Dataflow Process Network

Goals

- Formalize dataflow process network
  - Widely used in signal processing community
  - SPW, COSSAP, Khoros, Ptolemy, etc
- Good basis for programming language
  - Hierarchy, higher order function, recursion, etc

Kahn Process Networks

- MoC where concurrent processes communicate through unidirectional FIFO
- Process
  - Maps one or more input sequences to one or more output sequences
  - Usually constrained to be continuous
    - F(sup X) = sup F(X)

Dataflow Process Networks

- A process is sequence of firings of dataflow actors
  - F = map(f)
- Actor
  - Fires according to firing rules
  - Each firing consumes input tokens and produces output tokens
- Continuity
  - Functional
    - No side effects
  - Sequential
    - Firing rules can be tested in a predefined ordering

Firing Rules

- An actor with p inputs
  - N firing rules: $R = \{R_1, R_2, \ldots, R_N\}$
  - Patterns for each input: $R_{i\,=}\{R_{i,1}, R_{i,2}, \ldots, R_{i,p}\}$
- In order to fire
  - The patterns must be a prefix of the tokens at the inputs
  - Adder: $R_1 = \{[*], [*]\}$
  - Select: $R_1 = \{[*], [], [T]\}$, $R_2 = \{[], [*], [F]\}$

Execution Model

- Concurrent processes
  - Demand driven style
  - Processes with unavailable inputs are put to sleep with its input channels marked hungry
  - Writing to hungry channel suspends the writer and wakes the waiter
- Static/dynamic scheduling
  - Possible in dataflow process network b/c of actors
  - Avoids overhead of context switching
- Tagged-token model
  - Each token has a tag
  - Fire only when input tokens have matching tags
  - No need for FIFO, tags impose order

Language Design

- Ptolemy as a driving example
  - Visual and textual interface
  - No built in MoC
  - Supports 3 different dataflow process network domains
  - Extensible set of primitive actors

Hierarchy

- Subgraphs can be encapsulated into a single node
- Difficulties
  - Want hierarchical nodes to have the same properties as primitives
    - Firing rules, functional, etc.
    - State introduced from self loops on primitive actors
      - Reconciled: state is syntactic sugar for delay

Function Arguments

- Two types of arguments
  - Parameters
  - Input streams
- Why the distinction?
  - Parameters are constants
  - Do not need arcs for parameters
  - Simplifies work done by compiler/interpreter

Recursion

- Two examples
  - Sieve of Eratosthenes
  - FFT
- Sieve of Erathosthenes
  - Implemented with a hierarchical node “sift” that invokes itself when called
  - Graph is dynamically expanded
    - Mutates during execution
- FFT
  - Contrast to sieve of Erasthosthenes
  - Can be completely scheduled at compile time

Higher Order Function

- Map actor
  - Inputs:
    - Blockname
    - Input_map
    - Output_map
  - Replaces itself with one or more instances of the specified actor
- IfThenElse
  - Takes two replacement actors and a predicate

Datatypes, polymorphism

- Networks are typed
  - Type consistency is statically checked
- Polymorphism
  - Ptolemy supports parametric and ad-hoc
    - Parametric: behaves same way regardless of data type
    - Ad-hoc: behavior can be different

Parallelism

- Comes for free in dataflow process network
  - Dataflow graph exposes parallelism for hardware or compiler
  - Recursion can be evaluated during setup phase

Conclusion

- Formalization was useful