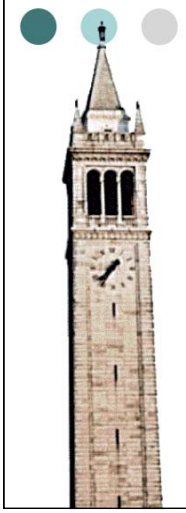


Concurrent Models of Computation

Edward A. Lee

Robert S. Pepper Distinguished Professor, UC Berkeley

EECS 290n – Advanced Topics in Systems Theory

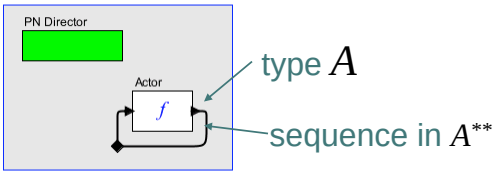
Concurrent Models of Computation

Spring 2009

Week 3: Execution Policies

Review:

Semantics of a PN Model is the Least Fixed Point of a Monotonic Function

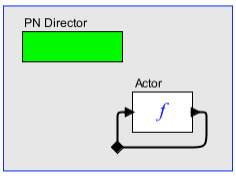


- Chain: $C = \{ f(\bot), f(f(\bot)), \ldots, f^n(\bot), \ldots \}$
- Continuity: $f(\vee C) = \vee \hat{f}(C)$

Limits

Composing Actors

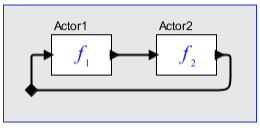
So far, our theory applies only to a single actor in a feedback loop:



What about more interesting models?

Cascade Composition

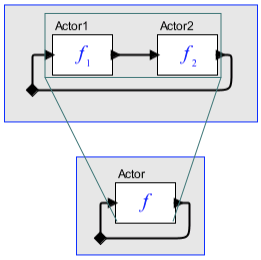
Consider cascade composition:



If $f_1 : A \to B$ and $f_2 : B \to C$ are monotonic (or continuous) functions on CPOs A, B, C, then $f_2 \circ f_1$ is monotonic (or continuous) (show this).

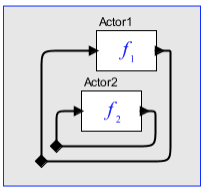
Hence, the execution procedure works for cascade composition.

Cascade Composition
Reduces to the Previous Case



Parallel Composition

Consider parallel composition:



If $f_1 : A \to A$ and $f_2 : B \to B$ are monotonic (or continuous) functions on CPOs A, B, then $f_1 \times f_2$ is monotonic (or continuous) on CPOs $A \times B$.

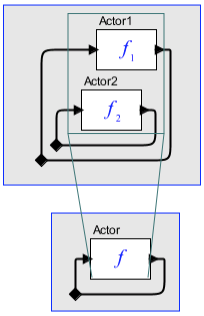
Cartesian Products of Functions

If $f_1 : A \to A$ and $f_2 : B \to B$ then the Cartesian product is $f_1 \times f_2 : A \times B \to A \times B$.

If A, B are CPOs then so is $A \times B$ under the pointwise order.

Exercise: Determine whether $A \times B$ is a CPO under the lexicographic order.

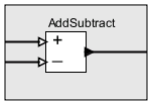
Parallel Composition Reduces to the Previous Case



Discussion: Does this composite actor have two streams in and out? Or streams of two-tuples?

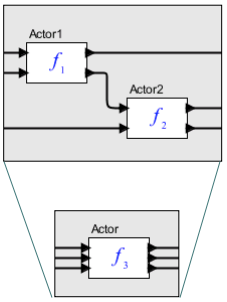
Multiple Inputs or Outputs

What about actors with multiple inputs or outputs?



More Interesting Feedback Compositions

Assuming f_1 and f_2 are monotonic, is f_3 monotonic?
Assuming f_1 and f_2 are continuous, is f_3 continuous?
Assuming f_1 and f_2 are sequential, is f_3 sequential?



Sequential Functions

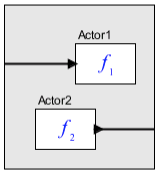
Let $F : S^p \rightarrow S^q$ be a function mapping p-ary inputs to q-ary inputs.

F is *sequential* if for any $\boldsymbol{X} = \{ X_1, X_2 \ldots X_p \}$, there exists an i, $1 \leq i \leq p$, such that for any $\boldsymbol{X}'$ where $\boldsymbol{X} \sqsubseteq \boldsymbol{X}'$ and $X_i = X_i'$, it is $F(X_i) = F(X_i')$.

Hence $\boldsymbol{X}'$ extends the streams (sequences) in $\boldsymbol{X}$, except the one stream X_i , which is not extended. The process is sequential if it cannot extend the output before X_i is extended. Intuitively, this corresponds to a blocking read on X_i , the outcome of which determining further computation of F.

Note that sequentiality implies continuity, which in turn implies monotonicity.

Source and Sink Actors

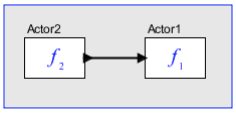


Consider Actor1. Its function is $f_1 : A^1 \rightarrow A^0$ where A^0 is a *singleton set* (a set with one element). Such a function is always monotonic (and continuous, and sequential).

Consider Actor2. Its function is $f_1 : A^0 \rightarrow A^1$. Such a function is again always monotonic (and continuous, and sequential). In fact, the function can only yield one possible output sequence, since its domain has size 1.

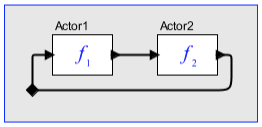
Composing Sources and Sinks

What about the following interconnection?

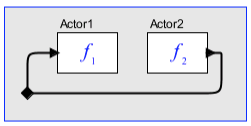


Composing Sources and Sinks

Recall cascade composition:



Reorganized, this looks like cascade composition:



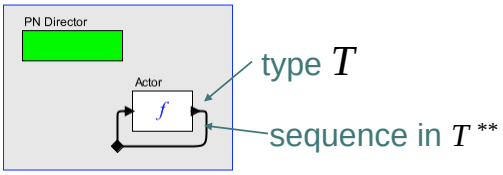
The codomain of f_1 and domain of f_2 are singleton sets, so there is no need to show any signal.

Complicated Compositions

Simple procedure:

- Bring all n signals out as outputs.
- Feed back all n signals as inputs.
- The resulting $f : A^n \rightarrow A^n$ will be continuous if the component functions are continuous.
- Hence the model will have a least fixed point that can be found by starting with all sequences being empty and repeatedly applying the function f.

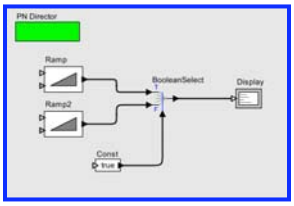
Taking Stock: Semantics of a PN Model is the Least Fixed Point of a Monotonic Function



- Chain: $C = \{ f(\bot), f(f(\bot)), \ldots, f^n(\bot), \ldots \}$
- Continuity: $f(\vee C) = \vee \hat{f}(C)$

Limits

Applying This In Practice

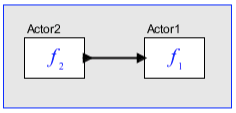


- Model is a composition of actors
- Each actor implements a monotonic function
- The composition is a monotonic function
- All signals are part of the “feedback”
- Execution approximates the semantics by
  - starting with empty sequences on all signals
  - allowing actors to react to inputs and build output signals
- Actors execute in their own thread.
- Reads of empty inputs block.

Practical Questions

- When a process suspends, how should you decide which process to activate next?
- If a process does not (voluntarily) suspend, when should you suspend it?
- How can you ensure “fairness”? In fact, what does “fairness” mean?
  - All inputs to a process are eventually consumed?
  - All outputs that a process can produce are eventually produced?
  - All processes are given equal opportunity to run? What does “equal opportunity” mean?

Consider a Simple Example



How can we prevent Actor2 from never suspending, thus starving Actor1 and causing memory usage to explode?

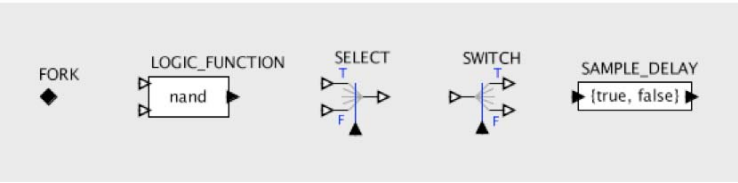
How can we prevent buffers from growing infinitely (data is produced a higher rate than it is consumed)?

Naïve answers:

- Fair execution: Give both actors equal time slices
- Data-driven execution: When Actor2 produces, execute Actor1
- Demand-driven execution: When Actor1 needs, execute Actor2
- Bound the buffer between them and implement blocking writes.

Undecidability [Buck, 1993]

Given the following four actors, and boolean data types on the ports, you can construct a universal Turing machine:



Consequence: The following questions are undecidable:

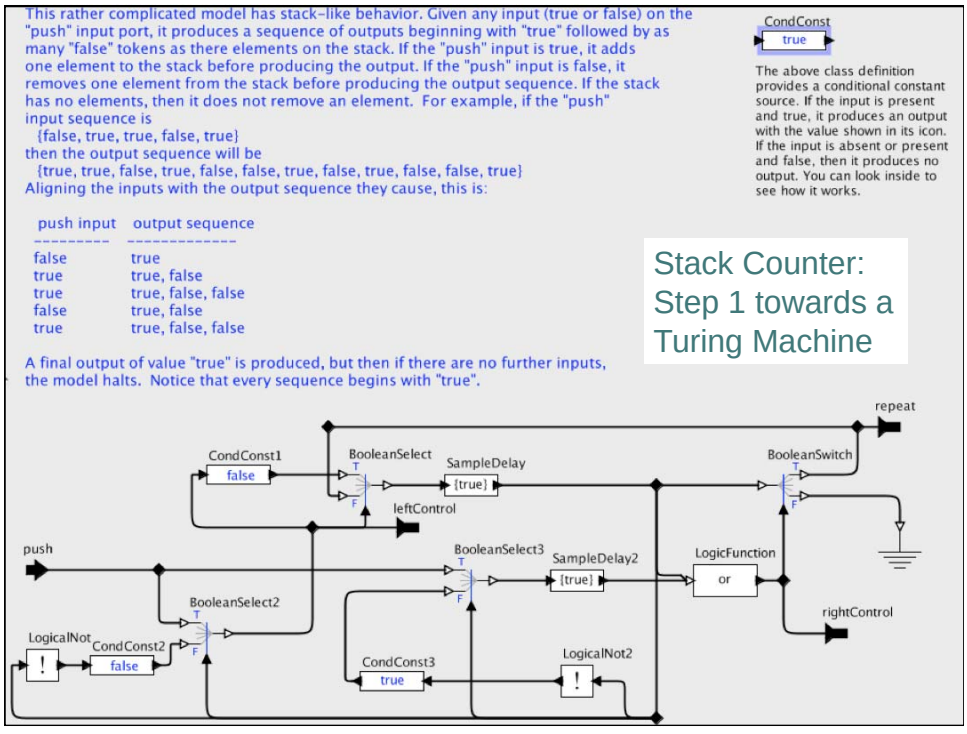
- Will a PN model deadlock?
- Can a PN model be executed in bounded memory?

Consequences

It is undecidable whether a PN model can execute in bounded memory, so no terminating algorithm can identify (for all PN models) bounds that are safe to use on the channels.

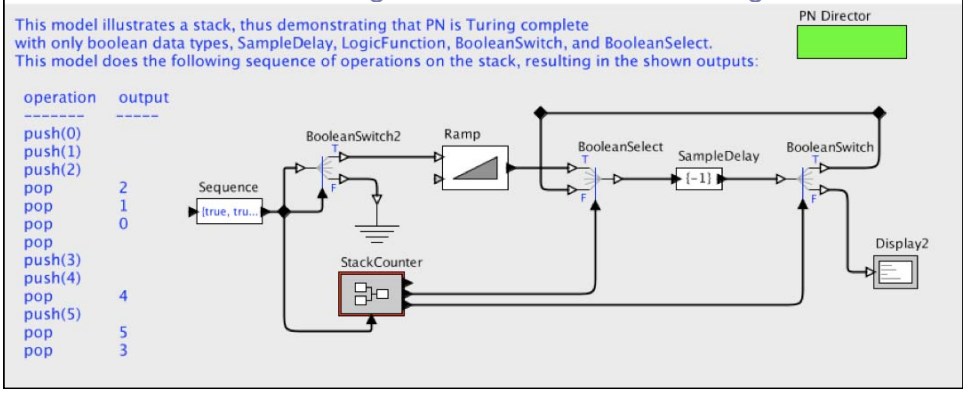
A PN model *terminates* if every signal is finite in the least fixed point semantics.

It is undecidable whether a PN model terminates.



Turing Machines from PN

The model below shows how to implement a stack. Given a stack, you can make two stacks, one representing the tape to the left and one representing the tape to the right on a Turing machine. With some logic, voila, a universal Turing machine.



A Practical Policy

- Define a *correct execution* to be any execution for which after any finite time every signal is a prefix of the LUB signal given by the semantics.
- Define a *useful execution* to be a correct execution that satisfies the following criteria:
  1. For every non-terminating PN model, after any finite time, a useful execution will extend at least one signal in finite (additional) time.
  2. If a correct execution satisfying criterion (1) exists that executes with bounded buffers, then a useful execution will execute with bounded buffers.

Parks' Strategy [Parks, 1995]

- Start with an arbitrary bound on the capacity of all buffers.
- Execute with both blocking reads and blocking writes (which prevent buffers from overflowing).
- If deadlock occurs and at least one actor is blocked on a write, increase the capacity of the smallest buffer to unblock at least one write.
- Continue executing, repeatedly checking for deadlock.

This is the strategy implemented in the PN domain in Ptolemy II. Notice that it "solves" two undecidable problems, but does so in infinite time.

More Execution Policy Considerations

Fairness: If a process at some point becomes able to send a message, then the message is eventually sent. If a process at some point becomes able to read a message that has been sent, then it eventually reads the message.

Maximality: An execution is maximal if it is finite (all processes terminate) or when it halts, all processes that have not terminated are blocked on reads.

Proposition (the Kahn Principle): Any two fair and maximal executions of a process network produce the same sequences of messages, matching the least fixed point in the Kahn semantics.

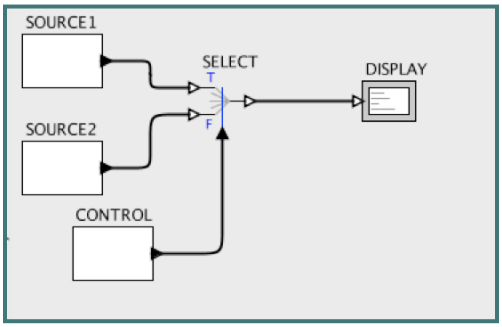
Geilen & Basten's Elaboration [2003]

Execute in a "data-driven" fashion with bounded buffers until the process network terminates or an *artificial local deadlock* occurs, defined to be a cycle of blocked processes waiting on each other in a chain where at least one process is blocked on a write.

Resolve any such artificial deadlock by increasing the capacity of the smallest full buffer in the cycle.

This strategy is fair and maximal for *effective* PNs (where every message sent is read) and will execute in bounded memory if this is possible.

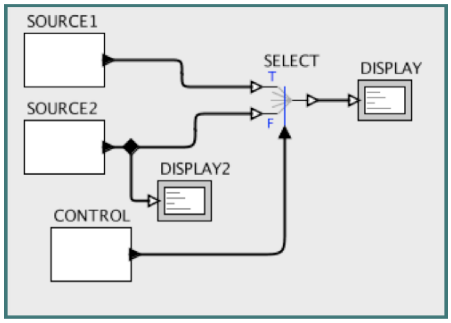
Questions 1 & 2: (from first week)
Is "Fair" Thread Scheduling a Good Idea?



Suppose the CONTROL output is always *true*. (In this case, is the network effective?)

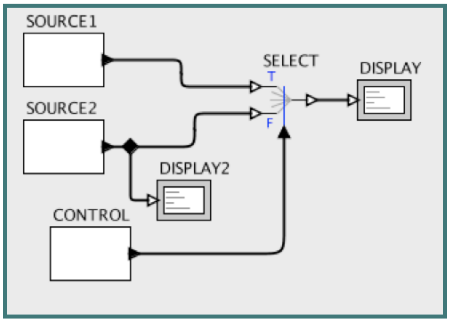
A "useful execution" will allow SOURCE2 to produce only finite output. This is unfair. Both strategies do this.

Question 3: (from first week)
When are Outputs Required?



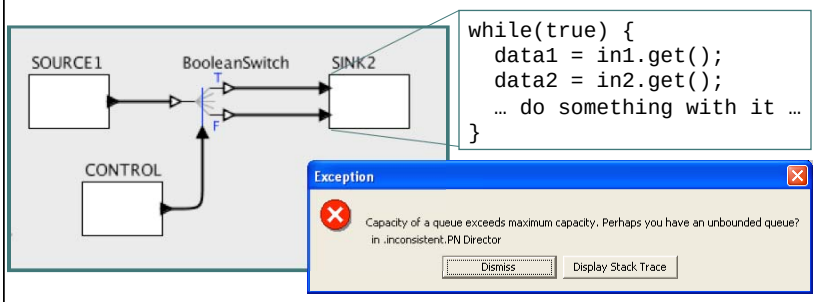
The “useful execution” is not changed by the mere act of observing a signal. Again, both strategies are unfair.

Question 4: (from last week)
Will Data-Driven Execution work?



A useful execution does not execute the sources merely because they have input data to depend on. You still need to bound the buffers. Geilen and Basten assume in the above network that any bound >= 1 will not lead to deadlock.

Question 5: (from first week)
What is the "Correct" Execution of This Model?

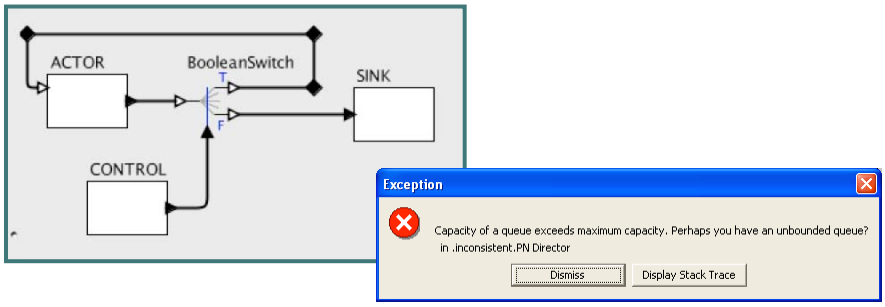


The PN Director optionally allows you to specify an overall bound on buffer sizes as a debugging tool.

Both strategies execute in bounded memory if the output of the CONTROL process makes this possible.

In G&B, if the BooleanSwitch blocks on a write to a full output buffer, then SINK must be blocked on a read from its other output, so we have a cycle.

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



Any maximal execution of this model requires unbounded buffers, assuming ACTOR reads its input first. This *local deadlock* is not an artificial deadlock.

Convergence

The Kahn principle states that maximal and fair executions produce sequences that match that the least fixed point.

But what can this mean? Every execution is, at all times, finite, and can only have produced a prefix of the least fixed point.

We can define a notion of convergence for sequences.

Limit of a Sequence of Reals

Consider a sequence of real numbers:

$$s : N \rightarrow \Re$$

This sequence is said to *converge* to a real number a if for all open sets A containing a there exists an integer n such that for all $m > n$ the following holds:

$$s(m) \in A$$

Standard Topology in the Reals

An *open neighborhood* around a in the reals is

$$\{ x \in \Re \mid a - \varepsilon < x < a + \varepsilon \}$$

for some positive real number ε.

An *open set* A in the reals is a subset of $\Re$ such that for all $a \in A$, there is an open neighborhood around a that is a subset of A.

The collection of open sets in the reals is called a *topology*.

Topology

Let X be any set. A collection τ of subsets of X is called a *topology* if:

- X and $\varnothing$ are members of τ
- The intersection of any two members of τ is in τ
- The union of any family of members of τ is in τ

The set of open sets in the reals is a *topology*.

For any topology τ, the members of τ are called its open sets.

A Scott Topology for Sequences

Consider a set T and the set T^{**} of all finite and infinite sequences of elements of T.

Given a *finite* sequence $s \in T^{**}$, an *open neighborhood* around s is the set

$$N_s = \{ s' \in T^{**} \mid s \sqsubseteq s' \}$$

It is the set of sequences with prefix s.

Let τ be the collection of all sets that arbitrary unions of open neighborhoods. Fact: τ is a topology.

Limit of a Sequence of Sequences (Convergence in the Scott Topology)

Consider a sequence of sequences:

$$S : N \to T^{**}$$

This sequence is said to *converge* to a sequence a if for all open sets A containing a there exists an integer n such that for all $m > n$ the following holds:

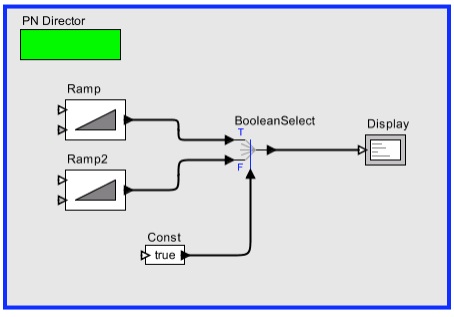
$$S(m) \in A$$

Intuition: For any finite prefix $p \sqsubseteq a$, the sequences in S eventually all have prefix p .

Consequences for Process Networks

- "Correct" executions of process networks do not necessarily converge to the Kahn least fixed-point semantics.
- This is because "correct" executions allow any signal to be evaluated only to a finite prefix of the LUB semantics.
- Maximal and fair executions, however, do converge.
- For networks that not effective, however, such executions may not be the ones you want!

Convergent Execution vs. Correct Execution



- A "convergent" execution of the above model is impossible with finite memory.
- A "correct" execution is possible and practical.

Which do you prefer?

Nondeterminism

Gordon Plotkin postulated that the Kahn principle could be extended to nondeterminate systems by modeling processes as functions on powersets of sequences.

That is, a nondeterminate process with n inputs and one output, for example, could be defined by a function of the form $F : (T^*)^n \rightarrow P(T^{**})$, where $P(T^{**})$ is the set of all subsets of T^{**}.

Brock and Ackermann put an end to this with an example where two components characterized by exactly the same such function nonetheless exhibited observably different behavior.

Summary

- Deadlock and memory requirements are undecidable for PN.
- Correct and useful executions can be practically achieved despite this fact using Parks' strategy.
- The result does not converge to the Kahn semantics, but always delivers a finite prefix of that semantics.