Schedulability Analysis of AADL Models

Oleg Sokolsky         Insup Lee
University of Pennsylvania
Duncan Clarke
Fremont Associates

Overview

- AADL modeling language
  - Why is it useful and what it has
- Formal schedulability analysis
  - Introduction to ACSR
  - Modeling task sets
- Translating AADL into ACSR
Embedded system architectures

- Both hardware and software aspects are important
  - Increasingly distributed and heterogeneous
- Analysis is important
  - Fast design space exploration
- Some behavioral information needed for analysis
- Tight resource and timing constraints
- Multimodal behaviors
  - E.g., fault recovery

AADL – ADL for embedded systems

- Architecture Analysis and Design Language
- Oriented towards modeling embedded and real-time systems
  - Hardware and software components
    - Threads, data, processors, buses, memory
  - Control, data, and access connections
- Semi-formal execution semantics in terms of hybrid automata
- SAE standard AS-5506
Component interfaces (types)

- **Features**
  - Points for external connections
    - E.g., data ports

- **Flows**
  - End-to-end internal connections

- **Properties**
  - Attributes useful for analysis

Component implementations

- **Internal structure of the component**
  - Subcomponents are type references
  - Connections conform with flows in the type
  - External features conform with the type
  - Internal features conform with subcomponent types
Features and connections

- Communication
  - Ports and port groups
  - Port connections
- Resource access
  - Required and provided access
  - Access connections
- Control
  - Subprogram features
  - Parameter connections

Thread components

- Thread represents a sequential flow of control
  - Can have only data as subcomponents
- Threads are executable components
  - Execution goes through a number of states
    - Active or inactive
  - Behaviors are specified by hybrid automata
Thread states

Thread Hybrid Automata
Thread dispatch

- Periodic threads are dispatched periodically
  - Event arrivals are queued
- Non-periodic threads are dispatched by incoming events
- Pre-declared ports
  - Event in port Dispatch
    - If connected, all other events are queued
  - Event out port Complete
    - Can implement precedence

Component properties

- Thread
  - Dispatch protocol
    - periodic, aperiodic, sporadic, or background
  - Period
    - For periodic and sporadic threads
  - Execution time range and deadline
    - for all execution states separately (initialize, compute, activate, etc.)
- Processor
  - Scheduling protocol
Component bindings

- Software components are bound to platform components
- Binding mechanism:
  - Properties specify allowed and actual bindings
    - Allows for exploration of design alternatives

Formal schedulability analysis

- Translation of AADL model into ACSR
- Search for deadlocks in ACSR model

State space exploration
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 an event and becomes $P_1'$;
  - $P_1'$ performs an action 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

$P_1 \rightarrow \text{go?} \rightarrow \{\text{compute}\}$

$P_2 \rightarrow \text{go?} \rightarrow \{\} \rightarrow \{\text{compute}\}$

Timeouts and interrupts

- Execution can be abandoned by time progress or external events

$P_2 \rightarrow \text{go?} \rightarrow \{\} \rightarrow \{\text{compute}\}$

$P_i \rightarrow \text{stop?} \rightarrow P_i$
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

  \[
  P_1 \xrightarrow{\text{go!}} P_1' \quad \parallel \quad P_2 \xrightarrow{\text{go?}} P_2'
  \]

  \[
  P_1 || P_2 \xrightarrow{\tau} P_1'||P_2'
  \]

• Time synchronization

  \[
  P_1 \{\text{cpu}\} \xrightarrow{\ } P_1' \quad \parallel \quad P_2 \{\text{bus}\} \xrightarrow{\ } P_2'
  \]

  \[
  P_1 || P_2 \{\text{cpu,bus}\} \xrightarrow{\ } P_1'||P_2'
  \]
Resource conflicts

- **Resources are used exclusively**

    ![Diagram illustrating resource conflicts]

- **Alternatives must be provided**

    ![Diagram illustrating alternative provision]

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** -
  - \( ((cpu,1),(bus,2)) \rightarrow ((cpu,2)) \)
  - \( ((cpu,1),(bus,2)) \rightarrow (\tau,1) \)

    ![Diagram illustrating priorities and preemption]

2/16/09  WPDRTS 2006 *RTG
Scheduling with priorities

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

Enforcing progress: resource closure

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

Translation of AADL into ACSR

- For each thread
  - generate skeleton
    • thread states
    • resources and dependencies (thread connections)
  - populate skeleton
    • timing: period, deadlines (thread properties)
    • events to raise (out event connections)
  - generate activator (dispatch policy property)
- For each processor
  - generate priorities for mapped threads
    • scheduling policy (processor property)
Summary

- AADL models hardware/software architectures for embedded systems
- 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