Heterochronous Dataflow in Ptolemy II

Brian K. Vogel
vogel@eecs.berkeley.edu

EE249 Project Presentation, Dec. 4, 1999
Outline

- Project goals
- Ptolemy II background
- Heterochronous dataflow (HDF) semantics
- Current results
- Future work
- Demo
- References
Project Goals

- Implement the heterochronous dataflow (HDF) MOC in Ptolemy II
  - Extend the FSM domain to allow interaction with SDF and HDF domains
  - Implement *charts interaction semantics for SDF with hierarchical FSMs
  - Implement *charts interaction semantics for HDF with hierarchical FSMs
- Create an interesting Ptolemy demo using SDF/HDF with hierarchical FSMs
Ptolemy II Background

- Software supporting heterogeneous, concurrent modeling and design.
- Component view of design
- Models of computation are implemented in Ptolemy II domains
- Written in Java
Heterochronous dataflow (HDF) semantics

- HDF [1] is a generalization of cyclo-static dataflow (CSDF) and also of static dataflow (SDF)
  - An HDF actor has a finite number of type signatures
  - The order in which type signatures are used is, in general, not cyclic
Heterochronous dataflow (HDF) semantics (cont.)

- An HDF actor has an initial type signature when execution starts.
- Can solve balance equations => find an iteration.
- The type signature is only allowed to change after the last firing of an iteration.
Heterochronous dataflow (HDF) semantics (cont.)

- It may be interesting to combine HDF with FSMs. An interaction semantics of HDF with FSMs is specified in *charts* [1]
- *charts* is a family of MOCs describing the semantics of hierarchical FSMs combined with multiple concurrency models
- In *charts*, an HDF actor can refine to an FSM
  - The type signature is determined by the current state
  - The type signature may change when a state transition occurs
  - A state may refine to an SDF diagram, HDF diagram, or another FSM (no restriction on number of levels in the hierarchy)
Heterochronous dataflow (HDF) properties

- HDF is more expressive than SDF but retains desirable decidability properties
  - Static schedules are possible
  - Deadlock, bounded memory are decidable
- A potential drawback is that a global solution to the balance equations determines when a state transition can occur (could be difficult/non-intuitive to use)
HDF example

HDF

FSM

SDF

HDF

FSM

SDF
Current results

- Completed a preliminary implementation in Ptolemy II
  - SDF with hierarchical FSMs
  - HDF with hierarchical FSMs
- Appears to be fully functional, but more testing is needed to verify this
  - Follows the semantics of [1]
  - HDF, SDF, FSM can be combined, with arbitrary level of hierarchy
  - Ptolemy II expression language is used to specify state transition guard expressions
Future work

- **Optimizations**
  - Investigate precomputing all schedules before execution begins (currently construct schedules dynamically)
  - Optimize token transfers
- Allow HDF actors to have 0-rate ports
- Construct more test cases to verify that the implementation is correct
- Create interesting demo applications that are a good match for HDF
Demo

- Hierarchical FSMs with SDF: Hysteresis
References