Java, Threads, and Ptolemy

A New Direction for Ptolemy

Ptolemy Miniconference

14 March 1997

Dick Stevens
Naval Research Laboratory
Washington, DC

stevens@ait.nrl.navy.mil
dstevens@eecs.berkeley.edu

Why Java?

• Because Java is
  • Portable
  • Distributable (Applets)
  • Cleaner than C++
  • Threaded
  • Buzzword Compliant

• Itcl complements Java
  • Java is low-level
  • Itcl is high-level & scripted
Building on the Ptolemy Experience

• **Separate Domains (current)**
  - Strict Information Hiding
  - Hierarchical Nesting of Models of Computation
  - Simulation vs Code Generation

• **Integrated Domains (future)**
  - More seamless Programming Model
  - More seamless User Interface
  - Potential for Better Synthesis
  - Modularized Programming Environment
  - Deployable, Modular Design Tools

---

Integrated Domains

• **We have learned a lot**
  - Domains have helped
  - We can exploit what we have learned

• **Models of Computation**
  - Kahn Process Network (PN)
  - Dynamic Data Flow (DDF)
  - Boolean Data Flow (BDF)
  - Synchronous Data Flow (SDF)
  - Finite State Machine (FSM)

Increasing specialization:
More efficient execution

Control
Models of Computation to be Integrated

- **PN**: Sequential processes
  - communication via FIFO channels
  - blocking read when channel is empty
- **DDF**: Run time firing of actors, mutable graph
- **BDF**: Compile time analysis => firing sequence
- **SDF**: Periodic firing sequence

Implementing PN in Java

<table>
<thead>
<tr>
<th>Icon</th>
<th>Object</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>Process</td>
<td>Thread</td>
</tr>
<tr>
<td>Connecting Line</td>
<td>FIFO Channel</td>
<td>List</td>
</tr>
<tr>
<td></td>
<td>- Unbounded</td>
<td>- Expandable</td>
</tr>
<tr>
<td></td>
<td>- Blocking Read</td>
<td>- Wait &amp; Notify</td>
</tr>
</tbody>
</table>

**PN vs Java Threads**

- Threads are low-level & non-determinate
- PN is high-level & determinate
- Use Java Threads to implement higher level concurrence model
Prototype PN in Java

Student Project for EE290N, Fall ’96

- Participants
  - Peggy Laramie
  - Marlene Wan
  - Dick Stevens

- Define Processes
- Connect Processes
- Run

Technical Issues

- Will a PN eventually halt?
  - In SDF - Decidable in finite time
  - In general - Undecidable

- If not, will it execute in bounded memory?
  - In SDF - Decidable
  - In general - Undecidable
  - Tom Parks: Execute in bounded memory if possible
Executing a PN in Bounded Memory

**Tom Parks’ Algorithm**

- Set a capacity on each channel
- Block write when channel is full
- Repeat
  - Run until deadlock occurs
  - If never deadlocks, then problem solved
  - If deadlock and no blocking writes, then PN halts
  - Among channels that block writes, select one with lowest capacity
  - Increase capacity of selected channel
- **Infinite time to decide**
  - Whether PN halts
  - Whether PN executes forever in bounded memory

Applet to Generate Primes

- Sieve of Eratosthenes
- Mutable Graph
  - Processes to generate 2, 3, 4, ...
  - Process to filter multiples of 2
  - Dynamically add process for each new prime
- Run from within Tycho using the Tcl/Java interface
Status

- Project: Prototype demonstration
  - Process Networks: Blocking Reads
  - Implemented in Java
  - Execute in Bounded Memory when possible
- Tycho/Java Environment
  - Applet to generate primes
- Currently Building Java Kernel

Summary

- Java implementation for Ptolemy
  - Portable, Distributable, Multi-Threaded
  - Build on past experience
  - Use same Itcl interface
  - Merge simulation, code generation
  - Merge hierarchical models of computation
- Approach
  - Generic Kernel to support multiple domains
  - Start with PN domain, PN model
  - Add more specialized models DDF, BDF, SDF
  - Add other models DE, VHDL, SR, ...