Process Algebraic Schedulability Analysis of Real-Time Systems

Oleg Sokolsky           Insup Lee
Real-Time System Group
University of Pennsylvania

SAE AADL Working Group Meeting
October 19, 2005
Outline

• Motivation
  - Response time vs. state-space exploration
• Introduction to ACSR
  - Recipe for real-time modeling
• Schedulability analysis with ACSR
  - Tasks
  - Scheduling policies
Motivation

• **Schedulability analysis based on response time**
  - Schedulable if $r_i \cdot d_i$ for every task $i$
    • [Joseph and Pandya 1986]
    • [Audsley et al. 1993]
  - Calculating response time becomes more complex with interdependencies between tasks

• **Alternative: state space exploration**
  - Schedulable if no state violating timing constraints is reachable
    • [Lee et al. 1995]
  - Incorporate interdependencies in a uniform way
  - Efficient state exploration engines exist
Ingredients for real-time modeling

- Task structure
  - Task states
  - Required resources
  - Input/output signals

- Timing constraints
  - Dispatch policy, period
  - Deadlines

- Priorities and scheduling
Modeling basics: events and actions

- Process: a modeling unit
- Steps of a process
  - (Logically) instantaneous events
  - Timed actions
- Events are used for communication
  - Inputs, outputs, and internal: $a? b! \tau$
- Actions require resource access
  - Take one or more units of time
Modeling basics: processes

• Sequential execution
  - $P_1$ performs a step and becomes $P_1'$

• Choice of steps
  - $P_2$ can input an event or idle
Modeling basics: time progress

- Timing model
  - Time is global
  - All concurrent processes need to pass time together
  - Passing time is an explicit choice
    - $P_1$ cannot pass time, but $P_2$ can
Timeouts and interrupts

• Execution can be abandoned by time progress or external events
Task skeleton

- A preemptable task $T$ with execution time $[c_{\text{min}}, c_{\text{max}}]$
Task skeleton

- A non-preemptable task $T$ with execution time $[c_{\text{min}}, c_{\text{max}}]$
Task activation

- An activator process invokes the task and keeps track of deadlines
  - Periodic activation with period $p$ and deadline = period
  - Aperiodic activation by the completion of task $T'$ with deadline $d$
Parallel composition

- Event synchronization

\[
P_1 \xrightarrow{\text{go!}} P_1' \parallel P_2 \xrightarrow{\tau} P_1'||P_2'^\prime
\]

- Time synchronization

\[
P_1 \xrightarrow{\{\text{cpu}\}} P_1' \parallel P_2 \xrightarrow{\{\text{bus}\}} P_1'||P_2'^\prime
\]
Resource conflicts

- Resources are used exclusively

- Alternatives must be provided

\[ P_1 \rightarrow \{cpu\} \rightarrow P_1' || P_2 \rightarrow \{cpu\} \rightarrow P_2' \]

\[ P_1 \rightarrow \{cpu\} \rightarrow P_1' || P_2 \rightarrow \{cpu\}, \{bus\} \rightarrow P_2' \]

\[ P_1 || P_2 \rightarrow \{cpu\}, \{bus\} \rightarrow P_1'||P_2'' \rightarrow \{cpu\}, \{bus\} \rightarrow P_2'' \]

\[ P_1 || P_2 \rightarrow \{cpu\} \rightarrow P_1'||P_2'' \rightarrow \{bus\} \rightarrow P_1'||P_2'' \]

\[ P_1 || P_2 \rightarrow \{cpu\} \rightarrow P_1'||P_2'' \rightarrow \{bus\} \rightarrow P_1'||P_2'' \]

\[ P_1 || P_2 \rightarrow \{cpu\} \rightarrow P_1'||P_2'' \rightarrow \{bus\} \rightarrow P_1'||P_2'' \]

\[ P_1 || P_2 \rightarrow \{cpu\} \rightarrow P_1'||P_2'' \rightarrow \{bus\} \rightarrow P_1'||P_2'' \]

\[ P_1 || P_2 \rightarrow \{cpu\} \rightarrow P_1'||P_2'' \rightarrow \{bus\} \rightarrow P_1'||P_2'' \]

\[ P_1 || P_2 \rightarrow \{cpu\} \rightarrow P_1'||P_2'' \rightarrow \{bus\} \rightarrow P_1'||P_2'' \]

\[ P_1 || P_2 \rightarrow \{cpu\} \rightarrow P_1'||P_2'' \rightarrow \{bus\} \rightarrow P_1'||P_2'' \]
Priorities and preemption

- **Access to resources in action steps and to event channels is controlled by priorities:**
  \[ \{(r_1,p_1),(r_2,p_2)\} \quad (e?,p) \]

- **Preemption relation on events and actions** -
  - \( \{(cpu,1),(bus,2)\} - \{(cpu,2)\} \)
  - \( \{(cpu,1),(bus,2)\} - (\tau,1) \)
Scheduling with priorities

- Priorities in a task reflect scheduling policy
- Static or dynamic priorities
  - A task with EDF priorities:

\[
T \xrightarrow{\text{Await\_Dispatch}} \xrightarrow{\text{dispatch}_T?} \xrightarrow{e := 0, t := 0} \xrightarrow{\text{compute}_T!} \xrightarrow{\{\text{cpu}\}}
\]

\[
\text{Compute}_{e,t} \xrightarrow{t := t + 1} \xrightarrow{\{\}} \xrightarrow{e < c_{\text{max}} - 1} \xrightarrow{c_{\text{min}} \leq e < c_{\text{max}} + 1} \xrightarrow{t := t + 1, e := e + 1} \xrightarrow{\{\text{cpu}, d_{\text{max}} - (d_T - t)\}}
\]
Enforcing progress: resource closure

- Resource-constrained progress
  - Processes should not wait unnecessarily
- In a closed system, processes have exclusive use of system resources

\[
\begin{align*}
\text{P}_1 & \rightarrow \text{P}_1' \\
\text{P}_2 & \rightarrow \text{P}_2'
\end{align*}
\]
Schedulability analysis

• Detect two kinds of problems:
  - Resource conflicts
  - Timing violations
• Schedulable systems are deadlock-free
• Analysis method:
  - Deadlock detection
  - Efficient methods for state-space exploration exist
  - Execution trace to a deadlocked state is produced
Summary

• Formal modeling based on ACSR allows schedulability analysis of different task models and scheduling approaches
  - Complicated precedence constraints
  - Static and dynamic priorities, priority inheritance, etc.
  - End-to-end timing constraints

• If task processes are detailed enough, functional verification can be done, too