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]SDRL (Systems Design Research Lab)
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 reliable and robust 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
- Factors that affect temporal behavior are
  - Synchronization and communication
  - Resource limitations and availability/failures
  - Scheduling algorithms
  - End-to-end temporal constraints
- An integrated framework to bridge the gap between concurrency theory and real-time scheduling

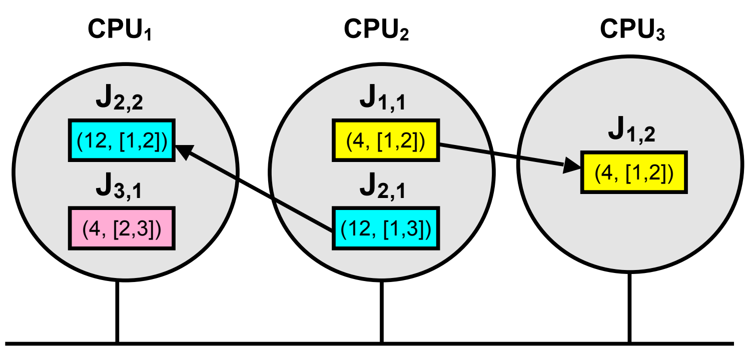
Scheduling Problems

- Priority Assignment Problem
- Schedulability Analysis Problem
- 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

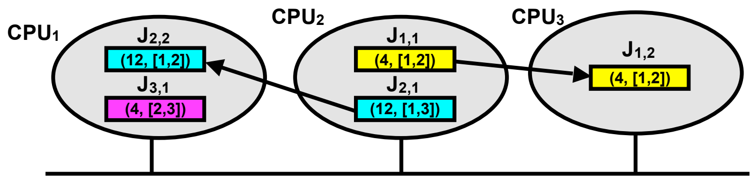
- Static priority vs dynamic priority
  - Cyclic executive, RM (Rate Monotonic), EDF (Earliest Deadline First)
- Priority inversion problem
- Independent tasks vs. dependent tasks
- Single processor vs. multiple processors
- Communication delays
- Uncertainty in execution times
- Resource use tradeoffs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

Example: Simple Scheduling Problem

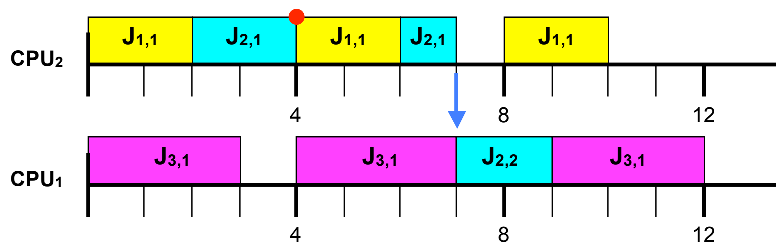


- (period, [e^-, e^+]), where e^- and e^+ are the lower and upper bound of execution time, respectively.
- Goal is to find the priority of each job so that jobs are schedulable
- Considering only worst case leads to scheduling anomaly

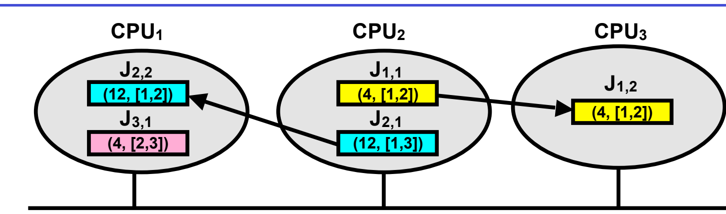
Example (2)



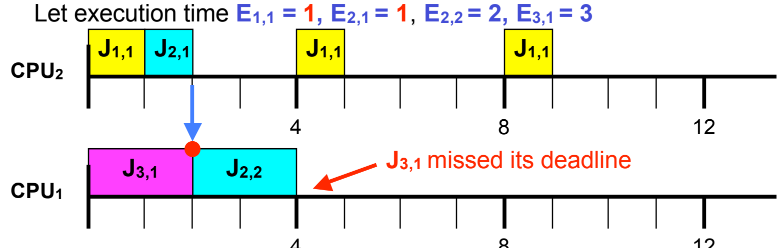
Let $J_{1,1} > J_{2,1}$ and $J_{2,2} > J_{3,1}$
Consider worst case execution time for all jobs, i.e.,
Execution time $E_{1,1} = 2$, $E_{2,1} = 3$, $E_{2,2} = 2$, $E_{3,1} = 3$



Example (3)



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



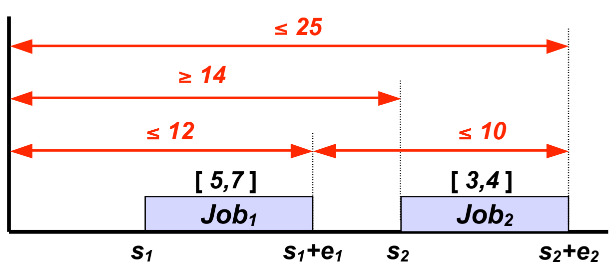
So with the priority assignment of $J_{1,1} > J_{2,1}$ and $J_{2,2} > J_{3,1}$,
jobs cannot be scheduled and scheduling problems are in general NP-hard

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

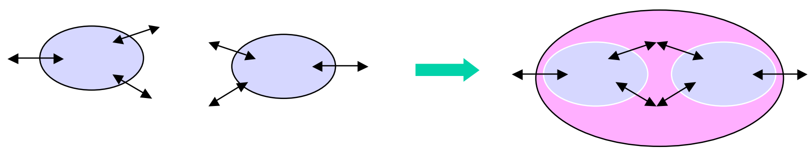
- Computational systems are always constrained in their behaviors
- Resources capture physical constraints
- Resources should be supported as a first-class notion in modeling and analysis
- Resource-bound computation is a general framework of wide applicability

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

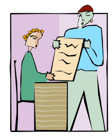
Events

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

  - point-to-point synchronization

Events

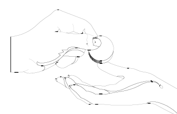
- Events
  - have priorities: $(\mathbf{job}, 10^{10})$



  - have input and output capabilities

or $(\mathbf{e}, \mathbf{p}_1)$ $(\overline{\mathbf{e}}, \mathbf{p}_2)$

$(\mathbf{e}\ ?, \mathbf{p}_1)$ $(\mathbf{e}!, \mathbf{p}_2)$



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:
  $\{(\mathrm{cpu},2)\}$, $\{(\mathrm{cpu}_1,3),(\mathrm{cpu}_2,4)\}$,
  $\{(\mathrm{semaphore},5)\}$

Syntax for ACSR processes

- Process terms
- Process names

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

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

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

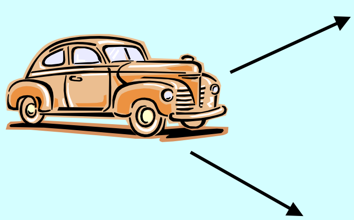
$\text{Operator} \stackrel{def}{=} (\text{ring},1).(\text{pickup},1).\text{Talk}$

$\text{Talk} = \{(\text{phone},2)\} : (\text{hangup},1).\text{Operator}$

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

$P = Q \parallel R$

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

EXAMPLE

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

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

$$Converse \stackrel{def}{=} Operator \parallel Caller$$

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

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

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

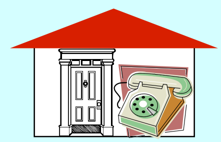
Hiding/Restriction

$P = [Q]_I$ — 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

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



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:

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

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

$$\textbf{ActT} \quad \frac{-}{A : P \xrightarrow{A} P} \qquad \textbf{ActI} \quad \frac{-}{(a,p) : P \xrightarrow{(a,p)} P}$$

- Choice

$$\textbf{ChoiceL} \quad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

- Parallel

$$\textbf{ParIL} \quad \frac{P \xrightarrow{(a,p)} P'}{P \| Q \xrightarrow{(a,p)} P' \| Q}$$

Unprioritized transition relation (II)

- Resource-constrained execution

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

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

Examples

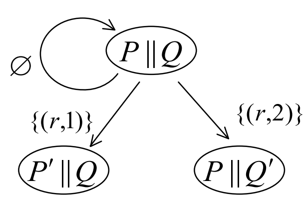
- Resource conflict

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

- Processes must provide for preemption

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

- Unprioritized transitions:



Unprioritized transition relation (III)

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

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

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

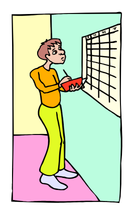
$$\textbf{ScopeT}\quad \frac{R \xrightarrow{\alpha} R'}{P\Delta^a_t(Q,R,S) \xrightarrow{\alpha} R'}\quad (t=0)$$

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

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 α preempts action β 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

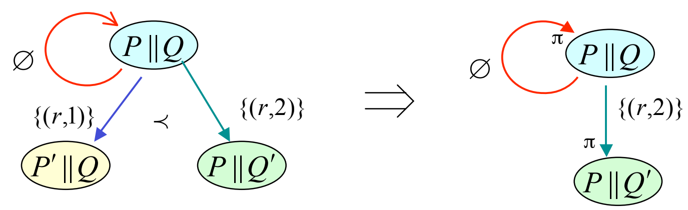
    $$P \xrightarrow{\alpha} P'$$

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

Example

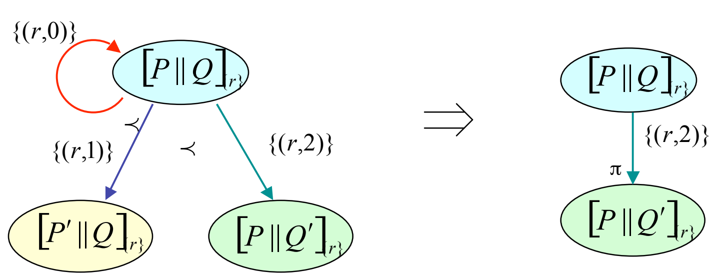
- Unprioritized and prioritized transitions:

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



Example (cont.)

- Resource closure enforces progress



Compositionality of preemption relation

- Given

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

- Given P_1 and P_2, can they be treated as equivalent? That is, for all Q, $P_1 \mid\mid Q = P_2 \mid\mid Q$?

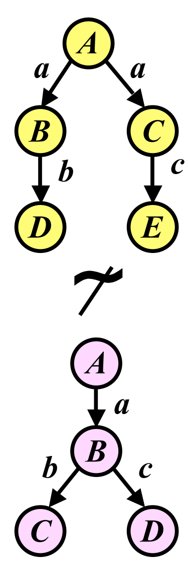
- How about R_1 and R_2?

Bisimulation

- Observational equivalence is based on the idea that two equivalent systems exhibit the same behavior at their interfaces with the environment.

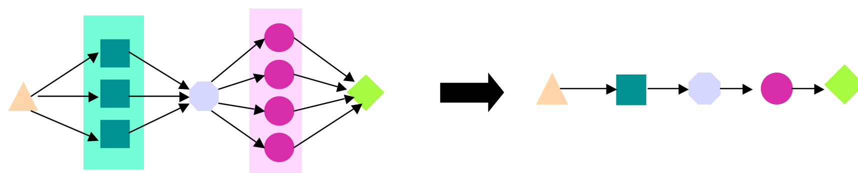
- This requirement was captured formally through the notion of bisimulation, a binary relation on the states of systems.

- 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:

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

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

...

Equational Laws

- ACSR-specific laws for scope and resource closure:

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

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)	$A : P\Delta^b_t(Q,R,S) = A : (P\Delta^b_{t-1}(Q,R,S)) + S$ if $t > 0$
Scope(2)	$e.P\Delta^b_t(Q,R,S) = e.(P\Delta^b_{t-1}(Q,R,S)) + S$ if $t > 0 \wedge \overline{l(e)} \neq b$
Scope(3)	$e.P\Delta^b_t(Q,R,S) = (\tau,\pi(e)).Q + S$ if $t > 0 \wedge \overline{l(e)} = b$
Scope(4)	$P\Delta^b_0(Q,R,S) = R$
Scope(5)	$(P_1 + P_2)\Delta^b_t(Q,R,S) = P_1\Delta^b_t(Q,R,S) + P_2\Delta^b_t(Q,R,S)$
Scope(6)	$\mathrm{NIL}\Delta^b_t(Q,R,S) = S$ 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)	$rec\ X.(P + \sum_{i \in J} [X \setminus E_i]_{U_i}) = rec\ X.(\sum_{J \subseteq I} [P \setminus E_J]_{U_J})$
	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
	$\mid$	$A : P$	timed action prefix
	$\mid$	$e.P$	instantaneous event prefix
	$\mid$	$be \rightarrow P$	conditional process (be : boolean expression)
	$\mid$	$P_1 + P_2$	choice
	$\mid$	$P_1 \parallel P_2$	parallel composition
	$\mid$	$[P]_I$	resource close
	$\mid$	$P \backslash F$	event restriction
	$\mid$	$P \backslash\backslash I$	resource hiding
	$\mid$	$C(\underline{x})$	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

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

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

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

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	$\dfrac{P \xrightarrow{(l!\langle \underline{k} \rangle, m)} P', Q \xrightarrow{(l?\langle \underline{k} \rangle, n)} Q'}{P \Vert Q \xrightarrow{(\tau, m+n)} P' \Vert Q'}$

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:

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

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

- Unprioritized transition : $\begin{cases} P \xrightarrow{\{(cpu,2)\}} P_1 \\ P \xrightarrow{\{(cpu,3)\}} P_2 \end{cases}$
- Prioritized transition : $P \xrightarrow{\{(cpu,3)\}}_{\pi} P_2$

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**:

$$\mathrm{RTS} \stackrel{def}{=} \left[\mathrm{T_1} \parallel \mathrm{T_2} \parallel \cdots \parallel \mathrm{T_n} \right]_{\pi}$$

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)

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

Activator 2. An aperiodic task with $(\mathrm{b, d, p_1, p_2})$

$$\begin{array}{lcl} \mathrm{Activator} & \stackrel{def}{=} & \varnothing^{\mathrm{b}} : \mathrm{Activator'} \\ \mathrm{Activator'} & \stackrel{def}{=} & (start!,1).\varnothing^{\mathrm{d}} : (end!,2). \\ & & \varnothing^{\mathrm{p_1\text{-}d}\ldots\mathrm{p_2}-d} : \mathrm{Activator'} \end{array}$$

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:

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

- *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:

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

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

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

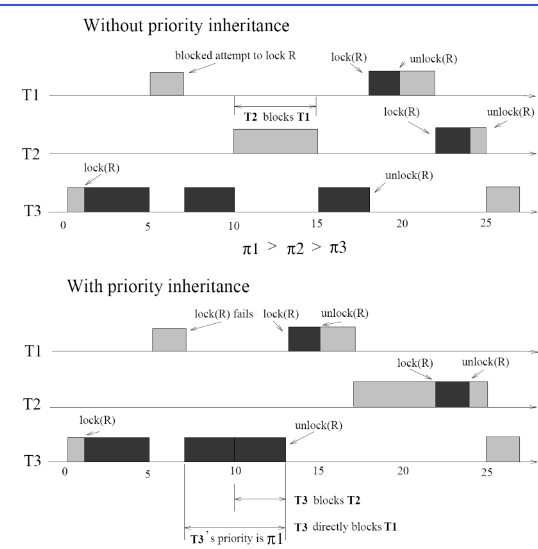
$$
\begin{array}{lcl}
\text{EDFSys} & \stackrel{def}{=} & \left[\text{T}_1 \,\|\, \text{T}_2 \,\|\, \cdots \,\|\, \text{T}_n \right]_{\{cpu\}} \\
\text{T}_i & \stackrel{def}{=} & (\text{Job}_i \,\|\, \text{Activator}_i) \setminus \{start, end\} \\
\hline
\text{Job}_i & \stackrel{def}{=} & \emptyset :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

Other Time-Driven Scheduling Disciplines

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

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

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

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

$$\mathrm{PIPSys} \stackrel{def}{=} \left[\left(\mathrm{T}_1 \| \mathrm{T}_2 \| \mathrm{T}_3 \| \mathrm{P} \right) \backslash \{req, chan, p, v\} \right]_{\{cpu\}}$$

$$\mathrm{T}_i \stackrel{def}{=} \left(\mathrm{Job}_i \| \mathrm{Activator}_i \right) \backslash \{start, end\}$$

$$\begin{aligned}
\mathrm{Job}_i &\stackrel{def}{=} \varnothing : \mathrm{Job}_i + (start?,1).\mathrm{Exec}_i(0,0) \\
\mathrm{Exec}_i(s) &\stackrel{def}{=} (s < \mathrm{c}_i \wedge s \neq \mathrm{cs}_i) \rightarrow (\{cpu, \pi_i)\} : \mathrm{Exec}_i(s+1) \\
&\qquad + \varnothing : \mathrm{Exec}_i(s)) \\
&\quad + (s = \mathrm{cs}_i) \rightarrow ((req!\pi_i, \pi_i).\mathrm{Req}_i(s) + \varnothing : \mathrm{Exec}_i(s)) \\
&\quad + (s = \mathrm{c}_i) \rightarrow \mathrm{Wait} \\
\mathrm{Wait}_i &\stackrel{def}{=} \varnothing : \mathrm{Wait}_i + (end?,1).\mathrm{Job}_i
\end{aligned}$$

$$\begin{aligned}
\mathrm{Req}_i(s) &\stackrel{def}{=} (p!\pi_i, \pi_i).\mathrm{CS}_i(s, \pi_i) + \varnothing : \mathrm{Req}_i(s) \\
\mathrm{CS}_i(s,\pi) &\stackrel{def}{=} (s < \mathrm{c'}_i + \mathrm{cs}_i) \rightarrow (\{cpu, \pi)\} : \mathrm{CS}_i(s+1, \pi) \\
&\qquad + (chan?new,1).\mathrm{CS}_i(s, new) \\
&\qquad + \varnothing : \mathrm{CS}_i(s, \pi)) \\
&\quad + (s = \mathrm{c'}_i + \mathrm{cs}_i) \rightarrow (v?,1).\mathrm{Exec}_i(s)
\end{aligned}$$

$$\mathrm{Activator}_i \stackrel{def}{=} \varnothing^{\mathrm{r}_i} : (start!,1).\varnothing^{\mathrm{d}_i} : (end!,2).\varnothing^{\infty}$$

$$\begin{aligned}
\mathrm{P} &\stackrel{def}{=} (p?x,1).\mathrm{V}(x) + (req?x, \pi_{max}).(p?x,1).\mathrm{V}(x) + \varnothing : \mathrm{P} \\
\mathrm{V}(max) &\stackrel{def}{=} (v!,1).\mathrm{P} + \varnothing : \mathrm{V}(max) \\
&\quad + (req?x,1).((x > max) \rightarrow (chan!x,1).\mathrm{V}(x) \\
&\qquad + (x \leq max) \rightarrow \mathrm{V}(max))
\end{aligned}$$

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 $E \stackrel{t}{\Rightarrow} E'$ 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', $Q \stackrel{\alpha}{\Rightarrow} Q'$ and $(P',Q') \in r$, and
2. if $Q \xrightarrow{\alpha} Q'$, then, for some P', $P \stackrel{\alpha}{\Rightarrow} P'$ 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 **EDFSys** and **PIPSys** 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***

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

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

Schedulability Analysis

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

Lemma 2 *If*

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

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

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

Lemma 4 *If*

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

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

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

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

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

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

Probabilistic ACSR for soft real-time scheduling analysis

PACSR (Probabilistic ACSR)

- ACSR extension for probabilistic behaviors.
- Objective :
  - formally describe behavioral variations in systems that arise due to failures in physical devices.
- Since failing devices are modeled by resources we associate a failure probability `p(r)` with every resource `r`
  - at any time unit, `r` is down with probability `p(r)` or up with probability `1-p(r)`
  - failures are assumed to be independent

Syntax for PACSR processes

- Similar to ACSR
- Process terms

$$P ::= NIL \quad | \quad A:P \quad | \quad (a,n).P \quad | \quad P+P \quad | \quad P \| P$$
$$\quad | \quad P\Delta_t^a(Q,R,S) \quad | \quad [P]_I \quad | \quad P \setminus F \quad | \quad b \rightarrow P \quad | \quad C$$

- Process names

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

- Distinction: For all resources r we write $\bar{r}$ for the *failed occurrence of resource* r. Thus, an action can specify access to failed resources.

Resource failures and recoveries

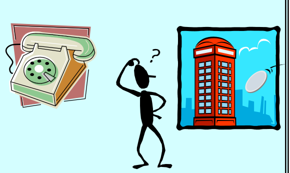
- An action containing resource r cannot be taken when r is failed, i.e.,

r is failed, $r \in \rho(A) \Rightarrow A:P = NIL$

- Failed resources: $\bar{r}$, $\mathbf{pr}(\bar{r}) = 1 - \mathbf{pr}(r)$
- Recoveries are modeled by using failed resources in actions

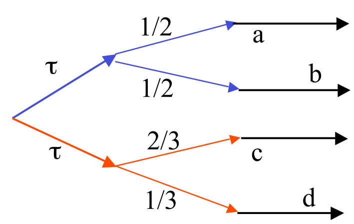
EXAMPLE

$$\{(phone,1)\}:PlaceCall$$
$$+\{(\overline{phone},1)\}:UsePayPhone$$



PACSR Semantics

- Semantics of a PACSR process is given in terms of *probabilistic transition systems*: some transitions are labeled with probabilities and others with actions/events.
- Labeled Concurrent Markov Chain (LCMC)



PACSR Semantics

- Configurations are pairs of the form (P,W), where
  - *P* is a PACSR process, and
  - *W* is a world capturing the state of resources as follows

$$\forall r, r \in W \Rightarrow \bar{r} \notin W \quad \text{and} \quad \forall r, \bar{r} \in W \Rightarrow r \notin W$$

- A configuration (P,W) is characterized as
  - Probabilistic, if P requires resources whose state is not in W.

    Example: ($\{r_1,1\}$:Q , $\{r_2\}$)
  - Nondeterministic, if all resource information required by P is in W.

    Example: ((a,1):NIL , ∅)

PACSR semantics (II)

- The semantics is given via a pair of transition relations:
  - Probabilistic transition relation,
    $$(P,W) \xrightarrow{pr}_p (P,W')$$
  - Nondeterministic transition relation,
    $$(P,W) \xrightarrow{\alpha} (Q,W)$$

- Let imr(P) be resources that can be used in the first step:
  $$\{r \mid P \xrightarrow{A} P', r \in \rho(A)\}$$

Operational semantics

- The probabilistic transition relation is as follows:
  $$\frac{P \in S_p, Z_1 = \mathrm{imr}(P) - (W \cup \overline{W}), Z_2 \in \mathcal{W}(Z_1)}{(P,W) \xrightarrow{pr(Z_2)}_p (P, W \cup Z_2)}$$

  W(Z) is a set of all possible scenarios of resources; e.g.,
  $$W(\{r_1, \overline{r_2}\}) = \{(\overline{r_1}, \overline{r_2}), (\overline{r_1}, r_2), (r_1, \overline{r_2}), (r_1, r_2)\}$$

- The nondeterministic transition relation is taken from ACSR, with one exception:
  $$\textbf{ActT} \quad \frac{-}{(A:P,W) \xrightarrow{A} (P,\emptyset)} \quad \rho(A) \subseteq W$$

Example

- Let $P = \{(r_1,2),(\overline{r_2},3)\} : Q$, pr(r_1) = ½ and pr(r_2) = 1/3.

  Then imr(P) = $\{r_1,r_2\}$ and W($\{r_1,r_2\}$)=$\{\{r_1,r_2\}, \{r_1,\overline{r_2}\}, \{\overline{r_1},r_2\}, \{\overline{r_1},\overline{r_2}\}\}$

- Thus by the probabilistic transition relation

$$(P,\phi) \xrightarrow{1/6}_p (P,\{r_1,r_2\}) \qquad (P,\phi) \xrightarrow{1/6}_p (P,\{\overline{r_1},r_2\})$$

$$(P,\phi) \xrightarrow{1/3}_p (P,\{r_1,\overline{r_2}\}) \qquad (P,\phi) \xrightarrow{1/3}_p (P,\{\overline{r_1},\overline{r_2}\})$$

- and by the nondeterministic transition relation

$$(P,\{r_1,r_2\}) \not\longrightarrow$$

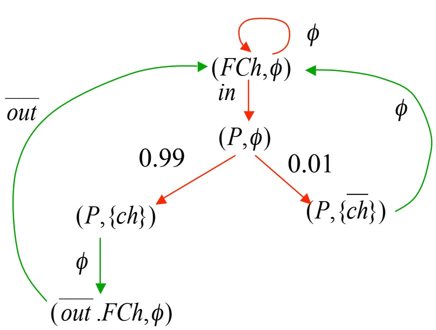
$$(P,\{\overline{r_1},r_2\}) \not\longrightarrow \qquad (P,\{r_1,\overline{r_2}\}) \xrightarrow{\{(r_1,2),(\overline{r_2},3)\}} (Q,\phi)$$

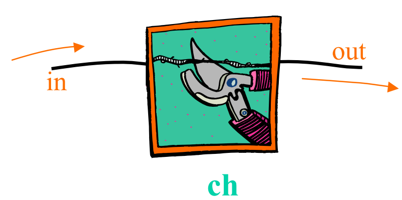
$$(P,\{\overline{r_1},\overline{r_2})\} \not\longrightarrow$$

Example: A faulty channel

$$\mathrm{FCh} = \phi : \mathrm{FCh}$$
$$+\,\mathrm{in}.(\{\mathrm{ch}\} : \overline{\mathrm{out}}!.\ \mathrm{FCh}$$
$$+\,\{\overline{\mathrm{ch}}\}.\ \mathrm{FCh}) \setminus \{\mathrm{ch}\}$$

where pr(ch) = 0.99





Probabilistic HML with until

- In order to analyze PACSR specifications we may want to check whether a specification satisfies a property written as a logical formula.
- We use a probabilistic HML with an 'until' operator
- The 'until' operator is parameterized with regular expressions over event names.
- Syntax

$$f ::= tt \mid \neg f \mid f \wedge f' \mid f \langle \Phi \rangle_{\omega p} f' \mid f \langle \Phi \rangle_{\omega p}^{t} f'$$

where Φ is a regular expression over actions and $\omega \in \{\leq, \geq\}$

The until operator

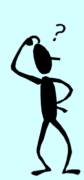
$$P \models f \langle \Phi \rangle_{\leq q}^{t} f'$$

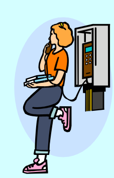
There is some execution with probability ≤ q for which f holds until f' becomes true within time t and observable behavior from Φ

EXAMPLE

$$\text{true} \langle \{\text{talk}, \text{wait}\}^{*} \text{hangup} \rangle_{0.01}^{20} \text{true}$$

≡ the probability that within 20 time units after any number of talk and wait actions action hangup arises is ≤ 0.01





Semantics for *until*

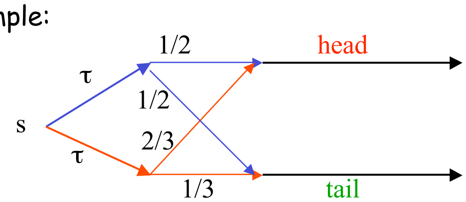
- $s \models f_1 \langle \Phi \rangle_{>\pi} f_2$

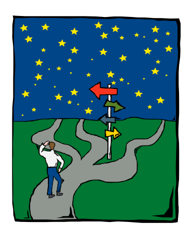
  if there exists a scheduler σ such that the set of computations that
  - start at s
  - contain only states (except the last) satisfying f_1
  - have observable content Φ
  - end in a state satisfying f_2

  have probability greater than π

Resolving non-determinism

- Analysis involves computing the probability of reaching a set of desired states (within a time period) via an acceptable set of behaviors.
- Example:





- What is the probability that event head takes place?
- Such probability depends on how the nondeterminism of s is resolved.

Model Checking

- Schedulers are used for resolving non-determinism. These are functions that given a computation ending in a nondeterministic state choose the next transition to take place.
- Given a scheduler σ of a system P, sets of states A and B, and a regular expression Φ, we may compute probabilities
  - $Pr_A(P \to B, \Phi, t, \sigma)$, the probability of reaching a state in B, passing only via states in A, via paths with observable content in Φ, and within t time units
- So for example:

$P \models f\langle\Phi\rangle^{t}_{\leq q} f'$ iff there is scheduler σ such that $q \geq Pr_A(P \to B, \Phi, t, \sigma)$
where $A = \{ P' \mid P' \models f \}$, $B = \{ P' \mid P' \models f' \}$

Model checking *until*

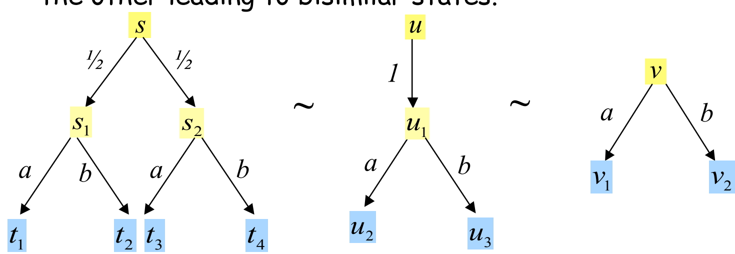
- To check $s \models f_1\langle\Phi\rangle_{>\pi} f_2$
  - Compute the least solution to the set of equations:

$$X^{s}_{f_1\langle\Phi\rangle f_2} = \begin{cases} \sum_{s \xrightarrow{\pi} s'} \pi \cdot X^{s'}_{f_1\langle\Phi\rangle^{\leq p} f_2} & s \in S_p \\ \max_{s \xrightarrow{\alpha} s'}(X^{s'}_{f_1\langle\Phi\rangle f_2}) & s \in S_n, s \models f_1 \\ 1 & s \in S_n, s \models f_2, \varepsilon \in \Phi, p \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

  - Return true if $X^{s}_{f_1\langle\Phi\rangle f_2} > \pi$

Equivalence Relations

- New notions of equivalence for the LCMC model taking account both action types and probabilities.
- In particular two LCMCs are strongly bisimilar if
  1. they reach sets of bisimilar states with the same probability, and
  2. for each nondeterministic step of one there exists a step of the other leading to bisimilar states.



3 October 2003 ESSES 2003 99

Equivalence Relations

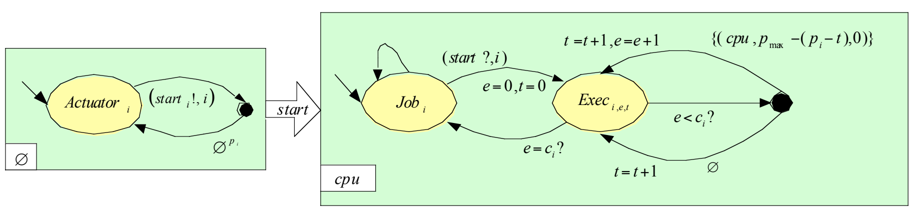
- There is a set of laws that completely axiomatizes strong bisimulation for PACSR processes.
- Other equivalence notions include *weak bisimulation* which relates systems that have the same observable behavior, that is, it ignores τ actions.

3 October 2003 ESSES 2003 100

Two Examples

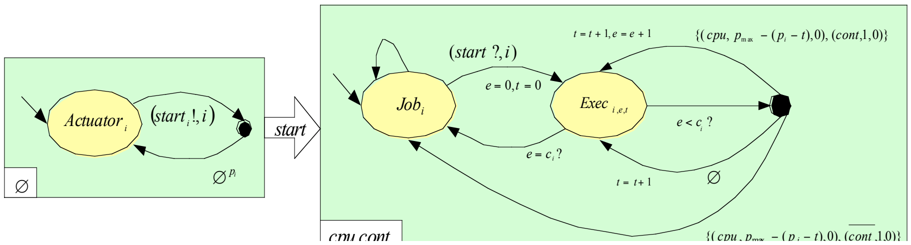
- EDF with probabilistic execution time
- Telecommunication application

EDF task scheduling



- Periodic process *Job*: Period p_i, computation time c_i
  - At each step, total time t increases, active time e increases only if resource *cpu* is available; complete when $e=c_i$
- Resource *cpu*: scheduling
  - Priority of a task dynamically increases closer to the deadline
- Process *Actuator* keeps timing deadlines
  - Every p_i seconds, signal *start* is sent to the task, which can accept it only if it has finished execution

EDF task with probabilistic completion

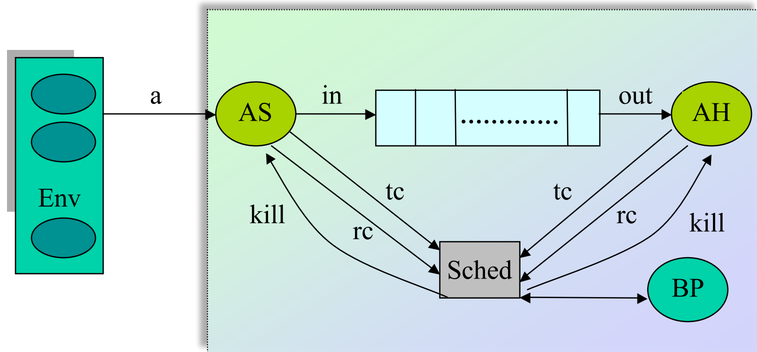


- The task may decide to become inactive after completing a computation step
- Resource *cont* controls probabilistic completion
  - failure means "terminate early"

A Telecommunication Application

- Based on the specification of a switching system considered in AJK97.
- The system consists of a number of concurrent processes with real-time constraints.
- Probabilistic behavior is present in the form of
  - probabilistic arrival of alarms, and
  - uncertain execution times of processes.

Example: A Telecommunication Application



PACSR Specification

- The System

$$Sys = (Env \,||\, B_0 \,||\, \phi : Sched \,||\, AS$$
$$||\, AH \,||\, BP) \setminus F \setminus\setminus I$$

The system in its initial state: *a parallel composition of all the components*

- The environment

$$Env = \Pi_{1 \le i \le n} P_i$$
$$P_i = \{r_i\} : P_i + \{\overline{r_i}\} : (P_i \,||\, Q_i)$$
$$Q_i = \overline{a} : NIL + \phi : Q_i$$

The environment provides probabilistic alarms: *at the failure of any of resources r_i an alarm is sent via channel a*

PACSR Specification

- Background Process

$$BP = (\overline{tc},0).BP'\Delta^{h}_{\infty}(NIL, NIL, kill.BP) + \phi : BP$$

$$BP' = (\{r\} : BP' + \{\overline{r}\} : \overline{rc}.BP) \setminus\setminus \{r\}$$

- The Scheduler

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

The background process competes for processor time managed by the scheduler. Its duration is geometrically distributed.

PACSR Specification

- The buffer

$$B_0 = in.B_1 + \phi : B_0$$

$$B_i = in.B_{i+1} + \Sigma_{1 \le i \le j} d_j.B_{i-j} + \phi : B_i + \overline{out}_i.B_i$$

$$B_n = in.\overline{overflow}.NIL + \Sigma_{1 \le i \le n} d_j.B_{n-j} + \phi : B_n + \overline{out}_n.B_n$$

- The Alarm Sampler and the Alarm Handler

$$AS = AS' \| (\phi^p : AS) \qquad AH = \Sigma_i out_i.AH_{n(i)} + \phi : AH$$

$$AS' = (\overline{tc},2).AS'' + \phi : AS' \qquad AH_i = (\overline{tc},2).AH_i^A + \phi : AH_i$$

$$AS'' = a.\overline{in}.AS'' + \phi : \overline{rc}.NIL \qquad AH_i^A = \phi^{pt(i)} : \overline{d}_i.\overline{rc}.AH$$

Two configurations

- Consider two versions of the system:
  S_1 with
  - Possibility of 1 alarm per time unit,
  - Buffer size of 3
  - Capability of processing 2 alarms per time unit, and

  S_2 with
  - Possibility of 2 alarms per time unit
  - Buffer size of 6
  - Capability of processing 4 alarms per time unit
- Comparison criterion: What is the probability of overflow in the alarm buffer?

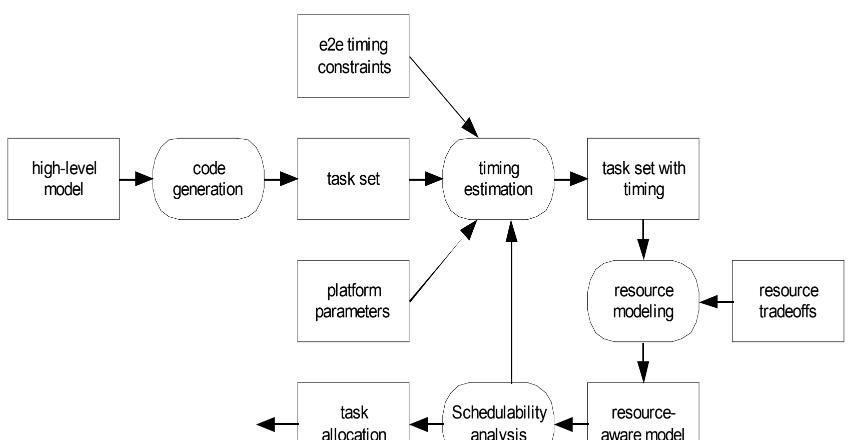
Checking $f = tt\langle overflow\rangle^{t}_{\leq q}\ tt$

T(time units)	S_1	S_2
10	2×10^{-6}	3×10^{-10}
20	5×10^{-6}	6×10^{-10}
30	9×10^{-6}	1.0×10^{-9}
40	1.2×10^{-5}	1.3×10^{-9}
50	1.5×10^{-5}	1.6×10^{-9}
60	1.9×10^{-5}	2.1×10^{-9}
70	2.2×10^{-5}	2.4×10^{-9}
80	2.5×10^{-5}	2.8×10^{-9}
90	2.9×10^{-5}	3.1×10^{-9}
100	3.2×10^{-5}	3.5×10^{-9}

The table shows for various values of t, the probability q that makes property f true for each of the systems.

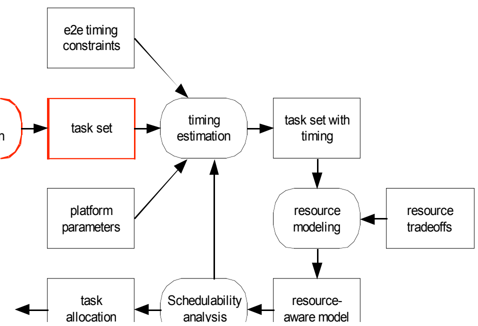
Design of Embedded Systems in a Resource-oriented Framework

Embedded systems design process



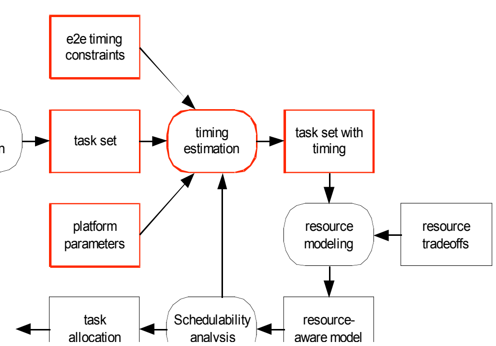
Modeling and code generation

- High-level model captures functionality of the system and assumptions about the environment
- Code generation breaks the functional behavior into a set of tasks



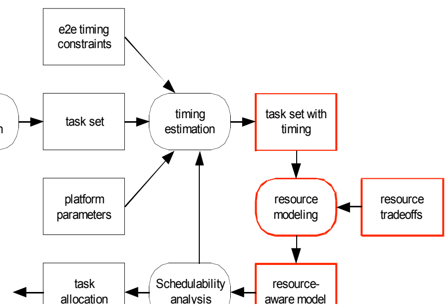
Timing parameter estimation

- Estimate the execution time for task on a given platform
- Assign task periods based on end-to-end timing constraints



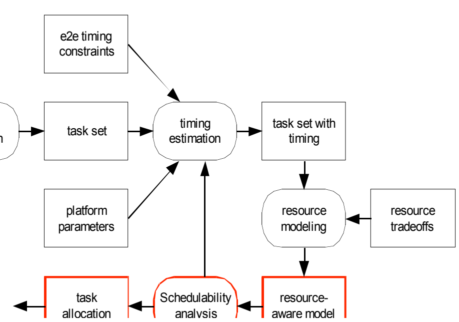
Resource modeling

- Resource is a critical notion in embedded and real-time system design, yet lacks systematic formal treatment
- Key idea: resource attributes capture tradeoffs



Formal schedulability analysis

- Resource conflicts introduce execution delays
- Violations of timing constraints lead to deadlocks in the model behavior
- Discovered by state-space exploration



Modeling and Analysis of Power-Aware Systems

Motivation

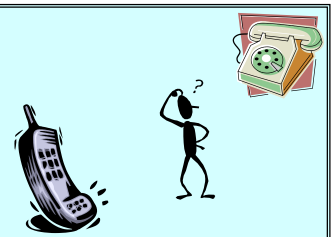
- Features of mobile embedded systems:
  - Resource constraints
    - Limited battery life
  - Uncertainty
    - changing communication delays, failures
- Solution:
  - a unified formal framework for designing and reasoning about power-constrained, timed systems with probabilistic behavior

P^2ACSR – A power-aware extension of PACSR

- A unified framework for modeling and analyzing power-aware real-time systems.
- We associate a further attribute to resource usage, that of power consumption.
- The syntax remains the same, except that actions are tuples of the form (r,p,c), where r is the *resource*, p is the *priority level* and c the *power consumption* of the resource usage.

EXAMPLE

$$\{(phone,1,0)\}:Call_1$$
$$+$$
$$\{(cellphone,1,3)\}:Call_2$$



P^2ACSR

- Semantics is given similarly to PACSR, as a LCMC.
- We can use various techniques to perform various analyses on P^2ACSR models including:
  - Model checking
    We may express temporal logic properties involving power consumption bounds and check that they are satisfied by P^2ACSR processes.
  - Probabilistic bounds on power consumption
    We may compute the probability that power consumption exceeds certain limits.
  - *Average power consumption*
    We may compute the average power consumption during intervals of interest.

P^2ACSR

- P^2ACSR is an extension of PACSR, a probabilistic real-time process algebra.
- In P^2ACSR:
  - system is a collection of concurrent processes
  - communication among processes is instantaneous
  - access to serially-reusable resources consumes time and power

Resources

- Resources capture constraints on executions
- Features of resources:
  - Serially reusable
    - processors, memory, communication channels
  - Unreliable
    - Fail with a fixed probability in each step
  - Require time and power
    - May allow different levels of power consumption

Actions

- Actions represent computation
  - actions take one unit of time
  - require access to resources
    - each resource r has priority of access p_r
    - each resource r has power use level c_r

$$A = \{(r_1, p_1, c_1), (r_2, p_2, c_2)\}$$

    - each resource can be used at most once
  - resources of action A: $\rho(A)$
  - power consumption of action A: $pc(A) = \sum_{r \in \rho(A)} c_r$

Power constraints

- Resource classes $\mathcal{R}_1, \ldots, \mathcal{R}_n$
  - correspond to different power sources
- Attributes of resource class $\mathcal{R}_i$:
  - capacity C_i – maximum amount of power in one step
  - charge $\mathcal{P}_i$ – total amount of power
- Valid actions satisfy capacity constraints:
  - for each $\mathcal{R}_i$, $pc_i(A) = \sum_{r \in \rho(A), r \in \mathcal{R}_i} c_r < C_i$

Processes

- Event and action steps
- Choice P_1+P_2
- Parallel composition $P_1||P_2$
- Temporal scope, time-outs, exceptions, …

- Structural operational semantic rules build behaviors of complex processes from behaviors of component processes

P²ACSR semantics

- Before steps of a process can be computed, status of relevant resources has to be determined
- Resource status is kept in a world
- Non-deterministic configurations S_n
  - world has complete knowledge of resources
- Probabilistic configurations S_p
  - incomplete knowledge
- Probabilistic steps: $S_p \to S_n$
  - acquire missing knowledge

Non-deterministic rules

- Action can happen if all resources are available and power constraints are obeyed:

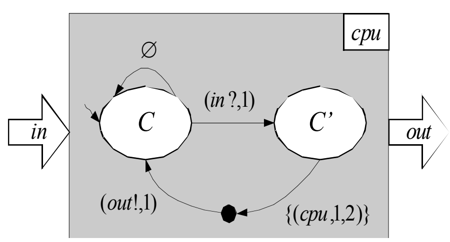
$$(A : P, W) \xrightarrow{A} (P, \varnothing), \quad \rho(A) \subseteq W, valid(A)$$

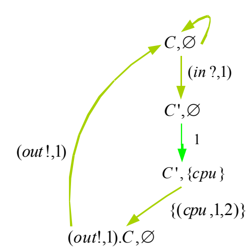
- Parallel processes can proceed if their actions do not conflict and the joint step does not violate constraints

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

- Model: Labeled Concurrent Markov Chains

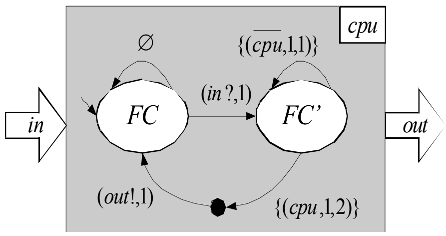
Example

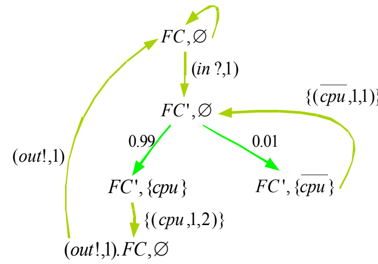




- C is a process that reliably translates messages from in to out in 1 time unit using 2 units of power per message
- $\pi(cpu) = 1$

Example





- *FC* (fault-tolerant *C*) accommodates for *cpu* failures
  - $\pi(cpu) = 0.99$
- If *cpu* fails, the message is not delivered, but less power is consumed
- Message is delivered with probability 1
  - What is the expected power consumption per message?

A logic for power constraints

- $\mathcal{L}^{pc}_{PHMLu}$: Power-aware probabilistic HML with *until*
  - Propositional operators $tt, \neg f, f_1 \wedge f_2$
  - *until* operators specify probabilistic bounds on power consumption along a set of paths
    - Basic variant: $f_1 \langle \Phi \rangle^{\leq p}_{>\pi} f_2$
    - With time constraints: $f_1 \langle \Phi \rangle^{\leq p}_{>\pi, t} f_2$
    - With resource class constraints: $f_1 \langle \Phi \rangle^{\leq p, \Re}_{>\pi} f_2$

Semantics for $\mathcal{L}^{pc}_{PHMLu}$: *until*

- $s \models f_1 \langle \Phi \rangle^{\leq p}_{>\pi} f_2$
  if there exists a scheduler σ such that the set of computations that
  - start at s
  - contain only states (except the last) satisfying f_1
  - have observable content Φ
  - consume no more power than p
  - end in a state satisfying f_2

  have probability greater than π

Model checking *until*

- To check $s \models f_1 \langle \Phi \rangle^{\leq p}_{>\pi} f_2$
  - Compute the least solution to the set of equations:

$$X^{s}_{f_1\langle\Phi\rangle^{\leq p} f_2} = \begin{cases} \sum_{s \xrightarrow{\pi} s'} \pi \cdot X^{s'}_{f_1\langle\Phi\rangle^{\leq p} f_2} & s \in S_p \\ \max_{s \xrightarrow{\alpha} s'} (X^{s'}_{f_1\langle\Phi\rangle^{\leq p - pow(\alpha)} f_2}) & s \in S_n, s \models f_1 \\ 1 & s \in S_n, s \models f_2, \varepsilon \in \Phi, p \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

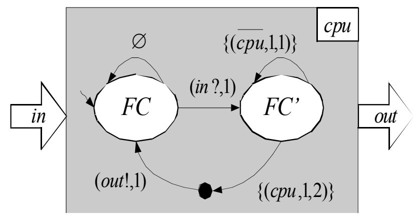
  - Additional annotation $\leq p$ in the variable set
  - Return true if $X^{s}_{f_1\langle\Phi\rangle^{\leq p} f_2} > \pi$

Example

- Power consumption per message:

$$FC,\varnothing \models tt\left\langle in\cdot\{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{>2}_{\geq 1} tt$$

$$FC,\varnothing \models tt\left\langle in\cdot\{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3}_{>0.999} tt$$



Example $FC,\varnothing \models tt\left\langle in\cdot\{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3}_{>0.999} tt$

$$X^{FC,\varnothing}_{tt\left\langle in\cdot\{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3} tt} = X^{FC',\varnothing}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3} tt}$$

$$X^{FC',\varnothing}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3} tt} = 0.99\cdot X^{FC',\{cpu\}}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3} tt} + 0.01\cdot X^{FC',\{\overline{cpu}\}}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3} tt}$$

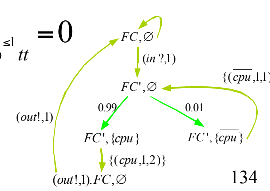
$$X^{FC',\{cpu\}}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3} tt} = X^{out!.FC,\varnothing}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3} tt} = X^{FC,\varnothing}_{tt\langle \varepsilon\rangle^{\leq 1} tt} = 1$$

$$X^{FC',\{\overline{cpu}\}}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3} tt} = X^{FC',\varnothing}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 2} tt}$$

$$X^{FC',\varnothing}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 2} tt} = 0.99\cdot X^{FC',\{cpu\}}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 2} tt} + 0.01\cdot X^{FC',\{\overline{cpu}\}}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 2} tt}$$

$$X^{FC',\{cpu\}}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 2} tt} = 1 \quad X^{FC',\{cpu\}}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 2} tt} = X^{FC',\varnothing}_{tt\left\langle \{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 1} tt} = 0$$

$$X^{FC,\varnothing}_{tt\left\langle in\cdot\{cpu,\overline{cpu}\}^{*}\cdot out\right\rangle^{\leq 3} tt} = 0.9999$$



Power-aware scheduling

- Trade-off: power vs. execution time
- CMOS-based processors can operate at reduced voltage levels
  - Power dissipation is proportional to V^2
  - StrongARM SA2:
    - 600 MHz / 500 mJ or 150 MHz / 160 mJ
- Tasks can take less than worst-case time
  - Adjust frequency dynamically to utilize "time slack"

Dynamic Voltage Scaling

- *Dynamic voltage scaling* is a technique proposed for making energy savings by dynamically altering the power consumed by a processor.
- Lower frequency execution implies longer processing of tasks.
- This may lead to violation of real-time constraints.
- [Pillai and Shin 01] propose extensions to real-time scheduling algorithms to make use of dynamic voltage scaling.

Case study: two kinds of resources

- Power-aware resources:
  - Attributes:
    - Priority (dynamic) – schedulability analysis
    - Power consumption (dynamic) – power calculations
- "abstract" resources:
  - Attributes:
    - Availability (static) – probabilistic completion
    - No power consumption, same priority

Power-Aware Real-Time Scheduling

- Let I be a set of tasks with periods p_i and worst-case execution times c_i, sharing the same CPU.
- In reality tasks often take much less time to execute.
- This probabilistic execution time may be modeled in PACSR as follows:

$$\mathbf{Task}_i = (start?,0) \,.\, \mathbf{Exec}_{i,0,0} + \varnothing : \mathbf{Task}_i \qquad i = \{1,\ldots,I\}$$

$$\begin{aligned}\mathbf{Exec}_{i,e,t} = e < c_i \rightarrow (\, & \varnothing : \mathbf{Exec}_{i,e,t+1} \\ & + \{(cpu,\, dmax-(p_i-t)),(cont,1) \ \ \} : \mathbf{Exec}_{i,e+1,t+1} \\ & +\{(cpu,dmax-(p_i-t)),(\overline{cont},1))\} : \mathbf{Task}_i\,) \\ + \; e = c_i \rightarrow \mathbf{Task}_i \end{aligned}$$

$i = \{1,\ldots,I\}$
$e= \{0,\ldots, c_i\}$
$t = \{0,\ldots, c_i\}$

potential for early termination (geometric distribution)

Power-Aware Real-Time Scheduling

- The algorithm of [Pillai and Shin] takes advantage of the possibility of early termination of a task by then executing the next task at the lowest possible frequency.
- Specifically, on every release or completion of a task it re-computes the sum

$$\alpha = \frac{c_1^{last}}{p_1} + \ldots + \frac{c_n^{last}}{p_n}$$

where c_i^{last} is the computation time of the last execution of task i or c_i if task i has just been released.

- Based on this value it decides the lowest frequency that is consistent with the current effective utilization.

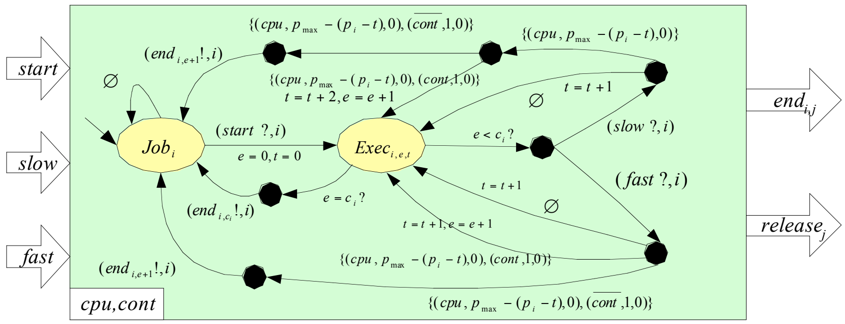
Power-Aware Real-Time Scheduling

- First we extend the model of a task with the ability of executing slower or faster. It responds to messages *fast* and *slow*. In the slow mode a computation step takes twice as long, i.e two time units. It also signals its release when execution commences and its completion time when it completes.

$$\mathbf{Task}_i = (start_i?,0)\ .\ (release_i!,\ \mathrm{i}).\ \mathbf{Exec}_{i,0,0} + \varnothing : \mathbf{Task}_i \qquad i = \{1,\ldots,I\}$$

$$\begin{aligned}
\mathbf{Exec}_{i,e,t} = e < c_i \rightarrow\ & ((fast?\ ,\ \mathrm{i})\ (\ \varnothing : \mathbf{Exec}_{i,e,t+1} \\
& \quad + \{(cpu,\ dmax-(p_i-t)),(cont,1)\} : \mathbf{Exec}_{i,e+1,t+1} \\
& \quad + \{(cpu,\ dmax-(p_i-t)),\ (\overline{cont},1)\} : (end_{i,e+1}!,\mathrm{i}).\ \mathbf{Task}_i\ ) \\
& + (slow?\ ,\ \mathrm{i})\ (\ \varnothing : \mathbf{Exec}_{i,e,t+1} \\
& \quad + \{(cpu,\ dmax-(p_i-t)),(cont,1)\} : \\
& \qquad (\{(cpu,\ dmax-(p_i-t)),(cont,1)\} : \mathbf{Exec}_{i,e+1,t+2} \\
& \qquad + \{(cpu,\ dmax-(p_i-t)),\ (\overline{cont},1)\} : (end_{i,e+1}!,\mathrm{i}).\ \mathbf{Task}_i\ ) \\
& + e = c_i \rightarrow \mathbf{Task}_i
\end{aligned}$$

Speed-sensitive task



- If operating frequency is *fast*, take one time unit per scheduling step
- If operating frequency is *slow*, take two time units per scheduling step

Power-Aware Real-Time Scheduling

- The DVS algorithm is represented as the P²ACSR process:

$$DVS = (Scale_{c_1,c_2,c_3} \parallel \mathrm{Proc}_{fast}) \setminus \{f_{up}, f_{down}\}$$

- Scale responds to release and completion signals and triggers the re-computation of α

$$\begin{aligned} Scale_{e_1,e_2,e_3} = &(release_1\,?,0).SetNew_{c_1,e_2,e_3} \\ &+(release_2\,?,0).SetNew_{e_1,c_2,e_3} \\ &+(release_3\,?,0).SetNew_{e_1,e_2,c_3} \\ &+\ldots \\ &+(end_{1,c}\,?,0).SetNew_{c,e_2,e_3} \\ &+(end_{2,c}\,?,0).SetNew_{e_1,c,e_3} \\ &+\ldots \end{aligned}$$

Power-Aware Real-Time Scheduling

- SetNew decides the lowest frequency to the current effective utilization and sends the appropriate signal

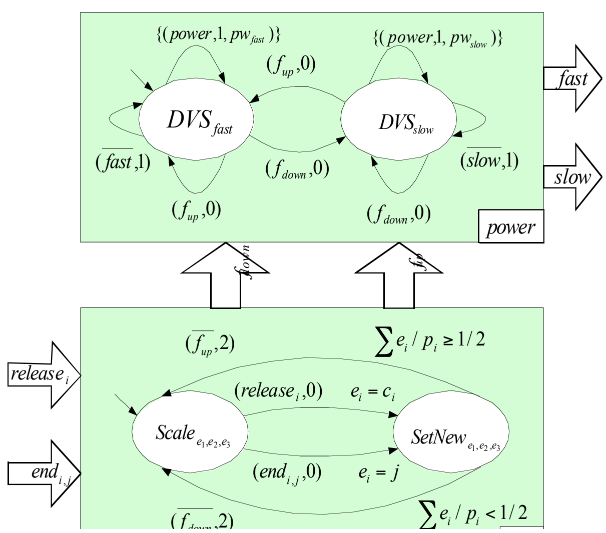
$$\mathbf{SetNew}_{e1,e2,e3} = e_1/p_1 + e_2/p_2 + e_3/p_3 < ½ \rightarrow (f_{down}!,4).\ \mathbf{Scale}_{e1,e2,e3} + e_1/p_1 + e_2/p_2 + e_3/p_3 \geq ½ \rightarrow (f_{up}!,4).\ \mathbf{Scale}_{e1,e2,e3}$$

- DVS_{fast} and DVS_{slow} describe the processor operating in the high and low frequency, respectively

$$\mathbf{DVS}_{fast} = \{(power,1,pw_{fast})\}:\mathbf{DVS}_{fast} + (fast!,1).\mathbf{DVS}_{fast} + (f_{down}?,0).\mathbf{DVS}_{slow} + (f_{up}?,0).\mathbf{DVS}_{fast}$$

$$\mathbf{DVS}_{slow} = \{(power,1,pw_{slow})\}:\mathbf{DVS}_{slow} + (slow!,1).\mathbf{DVS}_{slow} + (f_{down}?,0).\mathbf{DVS}_{slow} + (f_{up}?,0).\mathbf{DVS}_{fast}$$

Operating frequency manipulation



- Recompute frequency each time a task is released or completed
- Consume pw_{fast} in fast mode and pw_{slow} in slow mode

Analysis of DVS

- We considered the following set of tasks:

Task	Execution time	Period
1	3	8
2	3	10
3	1	14

- The algorithm guarantees the task set remains schedulable.
- We computed the expected power consumption for one major frame ($t=p_1 \cdot p_2 \cdot p_3$) for pr(cont)=1/3 and $pw_{fast}=2$, $pw_{slow}=1$.
  - With DVS *minimum power consumption = 1906.66* and *maximum power consumption = 1922.65*
  - Without DVS *power consumption = 2240*
  - Thus expected savings between 14% and 14.8%.

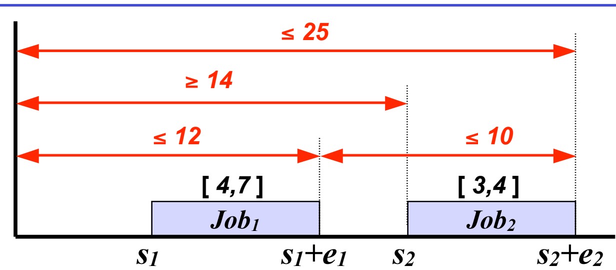
Conclusions

- We have developed a timed, probabilistic, process algebra that allows modeling the power consumption of system resources
- Various techniques for quantitative analysis of power properties have been developed and implemented in the PARAGON toolset
  - Probabilistic bounds computation
  - Model checking
- Research direction:
  - Uniform resource attribute model

ACSR-VP
for design synthesis and parametric analysis

3 October 2003 ESSES 2003 147

Example: A Start-time Assignment Problem



- Start-time Assignment Problem with Inter-job Temporal Constraints
- The order of execution of job is not known
- 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.

3 October 2003 ESSES 2003 148

ACSR-VP (ACSR With Value-passing)

- Extends ACSR with
  - variables: (a?x,1).(c!x,1)...
  - value passing communications: (c!7,1)... || (c?x,1)...
  - parameterized processes: P(x) = (x > 1) → (a!x,1).nil
- Priorities can be specified using expressions
  - timed actions: {(data, y+1)}
  - instantaneous events: (signal!8, x+3)
- Syntax

$$
\begin{array}{lll}
P & ::= & NIL \mid a.P \mid A:P \mid P+P \mid P \| P \\
 & & b \rightarrow P \mid P \backslash F \mid [P]_I \mid C \\
a & ::= & (\tau, e) \mid (c?x, e) \mid (c!e_1, e_2) \\
A & ::= & \varnothing \mid \{S\} \\
S & ::= & (r, e) \mid (r, e), S \\
C & ::= & X \mid X(\vec{v})
\end{array}
$$

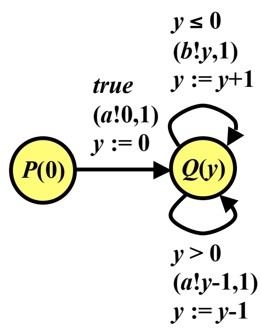
Symbolic Graph With Assignment (SGA)

SGA is a directed graph with edges labeled with b,α, and θ, where b is a Boolean condition, α is an action, and θ is an assignment.

We use SGA to capture the semantics of ACSR-VP

$$
\begin{aligned}
P(x) &= (a!x,1).Q(x) \\
Q(y) &= (y \le 0) \rightarrow (b!y,1).Q(y+1) \\
&\quad + (y > 0) \rightarrow (a!y\text{-}1,1).Q(y\text{-}1)
\end{aligned}
$$

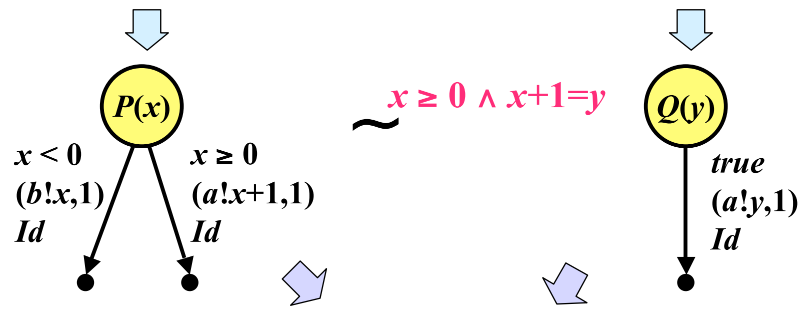
$P(0) \Rightarrow (a!0,1).(b!0,1).(a!0,1)\ldots$



Symbolic Bisimulation (Informal Description)

$P(x) = (x < 0) \rightarrow (b!x,1).nil$
$\quad + (x \geq 0) \rightarrow (a!x+1,1).nil$

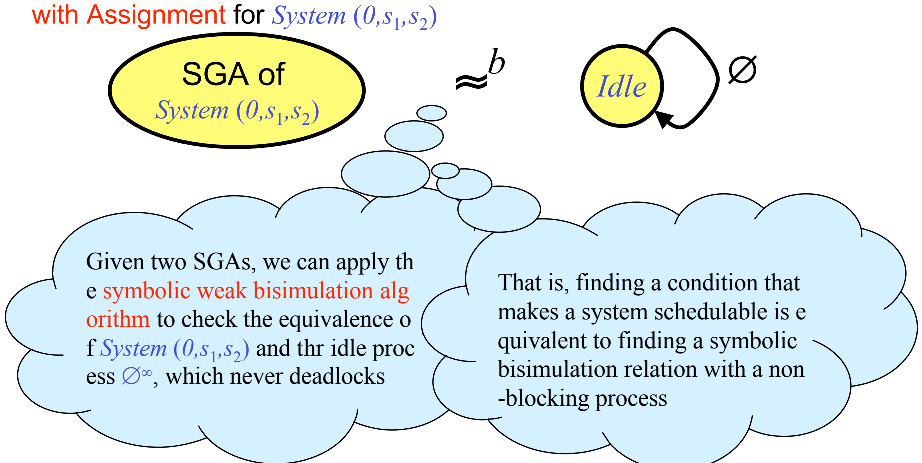
$Q(y) = (a!y,1).nil$



$$X_{PQ}(x,y) = (x < 0 \Rightarrow false) \wedge (x \geq 0 \Rightarrow (true \wedge x+1 = y)) \wedge (true \Rightarrow (x \geq 0 \wedge y = x+1))$$

Schedulability Analysis Using Symbolic Bisimulation

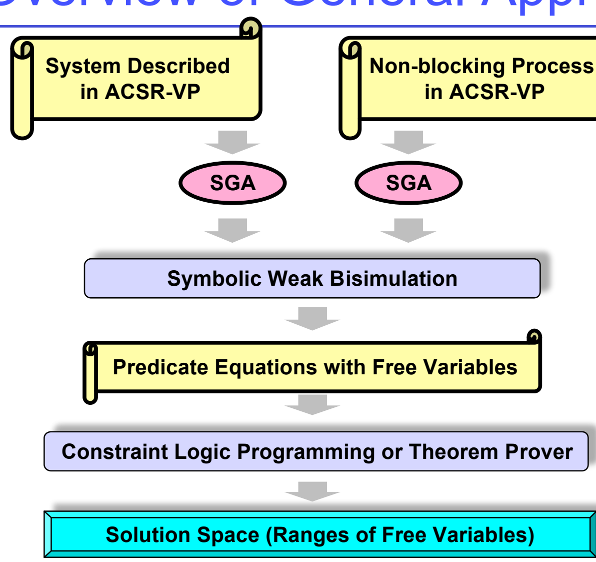
Suppose we have an ACSR-VP term *System* $(0,s_1,s_2)$ that model a real-time system or a scheduling problem. We generate the Symbolic Graph with Assignment for *System* $(0,s_1,s_2)$



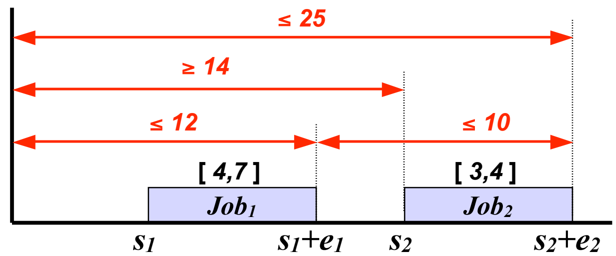
ACSR-VP approach

- Provides a formal framework for modeling real-time systems, especially for real-time scheduling problems such as
  - Priority Assignment Problem
  - Execution Synchronization Problem
    - Start-time assignment problem
    - Period assignment problem
- Deals with unknown parameters in the problems rather than "yes/no" answer (i.e., parametric approach)
- Provides a fully automatic method for the analysis of real-time scheduling problems
- Takes advantages of existing techniques such as integer programming and BDD

Overview of General Approach



Example: Start-time Assignment 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.

Modeling With ACSR-VP

- The following fragments of ACSR-VP describe the start time assignment problem with inter-job temporal constraints

$$Job_i(t,s) = (\, t < s \,) \rightarrow \varnothing : Job_i(t+1,s)$$
$$+ (\, t = s \,) \rightarrow (Start!,1).Job'_i\,(0,t,s)$$

$$Job'_i(e,t,s) = (\, e < e_i^- \,) \rightarrow \{(cpu,1)\}: Job'_i(e+1,t+1,s)$$
$$+ (\, e = e_i^- \,) \rightarrow Job''_i\,(e,t,s)$$

$$Job''_i(e,t,s) = (\, e < e_i^+ \,) \rightarrow \{(cpu,1)\}: Job''_i(e+1,t+1,s)$$
$$+ (\, e \leq e_i^+ \,) \rightarrow (Finished!,1).Idle$$

$$Constraint(t) = (start?,1).Constraint_1(t) + \varnothing : Constraint(t+1)$$
$$Constraint_1(t) = (Finished?,1).Constraint_2(t) + \varnothing : Constraint_1(t+1)$$
$$Constraint_2(t) = (\, t \leq 12 \,) \rightarrow Constraint_3(t,0)$$
$$Constraint_3(t) = \ldots$$

$$System(s_1,\ldots,s_n) = (Job_1(0,s_1)||\ldots||\, Job_n(0,s_n)||Constraint(0))\backslash\{Start,Finished\}$$

Predicate Equations

- The following fragments of predicate equations are generated from the symbolic weak bisimulation algorithm with the infinite idle process

$$
\begin{aligned}
X_0\,(t, s_1, s_2) = {} & (t \le 5 \wedge t < s_2) \rightarrow X_1\,(t{+}1, s_1, s_2) \\
& \wedge (t \le 5 \wedge t = s_1) \rightarrow X_2\,(0, t{+}5, s_2) \\
& \wedge (\ (t \le 5 \wedge t < s_1 \wedge X_1\,(t{+}1, s_1, s_2)) \\
& \quad \vee (t < 5 \wedge t = s_1 \wedge X_2\,(0, t{+}5, s_2))\ ) \\
X_1\,(t, s_1, s_2) = {} & \ldots\ X_2 \ldots \\
X_2\,(e, s_1, s_2) = {} & \ldots\ X_1 \ldots
\end{aligned}
$$

To get the values of s_1 and s_2, we can ask a query $X_0\,(0, s_1, s_2)$

Solution Space

- The solutions to the predicate equations can be obtained using linear/integer programming techniques, constraint logic programming techniques, or a theorem prover.
- The solutions for the previous example are:

Start time S_1	3	4	4	5	5	5
Start time S_2	14	14	15	14	15	16

An Automatic Approach

- The disadvantage of symbolic weak bisimulation is that it requires to add new τ edges into SGA. This will increase the size of predicate equations
- The disadvantage of CLP is that there is no guarantee that it terminates

- Reachability Analysis: Finding a condition that makes a system schedulable is equivalent to finding a condition that guarantees there is always a cycle in an SGA regardless of a path taken
  - No need to add new τ edges
- Restricted ACSR-VP
  - Give syntactic restriction to identify a decidable subset of ACSR-VP
    - Control Variable : in finite range; Values can be changed
    - Data Variable : could be in infinite range; Values cannot be changed
    - $P(x{:}0..100, y) = (x<0 \wedge x+y>10) \rightarrow \emptyset{:}Q(x+3, y)$
  - Generate a boolean expression or boolean equations (i.e., no need to use CLP)

Conclusions: resources

- We have presented a family of resource-bound process-algebraic formalisms
  - the notion of a resource plays central role
    - Abstractions of physical resources
    - Resource sharing: coordination and synchronization
    - Resource consumption takes time: real-time behavior
    - Resource failures: probabilistic behavior
- Sample application domain: analysis of scheduling problems
  - Other domains: protocol analysis, rapid prototyping

Conclusions: analysis techniques

- Analysis of safety properties by means of deadlock detection
- Conformance analysis by means of equivalence and preorder checking
- Probabilistic analysis techniques:
  - Model checking
  - Resource utilization
- Parametric analysis in ACSR-VP

Extensions

- Presented: serially reusable resources with access constraints
- Other types of resources:
  - Consumable resources: each resource use depletes resource stock
  - Multi-capacity resources: allow simultaneous access by a limited number of processes
- Other kinds of resource constraints:
  - non-functional constraints such as memory, power consumption, weight, etc.

References

- "A Process Algebraic Approach to the Specification and Analysis of Resource-Bound Real-Time Systems," Insup Lee, Patrice Br\'emond-Gr\'egoire and Richard Gerber, Proceedings of the IEEE, Jan 1994, pp. 158-171.
- "A Complete Axiomatization of Finite-state ACSR Processes," Patrice Br\'emond-Gr\'egoire, Jin-Young Choi and Insup Lee, Information and Computation, 138 (2), Nov 1997, pp. 124-159.
- " Process Algebra of Communicating Shared Resources with Dense Time and Priorities," by Patrice Br\'emond-Gr\'egoire and Insup Lee, Theoretical Computer Science, 189, 1997, pp. 179-219.
- "A Process Algebraic Approach to the Schedulability Analysis of Real-Time Systems," Han\^ene Ben-Abdallah, Jin-Young Choi, Duncan Clarke, Young Si Kim, Insup Lee and Hong-Liang Xie, Real-time Systems, 15, 1998, pp. 189-219.
- "Probabilistic Resource Failure in Real-Time Process Algebra," Anna Philippou, Oleg Sokolsky, Insup Lee, Rance Cleaveland, and Scott Smolka, CONCUR '98, Sept 1998.
- "Specification and Analysis of Real-Time Systems with PARAGON," Oleg Sokolsky, Insup Lee, and Han\^ene Ben-Abdallah, Annals of Software Engineering, 7, 1999, pp. 211-234.
- "Modeling and Analysis of Power-Aware Systems," Oleg Sokolsky, Anna Philippou, Insup Lee, and Kyriakos Christou, TACAS 2003, April 2003.
- These papers are also available on-line from www.cis.upenn.edu/~rtg/papers.php3.

Thanks

- … for invitation to ESSES 2003
- … for fundamental work done by my former Ph.D. students:
  - Amy Zwarico
  - Rich Gerber
  - Patrice Bremond-Gregoire
  - Hanene Ben-Abdallah
  - Duncan Clark
  - Hee Hwan Kwak
- … for support from ARO, NSF, ONR over a number of years

Q & A