



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

2/16/09



WPDRS 2006



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

2/16/09

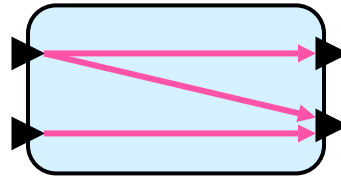


WPDRS 2006



Component interfaces (types)

- Features
 - Points for external connections
 - E.g., data ports
- Flows
 - End-to-end internal connections
- Properties
 - Attributes useful for analysis



2/16/09

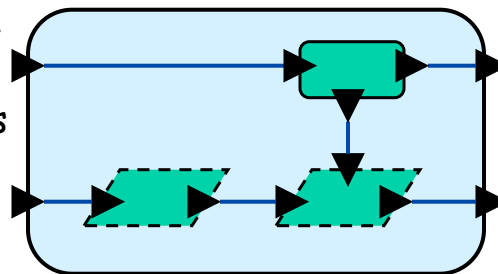


WPDRTS 2006



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



2/16/09

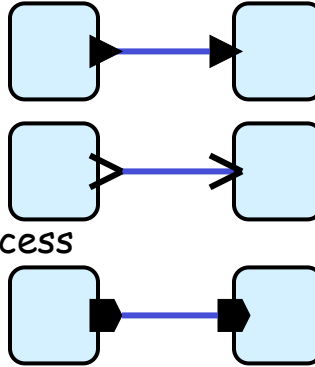


WPDRTS 2006



Features and connections

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



2/16/09



WPDRS 2006



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

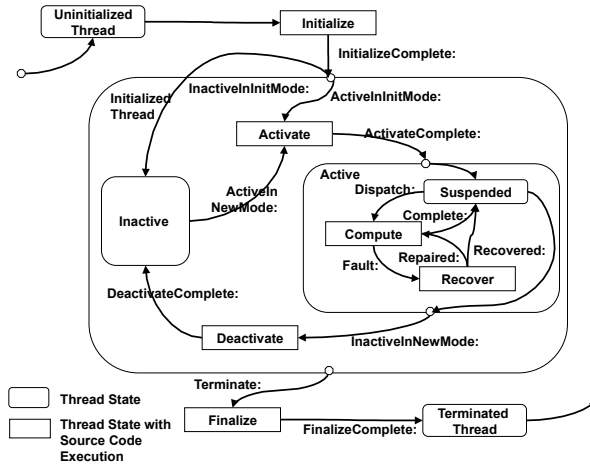
2/16/09



WPDRS 2006



Thread states



Courtesy Peter Feiler

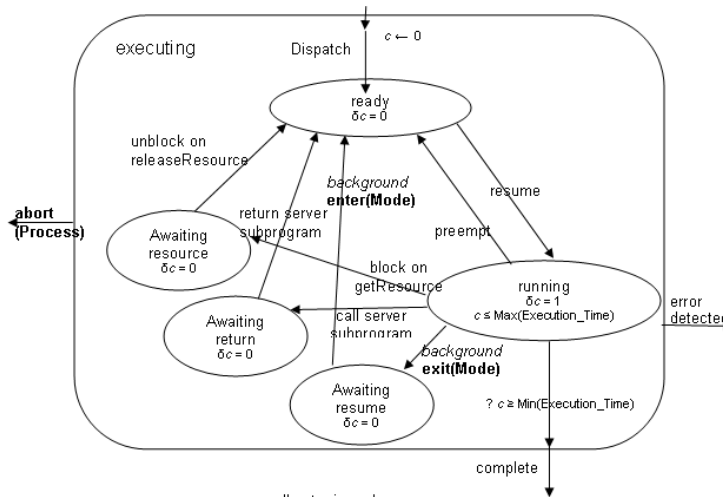
2/16/09



WPDRS 2006



Thread Hybrid Automata



on all outgoing edges:
assert $t \leq \text{Deadline}$

2/16/09



WPDRS 2006



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



2/16/09



WPDRTS 2006



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

2/16/09

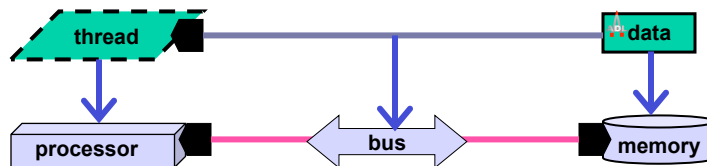


WPDRTS 2006



Component bindings

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



2/16/09

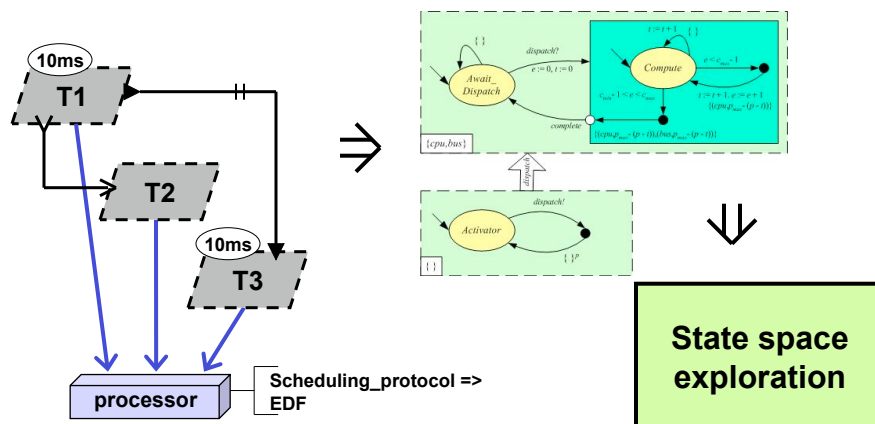


WPDRTS 2006



Formal schedulability analysis

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



2/16/09



WPDRTS 2006



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: $a?$ $b!$ τ
- Actions require resource access
 - Take one or more units of time

2/16/09

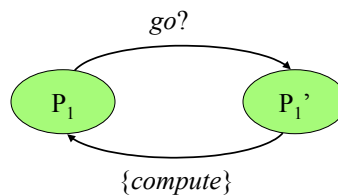


WPDRS 2006

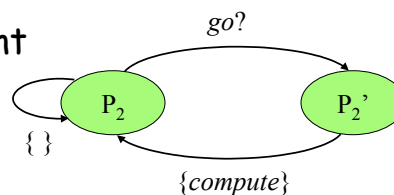


Modeling basics: processes

- 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



2/16/09

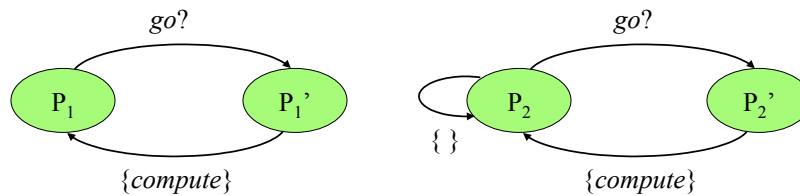


WPDRS 2006



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



2/16/09

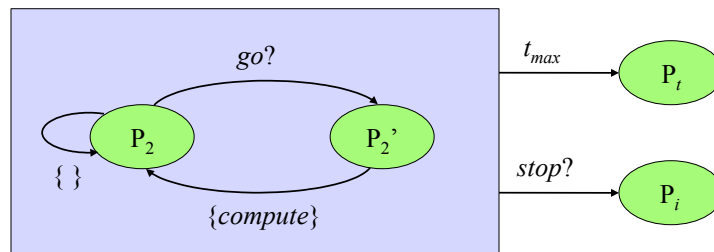


WPDRTS 2006



Timeouts and interrupts

- Execution can be abandoned by time progress or external events



2/16/09

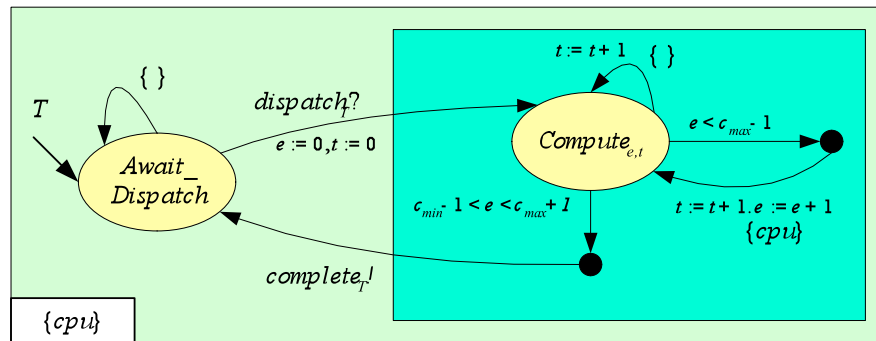


WPDRTS 2006



Task skeleton

- A **preemptable** task T with execution time $[c_{min}, c_{max}]$



2/16/09

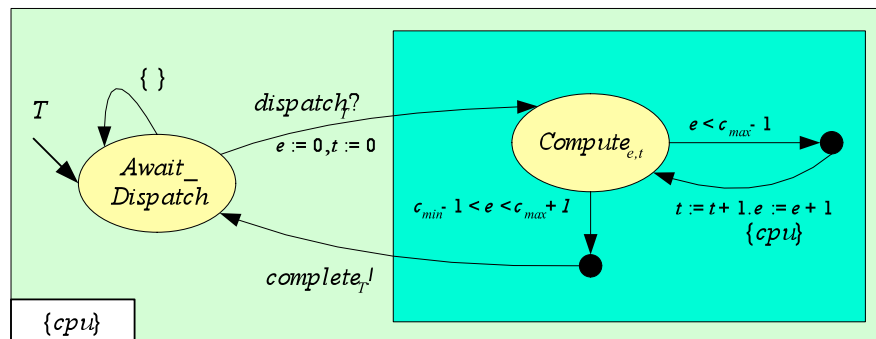


WPDRS 2006



Task skeleton

- A **non-preemptable** task T with execution time $[c_{min}, c_{max}]$



2/16/09



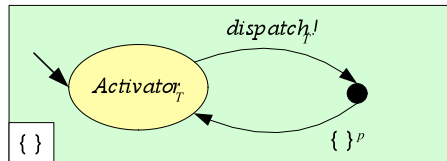
WPDRS 2006



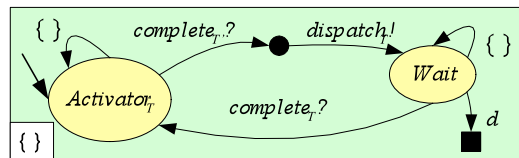
Task activation

- 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



2/16/09

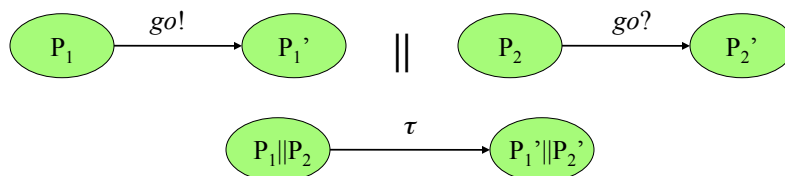


WPDRS 2006

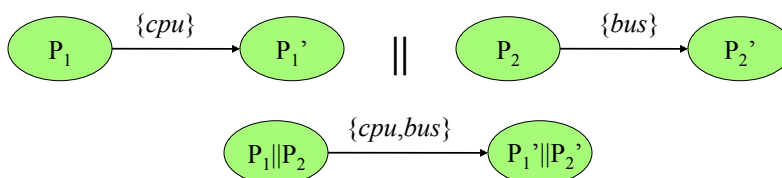


Parallel composition

- Event synchronization



- Time synchronization



2/16/09

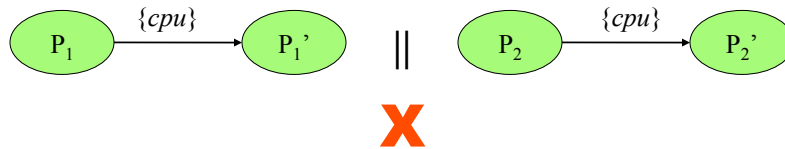


WPDRS 2006

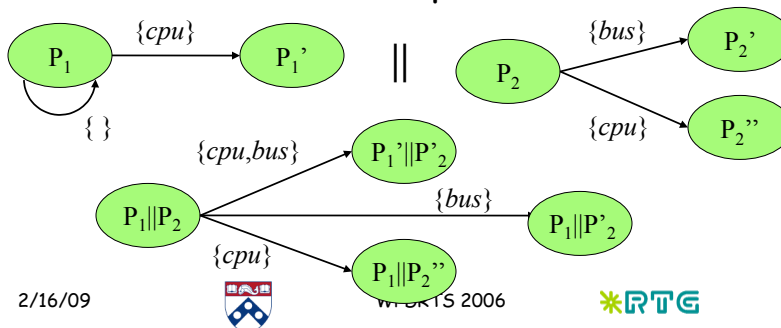


Resource conflicts

- Resources are used exclusively



- Alternatives must be provided



2/16/09

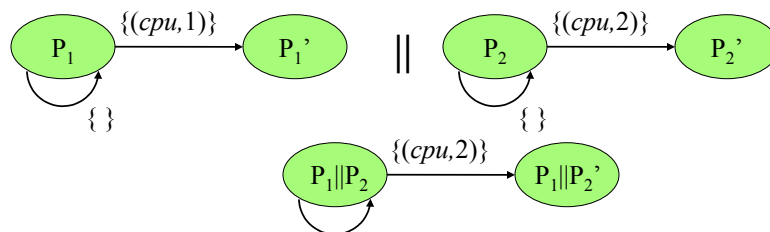


WADRTS 2006



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 -
 - $\{(cpu, 1), (bus, 2)\} - \{(cpu, 2)\}$
 - $\{(cpu, 1), (bus, 2)\} - (\tau, 1)$



2/16/09

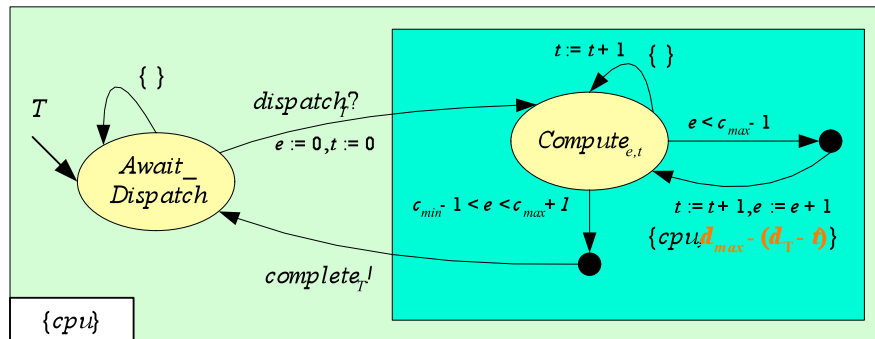


WADRTS 2006



Scheduling with priorities

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



2/16/09

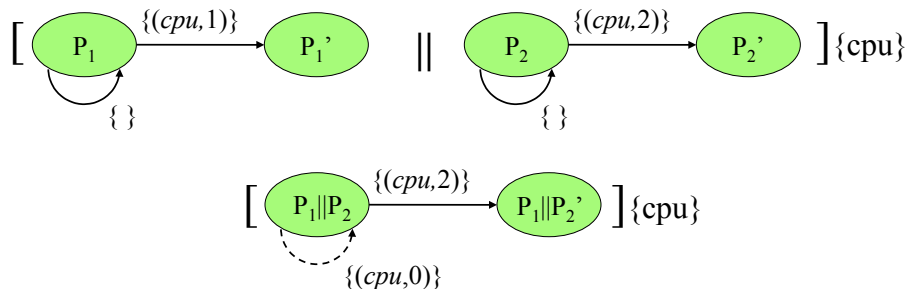


WPDRS 2006



Enforcing progress: resource closure

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



2/16/09



WPDRS 2006



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

2/16/09



WPDRS 2006



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)

2/16/09



WPDRS 2006



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

2/16/09



WPDRTS 2006

