Concurrent Models of Computation

Edward A. Lee
Robert S. Pepper Distinguished Professor, UC Berkeley
EECS 290n – Advanced Topics in Systems Theory
Concurrent Models of Computation
Spring 2009

Week 1: Process Networks

Logistics

Class web page:
http://embedded.eecs.berkeley.edu/concurrency
Project
Paper (and paper review)
Homework
Reading
Study group
Technology:
- Ptolemy II
- Java
- Eclipse
- LaTeX

Homework

- Issued roughly every two weeks
- Will leverage a common technology base:
  - Java
  - Eclipse
  - Ptolemy II
- First assignment is on the web

Project

- Conference (workshop) paper quality expected
- Papers will be “submitted” and “reviewed” by you
- Presentations will be workshop like
- Teams up to two are encouraged
- Leveraging the technology base is encouraged
- Many project suggestions are on the web
  - In almost all cases, I have a fairly clear idea of how to start. Come talk to me if one of these looks interesting

Study Group

- A mechanism for reading and discussing papers.
- Hopefully will meet Fridays, 4-5PM, 540 A/B Cory.
- Each week, 2-3 students are assigned to lead the discussion. One of those will be selected as the overall coordinator.
- All are expected to have read the paper before the study group meets.
- All are encouraged to comment, as questions, and participate in discussion.
- Come prepared with a hard or soft copy of the paper.

Introduction to Edward A. Lee

- Working in embedded software since1978, when I was writing assembly code for 8-bit microcontrollers to control biomedical robotic machines. From 1980-82, I was writing assembly code for the AT&T DSP-1 to implement modems at Bell Labs.
- BS '79 (Yale, Double major: CS and EE)
  SM '81 (MIT, EECS)
  PhD '86 (Berkeley, EECS)
- Berkeley EECS faculty since 1986
- One of four directors of Chess, the Berkeley *Center for Hybrid and Embedded Software Systems*
- Director of the Berkeley Ptolemy project
- Co-author of five books (on digital communications, signals and systems, and dataflow)
- Chair of EE, then EECS, from Jan. 2005- June 2008.
- Co-founder of BDTI, Inc., a 12 year old technology company
- Key awards:
  - Robert S. Pepper Distinguished Professor
  - NSF Presidential Young Investigator
  - Terman Award for Engineering Education.
  - Fellow of the IEEE

Who are you?

Model of Computation

NIST:

"A formal, abstract definition of a computer."

Examples: Turing machine, random access machine, primitive recursive, cellular automaton, finite state machine, …

Wikipedia (on 1/18/09):

"the definition of the set of allowable operations used in computation and their respective costs."

"In model-driven engineering, the model of computation explains how the behaviour of the whole system is the result of the behaviour of each of its components."

Google | define:Model of computation | Search | Advanced Search | Preferences

Web

Definitions of **Model of computation** on the Web:

- In computability theory and computational complexity theory, a model of computation is the definition of the set of allowable operations used in ...
  en.wikipedia.org/wiki/Model_of_computation
- The rules that govern the interaction, communication, and control flow of a set of components.
  ptolemy.eecs.berkeley.edu/publications/papers/99/HMAD/html/glossary.html

-Ptolemy II -

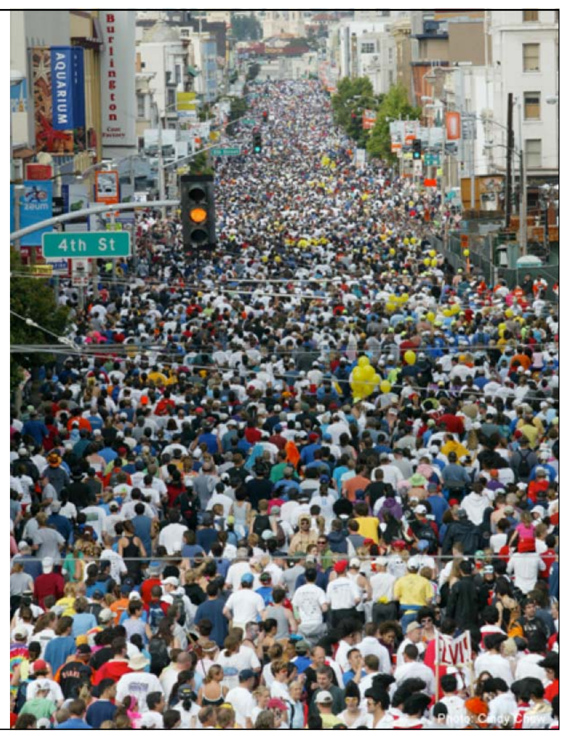
Heterogeneous Concurrent Modeling and Design in Java

Lee 01: 9

Concurrency

From the Latin, concurrere, “run together”

Discussion:
Is concurrency hard?



Google | define:Concurrency | Search | Advanced Search | Preferences

Web

Related phrases: concurrency control wrongway concurrency wrong way concurrency optimistic concurrency control optimistic concurrency java concurrency p2c/c2p concurrency non lock concurrency control non concurrency

Definitions of **Concurrency** on the Web:

- concurrence: agreement of results or opinions
- concurrence: acting together, as agents or circumstances or events
  wordnet.princeton.edu/perl/webwn
- In computer science, concurrency is a property of systems in which several computational processes are executing at the same time, and potentially interacting with each other. ...
  en.wikipedia.org/wiki/Concurrency_(computer_science)
- A concurrency, overlap, or coincidence in a road network is an instance of one physical road bearing two or more different highway, motorway, or other route numbers. When it is two freeways that share the same right-of-way, it is sometimes called a common section or commons.
  en.wikipedia.org/wiki/Concurrency_(road)
- The property or an instance of being concurrent; something that happens at the same time as something else
  en.wiktionary.org/wiki/concurrency
- Execution of two processes or operations simultaneously.
  www.iso.port.ac.uk/~mike/interests/chistory/documents/cpm-22-manual/axh.html

Potential Confusion

- Concurrent vs. parallel
- Concurrent vs. determinate

Kahn Process Networks (PN)
A Concurrent Model of Computation (MoC)

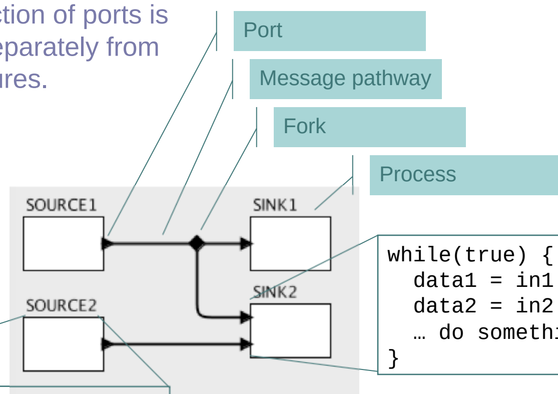
- A set of components called *actors.*
- Each representing a sequential procedure.
- Where steps in these procedures receive or send messages to other actors (or perform local operations).
- Messages are communicated asynchronously with unbounded buffers.
- A procedure can always send a message. It does not need to wait for the recipient to be ready to receive.
- Messages are delivered reliably and in order.
- When a procedure attempts to receive a message, that attempt blocks the procedure until a message is available.

Coarse History

- Semantics given by Gilles Kahn in 1974.
  - Fixed points of continuous and monotonic functions
- More limited form given by Kahn and MacQueen in 1977.
  - Blocking reads and nonblocking writes.
- Generalizations to nondeterministic systems
  - Kosinski [1978], Stark [1980s], …
- Bounded memory execution given by Parks in 1995.
  - Solves an undecidable problem.
- Debate over validity of this policy, Geilen and Basten 2003.
  - Relationship between denotational and operational semantics.
- Many related models intertwined.
  - Actors (Hewitt, Agha), CSP (Hoare), CCS (Milner), Interaction (Wegner), Streams (Broy, …), Dataflow (Dennis, Arvind, …)...

Syntax

- Processes communicate via *ports*.
- Ports are connected to one another, indicating message pathways.
- Interconnection of ports is specified separately from the procedures.

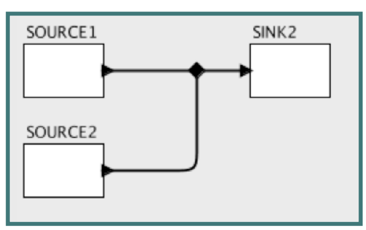



```
while(true) {
  data1 = in1.get();
  data2 = in2.get();
  … do something with it …
}
```

```
while(true) {
  data = …
  outputPort.send(data);
}
```

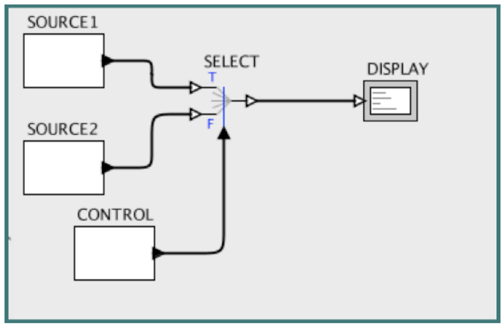
Discussion: What should a fork do?

What should this mean?



Question 1:
Is “Fair” Thread Scheduling a Good Idea?

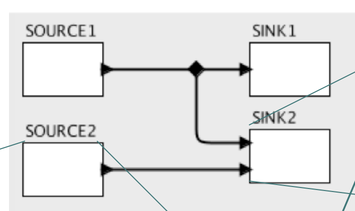
In the following model, what happens if every thread is given an equal opportunity to run?



Lee 01: 17

Rendezvous: An Alternative Communication Mechanism with Bounded Buffers

Rendezvous underlies CSP (Hoare), CCS (Milner), and Statecharts (Harel)




```
while(true) {
  data1 = in1.get();
  data2 = in2.get();
  … do something with it …
}
```

```
while(true) {
  data = …
  outputPort.send(data);
}
```

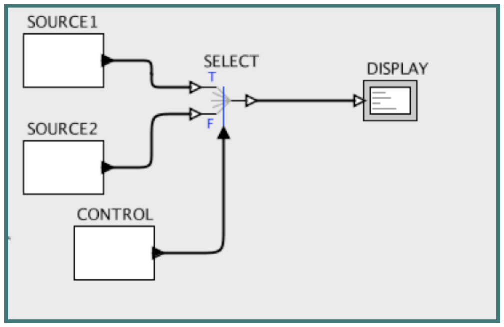
Processes must both reach this point before either can continue.

Discussion: What should a fork do?

Lee 01: 18

Discussion

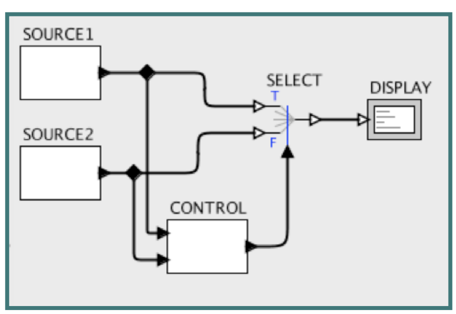
How does this program compare under rendezvous communication vs. process networks?



Lee 01: 19

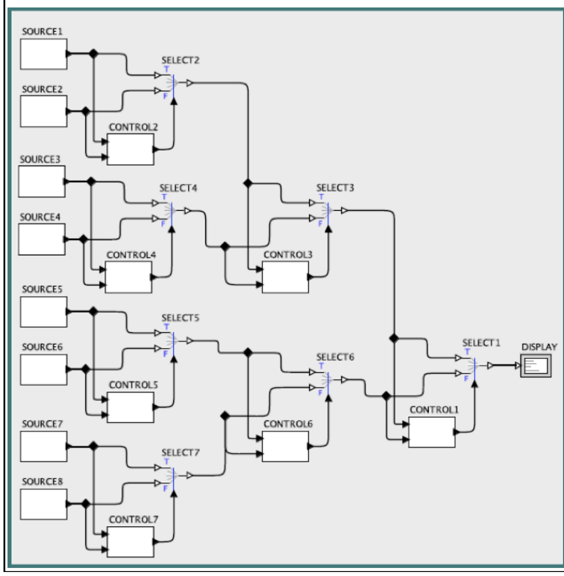
Question 2: Should we use Rendezvous Here?

The control signal now depends on the source data.



Lee 01: 20

A Practical Application with this Structure

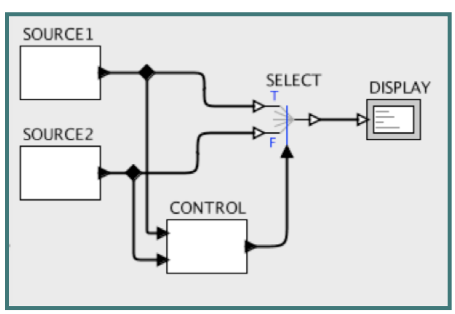


Consider collecting time-stamped trades from commodities markets around the world and merging them into a single time-stamped stream. The CONTROL actors could compare time stamps, with logic like this:

```
data1 = topPort.get();
data2 = bottomPort.get();
while (true) {
  if (data1.time < data2.time)) {
    output.send(true);
    data1 = topPort.get();
} else {
    output.send(false);
    data2 = bottomPort.get();}
}
```

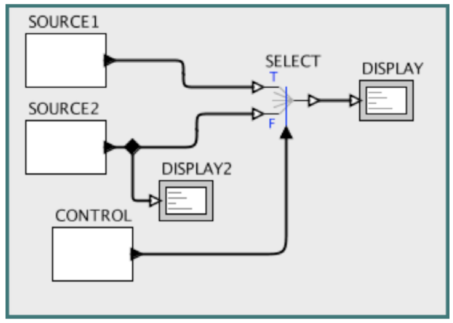

Question 3: How about Demand-Driven (Lazy) Execution?

In demand-driven execution, a process is stalled unless its outputs are required by a downstream process.



The DISPLAY process has nothing downstream. When should it be allowed to run?

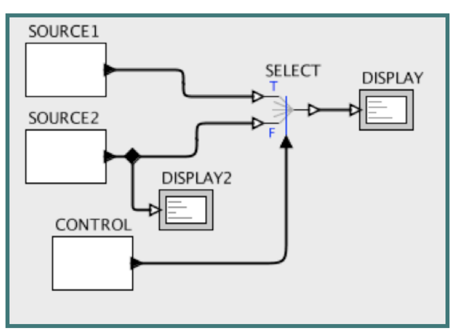
Will Demand-Driven Execution Work Here?



Lee 01: 23

Question 4:
Will Data-Driven Execution Work?

In data-driven execution, a process is stalled unless it has input data. What about the processes with no inputs?



Lee 01: 24

Things are not looking good…

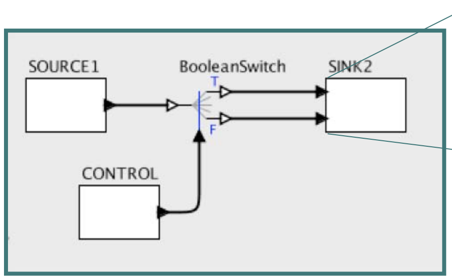
We have ruled out:

- Fair execution.
- Rendezvous communication.
- Demand-driven execution.
- Data-driven execution.

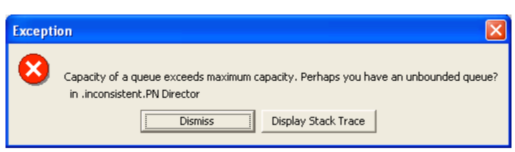
For all the examples given so far, there is an obvious execution policy that does what we want. Is there a general policy that will always deliver that obvious policy?

Are there models for which the policy is not so obvious?

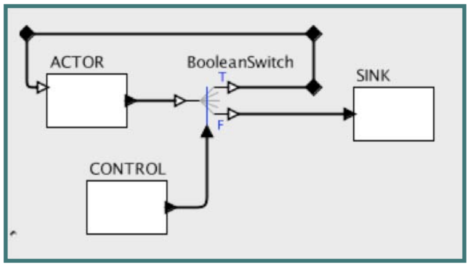
Question 5: What is the "Correct" Execution of This Model?

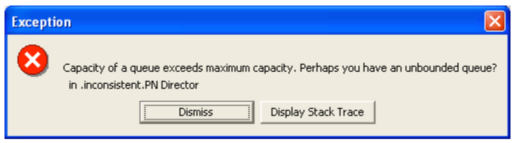



```
while(true) {
  data1 = in1.get();
  data2 = in2.get();
  … do something with it …
}
```

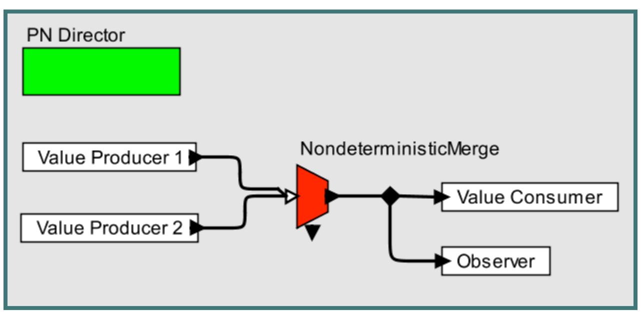


Question 6:
What is the Correct Behavior of this Model?





Question 7:
How to support nondeterminism?



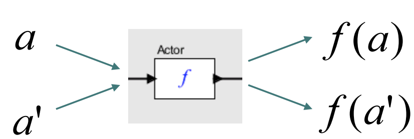
Merging of streams is needed for some applications. Does this require fairness? What does fairness mean?

Properties of PN (Two Big Topics)

- Assuming "well-behaved" actors, a PN network is determinate in that the sequence of tokens on each arc is independent of the thread scheduling strategy.
  - Making this statement precise, however, is nontrivial.
- PN is Turing complete.
  - Given only boolean tokens, memoryless functional actors, Switch, Select, and initial tokens, one can implement a universal Turing machine.
  - Whether a PN network deadlocks is undecidable.
  - Whether buffers grow without bound is undecidable.

Lee 01: 29

PN Semantics
Where This is Going



A signal is a sequence of values

Define a prefix order:

$$a \sqsubseteq a'$$

means that a is a prefix of a'.

Actors are *monotonic* functions:

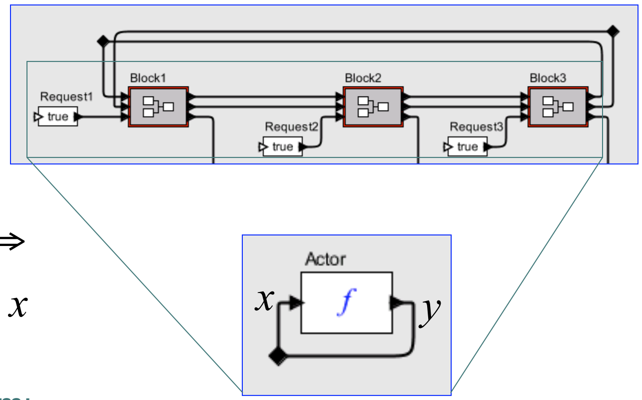
$$a \sqsubseteq a' \quad \Rightarrow \quad f(a) \sqsubseteq f(a')$$

Stronger condition: Actors are *continuous* functions (intuitively: they don't wait forever to produce outputs).

Lee 01: 30

PN Semantics of Composition (Kahn, '74) This Approach to Semantics is "Tarskian"

If the components are deterministic, the composition is deterministic.



$$x = y \Rightarrow$$
$$f(x) = x$$

Fixed point theorem:

- Continuous function has a unique least fixed point
- Execution procedure for finding that fixed point
- Successive approximations to the fixed point

What is Order?

Intuition:

1. 0 < 1
2. 1 < ∞
3. child < parent
4. child > parent
5. 11,000/3,501 is a better approximation to π than 22/7
6. integer n is a divisor of integer m.
7. Set A is a subset of set B.

Which of these are *partial* orders?

Relations

- A *relation* R from A to B is a subset of $A \times B$
- A *function* F from A to B is a relation where
  $(a, b) \in R$ and $(a, b') \in R \Rightarrow b = b'$
- A *binary relation* R on A is a subset of $A \times A$
- A *binary relation* R on A is *reflexive* if
  $\forall\ a \in A,\ (a, a) \in R$
- A *binary relation* R on A is *symmetric* if
  $(a, b) \in R \Rightarrow (b, a) \in R$
- A *binary relation* R on A is *antisymmetric* if
  $(a, b) \in R$ and $(b, a) \in R \Rightarrow a = b$
- A *binary relation* R on A is *transitive* if
  $(a, b) \in R$ and $(b, c) \in R \Rightarrow (a, c) \in R$

Infix Notation for Binary Relations

- $(a, b) \in R$ can be written $a\ R\ b$
- A symbol can be used instead of R. For examples:
  - $\leq \subset N \times N$ is a relation.
  - $(a, b) \in \leq$ is written $a \leq b$
- A function $f \in (A, B)$ can be written $f : A \rightarrow B$

Partial Orders

A *partial order* on the set A is a binary relation $\leq$ that is:

For all $a, b, c \in A$,

- reflexive: $a \leq a$
- antisymmetric: $a \leq b$ and $b \leq a \Rightarrow a = b$
- transitive: $a \leq b$ and $b \leq c \Rightarrow a \leq c$

A *partially ordered set* (*poset*) is a set A and a binary relation $\leq$, written $(A, \leq)$.

Strict Partial Order

For every partial order $\leq$ there is a *strict partial order* $<$ where $a < b$ if and only if $a \leq b$ and $a \neq b$.

A *strict poset* is a set and a strict partial order.

Total Orders

Elements a and b of a poset $(A, \leq)$ are *comparable* if either $a \leq b$ or $b \leq a$. Otherwise they are *incomparable*.

A poset $(A, \leq)$ is *totally ordered* if every pair of elements is comparable.

Totally ordered sets are also called *linearly ordered sets* or *chains*.

Quiz

1. Is the set of integers with the usual numerical ordering a well-ordered set? (A *well-ordered set* is a set where every non-empty subset has a least element.)

2. Given a set A and its *powerset* (set of all subsets) $P(A)$, is $(P(A), \subseteq)$ a poset? A chain?

3. For $A = \{a, b, c\}$ (a set of three letters), find a well-ordered subset of $(P(A), \subseteq)$.

Answers

1. Is the set of integers with the usual numerical ordering a well-ordered set?
   No. The set itself is a chain with no least element.

2. Given a set A and its *powerset* (set of all subsets) $P(A)$, is $(P(A), \subseteq)$ a poset? A chain?
   It is a poset, but not a chain.

3. For $A = \{a, b, c\}$ (a set of three letters), find a well-ordered subset of $(P(A), \subseteq)$.
   One possibility: $\{\emptyset, \{a\}, \{a, b\}, \{a, b, c\}\}$

Lee 01: 39

Pertinent Example: Prefix Orders

Let T be a type (a set of values).
Let T^{**} be the set of all finite and infinite sequences of elements of T, including the empty sequence $\perp$ (bottom).

Let $\sqsubseteq$ be a binary relation on T^{**} such that $a \sqsubseteq b$ if a is a *prefix* of b. That is, for all n in N such that $a(n)$ is defined, then $b(n)$ is defined and $a(n) = b(n)$.

This is called a *prefix order*.

During execution, the outputs of a PN actor form a well-ordered subset of $(T^{**}, \sqsubseteq)$.

Lee 01: 40

Summary

- Concurrent models of computation
- Process networks as an example
- Intuitive model, but many subtle corner cases
- Need a solid theory underlying it
- Posets

- *Next time:*
  - *give meaning to all programs*
  - *develop an execution policy*