



Concurrent Models of Computation for Embedded Software

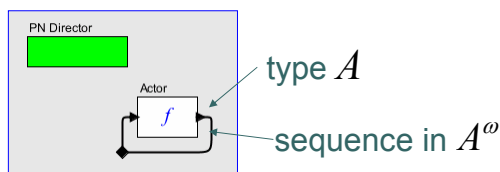
Edward A. Lee

Professor, UC Berkeley
EECS 290n – Advanced Topics in Systems Theory
Fall, 2004

Copyright © 2004, Edward A. Lee, All rights reserved

Lecture 8: Execution of Process Networks

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

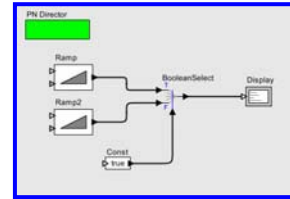


- Chain: $C = \{f(\perp), f(f(\perp)), \dots, f^n(\perp), \dots\}$

- 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.

Lee 08: 3

Kahn-MacQueen Blocking Reads

Following Kahn-MacQueen [1977], actors are threads that implement *blocking reads*, which means that when they attempt to read from an empty input, the thread stalls.

- This restricts expressiveness more than continuity
- This still leaves open the question of thread scheduling

Lee 08: 4

Blocking Reads Realize Sequential Functions [Vuillemin]

Let $f: A^n \rightarrow A^m$ be an n input, m output function.

Then f is *sequential* if it is continuous and for any $a, b \in A^n$ where $a \leq b$ there exists an $i \in \{1, \dots, n\}$, where:

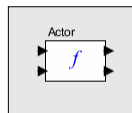
$$\pi_i(a) = \pi_i(b) \Rightarrow f(a) = f(b)$$

Intuitively: At all times during an execution, there is an input channel that blocks further output. This is the Kahn-MacQueen blocking read!

Lee 08: 5

Continuous Function that is not Sequential

Two input identity function is not sequential:



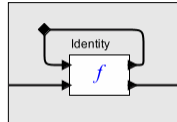
Let $f: A^2 \rightarrow A^2$ such that for all $a \in A^2$, $f(a) = a$.

Then f is not sequential.

Lee 08: 6

Cannot Implement the Two-Input Identity with Blocking Reads

Consider the following connection:

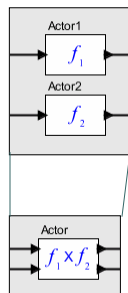


This has a well-defined behavior, but an implementation of the two-input identity with blocking reads will fail to find that behavior.

Lee 08: 7

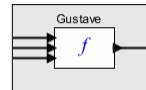
Sequential Functions do not Compose

If $f_1 : A \rightarrow B$ and $f_2 : C \rightarrow D$ are sequential then $f_1 \times f_2$ may or may not be sequential. Simple example: suppose f_1 and f_2 are identity functions in the following:



Lee 08: 8

Gustave Function Non Sequential but Continuous



Let $A = T^{**}$ where $T = \{t, f\}$.

Let $f: A^3 \rightarrow N^{**}$ such that for all $a \in A^3$,

$$f(a) = \begin{cases} (1) & \text{if } ((t), (f), \perp) \sqsubseteq a \\ (2) & \text{if } (\perp, (t), (f)) \sqsubseteq a \\ (3) & \text{if } ((f), \perp, (t)) \sqsubseteq a \end{cases}$$

This function is continuous but not sequential.

Lee 08: 9

Linear Functions [Erhard]

Function $f: A \rightarrow B$ on CPOs is *linear* if for all joinable sets $C \sqsubseteq A$, $\hat{f}(C)$ is joinable and

$$\vee \hat{f}(C) = f(\vee C)$$

Intuition: If two possible inputs can be extended to a common input, then the two corresponding outputs can be extended to the common output.

Fact: Sequential functions are linear.

Fact: Linear functions are continuous (trivial)

Lee 08: 10

Stable Functions [Berry]

Function $f: A \rightarrow B$ on CPOs is *stable* if it is continuous and for all joinable sets $C \subseteq A$, $\hat{f}(C)$ is joinable and

$$\bigwedge \hat{f}(C) = f(\bigwedge C) \quad \longleftarrow \text{NOTE: meet! not join!}$$

Intuition: If two possible inputs do not contain contradictory information, then neither will the two corresponding outputs.

Fact: Sequential functions are stable.

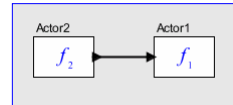
Lee 08: 11

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?

Lee 08: 12

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)?

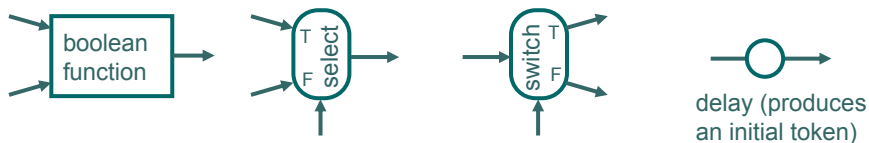
Naïve answers:

- 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.

Lee 08: 13

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?

Lee 08: 14

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.

Lee 08: 15

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.

Lee 08: 16

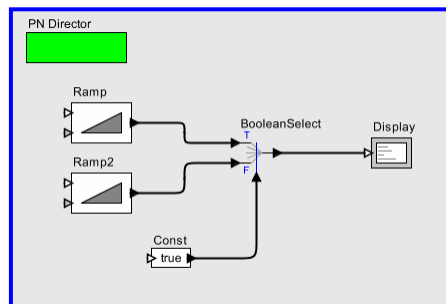
Parks' Strategy [Parks, 1995]

- 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 at least one 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.

Lee 08: 17

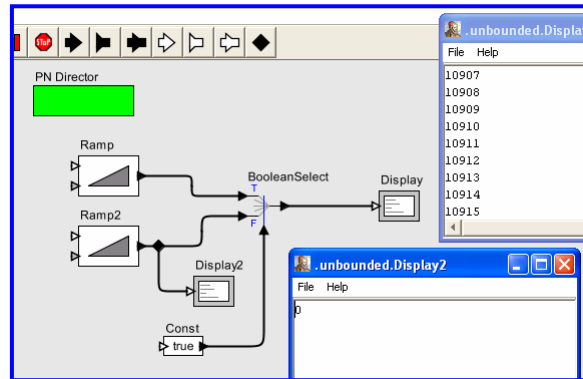
Questions 1 & 2: (from lecture 4) Is “Fair” Thread Scheduling a Good Idea?



A “useful execution” will allow Ramp2 to produce only finite output.

Lee 08: 18

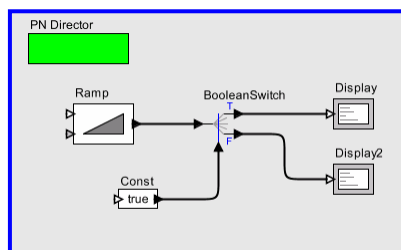
Question 3: (from lecture 4) When are Outputs Required?



The “useful execution” is not changed by the mere act of observing a signal.

Lee 08: 19

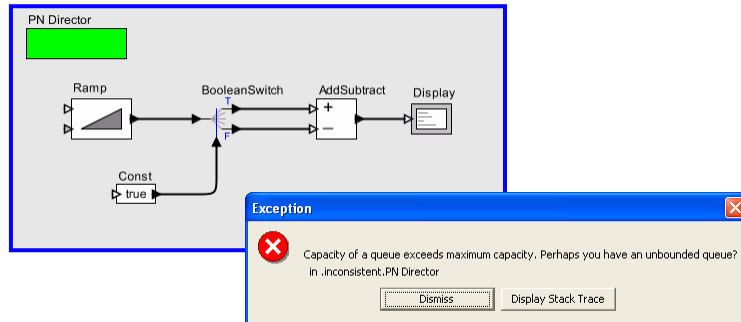
Question 4: (from lecture 4) Is “Demand-Driven” Execution a Good Idea?



A useful execution of this is not frustrated by the lack of data to Display2.

Lee 08: 20

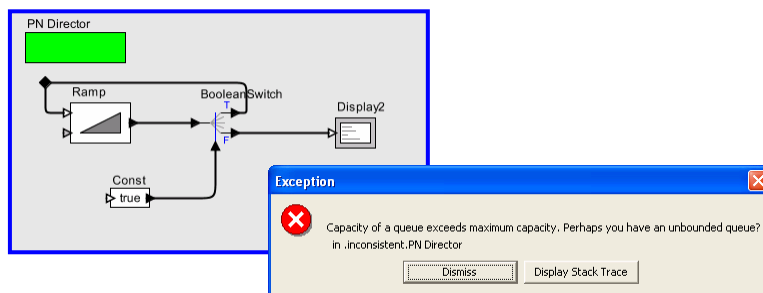
Question 5: (from lecture 4) What is the “Correct” Execution of This Model?



The PN Director optionally allows you to specify an overall bound on buffer sizes. **This is a debugging tool, not a change in the semantics!**

Lee 08: 21

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

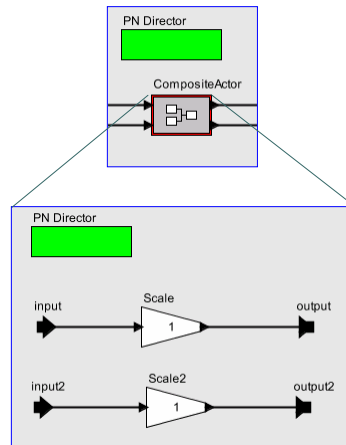


A correct behavior of this model (like the previous one) requires unbounded buffers.

Lee 08: 22

A Deeper Question

How can process networks be composed?



Lee 08: 23

Conclusion

- Processes with blocking reads realize sequential functions, a subset of monotonic functions.
- Sequential functions are (regrettably) not compositional.
- Deadlock and memory requires are undecidable for PN.
- Correct and useful executions can be practically achieved despite this fact using Parks' strategy.
- Compositionality questions still have to be addressed.

Lee 08: 24