\}$	$\{\}$	$req!1, p!1, \{(cpu,1)\}$ •	$req?1, p?1$, V(1)
2	$\{\}$	$\{\}$	$\{(cpu,1)\}$ •	V(1)
3	$\{\}$	$\{\}$	$\{(cpu,1)\}$ •	V(1)
4	$\{\}$	$\{\}$	$\{(cpu,1)\}$ •	V(1)
5	$start?,\{(cpu,3)\}$	$\{\}$	$\{\}$	V(1)
6	$\{(cpu,3)\}$	$\{\}$	$\{\}$	V(1)
7	$req!3,\{\}$	$\{\}$	$chan?3,\{(cpu,3)\}$ •	$req?3, chan!3$, V(3)
8	$\{\}$	$\{\}$	$\{(cpu,3)\}$ •	V(3)
9	$\{\}$	$\{\}$	$\{(cpu,3)\}$ •	V(3)
10	$\{\}$	$start?,\{\}$	$\{(cpu,3)\}$ •	V(3)
11	$\{\}$	$\{\}$	$\{(cpu,3)\}$ •	V(3)
12	$\{\}$	$\{\}$	$\{(cpu,3)\}, v?$ •	$v!$, P
13	$p!3,\{(cpu,3)\}$ •	$\{\}$	$\{\}$	$p?3$, V(3)
14	$\{(cpu,3)\}, v?$ •	$\{\}$	$\{\}$	$v!$, P
15	$\{(cpu,3)\}$	$\{\}$	$\{\}$	P
16	$\{(cpu,3)\}$	$\{\}$	$\{\}$	P
17	$\{\}$	$\{(cpu,2)\}$	$\{\}$	P
18	$\{\}$	$\{(cpu,2)\}$	$\{\}$	P
19	$\{\}$	$\{(cpu,2)\}$	$\{\}$	P
20	$\{\}$	$\{(cpu,2)\}$	$\{\}$	P
21	$\{\}$	$\{(cpu,2)\}$	$\{\}$	P
22	$\{\}$	$req!2, p!2, \{(cpu,2)\}$ •	$\{\}$	$req?2, p?2$, V(2)
23	$\{\}$	$\{(cpu,2)\}, v?$ •	$\{\}$	$v!$P
24	$\{\}$	$\{(cpu,2)\}$	$\{\}$	P
25	$\{\}$	$\{\}$	$\{(cpu,1)\}$	P
26	$\{\}$	$\{\}$	$\{(cpu,1)\}$	P

(• : in critical section)

Weak Bisimulation

Def. If $t \in D^*$, then $\hat{t} \in (D - \{\tau\})^*$ is the sequence derived by deleting all occurrences of π from t.

Def. If $t = \alpha_1 \ldots \alpha_n \in D^*$, then $E \overset{t}{\Rightarrow} E'$ if

$$P(\xrightarrow{(\tau,\_)})^* \xrightarrow{\alpha_1} (\xrightarrow{(\tau,\_)})^* \cdots (\xrightarrow{(\tau,\_)})^* \xrightarrow{\alpha_n} (\xrightarrow{(\tau,\_)})^* P',$$

where "_" in $(\tau, \_)$ represents arbitrary integer.

Def. For a given transition system "$\rightarrow$", any binary relation r is a weak bisimulation if, for $(P,Q) \in r$ and for any action $\alpha \in D$,

1. if $P \xrightarrow{\alpha} P'$, then, for some Q', $Q \overset{\alpha}{\Rightarrow} Q'$ and $(P',Q') \in r$, and
2. if $Q \xrightarrow{\alpha} Q'$, then, for some P', $P \overset{\alpha}{\Rightarrow} P'$ and $(P',Q') \in r$.

Def. $\approx_\pi$ is the largest weak bisimulation over "$\rightarrow_\pi$". It is an equivalence relation (though not a congruence) for ACSR.

Analyzing Real-Time Systems in ACSR-VP

- Two types of analyses
  - Validation
  - Schedulability analysis
- Basic idea
  - Checking weak bisimulation $\approx_\pi$
  - Searching deadlocked states
- Practical Issues
  - Ensure that the **EDFS**ys and **PIPS**ys processes are finite state
  - Translate ACSR-VP processes to ACSR processes and use VERSA, the toolkit for ACSR

Validating the EDFSys Specification

Construct a correctness specification, EDFSpec, that is sequential and easy to inspect

Verify that **EDFSys $\approx_\pi$ *EDFSpec***

$$\mathrm{EDFSpec} \stackrel{def}{=} [\mathrm{S}(0,\cdots,0,0)]_{\{cpu\}}$$

$$\mathrm{S}(s_1,t_1,\cdots,s_n t_n) \stackrel{def}{=} \sum_{1\le i\le n} \begin{cases} (s_i = \mathrm{c}_i \wedge t_i = \mathrm{p}_i) \\ \quad \rightarrow (\tau,1).\mathrm{S}(\cdots,s_{i-1},t_{i-1},0,0,s_{i+1},t_{i+1},\cdots) \\ + \ (s_i < \mathrm{c}_i \wedge t_i = \mathrm{d}_i) \\ \quad \rightarrow (\tau,1).\mathrm{NIL} \\ + \ (s_i = \mathrm{c}_i \wedge t_i < \mathrm{p}_i) \\ \quad \rightarrow \varnothing : \mathrm{S}(\cdots,s_{i-1},t_{i-1}+1,s_i,t_i+1,s_{i+1},t_{i+1}+1,\cdots) \\ + \ (s_i < \mathrm{c}_i \wedge t_i < \mathrm{d}_i) \\ \quad \rightarrow \{(cpu,\mathrm{d}_{max} - (\mathrm{d}_i - t))\} \\ \quad\quad : \mathrm{S}(\cdots,s_{i-1},t_{i-1}+1,s_i+1,t_i+1,s_{i+1},t_{i+1}+1,\cdots) \end{cases}$$

Schedulability Analysis

Lemma 1 *If* **EDFSys** *is deadlock free, then it is schedulable.*

Lemma 2 *If*

$$\text{EDFSys} \backslash\backslash \{cpu\} \approx_\pi \varnothing^\infty,$$

then **EDFSys** *is deadlock free.*

Lemma 3 *If* **PIPSys** *is deadlock free, then it is schedulable.*

Lemma 4 *If*

$$\text{PIPSys} \backslash\backslash \{cpu\} \approx_\pi \varnothing^\infty,$$

then **PIPSys** *is deadlock free*

Example 1

- Consider an instance **EDFSys**$_1$ of **EDFSys** where:

  Task T_1: $c_1 = 1$, $d_1 = 2$, $p_1 = 3$

  Task T_2: $c_2 = 2$, $d_2 = 3$, $p_2 = 3$
- The following sufficient condition for schedulability from [Liu and Lay 73] is not satisfied:

$$\frac{c_1}{d_1} + \frac{c_2}{d_2} \leq 1$$

- The following equation $\text{EDFSys} \backslash\backslash \{cpu\} \approx_\pi \varnothing^\infty,$

  is satisfied, i.e., the task system is *schedulable.*
  More specifically, we have

$$\text{EDFSys1} \xrightarrow{(\tau,2)}_\pi \xrightarrow{(\tau,2)}_\pi \xrightarrow{\{(cpu,2)\}}_\pi \xrightarrow{\{(cpu,3)\}}_\pi \xrightarrow{(\tau,3)}_\pi \xrightarrow{\{(cpu,3)\}}_\pi \xrightarrow{(\tau,3)}_\pi \text{EDFSys1}$$

Example 2

- Consider another instance **EDFSys**$_2$ of **EDFSys** where:

  Task T_1: $c_1 = 2$, $d_1 = 2$, $p_1 = 3$

  Task T_2: $c_2 = 2$, $d_2 = 3$, $p_2 = 3$
- The equivalence

$$\mathrm{EDFSys}_2 \setminus\setminus \{cpu\} \approx_\pi \varnothing^\infty,$$

  is *false* and the task system is therefore *not schedulable.*

- More specifically, we have

$$\mathrm{EDFSys2} \xrightarrow{(\tau,2)}_\pi \xrightarrow{(\tau,2)}_\pi \xrightarrow{\{(\mathrm{cpu},2)\}}_\pi \xrightarrow{\{(\mathrm{cpu},2)\}}_\pi \xrightarrow{(\tau,3)}_\pi$$
$$\xrightarrow{\{(\mathrm{cpu},3)\}}_\pi \mathrm{NIL}$$

Summary

- The ACSR paradigm:
  - Formalism for modular specification of real-time systems along with scheduling disciplines
  - Formal characterization of the schedulability analysis in process algebra
- Automated schedulability analysis
  - Provide techniques for detecting timing anomalies before an implementation is developed
  - Integrate into a methodology for engineering reliable real-time systems
- Tools:
  - GCSR (Graphical ACSR)
  - XVERSA: VERSA and GCSR