

## Synchronous Languages

- Esterel
- Lustre
- SCADE (visual editor for Lustre-ish/Esterel-ish lang.)
- Signal
- Statecharts (some variants)
- Ptolemy II SR domain

The model of computation is called synchronous reactive (SR). It has strong formal properties (many key questions are decidable).


## SR Domain in Ptolemy II

At each tick of a global "clock," every signal has a value or is absent.


The job of the SR director is to find the value at each tick.


## The Synchronous Abstraction

o "Model time" is discrete: Countable ticks of a clock.

- WRT model time, computation does not take time.
- All actors execute "simultaneously" and "instantaneously" (WRT to model time).
o There is an obviously appealing mapping onto real time, where the real time between the ticks of the clock is constant. Good for specifying periodic realtime tasks.


## Properties

- Buffer memory is bounded (obviously).
- Hence the model of computation is not Turing complete.
- ... or bounded memory would be undecidable ...
- Causality loops are possible, where at a tick, the value of one or more signals cannot be determined.


## Practical Application - Token Ring Arbitration



## Arbiter Design



## Simple Execution Policy

At each tick, start with all signals "unknown." Evaluate non-strict actors and source actors. Then keep evaluating any actors that can be evaluated until all signals become known or until no further progress can be made.

Q: How do we know this will work?

A: Least fixed point semantics.

## SR Domain in Ptolemy II

At each tick of a global "clock," every signal has a value or is absent.


The job of the SR director is to find the value at each tick.


## Cycles

Note that there are cycles in this graph, so that if you require that all inputs be known to find the output, then this cannot execute.

The "non strict" actors are key: They do not need to know all their inputs to determine the outputs.


## Non-Strict Logical Or



The non-strict or (often called the "parallel or") can produce a known output even if the input is not completely known. Here is a table showing the output as a function of two inputs:
input 1


## More Synchronous/Reactive Actors

Key SR Actors


Pre: When the input is present, the output is the previous present input value.

NonStrictDelay


NonStrictDelay: The output is equal to the input in the previous clock tick.


When: When the bottom input is present and true, the output equals the input. Otherwise, the output is absent.


Default: The output equals the left input, if it is present, and the bottom input otherwise.


Current: The output equals the most recent present input value.

EnabledComposite


EnabledComposite: Composite actor whose internal clock ticks only when the bottom input is present and true.

Use of some of these can be quite subtle.



## Subtleties: Pre vs. NonStrictDelay



Pre: When the input is present, the output is the previous present input value.

NonStrictDelay


NonStrictDelay: The output is equal to the input in the previous clock tick.

Pre: True one-sample delay. The behavior is not affected by insertion of an arbitrary number of ticks with "absent" inputs between present inputs.

NonStrictDelay: One-tick delay (vs. onesample). The output in each tick equals the input in the previous tick (whether absent or not).

## Illustration of this Subtlety

 is delayed by one click,

## Consequences: Pre vs. NonStrictDelay



Pre: When the input is present, the output is the previous present input value.

NonStrictDelay


NonStrictDelay: The output is equal to the input in the previous clock tick.

Pre: This actor is strict. It must know whether the input is present before it can determine the output. Hence, it cannot be used to break feedback loops.

NonStrictDelay: This actor is nonstrict. It need not know whether the input is present nor what its value is before it can determine the output. Hence, it can be used to break feedback loops.


## The Flat CPO

Consider a set of possible values $T=\left\{t_{1}, t_{2}, \ldots\right\}$. Let

$$
A=T \cup\{\perp, \varepsilon\}
$$

where $\perp$ represents "unknown" and $\varepsilon$ represents
"absent."

Let $(A, \leq)$ be a partial order where:

- $\quad \perp \leq \varepsilon$
- for all $t$ in $T, \perp \leq t$
- all other pairs are incomparable


## Hasse Diagram for the Flat CPO



Note that this is obviously a CPO
(all chains have a LUB)

All chains have length 2.

## Monotonic Functions on This CPO

In this CPO, any function $f: A \rightarrow A$ is monotonic if

$$
f(\perp)=a \neq \perp \Rightarrow f(b)=a \text { for all } b \in A
$$

I.e., if the function yields a "known" output when the input is unknown, then it will not change its mind about the output once the input becomes known.

Since all chains are finite, every monotonic function is continuous.

## Non-Strict Logical Or is Monotonic



The non-strict or is a monotonic function $f: A \times A \rightarrow A$ where $A=\{\perp, \varepsilon, \mathrm{T}, \mathrm{F}\}$ as can be verified from the truth table:

| input 1 |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\perp$ | $\varepsilon$ | F | T |
|  | $\perp$ | $\perp$ | $\perp$ | $\perp$ | T |
| $\stackrel{\square}{7}$ | $\varepsilon$ | $\perp$ | $\varepsilon$ | F | T |
| . | F | $\perp$ | F | F | T |
|  | T | T | T | T | T |

## Recall: Kleene Fixed Point Theorem

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

- If $\mathrm{v} C=f(\mathrm{v} C)$, then $\mathrm{v} C$ is the least fixed point of $f$
- If $f$ is continuous, then $\vee C=f(\vee C)$

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.

## Applying Kleene Fixed Point Theorem



At each tick of the clock

- Start with signal value $\perp$
- Evaluate $f(\perp)$
- Evaluate $f(f(\perp))$
- Stop when a fixed point is reached

Unlike PN, a fixed point is always reached in a finite number of steps (one, in this case).

## Causality Loops



What is the behavior in the following cases?

- $f$ is the identity function.
- $f$ is the logical NOT function.
- $f$ is the nonstrict delay function with initial value 0 .
- $f$ is the nonstrict delay function with no initial value.


## Causality Loops



What is the behavior in the following cases?

- $f$ is the identity function: $\perp$
- $f$ is the logical NOT function: $\perp$
- $f$ is the nonstrict delay function with initial value 0: 0
- $f$ is the nonstrict delay function with no initial value: $\varepsilon$


## Generalizing to Multiple Signals



- The Cartesian product of flat CPOs under pointwise ordering is also a CPO.
- All chains are still finite.
- Can now apply to any composition, as done with PN.


## Compositional Reasoning

So far, with both PN and SR, we deal with composite systems by reducing them to a monotonic function of all the signals.

An alternative approach is to convert an arbitrary composition to a continuous function.

## Example to Use for Compositional Reasoning

Consider an actor:


Assume $a \in A, b \in B, c \in C$, all CPOs.
Assume that the actor function $f: A \times B \rightarrow C$ is continuous
Consider the following composition:


We would like to consider this a function from $a$ to $c$.

## First Option: Currying

## (Named after Haskell Curry)

Given a function $f: A \times B \rightarrow C$, we can alternatively think of this in stages as $f_{1}: A \rightarrow[B \rightarrow C]$, where $[B \rightarrow C]$ is the set of all functions from $B$ to $C$.

For the following example, for each given value of $a$ we get a new function $f_{1}(a)$ for which we can find the least fixed point. That least fixed point is the value of $c$.


## Example: Non-Strict OR



Suppose $f$ is a non-strict logical OR function. Then:

- If $a=$ true, then the resulting function $f_{1}(a)$ always returns true, for all values of the input $b$.

In this case, the least fixed point yields $c=$ true.

- If $a=$ false, then the resulting function $f_{1}(a)$ is the identity function.

In this case, the least fixed point yields $c=\perp$.

## Second Option: Lifting (Named after Heavy Lifting)



Given a function $f: A \times B \rightarrow C$, we are looking for a function $g: A \rightarrow C$ such that

$$
c=g(a)
$$

In the model we have $b=c$ and $c=f(a, b)$ so

$$
g(a)=f(a, g(a))
$$

This looks like a fixed point problem, but the "unknown" on both sides is $g$, a function not a value. If we can find the function $g$ that satisfies this equation, then we can use it always to calculate $c$ given $a$.

## Posets of Functions

Suppose ( $A, \leq$ ) and ( $C, \leq$ ) are CPOs.
Consider functions $f, g \in[A \rightarrow C]$.
Define the pointwise order on these functions to be

$$
f \leq g \Leftrightarrow \forall a \in A, f(a) \leq g(a)
$$

Let $X \subset[A \rightarrow C]$ be the set of all continuous total functions from $A$ to $C$.

Theorem: $(X, \leq)$ is a CPO under the pointwise order.

Proof: See textbook.

## Least Function in the CPO of Functions

Let $X \subset[A \rightarrow C]$ be the set of all continuous total functions from $A$ to $C$. Since $X$ is a CPO, it must have a bottom. The bottom is a function $\perp_{X}: A \rightarrow C$ where for all $a \in A$,

$$
\perp_{X}(a)=\perp_{C} \in C
$$

## Consequence of this Theorem



Given a continuous function $f: A \times B \rightarrow C$, the function $g: A \rightarrow C$ such that

$$
c=g(a)
$$

is the least fixed point of a continuous function

$$
F: X \rightarrow X
$$

where $X \subset[A \rightarrow C]$ is the set of all continuous total functions from $A$ to $C$.

We need to now determine the continuous function $F$.

## Consequence of this Theorem (Continued)



We need to find a function that $g$ satisfies:

$$
g(a)=f(a, g(a))
$$

Let $X \subset[A \rightarrow C]$ be the set of all continuous total functions from $A$ to $C$ and let $F$ be a continuous function $F: X \rightarrow X$.

Then $g \in X$ is the least function such that $F(g)=g$ where for all $a \in A$,

$$
(F(g))(a)=f(a, g(a))
$$

The theorem, with the Kleene fixed point theorem, tells us that $F$ has a least fixed point, and tells us how to find it.

## Example: Non-Strict OR



Suppose $f$ is a non-strict logical OR function. Then:

$$
(F(g))(a)= \begin{cases}\text { true } & \text { if } a=\text { true } \\ \perp & \text { if } a=\perp \text { and } g(a)=\text { false } \\ g(a) & \text { otherwise }\end{cases}
$$

The least fixed point of this is the function $g$ given by:

$$
g(a)=\left\{\begin{array}{cc}
\text { true } & \text { if } a=\text { true } \\
\perp & \text { otherwise }
\end{array}\right.
$$

To find this, start with $F(\perp)$, then find $F(F(\perp)$ ), etc., until you get a fixed point (which happens immediately).

## Showing that $F$ is Continuous

Need to show that given a chain of continuous total functions $C=\left\{g_{1}, g_{2} \ldots\right\}$ that:

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

For all $a \in A$ :

$$
\begin{aligned}
(F(\vee C))(a) & =f(a,(\vee C)(a)) & & \\
& =f\left(a, \mathrm{v}\left\{g_{1}(a), g_{2}(a), \ldots\right\}\right) & & \begin{array}{l}
\text { because each } g_{i} \text { is } \\
\text { continuous }
\end{array} \\
& =\vee \hat{f}\left(a,\left\{g_{1}(a), g_{2}(a), \ldots\right\}\right) & & \text { because } f \text { is continuous } \\
& =(\vee \hat{F}(C))(a) & & \text { QED }
\end{aligned}
$$

## Summary

- In SR, fixed point semantics is simpler than in PN because the CPO has only finite chains.
- The fancier techniques of Currying and Lifting can be applied equal well to PN, but we introduce them here because the simpler CPO makes them easier to understand.
- The fixed point semantics of SR talks only about the behavior at a tick of the clock. The behavior across ticks of the clock will require a clock calculus.

