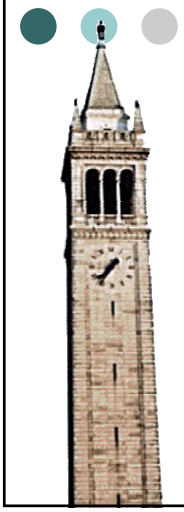
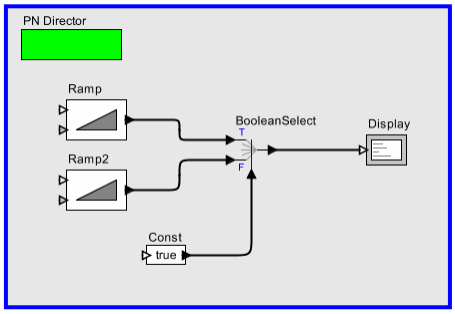


Concurrent Models of Computation for Embedded Software

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

Lecture 9: Convergence and Introduction to Synchronous Models

The Convergence Question



- *Correct execution:* after any finite time every signal is a prefix of the LUB signal given by the semantics.
- *Useful execution:* a correct execution that:
  1. Does not stop if at least one signal has not reach the LUB.
  2. Executes with bounded buffers if this is possible.

The Question: Does this execution "converge" to the LUB?

Convergence in the Reals

Consider a sequence of real numbers:

$$s : N \to \Re$$

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

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

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

Scott Topology

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

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

$$N_t = \{ t' \in T \mid t' \sqsubseteq t \}$$

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:

$$s(m) \in A$$

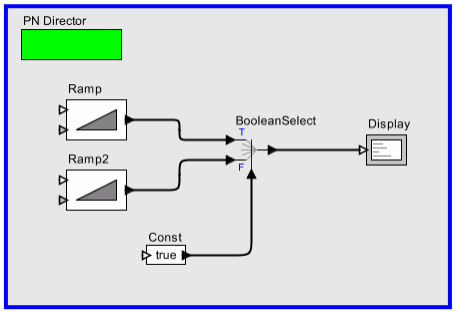
Intuition: For any finite prefix $p \sqsubseteq a$, the sequences in s eventually all have prefix p .

Consequences for Process Networks

- "Correct" executions of process networks do not necessarily converge to the LUB semantics.
- This is because "correct" executions allow any signal to be evaluated only to a finite prefix of the LUB semantics.
- But if leaving the execution at a finite prefix were "incorrect," then it would be incorrect for Ptolemy II to stop the execution when you push the stop button.

  This would be counterintuitive.

Convergent Execution vs. Correct Execution



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

Which do you prefer?

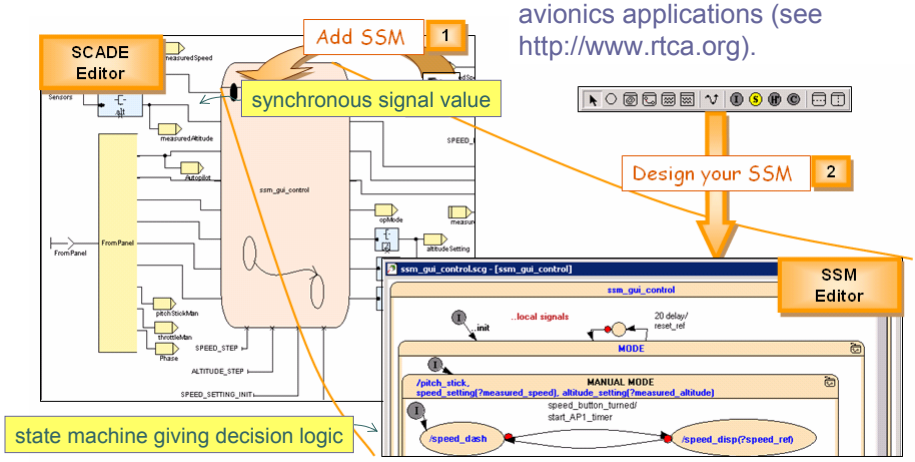
Synchronous Languages

- Esterel
- Lustre
- SCADE (visual editor for Lustre)
- 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).

Lustre/SCADE

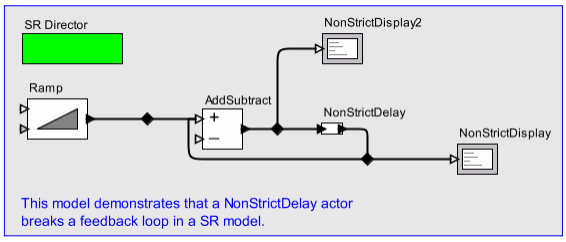
The SCADE tool has a code generator that produces C or ADA code that is compliant with the DO-178B Level A standard, which allows it to be used in critical avionics applications (see http://www.rtca.org).

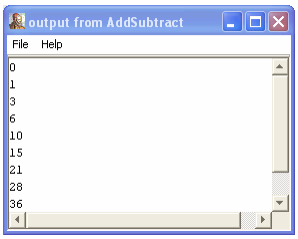


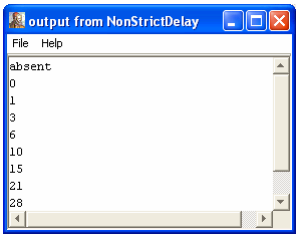
from http://www.esterel-technologies.com/

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.

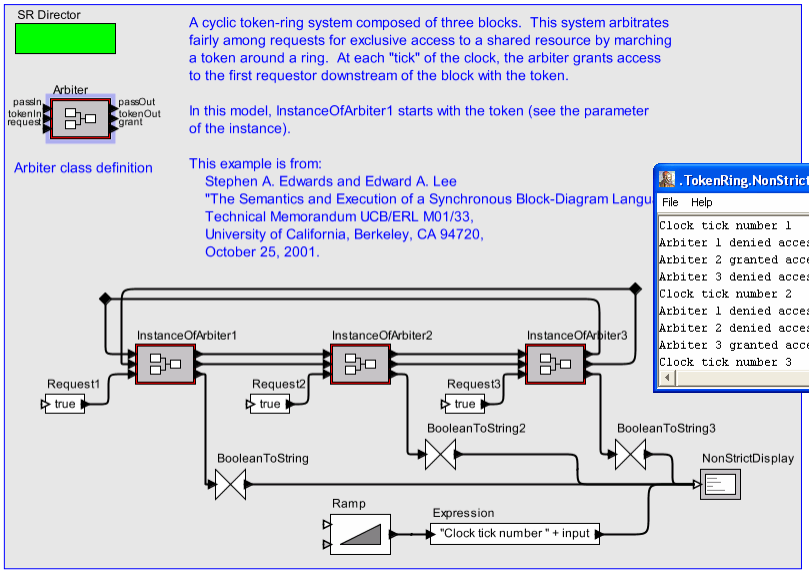
The Synchronous Abstraction

- "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).
- 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 real-time tasks.

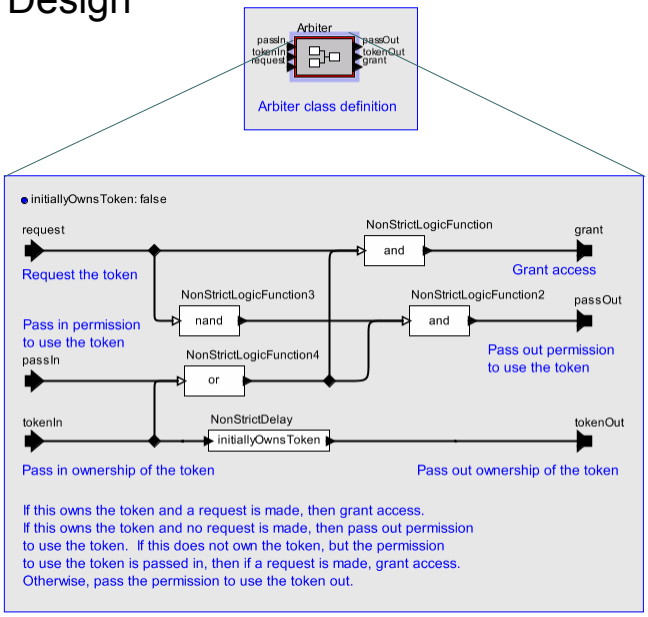
Properties

- 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



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.

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.

Conclusion and Open Issues

- "Correct" and "useful" executions of process networks do not necessarily converge to the denotational semantics of the model.
- But insisting on convergence may cause an execution to fail on a finite memory machine that could have executed forever.
- Synchronous/Reactive languages are promising alternatives where termination and boundedness are decidable.