Ptolemy II

Heterogeneous Modeling and Design in Java

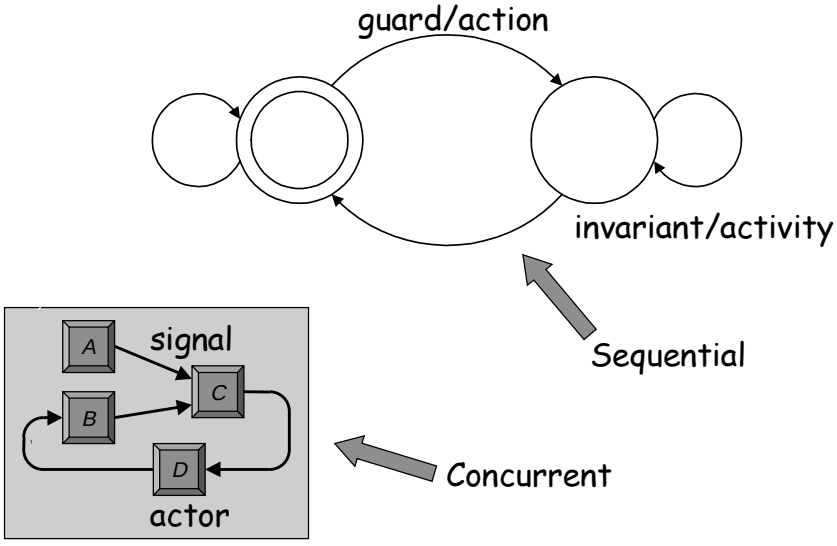
Principal Investigator
Edward A. Lee

Staff
Jennifer Basler
Christopher Hylands
Mary P. Stewart

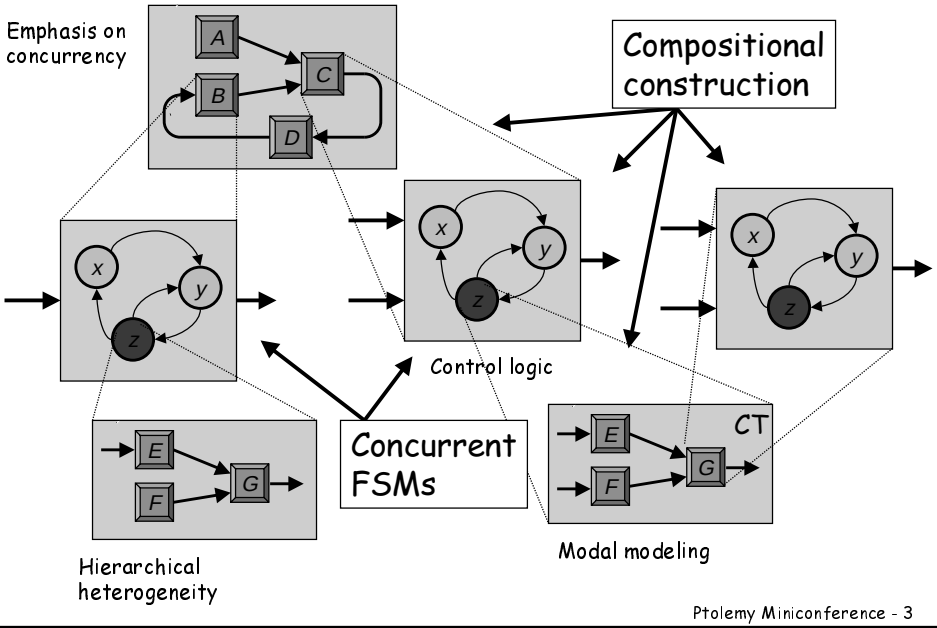
Postdocs
Bart Kienhuis
John Reekie

Students
John Davis, II
Ron Galicia
Mudit Goel
Bilung Lee
Michael Leung
Jie Liu
Xiaojun Liu
Lukito Muliadi
Steve Neuendorffer
Neil Smyth
Jeff Tsay
William Wu
Yuhong Xiong

State Machines & Concurrency

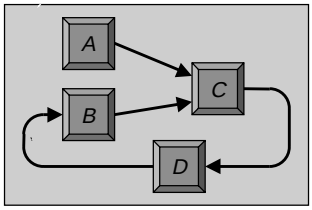


Concurrency + Control Logic



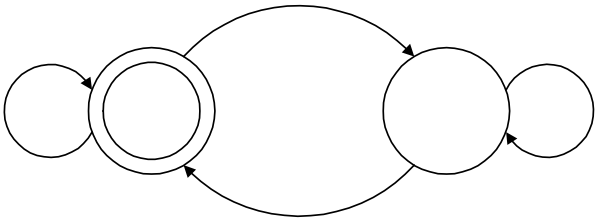
Useful Concurrent Semantics

- Analog computers (ODEs)
- Spatial/temporal models (PDEs)
- Discrete time (difference equations)
- Discrete-event systems (DE)
- Synchronous-reactive systems (SR)
- Sequential processes with rendezvous (CSP)
- Process networks (Kahn)
- Dataflow (Dennis)
- ...



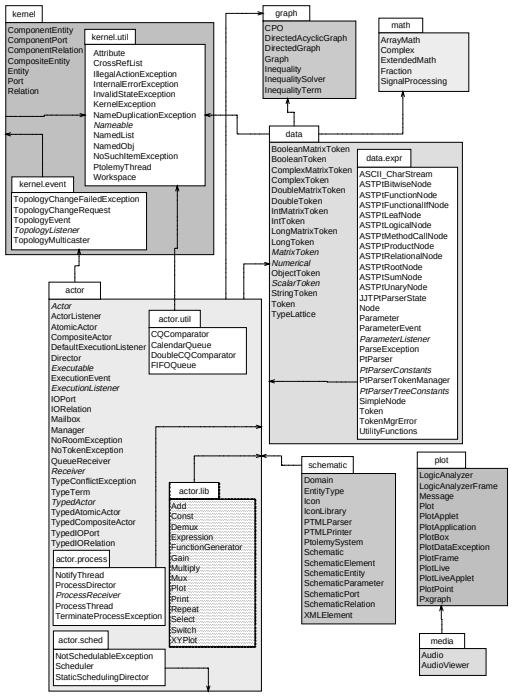
Useful State Machine Models

- Von-Neumann computers
- Imperative programming languages
- Finite state machines (FSMs)



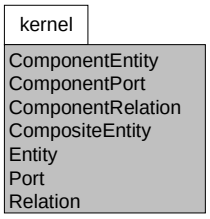
What's New in Ptolemy II?

- Applettable, migratable, Java code
- Multi-threaded kernel
- Modular package structure
- Better, richer abstract syntax (kernel)
- A modern type system, polymorphic actors
- Domain-polymorphic actors
- SW architecture based on object modeling
- An expression language and interpreter
- Math, graph, plot packages

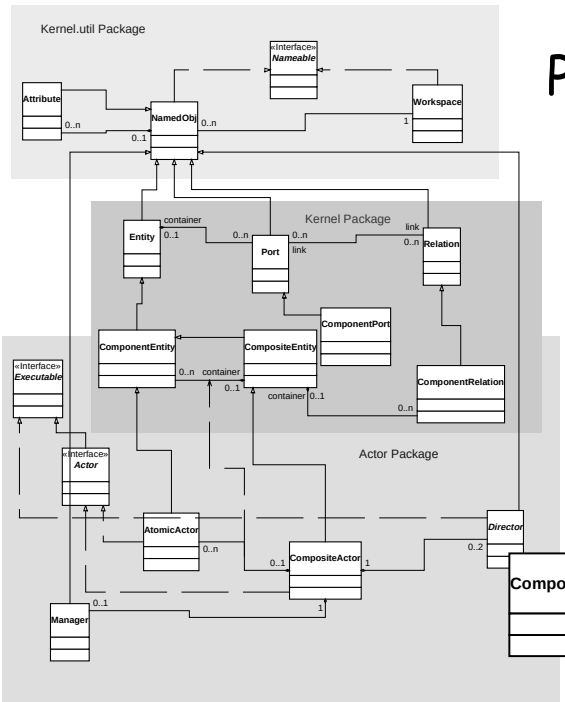


Ptolemy II Packages

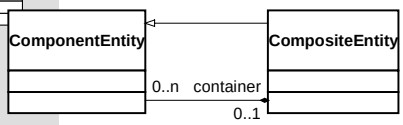
- kernel (clusterd graphs)
- actor (executable models)
- data (tokens, expressions)
- schematic (API for UIs)
- graph (graph algorithms)
- math (math algorithms)
- plot (plotting utilities)



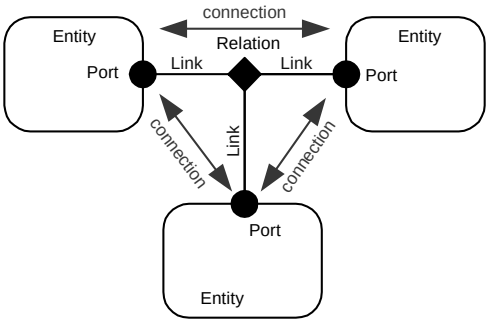
Ptolemy II Key Classes



UML static structure diagram for the key classes in the kernel, kernel.util, and actor packages.

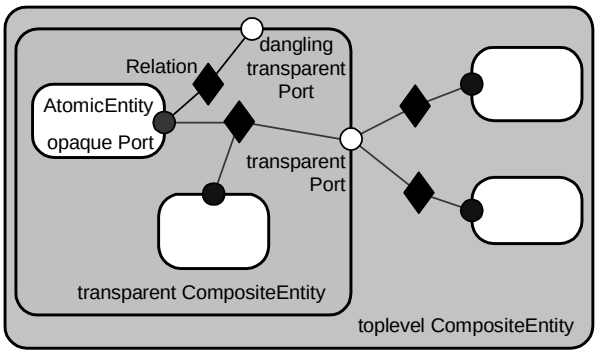


Kernel Package



The Ptolemy II kernel provides an *abstract syntax* - clustered graphs - that is well suited to a wide variety of domains, ranging from state machines to process networks. Here is a simple graph with three interrelated entities.

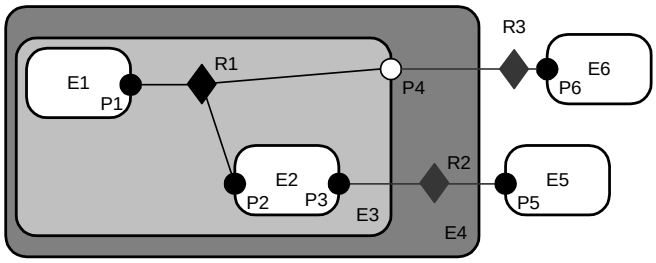
Clustering



Composite entities and ports in Ptolemy II provide a simple and powerful, domain-independent abstraction mechanism

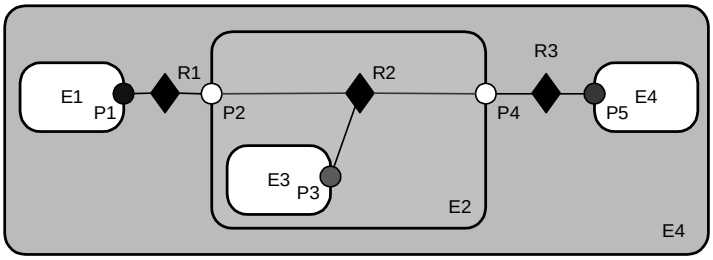
The ports deeply connected to the red port are the blue ones.

Level-Crossing Transitions



Level-crossing connections (in red) are supported, but discouraged. They may be needed for some models of computation (like Statecharts), but they break modularity. This, for example, makes cloning of subsystems impossible.

Tunneling Entities



Unlike Ptolemy Classic, no actor is required inside a composite entity to mediate a connection through it. In this example, P5 is visible to P1 (as is P3).

Among other consequences, this will make HOF actors easier to design.

Mutations

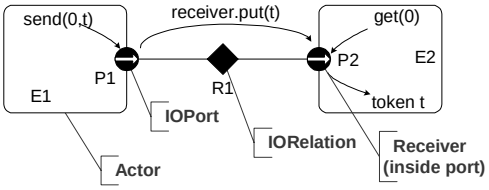
The kernel.event package provides support for

- Queueing requests for topology changes
- Processing requests for topology changes
- Registering listeners
- Notifying listeners of changes

Thus, models with dynamically changing topologies are cleanly supported, and the director in each domain can control when mutations are implemented.

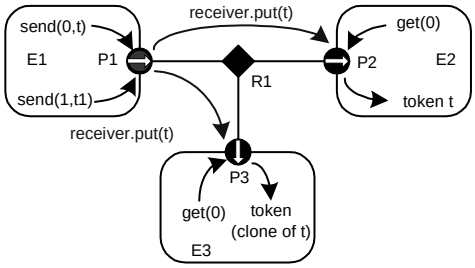
Actor Package

Basic Transport:



Services
- broadcast
- multicast
- busses
- cacheing topology info
- clustering
- parameterization
- typing
- polymorphism

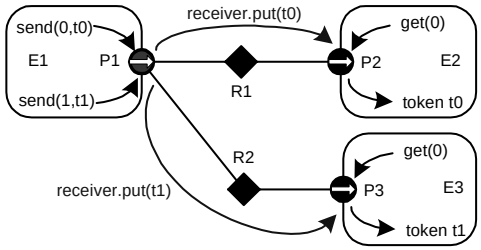
Broadcast



If a relation connects a single output to multiple inputs, then a *send* results in the token being broadcast to multiple destination.

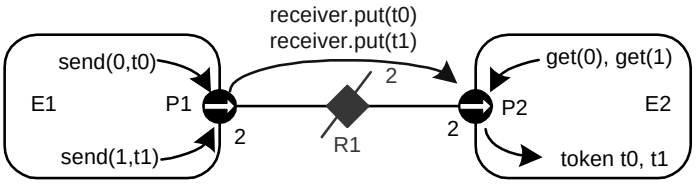
Tokens values are immutable, so it is safe for multiple destinations to have a reference to the same token, regardless of the domain.

Multicast



The red port is a *multiport*, meaning that it can be linked to multiple relations, and it can treat them each as a separate *channel*.

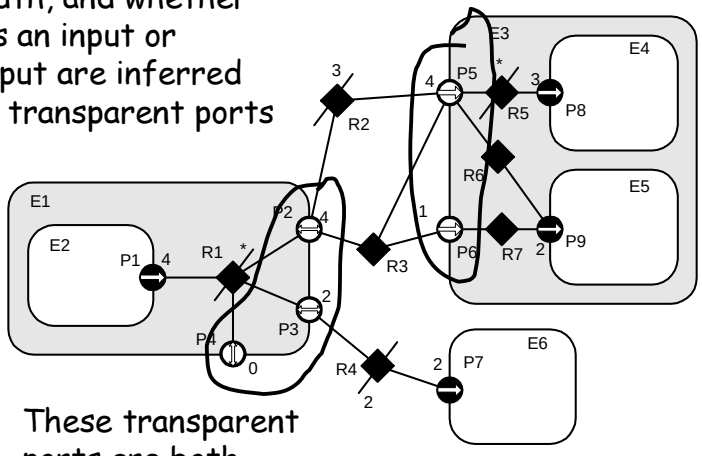
Busses



Relations and ports can have *width*, in which case they compactly represent multiple channels. The width of a port is inferred from the relations it is linked to, directly or through transparent ports.

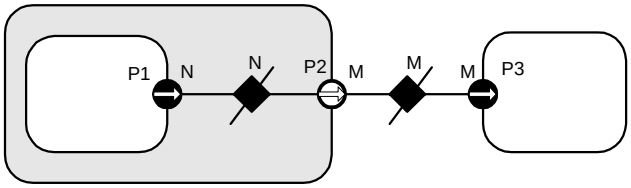
Example

Width, and whether it is an input or output are inferred for transparent ports



These transparent ports are both inputs and outputs

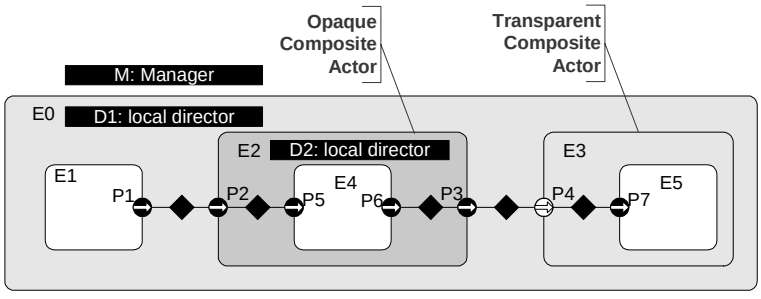
Width of a Transparent Port



The width of a transparent port is determined by its outside connections. If N > M, then N-M channels from P1 are dangling. If M > N, then N-M channels into P3 are dangling. Some domains tolerate dangling channels, and some do not.

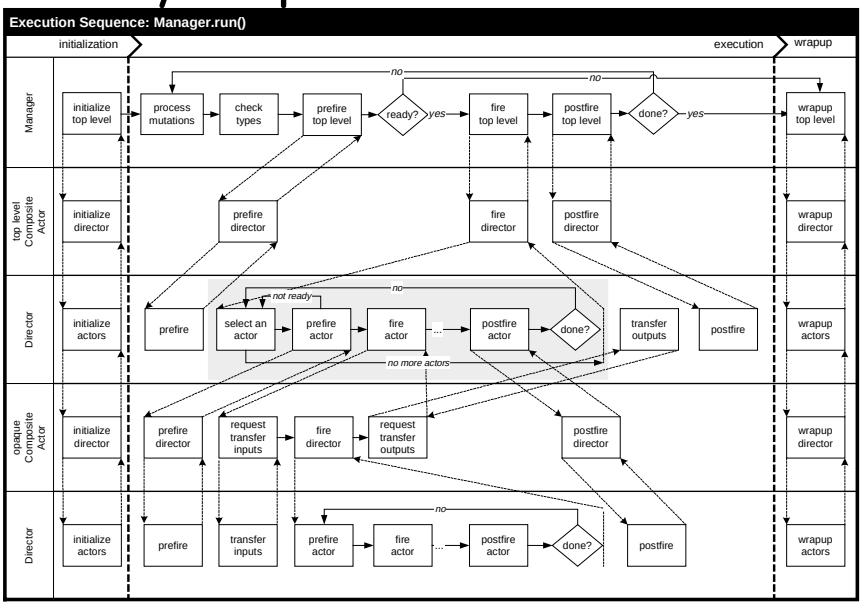
Manager and Directors

Hierarchical Heterogeneity:



Directors are domain-specific. A composite actor with a director becomes opaque (a "wormhole" in Ptolemy Classic). The Manager is domain-independent.

Polymorphic Flow of Control



Working Domains

- CSP - communicating sequential processes
- CT - continuous time
- DE - discrete event
- FSM - finite state machines
- PN - process networks
- SDF - synchronous dataflow

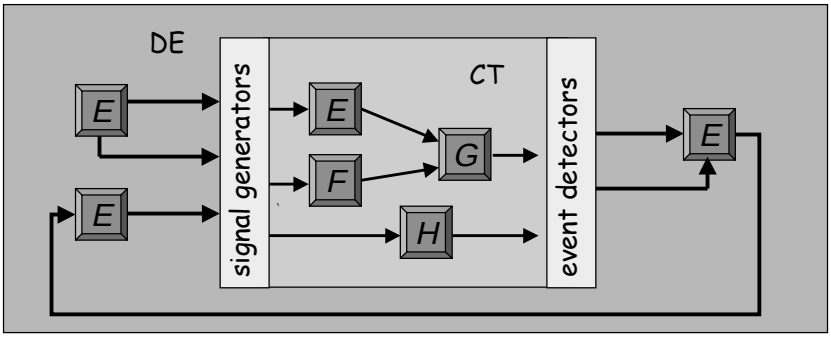
Innovations:

- Timed CSP
- Timed PN

Planned or In-Progress

- DFM - design flow management
  - emphasizing interfaces to external tools
- DT - discrete time
  - a timed version of SDF
- ODF - ordered dataflow
  - a distributed version of DE
- SR - synchronous/reactive
  - based on SR in Ptolemy Classic

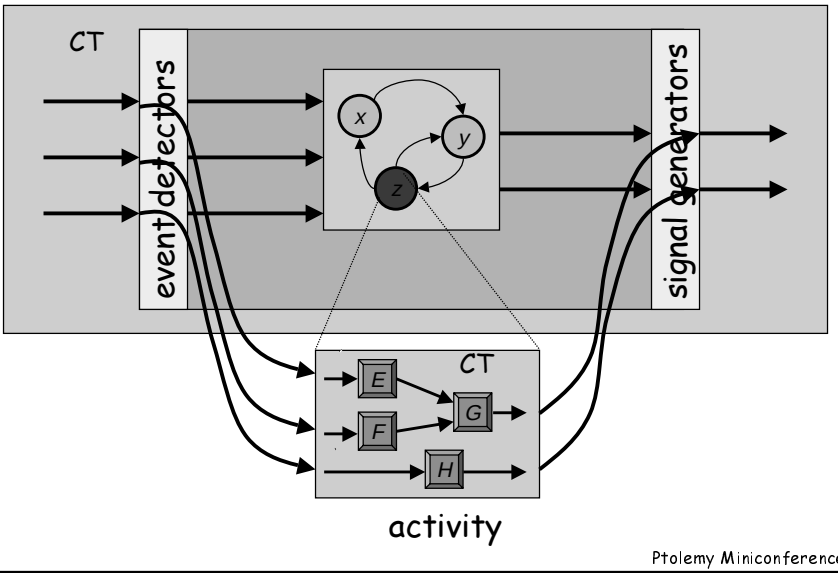
Mixed Domain Examples: CT +DE = Mixed Signals



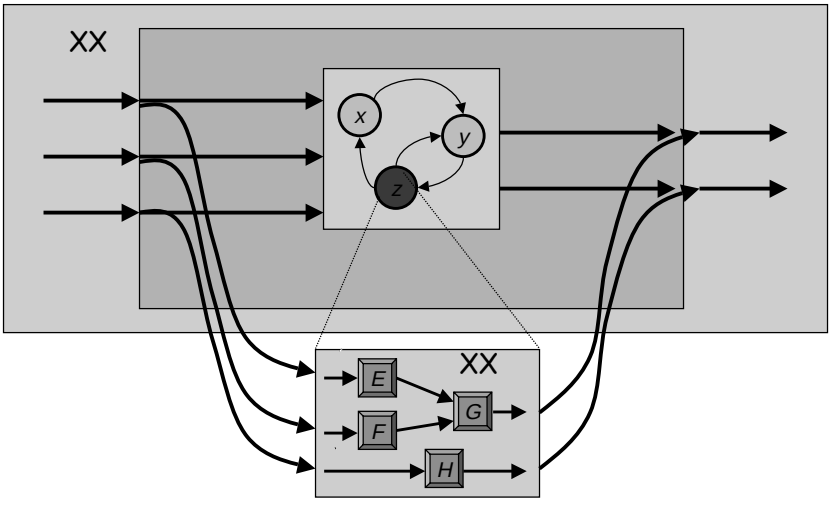
Feedback coupling of a discrete controller with a continuous-time system.

The reverse hierarchy, with CT on top, is also supported.

Mixed Domain Examples:
CT + FSM = Hybrid Systems



Mixed Domain Examples:
Anything + FSM = Modal Models

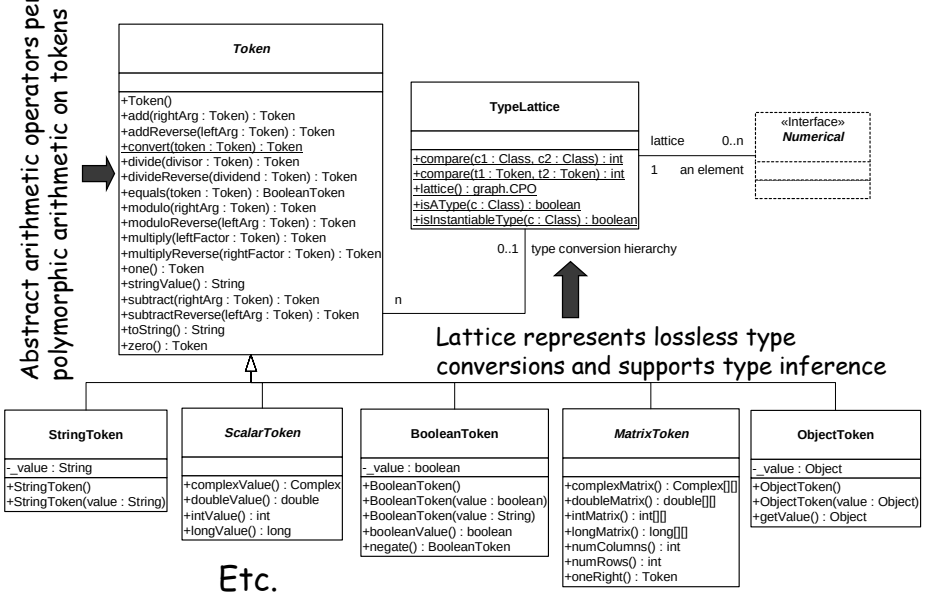


Other Combinations

- CSP for resource management
  - modeling real-time O/S
  - regulating access to shared data
  - modeling shared hardware resources
- PN for modeling concurrent hardware
  - message passing over networks (e.g. JINI)
- DE for performance modeling
  - abstracted timing models of hardware/software systems

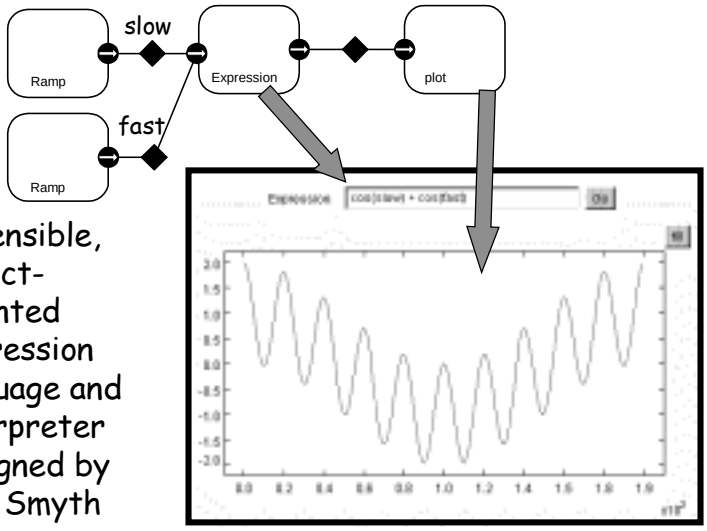
Data Package - Tokens

Abstract arithmetic operators permit polymorphic arithmetic on tokens



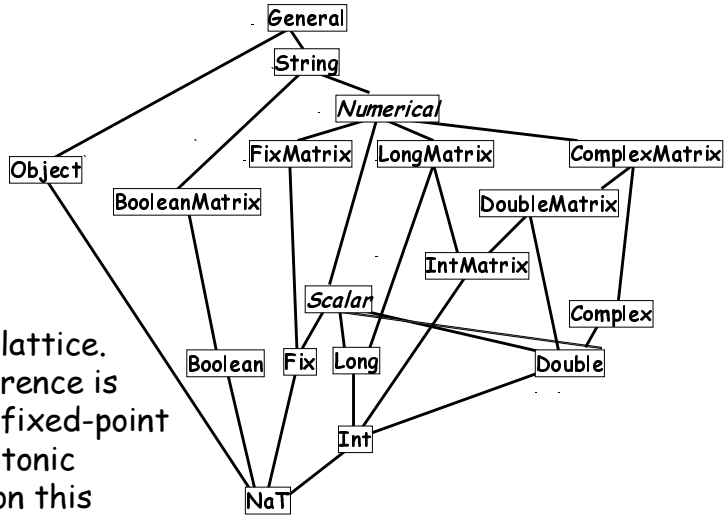
Etc.

Expression Language



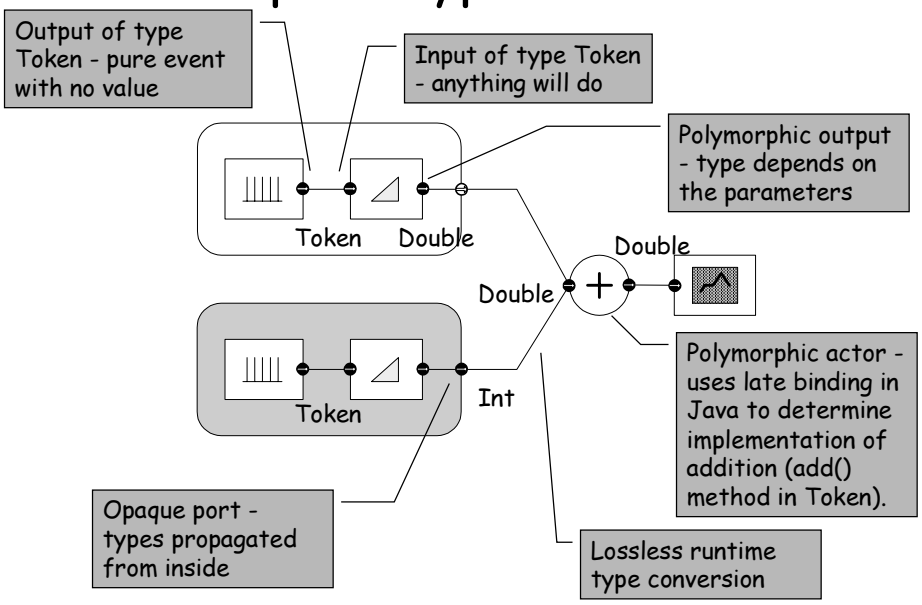
Extensible, object-oriented expression language and interpreter designed by Neil Smyth

The Data Type Hierarchy



The type lattice. Type inference is the least fixed-point of a monotonic function on this lattice.

Example - Type Inference



Ptolemy II Support Packages

- Graph package
  - topological sort, spanning trees, strongly-connected components, etc.
- Math algorithms
  - matrix and vector math and signal processing routines.
- Plotting utilities
  - signal plotter.