CAL Actor Language: Motivation

- Writing simple actors should be simple.
  - Ptolemy II API very rich
  - actor writing requires considerable skill
  - BUT: Actors have a lot of common structure.
- Models should allow efficient code generation.
  - actor descriptions contain a lot of "admin" code
  -> Generate actors from a more abstract description.
  - reduces amount of code to be written
  - makes writing actors more accessible
  - reduces error probability
  - makes code more versatile
  - retargeting (other platforms, new versions of the Ptolemy API)
  - analysis & composition

Simple actors

- action ID (In ==> Out):
  - action [a] ==> [a]

- actor A (k) Input1, Input2 ==> Output:
  - action [a], [b] ==> [k*[a + b]]

- Merge () Input1, Input2 ==> Output:
  - action Input1: [x] ==> [x]
  - action Input2: [x] ==> [x]

- non-deterministic example
  - action C () Double Input ==> Double Output:
    - action [a] ==> [a] where a >= 0
    - action [a] ==> [-a] where a < 0

- deterministic example
  - action Sum () Input ==> Output:
    - sum := 0
    - action [a] ==> [sum] do sum := sum + a; end

Action Guards

- action
  - input patterns
decaling variables
  - guard
  - specifying enabling conditions
  - output expressions
  - computing output tokens
  - body
  - modifying the actor state

Stateful Actors

- action
  - storing state variables
  - modifying the actor state

Action schedules

- action
  - schedule
  - regexpr

Src: J. Janneck, CAL – An actor language, 2003
Executing CAL: Discovering concurrency

References & Further Reading

- **Dataflow Process Networks**
  - Krzysztof Kuchcinski, Class notes, EDA 380, Design of Embedded Systems, 2002

- **Actor-oriented Modeling**

- **Caltrop Project/Cal Actor Language**
  - Janneck et al., [http://embedded.eecs.berkeley.edu/caltrop](http://embedded.eecs.berkeley.edu/caltrop)