Model-Based Scheduler Analysis

Bengt Jonsson
Leonid Mokrushin
Wang Yi
Uppsala University, Sweden

With contributions from
Tobias Amnell, Elena Fersman, Pavel Krcal, Paul Pettersson

Lorentz Workshop 2005, Leiden

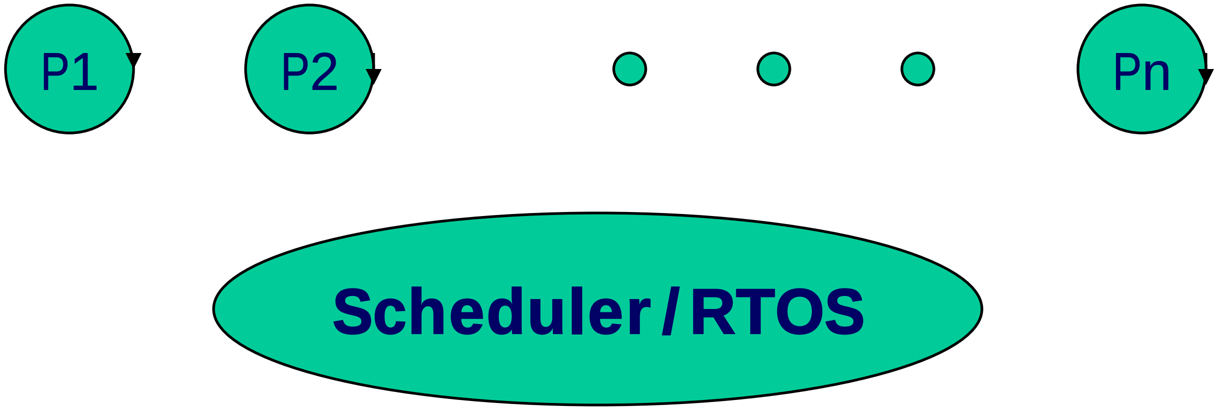
OUTLINE

Scheduler analysis as Model Checking of Timed Systems

- A Unified Model for Timed Systems
  - Timed automata with tasks
- Scheduling Analysis by Model Checking of Timed Systems (w. UPPAAL)
  - Additional trick to handle preemption
  - Limits to decidability
- TIMES tool
- Preliminary ideas on achieving modularity

Classical approach to Real Time Scheduling

- Controller = a set of periodic tasks + a scheduler



- Well-developed techniques, e.g., Rate-Monotonic Scheduling

The Periodic Task Model

+ Simple to analyze (Rate-Monotonic Analysis)

- Assumption too simplistic for many systems
  - May give too pessimistic analysis results
  - ”Real” systems have
    - Shared resources, process synchronization, communication, precedence constraints, complex timing (modes, jitter, ...)
  - Adding these features complicates the model, and leads to an explosion in ”special cases”

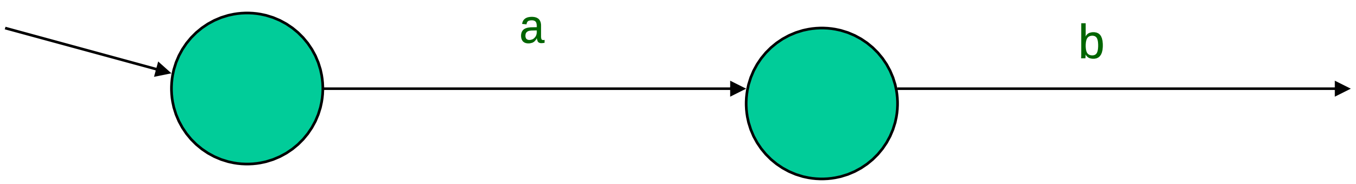
The Periodic Task Model

+ Simple to analyze (Rate-Monotonic Analysis)

- Assumption too simplistic for many systems
  - May give too pessimistic analysis results
  - ”Real” systems have
    - Shared resources, process synchronization, communication, precedence constraints, complex timing (modes, jitter, ...)
  - Adding these features complicates the model, and leads to an explosion in ”special cases”

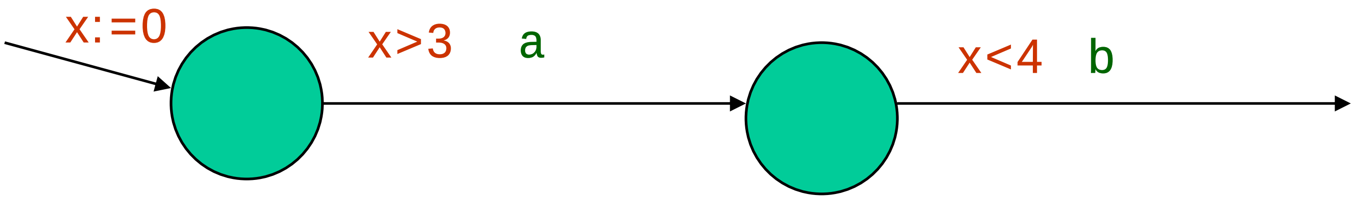
- Wanted: uniform framework to model a variety of patterns in timed systems.
- Proposal: Timed Automata

Timed Automata



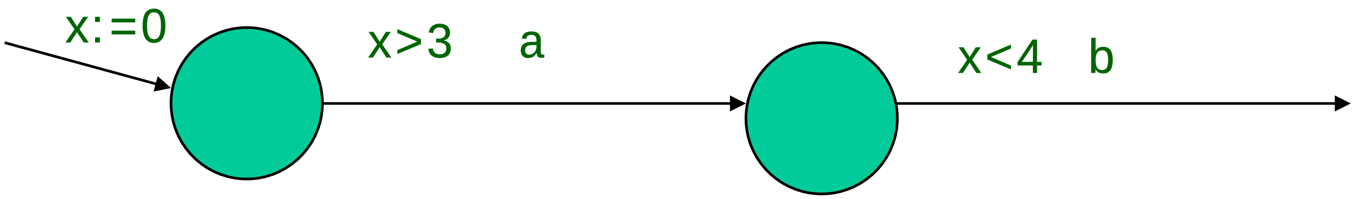
- Based on standard automata

Timed Automata



- Based on standard automata
- Clocks give upper and lower bounds on distance in time between occurrences of symbols.
- Temporal properties of Timed Automata (reachability, LTL, ...) can be model-checked (PSPACE-complete)
- Implemented in tools (UPPAAL, IF/Kronos)

Timed Traces of TA



(3.3, a) (3.4, b),

(6.5, a),

(3.6, a) (3.9, b),

(3.14, a) (3.14159, b)

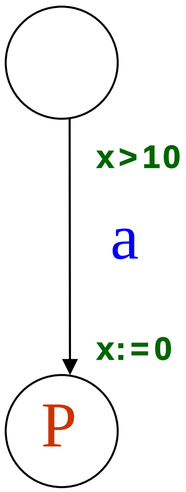
… …

Using Timed Automata to model Real Time Systems

- Arrival pattern of tasks modeled by Timed Automata
  - Extend TA with task spawning
- Computation time of tasks modeled by clock
  - Assume no preemption for now
- Deadlines modeled by clocks
  - Expiration leads to ”error state”
- Include processor and task queue in the analysis
- Precedence, ..., can be modeled by additional synchronizations

Timed Automata with Tasks

- Events
  - synchronization
  - interrupts,
  - passing of time
- Timing constraints
  - specifying event arrivals
  - e.g., periodic and sporadic
- Tasks (executable programs)
  - Internal computation (need not be modeled)
  - Released by a TA transition, and scheduled in the ready queue of RTOS

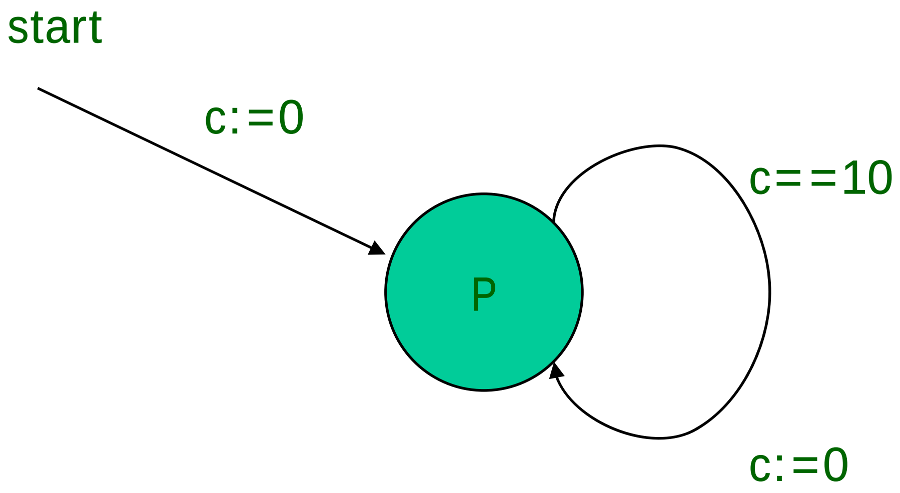


Timed Automaton

+ tasks

- Tasks have parameters:
  - C: WCET
  - D: Relative deadline
  - (other parameters for scheduling, e.g., priority)

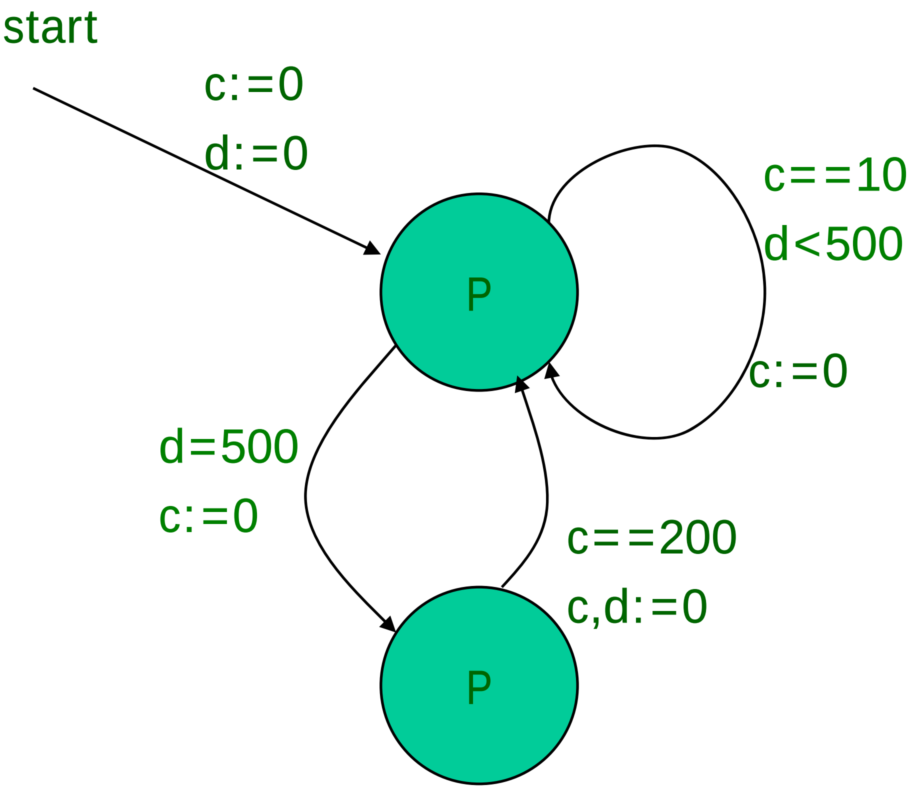
Example: periodic task



c: clock

P: task

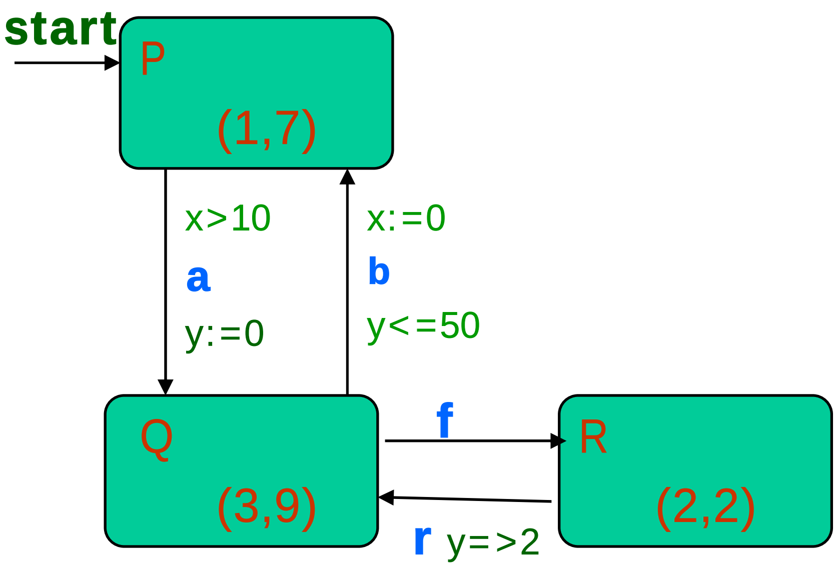
Example: periodic task with modes



c,d: clocks

P: task

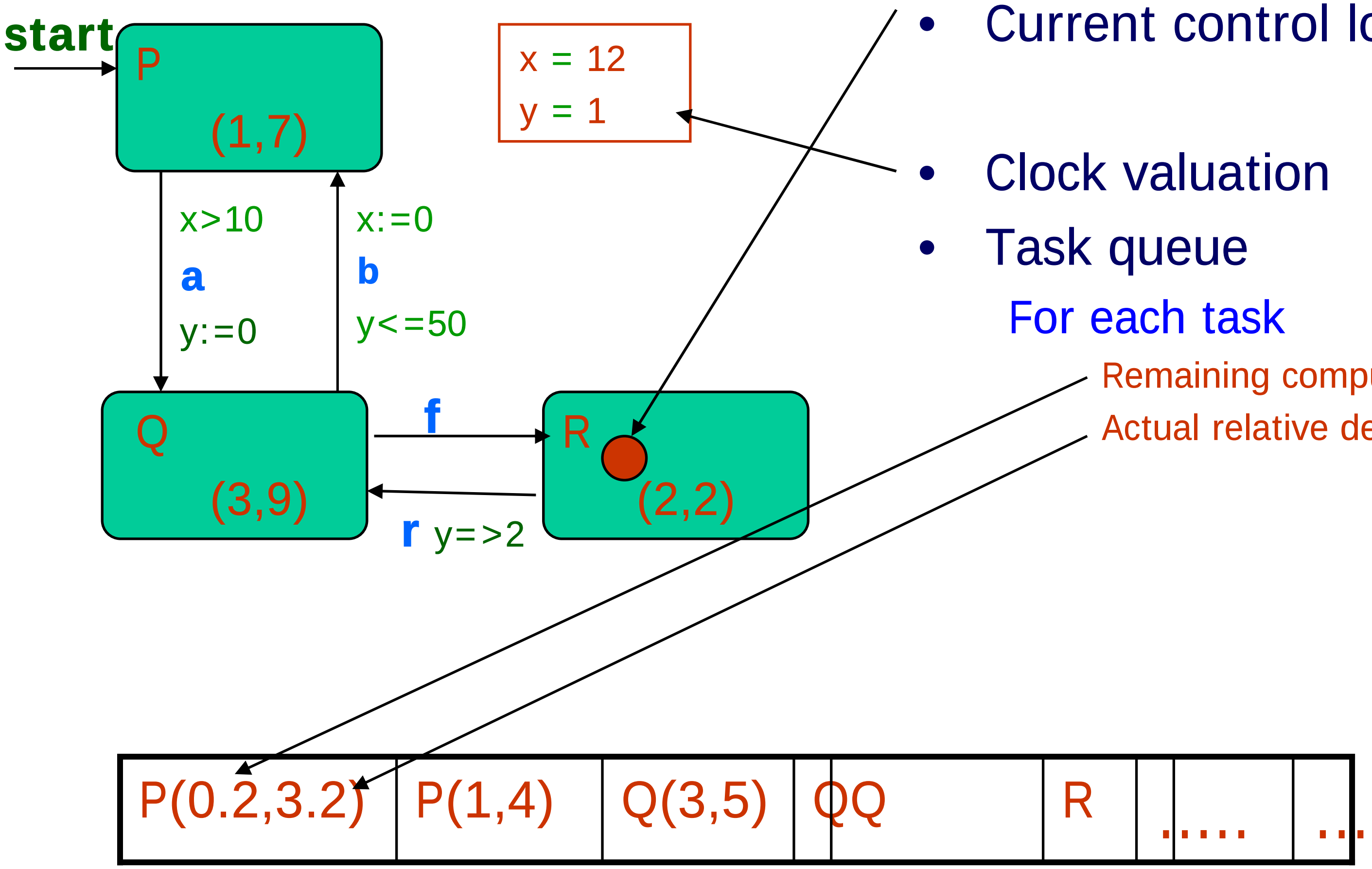
Timed Automata with Tasks (Structure of Operation)



- "Processor" 1 (task generator)
  - Initially, P is released
  - Forever, do
    - Whenever **a** is available and x>10, Q is released
    - Then
      - Whenever **b** is available and y<=50, P is released
      - Whenever **f** appears, it releases R
- "Processor" 2 (task handler)
  - Scheduling and Computing tasks in the queue

P	P	Q	Q	Q	R	...	...	...

States/Configurations of Model



P(0.2,3.2)	P(1,4)	Q(3,5)	QQ	R	.....	...

- Current control location
- Clock valuation
- Task queue

For each task

Remaining computation time

Actual relative deadline

Operations to Model Scheduling

- The scheduling algorithm (EDF, FP, FIFO, ...) is modeled by sorting policy on the task queue
- Task processing modeled by decreasing remaining computation times and relative deadlines

Example:

$$[Q(4, 7), P(2, 10)] \xrightarrow{\text{wait}(0.5)} [Q(3.5, 6.5), P(2, 9.5)]$$

SCHEDULING ANALYSIS

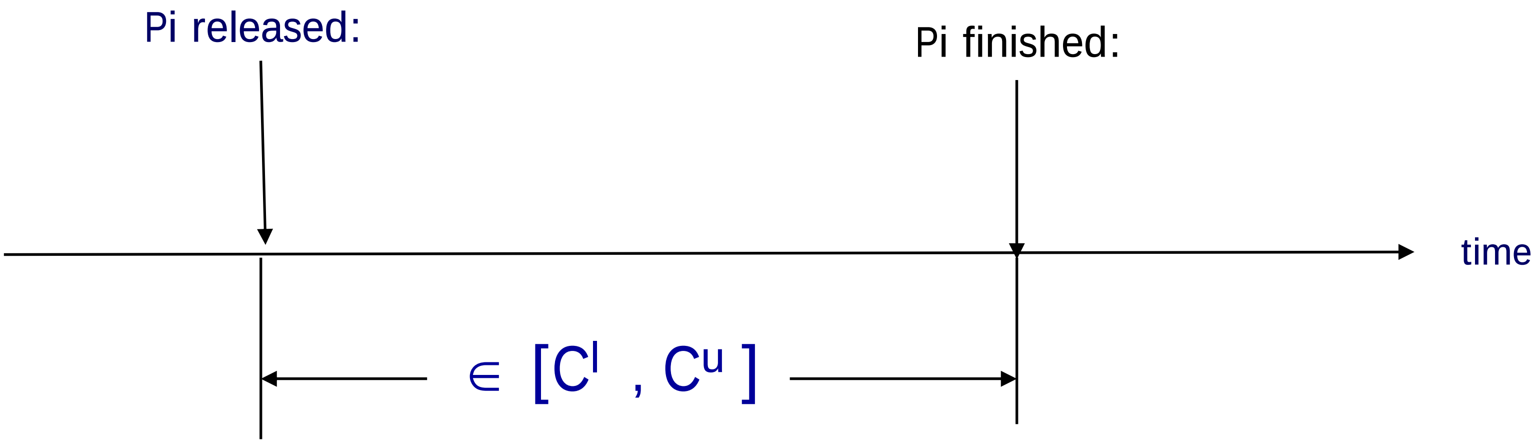
Schedulability by model checking

Assume a scheduling policy Sch:

- A configuration is schedulable with Sch if it is possible to meet all relative deadlines (simple calculation on occuring c_i d_i in task queue)
- An automaton is schedulable with Sch if all its reachable states are schedulable
- Schedulability checking == reachability analysis
  - set of schedulable configurations is bounded (modulo clocks)

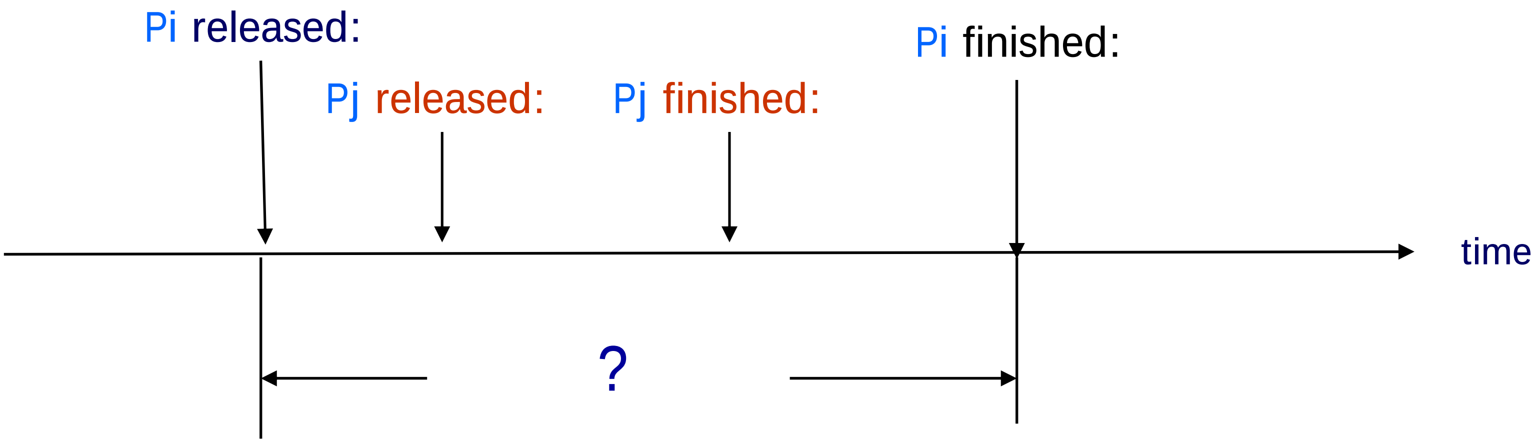
Handling preemption

- Assume Pj preempts Pi
- Assume computation time of Pi between C^l and C^u



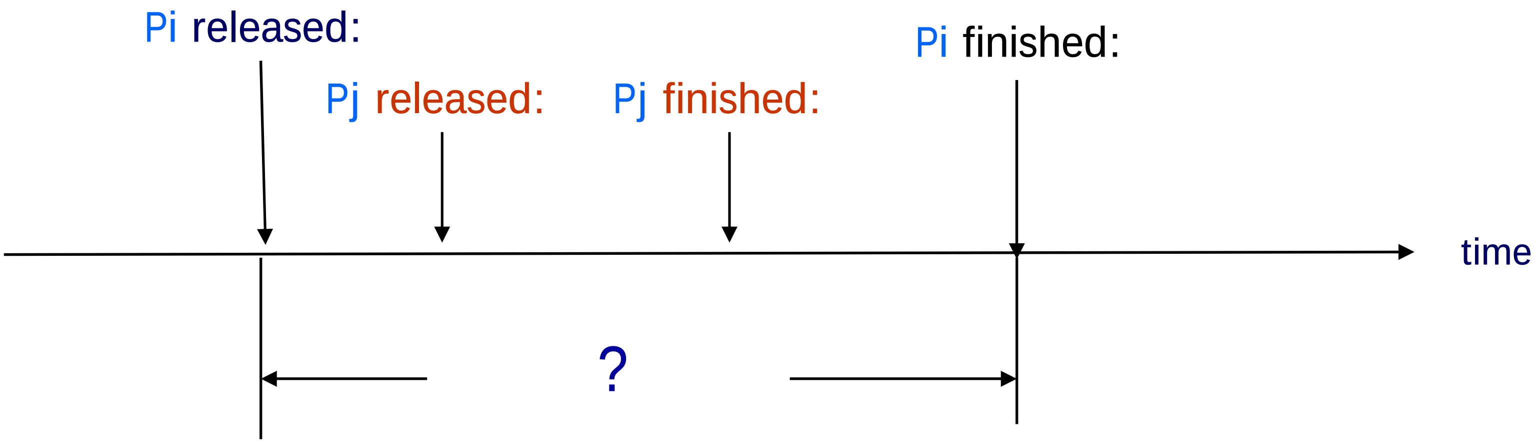
Handling preemption

- Assume Pj preempts Pi
- Assume computation time of Pi between C^l and C^u



Handling preemption

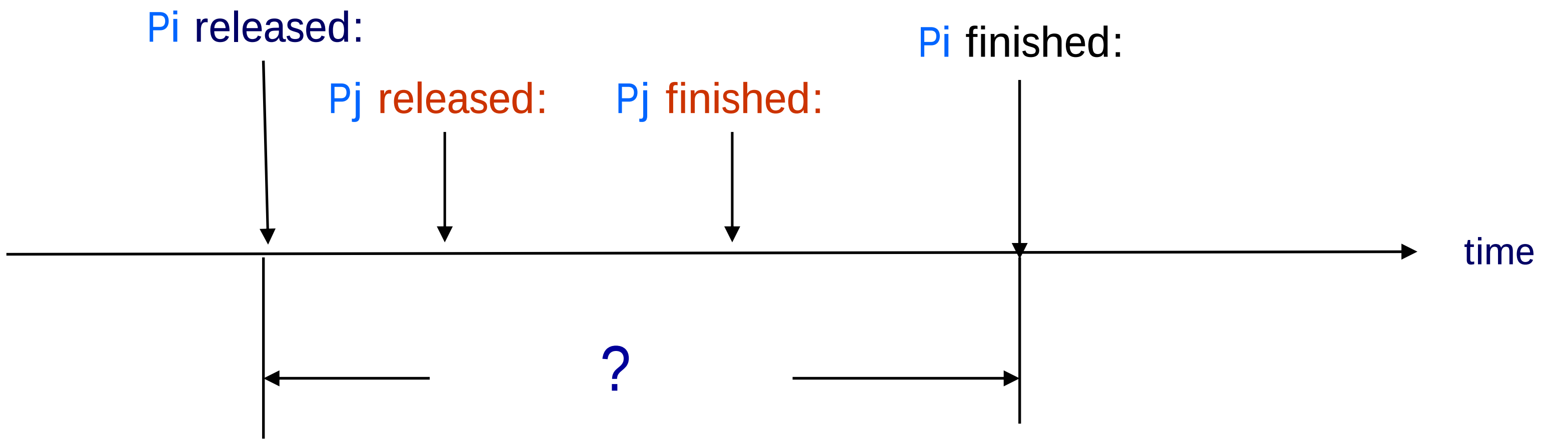
- Assume Pj preempts Pi
- Assume computation time of Pi between C^l and C^u



- ? Is an interval if computation time of Pj is known & constant

Handling preemption

- Assume Pj preempts Pi
- Assume computation time of Pi between C^l and C^u



- ? Is an interval if computation time of Pj is known & constant
- if computation time of Pj may vary, timing properties cannot be precisely modeled with timed automata

Decidability results (summary)

- For Non-preemptive scheduling, scheduling can be analyzed by model checking TAs. [Ericsson,Wall,Yi 98]
- For preemptive scheduling, the problem can be solved using BSA (Bounded Substraction Automata) [Fersman,Pettersson,Yi, TACAS02]
  - (#extra clocks needed is 2 x #instances = $2 \sum_i \lceil Di/Ci \rceil$)
- For fixed-priority scheduling, the problem can be solved using TA with only 2 extra clocks – similar to the classic RMA technique (Rate-Monotonic Analysis) [Fersman,Mokrushin, Pettersson,Yi, TACAS03]
- Problem becomes undecidable with preemption if both
  - the execution times of tasks are intervals,
  - task completion times influence task release times [Krcal,Yi, TACAS 04]