Higher-order functions

Jörn W. Janneck

The Ptolemy Group
UC Berkeley

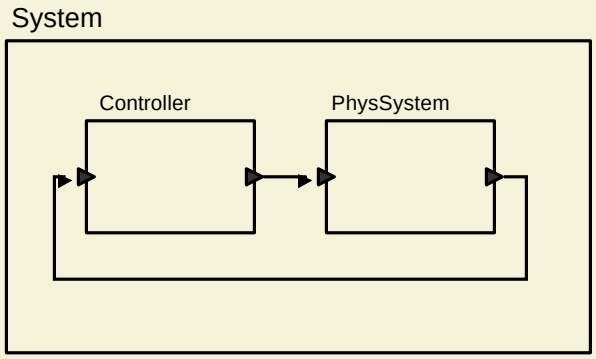
Ptolemy Miniconference
Berkeley, CA, March 22-23, 2001

What is it?

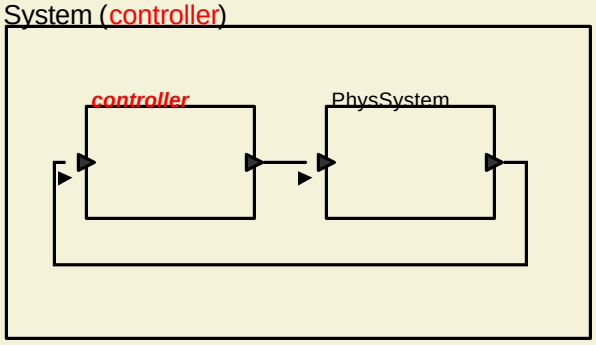
Essentially, it comprises all the language/modeling constructs that allow actors to be treated as data objects (tokens).

- **Some important cases:**
  - actors may be **parameters** (of actors, for example)
    - actors may be bound to variables
  - actors may be the **result of expressions**
  - actors may be **input and output tokens**
    - they may flow through the network
    - they can be hooked up in different places, dynamically

The classic case.

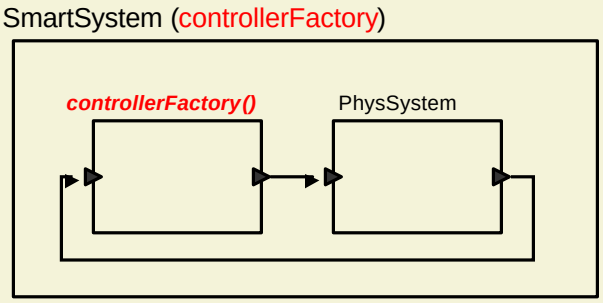


Applications
Actors may be parameters



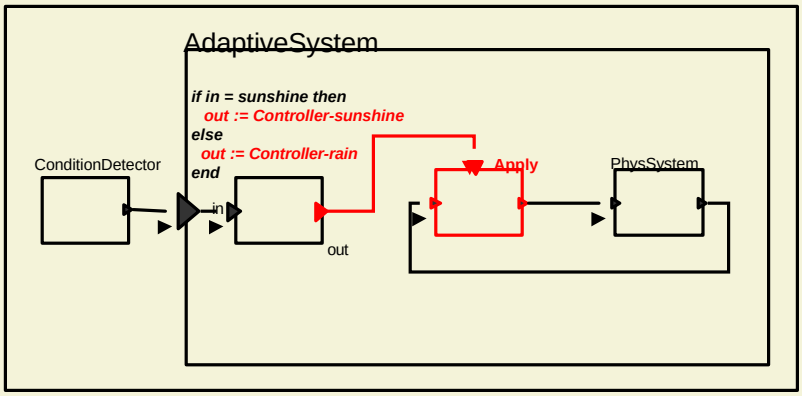
Applications

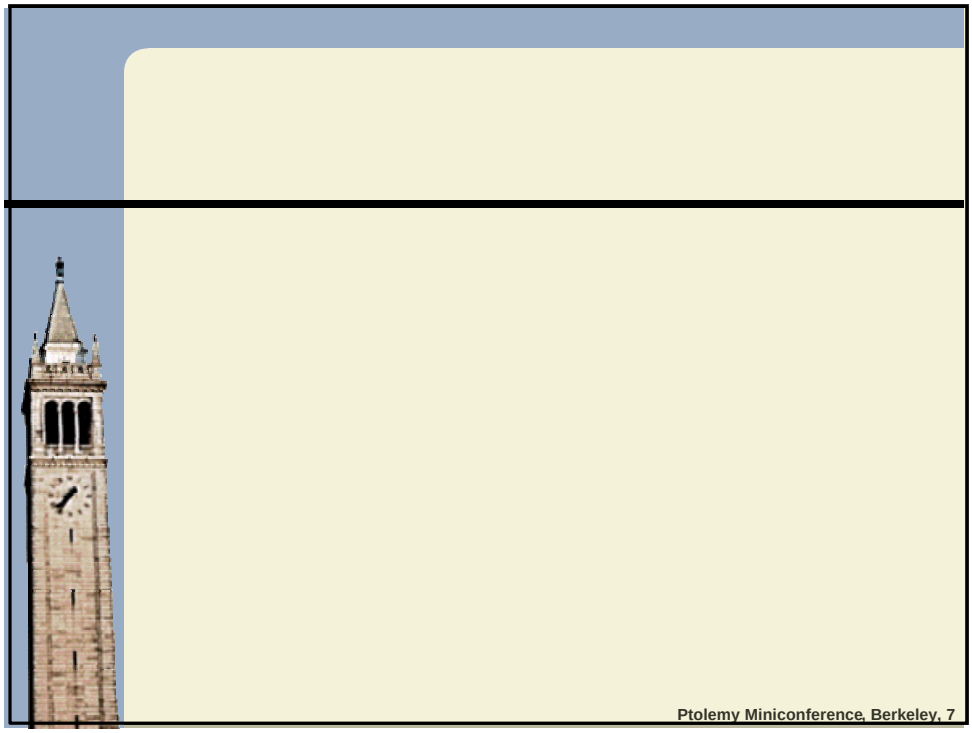
Actors may be result of expressions

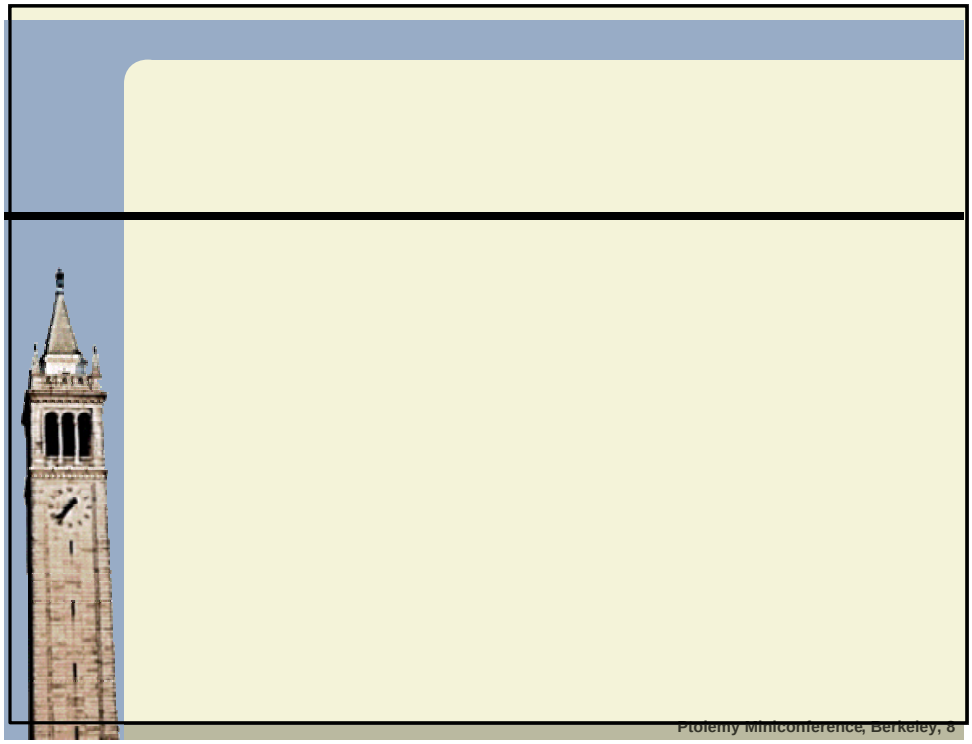


Applications

Actors may be tokens







What's it good for?

- Parametric static structure
  - a model a family of structures, depending on one or more parameters
  - Examples:
    - FFT (of parametric size)
    - FIR filter (of parametric length)
  - incomplete specifications (that will be completed only on instantiation, and then potentially in different ways)
  - Examples:
    - shop floor model (the controller is parametric)
  - this, in general, is the key to better factoring of component functionality:

    Make components incomplete.

What's it good for?

- Dynamic structures
  - dynamic networks, where participants are added and re/moved during runtime
  - Examples:
    - the Internet, many larger networks, mobile/wireless networks
    - traffic networks
    - changing scene graphs
  - building systems that manipulate models
  - Examples:
    - automatic simulation frameworks
    - Ptolemy?

The challenge.

- Undisciplined implementation of these features may undermine essentially all of Ptolemy's structuring facilities.
  - Hierarchy
- We want to make models more understandable, not less.
- We are working on ideas of how to achieve the best of both worlds.
  - Your input is appreciated – come and talk to us.

Higher-order functions

Jörn W. Janneck
UC Berkeley

Ptolemy Miniconference
Berkeley, CA, March 22-23, 2001

4-th Biennial Ptolemy Miniconference
March 22-23, 2001, Berkeley, CA

Why do we want it?

Better model reuse.

If there was more space, we would even use a bigger font – this is the key motivation behind many of our core concepts.

Dynamic model structures.

Dynamic model structures facilitate some advanced modeling constructs that are hard or impossible to realize without them. But they need to be controled. Higher-order actors/functions are a key to this.

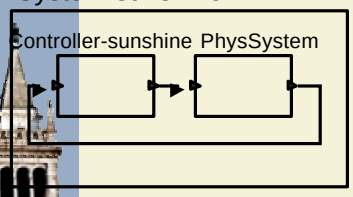
Applications

Actors may be parameters

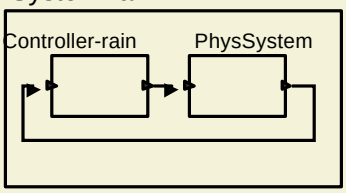
Applications

Actors may be result of expressions

System-sunshine



System-rain



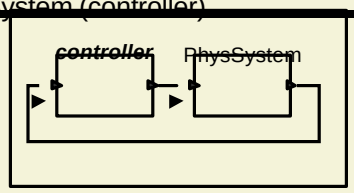
Different systems, for different application conditions.
How can we make the user choose between them?

We could preconfigure complete systems, one for each condition.
This potentially leads to many, many components.

Applications

Actors may be result of expressions

System (controller)



We could provide a parametric version, and have the user instantiate it appropriately:

System(controller = Controller-sunshine)
System(controller = Controller-rain)

But:

- This requires that the user knows the inside of out system.
  (bad encapsulation, IP, why should the user bother)
- If several things inside the component need to be instantiated consistently, we cannot rely on the user to do this.

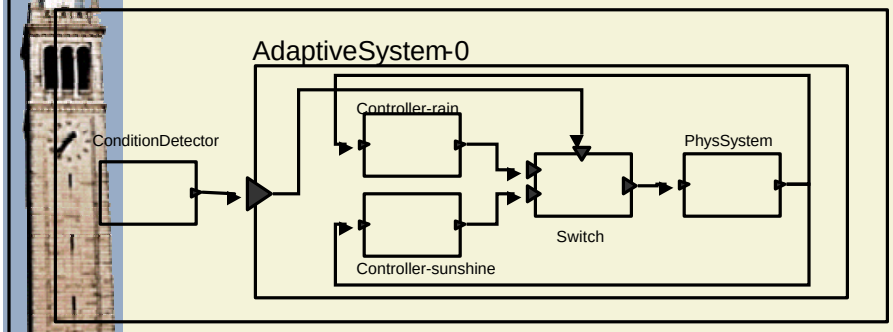
Applications

Actors may be tokens

Problem:

How do we change from sunshine to rain?

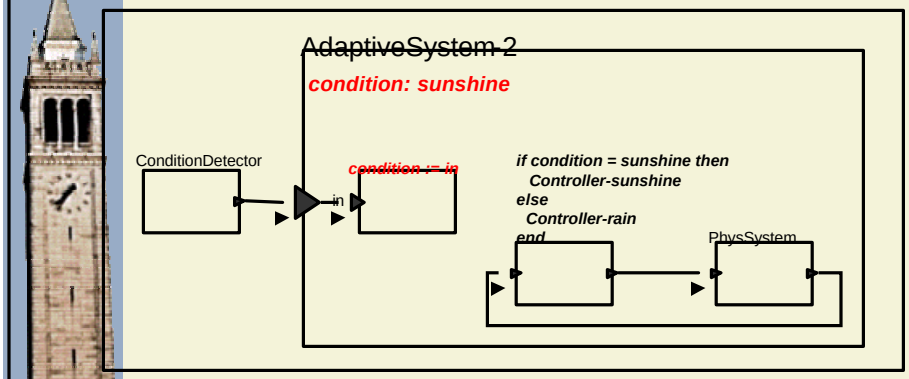
Switching is potentially wasteful, and leads to complicated models. It mixes the controlling structure with the switching structure.



Applications

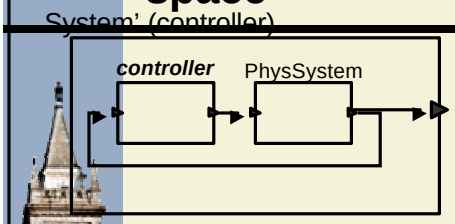
Actors may be tokens (but need not be)

Using conditional construction (the previous example), there may also be a less explicit way of addressing the problem.



Applications

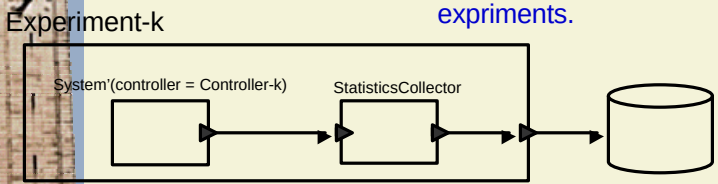
Actors as tokens – exploring design space



Now we would like to observe the system, to see which of the n controllers does the best job.

Assume n is big, and running the system takes a fair amount of time.

You don't want to waste your time setting up n essentially identical expriments.



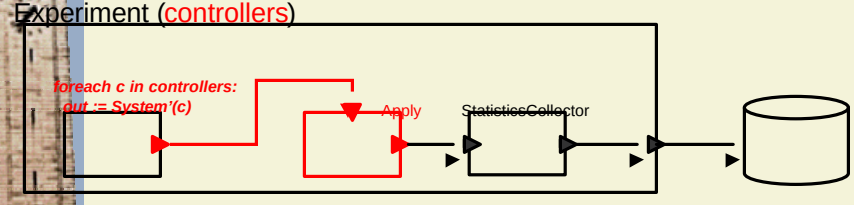
Applications

Actors as tokens – exploring design space

The solution is to set up an experimentation environment that iterates through a list of controllers (e.g.) and instantiates a new System'(...) for each. Now the entire experiment can be instantiated like this:

Experiment([Controller-1, Controller-2, ..., Controller-n])

After that, we only have to wait.



Challenges

- Undisciplined implementation of these features may undermine essentially all of Ptolemy's structuring facilities.
  - in particular, hierarchical structures
- So we need to make higher-order constructs well-behaved.

Summary

- Higher-order modeling facilities give us
  - better component reuse
    - parametric components
  - better encapsulation
    - actors as results of expressions
  - dynamic model structures
    - actors as tokens
- It is not trivial to integrate them into a structured modeling environment.
  - But it's worth the effort.

What is it?

Essentially, it comprises all the language/modeling constructs that make components/actors first-class citizen.

- **Some important cases:**
  - actors may be parameters (of actors, for example)
  - actors may be input and output tokens
    - they may flow through the network
    - they can be hooked up in different places, dynamically
  - actors may be the result of expressions, they may be bound to variables