# The SR Domain

Stephen Edwards
Edward A. Lee

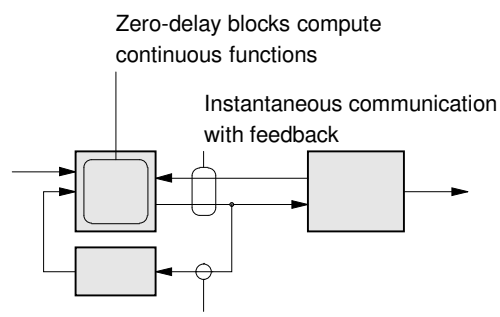
http://www.eecs.berkeley.edu/~sedwards/

University of California, Berkeley

## The SR Domain

- A specification scheme
  - Synchronous model of time
    - Predictable temporal behavior
    - Easier to design
    - Easier to analyze
  - Heterogeneous: compiler cannot see inside blocks
    - Mixing languages made easy
    - Allows separate compilation
    - Large designs are tractable
- Deterministic
  - Guaranteed by fixed-point semantics
- Fast, predictable execution time
  - Chaotic iteration-based scheme
  - Fully static scheduling

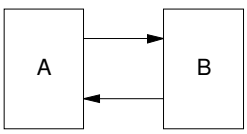
## SR Systems



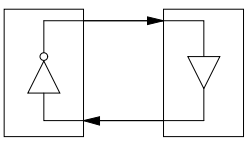

- Block functions may change between instants for time-varying behavior
- Block functions may be specified in any language

## Zero Delay and Feedback

How to maintain determinism?




**Which goes first?**
*Need an order-invariant semantics*



**Contradictory!**
*Need to attach meaning to such systems.*

## Dealing with Feedback

Why bother at all?
Answer: *Heterogeneity*

- Cycles are usually broken by delay elements *at the lowest level*
- Some schemes (e.g., Lustre) insist on this
- False feedback often appears at higher levels
- Data dependent cycles can appear when sharing resources
- *Virtually all cycles are "false," yet must be dealt with.*

## Fixed-point Semantics are Natural for Synchronous Specifications with Feedback

Why a fixed point?

Self-reference:
The essence of a cycle

$$f(x_t) = x_t$$

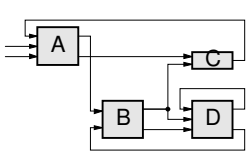
System function (composition of block functions)

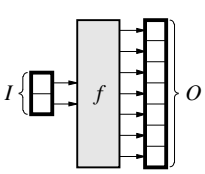
Vector of signals at time $t$ (zero delay)

fixed point $\Longleftrightarrow$ stable state

determinism $\Longleftrightarrow$ unique solution

## The Least Fixed Point of What?

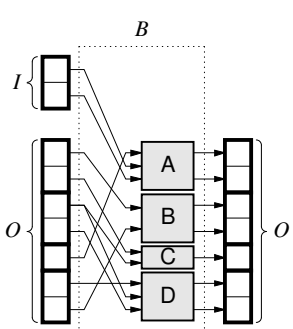



Interpret as ↘

↗ Take LFP

$B(I, f(I)) = f(I)$




## Unique Least Fixed Point Theorem

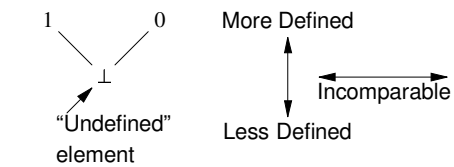
Recall:

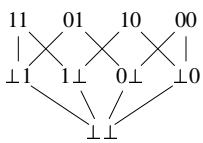
> A monotonic function on a complete partial order (with $\perp$) has a unique least fixed point.

*What does it mean to make the system function $f$ monotonic and the signal values a CPO?*

## Vector of Signals is a CPO

Values along an upward path grow more defined.



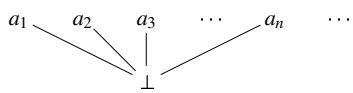




vector-valued extension

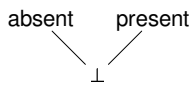
Formally, $x \sqsubseteq y$ if $y$ is at least as defined as $x$.

## Adding ⊥ Is Enough

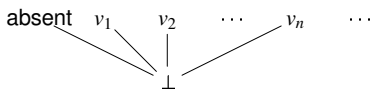
Any set $\{a_1, a_2, \ldots, a_n, \ldots\}$ can easily be "lifted" to give a flat partial order:




A CPO for signals with pure events:




A CPO for valued events:



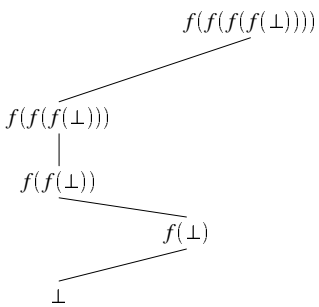

Why not absent ⊑ present?

```
present A then ... else ... end
```

Violates monotonicity

## Monotonic Block Functions

Giving a more defined input to a monotonic function always gives a more defined output.




Formally, $x \sqsubseteq y$ implies $f(x) \sqsubseteq f(y)$.

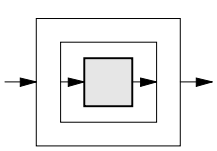
A monotonic function never recants ("changes its mind").

## Many Languages Use Strict Functions, Which Are Monotonic

A strict function:

$$g(\underbrace{\ldots, \bot, \ldots}_{\text{inputs}}) = (\underbrace{\bot, \ldots, \bot}_{\text{outputs}})$$

**Outside:** A strict monotonic function



**Inside:** Simple "function call" semantics

Most common imperative languages only compute strict functions.

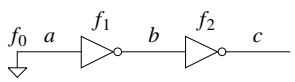
**Danger:** *Cycles of strict functions deadlock—fixed point is all ⊥—need some non-strict functions.*

## A Simple Way to Find the Least Fixed Point

$$\bot \sqsubseteq f(\bot) \sqsubseteq f(f(\bot)) \sqsubseteq \cdots \sqsubseteq \text{LFP} = \text{LFP} = \cdots$$

For each instant,

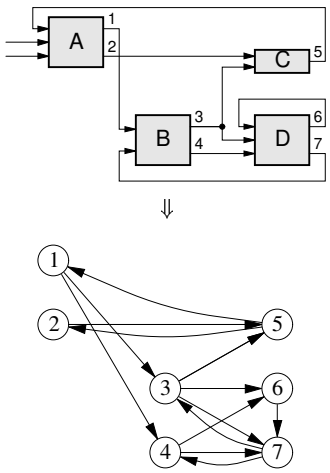
1. Start with all signals at $\bot$
2. Evaluate all blocks (in some order)
3. If any change their outputs, repeat Step 2




$$\begin{aligned}
(a,b,c) &= (\bot,\bot,\bot) \\
f_0(\bot,\bot,\bot) &= (0,\bot,\bot) \\
f_1(0,\bot,\bot) &= (0,1,\bot) \\
f_2(0,1,\bot) &= (0,1,0) \\
f_2(f_1(f_0(0,1,0))) &= (0,1,0)
\end{aligned}$$

## The Dependency Graph

Transform into single-output functions:

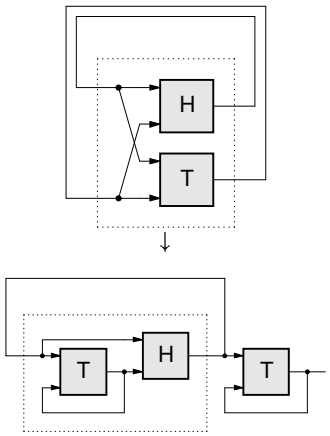



## The Scheduling Algorithm

1. Decompose into strongly-connected components
2. Remove a head (set of vertices) from each SCC, leaving a tail
3. Recurse on each tail

## Evaluating SCCs

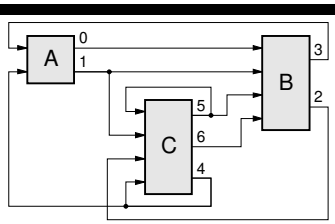
Split a strongly-connected graph into a head and tail:



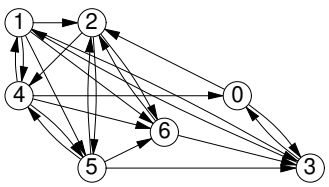

Good heads break T's strong connectivity.

## Example

System

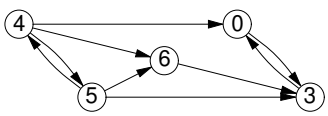



Graph

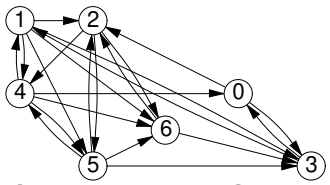


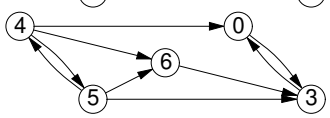

Head 1 2

Tail




## Schedules



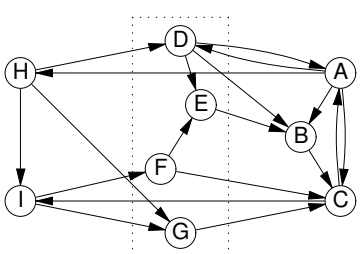

( 1 2 . ( 4 . 5 ) 6 ( 0 . 3 ) )

head tail; head tail head tail; SCC SCC

5 4 5 6 3 0 3 1 2 5 4 5 6 3 0 3 1 2 5 4 5 6 3 0 3

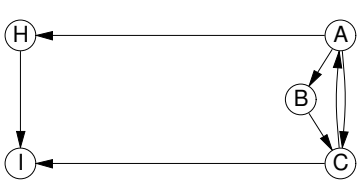
## Finding Good Heads

Must break strong connectivity—remove a border of a set of vertices:



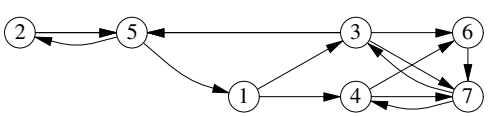

border of { A, B, C }

(vertices with incoming edges)




## Choosing Good Border Sets

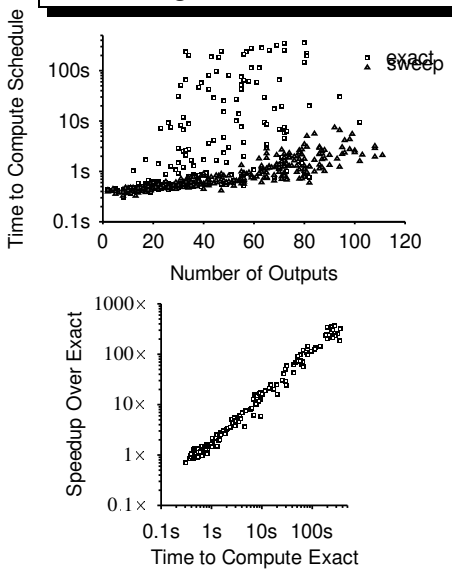
Heuristic: “Grow” a set starting from a vertex and greedily include the best border vertex:



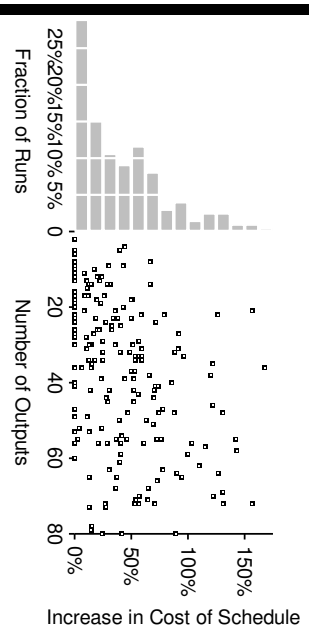

| Set | Border |
|---|---|
| 1 | 5 |
| 1 5 | 2 3 |
| 1 5 2 | 3 |
| 1 5 2 3 | 7 |
| 1 5 2 3 7 | 4 6 |
| 1 5 2 3 7 4 | 6 |

2 is better (3 would increase border)

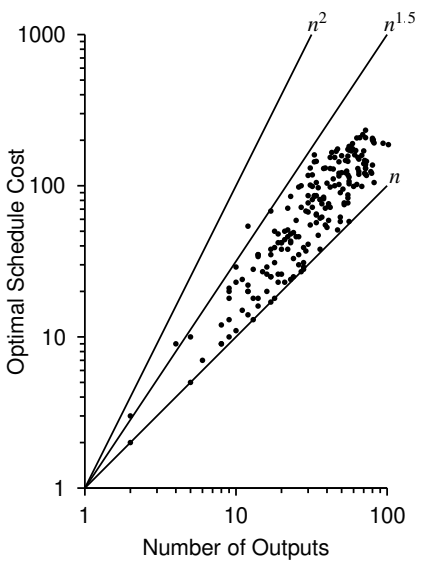
## Scheduling Results




## The Cost of Using the Heuristic




## Asymptotic Schedule Cost




## Conclusions

- Deterministic specification scheme combining synchrony and heterogeneity
- Semantics: the least fixed point of a continuous function on a CPO
- Iterative execution scheme based on recursive divide-and-conquer
- Exact scheduling practical for small graphs
- Heuristic practical for very large graphs
- Execution time for random graphs growing slower than $n^{1.5}$