Mixing Dataflow with Control

Wan-Teh Chang
Edward A. Lee
David G. Messerschmitt
EECS Department
UC Berkeley

Major collaborators:

Frederic Boulanger
Bilung Lee

Motivation

Objective: To develop specialized computational models for describing complex control functionality in Ptolemy, and mix them with other computational models like dataflow.

- Dataflow graphs represent numerical computation (DSP) tasks.
- Controllers control and sequence the dataflow tasks.

Issues:

- Better abstractions for control
- Semantics of interface between control and dataflow

Hierarchical Description of Control Functionality

Finite state machines (FSM):

- Intuitive, well-developed formal theory
- Flat and sequential, practical difficulty in describing large complex controllers.

Method: Augment the familiar event/state-based models with hierarchy and concurrency.

- Textual languages: Esterel
- Graphical languages: Statecharts, Argos

Textual Language: Esterel

A special-purpose programming language for reactive systems (controllers etc.)

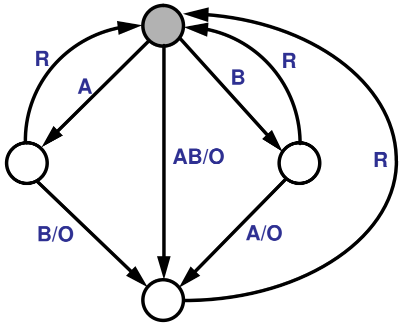
- Developed at INRIA, France
- Perfect synchrony hypothesis
- Can be compiled into C or C++

Basic features:

- Sequencing, testing, looping, and parallel constructs
- Communication mechanism: Instantaneous broadcast of signals
- Interrupt: do *stmt* watching *S*

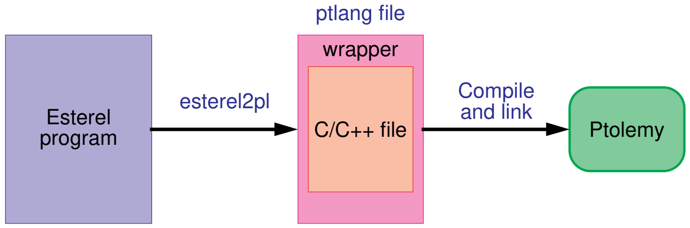
A Simple Esterel Program

```
module EsTest:
input A, B, R;
output O;
loop
   do
      [await A || await B];
      emit O;
      halt
   watching R
end loop
end module
```



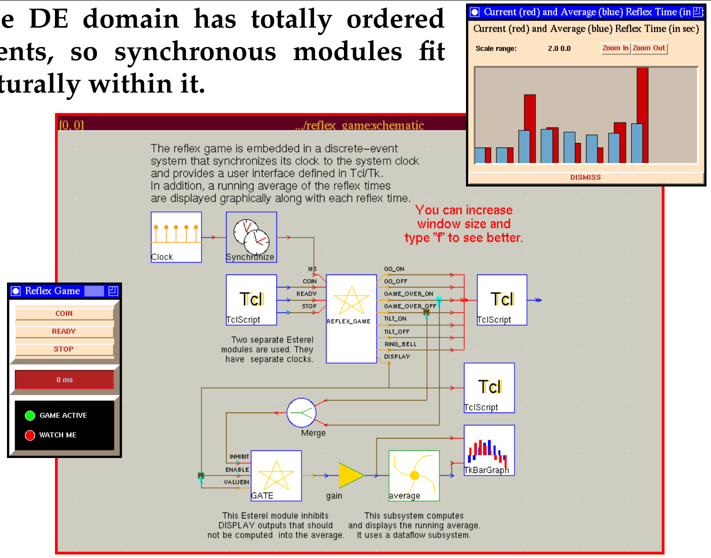
Ptolemy's Esterel Interface

Stars in SDF, DE, and CGC can be defined in Esterel.



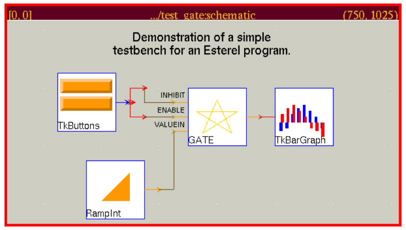
Mixing Esterel with Ptolemy Discrete Event

The DE domain has totally ordered events, so synchronous modules fit naturally within it.



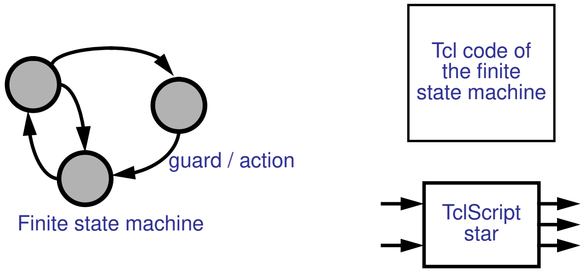
Mixing Esterel with Dataflow Process Networks

Synchronous dataflow using TRUE/FALSE encoding of pure signals, is one option.



Finite-State Machines

- A graphical entry tool for drawing state transition diagrams
- Each arc has a guard (enabling condition) and an action (code to execute when guard is true).
- Currently, guards are Tcl expressions, and actions are Tcl code.



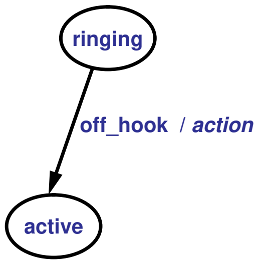
Tcl Code for FSM

```
proc fsm_phone_react {event} {
  global fsm_phone_state

  switch $fsm_phone_state {
    ...
    ringing {
      ...
      if {$event == off_hook} {
        action
        set fsm_phone_state active
      }
      ...
    }
    ...
  }

  return [list $out1 $out2]
}
```

Name: phone
Input: event
Output: out1, out2

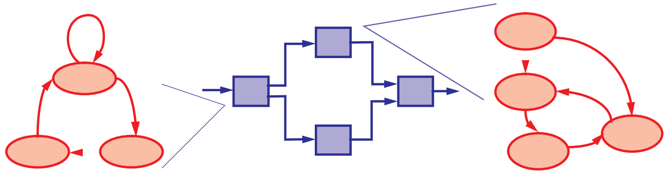


Hierarchy and Concurrency

Hierarchical state:

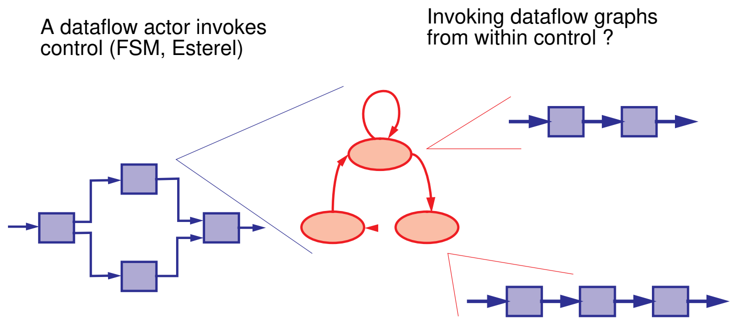
- containing complexity
- a compact way to describe interrupt behavior

Concurrency: can be (partially) achieved by having the concurrent finite-state controllers communicating with dataflow semantics.



Mixing Control with Dataflow

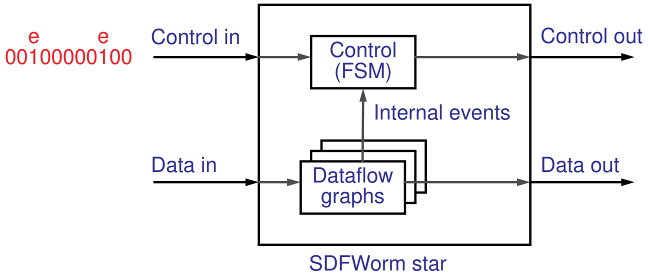
- Control inside a dataflow actor
- Dataflow graph inside control?



FSM Controls Invocation of Ptolemy Galaxies

Flexible wormhole: a star that is replaced by one of a set of galaxies. The choice of galaxy is controlled dynamically by a Tcl script.

- Preliminary demo works
- Semantic issues



Conclusions

- Approaches to introducing control into Ptolemy and their implementations
- Largely using control abstractions and languages developed elsewhere but seeking improvements
- Interesting semantic issues in mixing control with dataflow
- Invoking dataflow graphs from within control