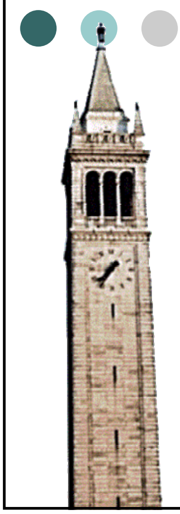


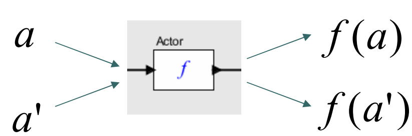
# Concurrent Models of Computation for Embedded Software

Edward A. Lee
Professor, UC Berkeley
EECS 290n – Advanced Topics in Systems Theory
Fall, 2004



Lecture 7: Continuous Functions and PN Composition

## PN Actors are Monotonic Functions on a CPO




Set of signals with the prefix order is a CPO.

Actors are *monotonic* functions:

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

This is a timeless *causality* condition.

## Continuous (Limit Preserving) Functions

Let $(A, \leq)$ and $(B, \leq)$ be CPOs.

A function $f: A \to B$ is called *continuous* if for all chains $C \subseteq A$,

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

*Notation*: Given a function $f: A \to B$, define a new function $\hat{f}: P(A) \to P(B)$, where for any $C \subseteq A$,

$$\hat{f}(C) = \{b \in B \mid \exists c \in C \text{ s.t. } f(c) = b\}$$



## Continuous vs. Monotonic

Fact: Every continuous function is monotonic.

- Easy to show (consider chains of length 2)

Fact: If every chain in $A$ is finite, then every monotonic function $f: A \to B$ is continuous.

But: If $A$ has infinite chains, the monotonic does not imply continuous.

## Counterexample Showing that Monotonic Does Not Imply Continuous

Let $A = (N \cup \{\infty\}, \leq)$ (a CPO).
Let $f : A \to A$ be given by

$$f(a) = \begin{cases} 1 & \text{if } a \text{ is finite} \\ 2 & \text{otherwise} \end{cases}$$

This function is obviously monotonic. But it is not continuous. To see that, let $C = \{1, 2, 3, \ldots\}$, and note that $\vee C = \infty$. Hence,

$$f(\vee C) = 2$$
$$\vee f(C) = 1$$

which are not equal.



## Intuition

Under the prefix order, for any monotonic functions that is not continuous, there is a continuous function that yields the same result for every finite input.

For practical purposes, we can assume that any monotonic function is continuous, because the only exceptions will be functions that wait for infinite input before producing output.

## Fixed Point Theorem 1

Let $(A, \leq)$ be a CPO with bottom $\perp$
Let $f: A \to A$ be a monotonic function
Let $C = \{ f^n(\perp), n \in N \}$

- If $f$ is continuous, then $\vee C = f(\vee C)$
- If $\vee C = f(\vee C)$, then $\vee C$ is the *least* fixed point of $f$

Intuition: The least fixed point of a continuous function is obtained by applying the function first to the empty sequence, then to the result, then to that result, etc.



## Proof (Continuous Part)

Note that $C$ is a chain in a CPO (show this) and hence has a LUB $\vee C$.

Let $C' = C \cup \{ \perp \}$ and note that $\vee C = \vee C'$.
Note further that $\hat{f}(C') = C$ and hence $\vee \hat{f}(C') = \vee C$
By continuity, $\vee \hat{f}(C') = f(\vee C') = f(\vee C)$
Hence $\vee C = f(\vee C)$

QED ( $\vee C$ is a fixed point of $f$ )

## Proof (Least Fixed Point Part)

NOTE: This part does not require continuity.

Let $a$ be another fixed point: $f(a) = a$

Show that $\vee C$ is the least fixed point: $\vee C \leq a$

Since $f$ is monotonic:

$$\perp \leq a$$

$$f(\perp) \leq f(a) = a$$

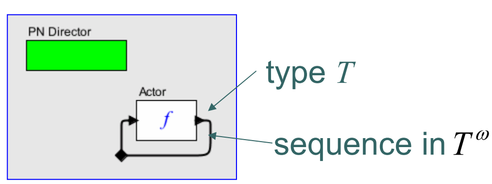
$$\ldots$$

$$f^k(\perp) \leq f^k(a) = a$$

So $a$ is an upper bound of the chain $C$, hence $\vee C \leq a$.



## Fixed Point Semantics




- Start with the empty sequence.
- Apply the (continuous) function.
- Apply the function again to the result.
- Repeat forever.

The result "converges" to the least fixed point.

## Fixed Point Theorem 2

Let $f : A \to A$ be a monotonic function on CPO $(A, \leq)$.
Then $f$ has a least fixed point.

Intuition: If a function is monotonic (but not continuous), then it has a least fixed point, but the execution procedure of starting with the empty sequence and iterating may not converge to that fixed point.

This is obvious, since monotonic but not continuous means it waits forever to produce output.

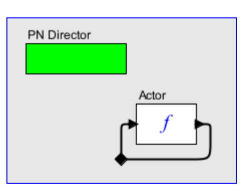


## Example 1: Identity Function

Let $A = T^{**}$ and $f : A \to A$ be such that $\forall a \in A$, $f(a) = a$.

This is obviously continuous (and hence monotonic) under the prefix order.

Then the model below has many fixed points, but only one least fixed point (the empty sequence).

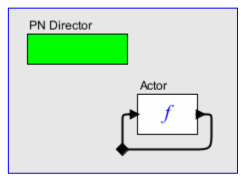


## Example 2: Delay Function

Let $A = T^{**}$ and $f : A \to A$ be such that $\forall a \in A$, $f(a) = t.a$ (concatenation), where $t \in T$.

This is obviously continuous (and hence monotonic) under the prefix order.

Then the model below has only one fixed point, the infinite sequence $(t, t, t, \ldots)$

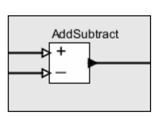



Why is this called a "delay?" In the feedback loop, it functions like Const.



## Multiple Inputs or Outputs

What about actors with multiple inputs or outputs?



## Cartesian Products of Posets

Let $(A, \leq)$ and $(B, \leq)$ be CPOs.
Then $A \times B$ is a CPO under the pointwise order.

Pointwise order: $(a_1, b_1) \leq (a_2, b_2) \Leftrightarrow a_1 \leq a_2$ and $b_1 \leq b_2$

Contrast with lexicographic order:
$(a_1, b_1) \leq (a_2, b_2) \Leftrightarrow a_1 \leq a_2$ or $a_1 = a_2$ and $b_1 \leq b_2$

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



## More Cartesian Products and Projections

Let $(A, \leq)$ be a CPO.
Let $A^n$ denote $A \times A \times \ldots \times A$ , $n$ times

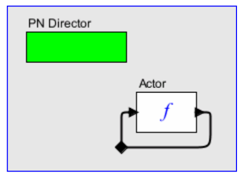
Then $(A^n, \leq)$ is a CPO under the pointwise order for any natural number $n$.

For any $a = \{a_1, \ldots, a_n\} \in A^n$ and $i \in \{1, \ldots n\}$, define the *projection* on $i$ to be:

$$\pi_i(a) = \{a_1, \ldots, a_n\}$$

## Composing Actors

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

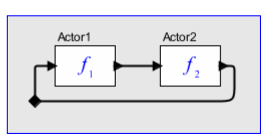



What about more interesting models?



## Cascade Composition

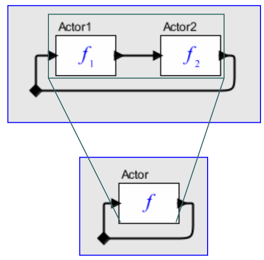
Consider cascade composition:




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

Hence, the execution procedure works for cascade composition.

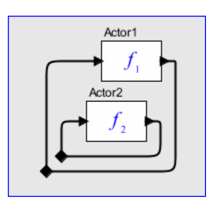
## Cascade Composition
## Reduces to the Previous Case






## Parallel Composition

Consider parallel composition:




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

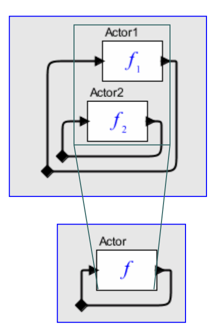
## Cartesian Products of Functions

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

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

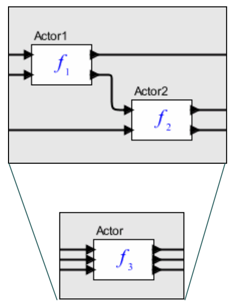


## Parallel Composition
## Reduces to the Previous Case



## More Interesting Feedback Compositions

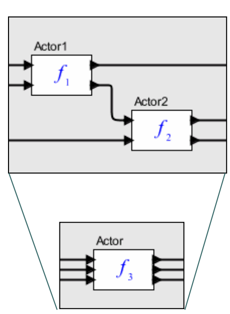
Assuming $f_1$ and $f_2$ are monotonic, is $f_3$ monotonic?
Assuming $f_1$ and $f_2$ are continuous, is $f_3$ continuous?
Assuming $f_1$ and $f_2$ are sequential, is $f_3$ sequential?



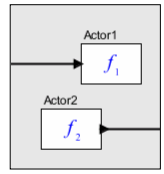



## More Interesting Feedback Compositions

Assuming $f_1$ and $f_2$ are monotonic, is $f_3$ monotonic? *yes*
Assuming $f_1$ and $f_2$ are continuous, is $f_3$ continuous? *yes*
Assuming $f_1$ and $f_2$ are sequential, is $f_3$ sequential? *no*



## Source and Sink Actors



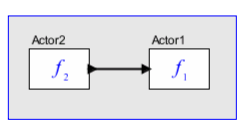

Consider Actor1. Its function is $f_1 \colon A^1 \to A^0$ where $A^0$ is a *singleton set* (a set with one element). Such a function is always monotonic (and continuous, and sequential).

Consider Actor2. Its function is $f_1 \colon A^0 \to A^1$. 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.



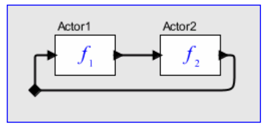
## Composing Sources and Sinks

What about the following interconnection?

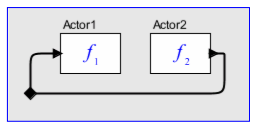


## Composing Sources and Sinks

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 $f: A^n \to A^n$ 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 $f$.

## Conclusion

Continuous functions compose, sequential functions do not.

Implementing sequential functions is easy (blocking reads). Implementing continuous functions can be hard.