Cycle-based Programming of Distributed Systems: The Synchrony Hypothesis

Michael Mendler

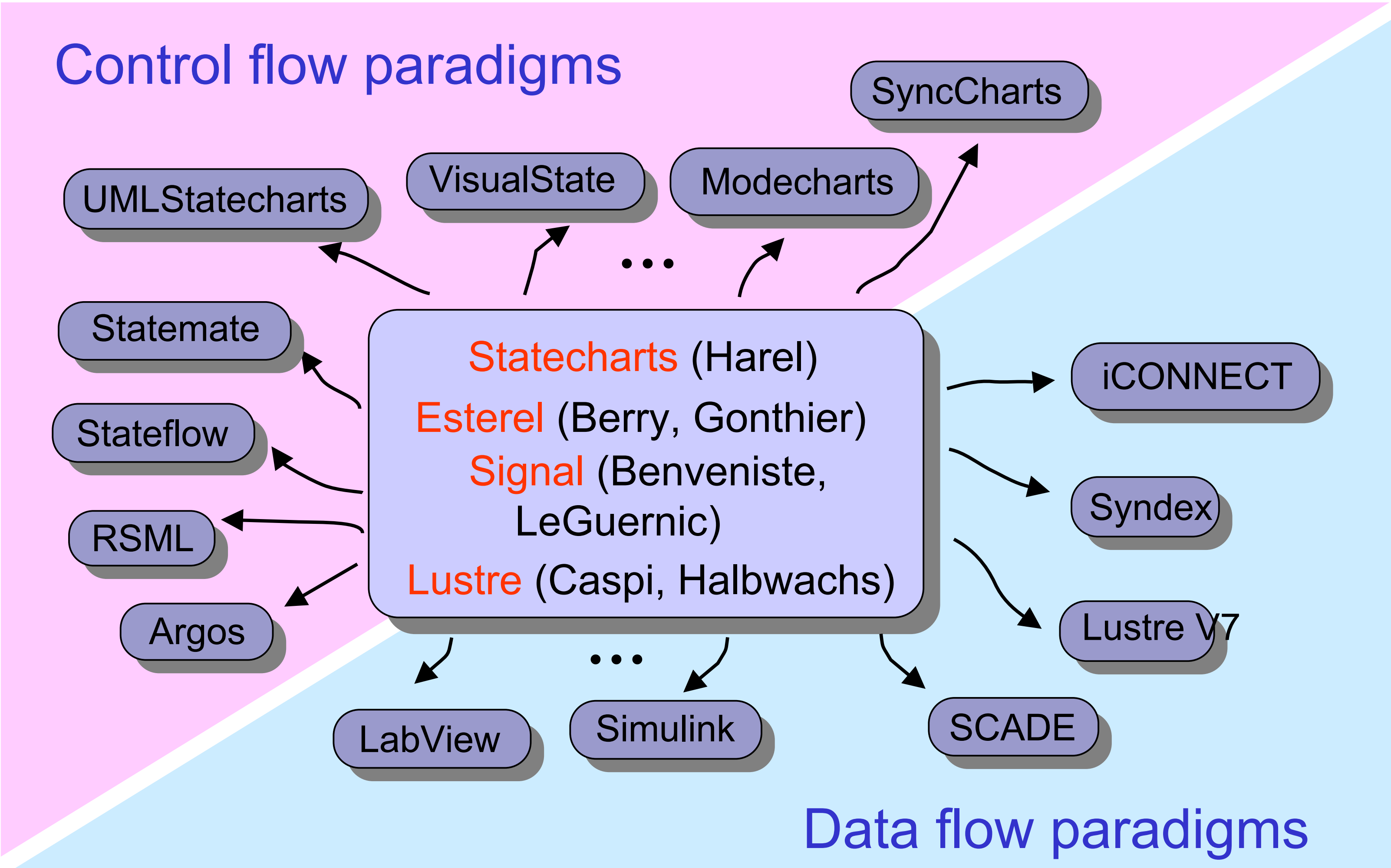
Faculty of Information Systems and Applied Computer Sciences
University of Bamberg, Germany

Overview

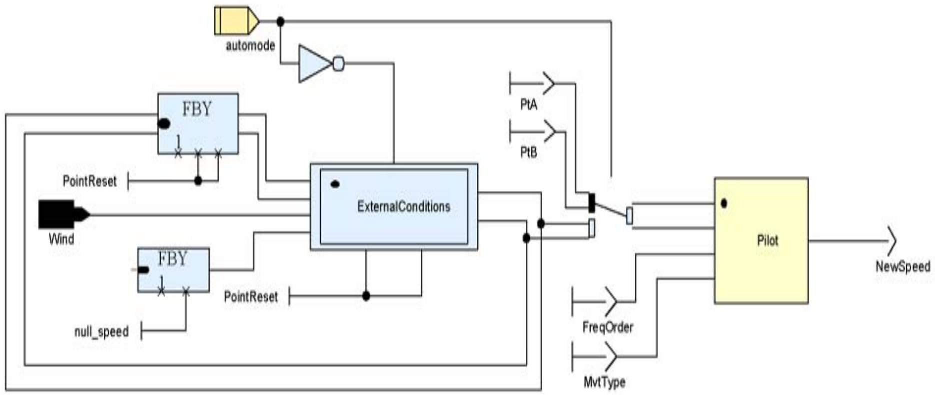
1. Synchronous Programming
2. The Synchrony Hypothesis
3. Causal Reaction = Fixed Point ?
4. What‘s in a Step ?: Notions of Causality
5. The Synchrony Hypothesis (Hypo-)Thesis

1. Synchronous Programming

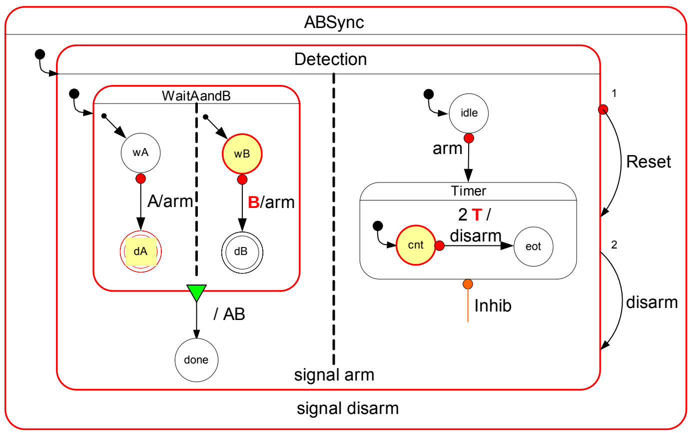
Synchronous Programming



Example: SCADE – Esterel Tech



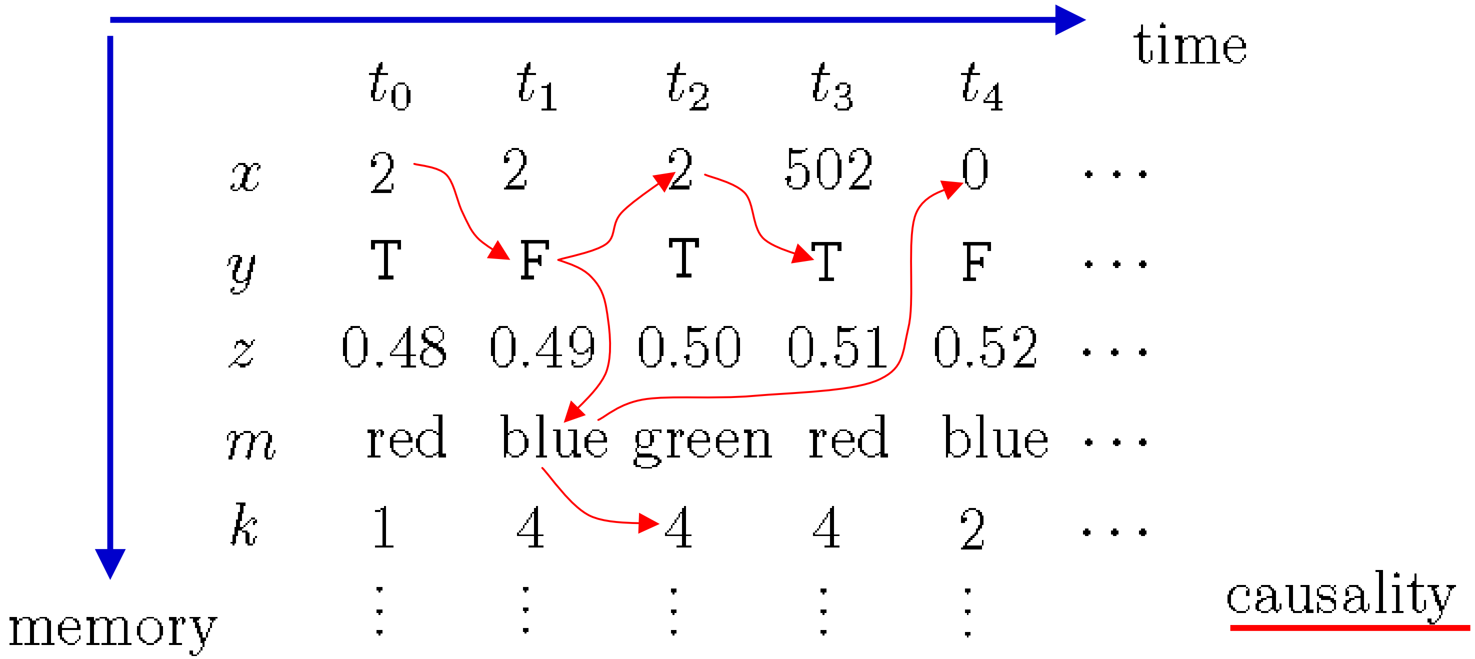
Data Flow: SCADE Lustre



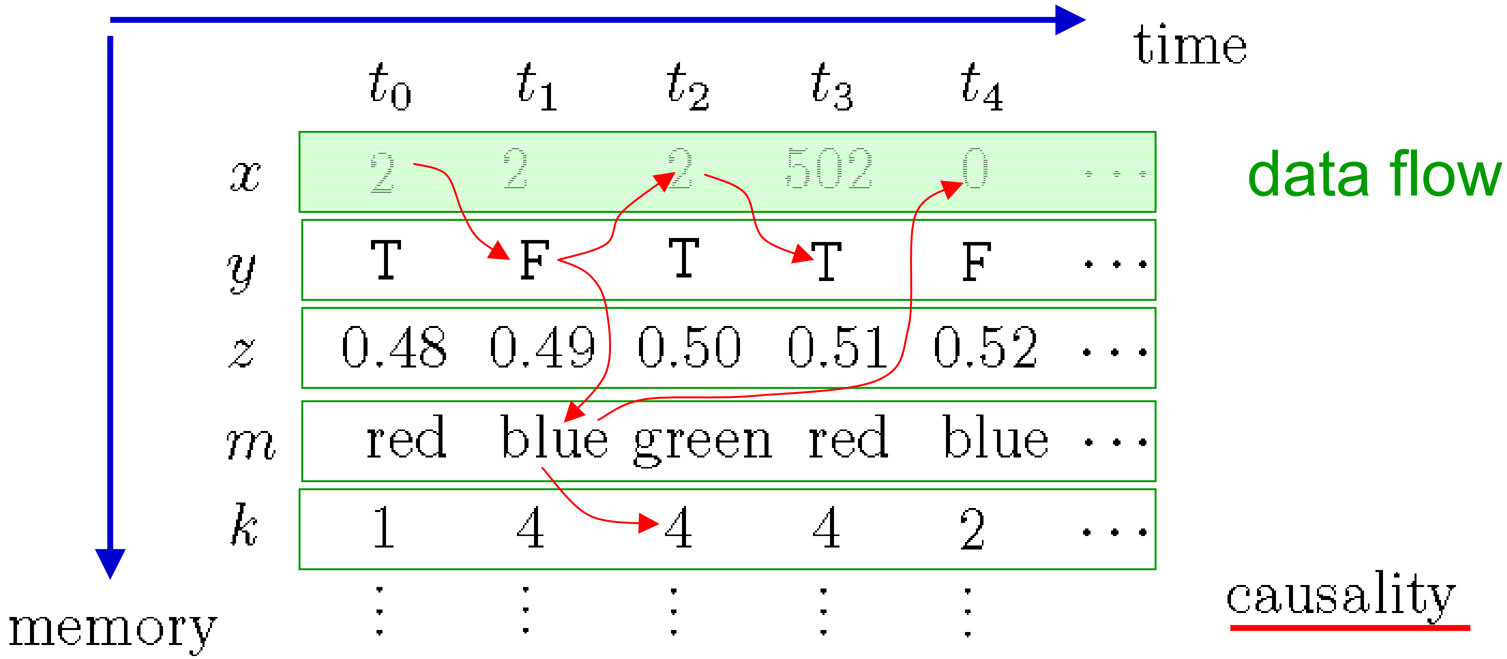
Control Flow: SCADE Safe State Machines

- embedded systems domain (avionics, automotive)
- rigorous semantics
- verification & testing (certification)
- code-generation
- hw/sw codesign

Orthogonality in Time and Space

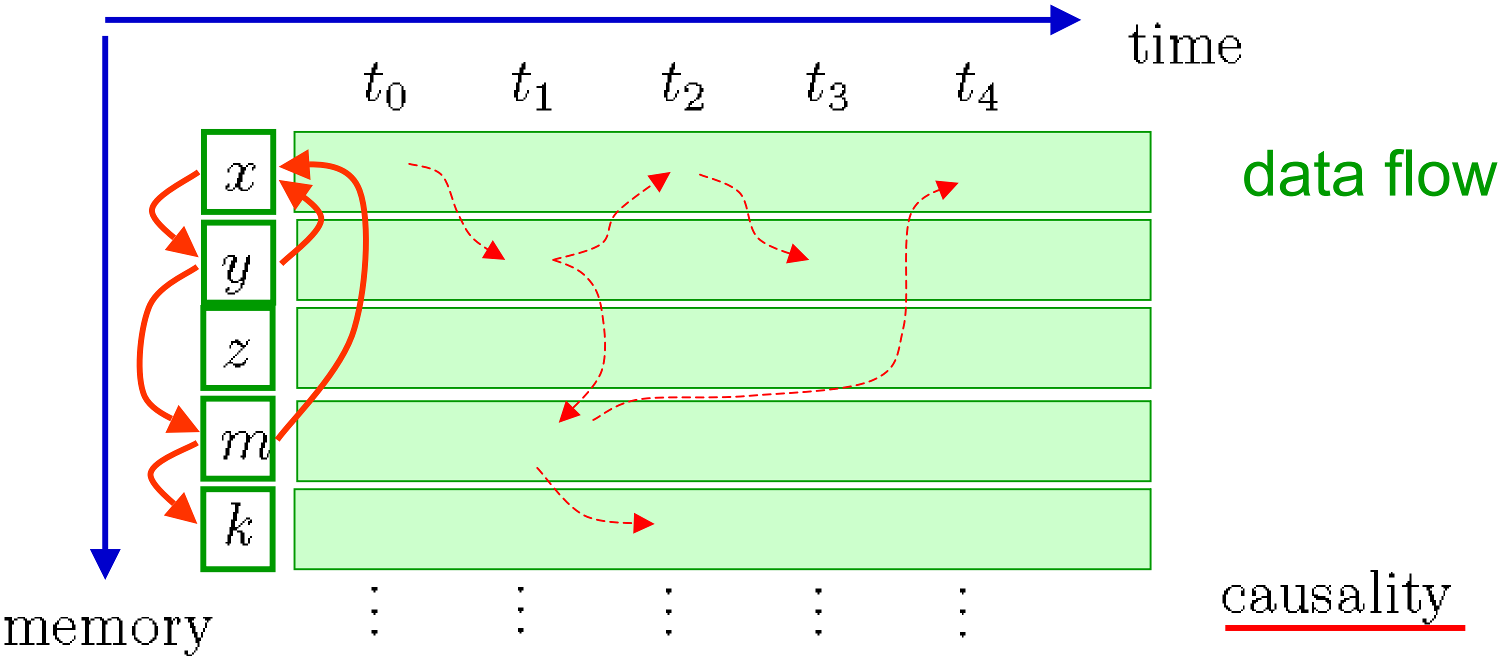


Data Flow


time
t_0 t_1 t_2 t_3 t_4
x 2 2 2 502 0 ...
y T F T T F ...
z 0.48 0.49 0.50 0.51 0.52 ...
m red blue green red blue ...
k 1 4 4 4 2 ...
memory
data flow
causality

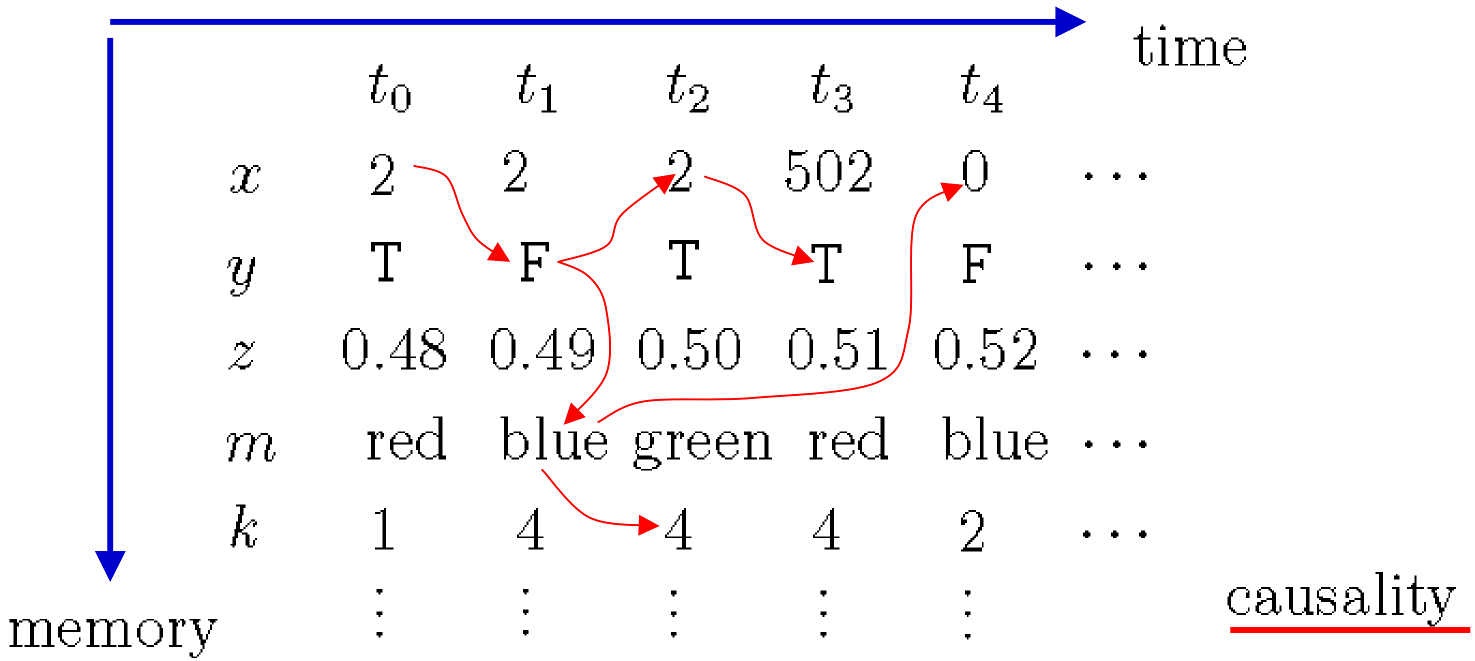
Data Flow

Q: How do we treat the cyclic DF dependencies ?

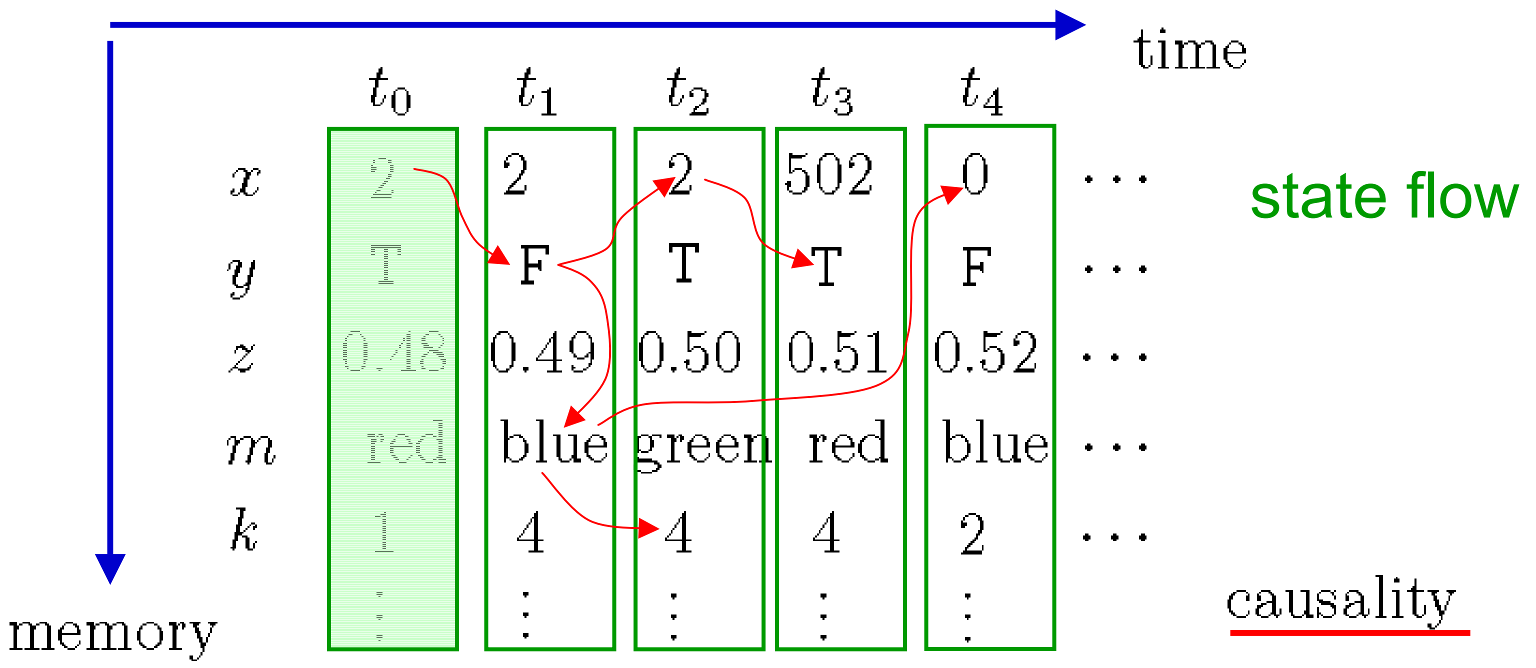


A: Continuitiy Hypothesis, Kahn stream semantics !

Orthogonality in Time and Space



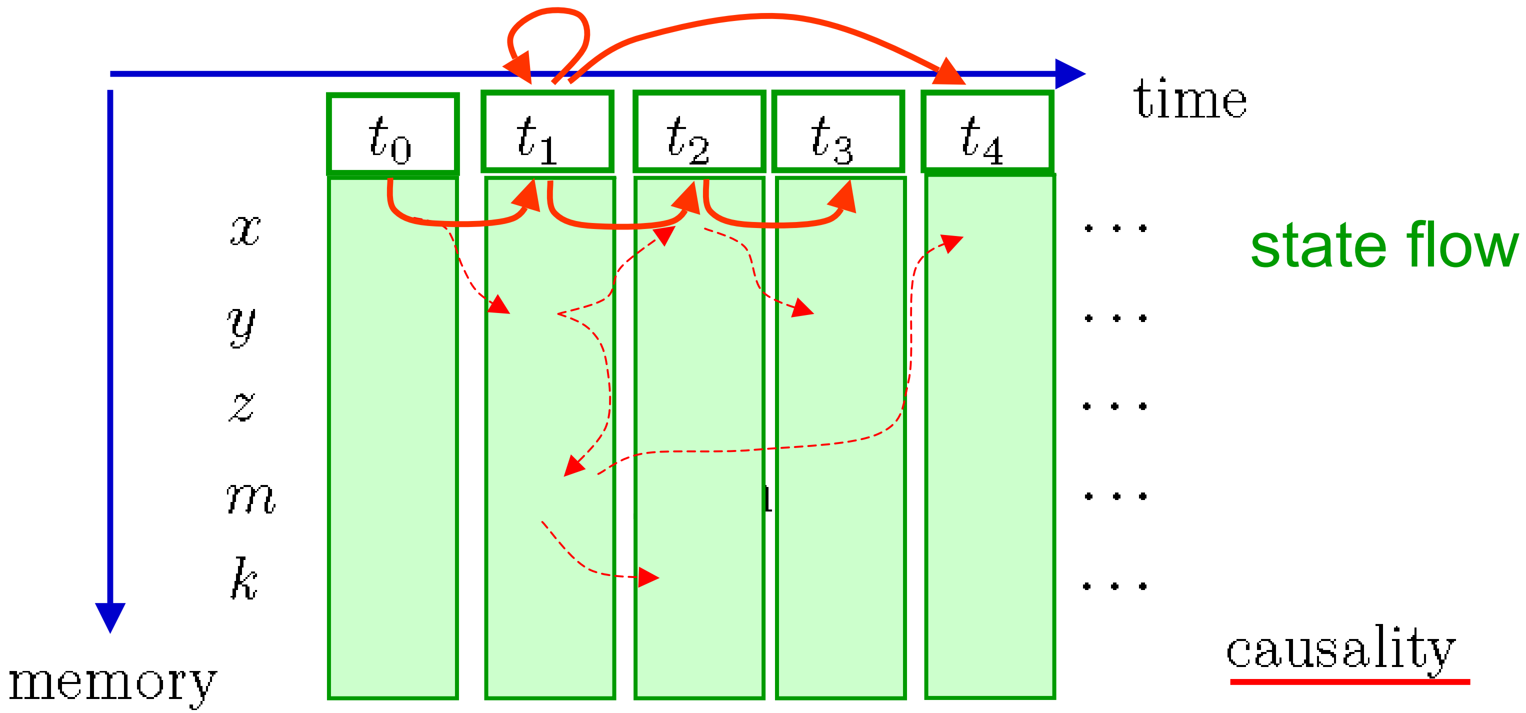
State Flow



time
memory
t_0 t_1 t_2 t_3 t_4
x 2 2 2 502 0 ...
y T F T T F ...
z 0.48 0.49 0.50 0.51 0.52 ...
m red blue green red blue ...
k 1 4 4 4 2 ...
state flow
causality

State Flow

Q: How do we treat the cyclic SF dependencies ?

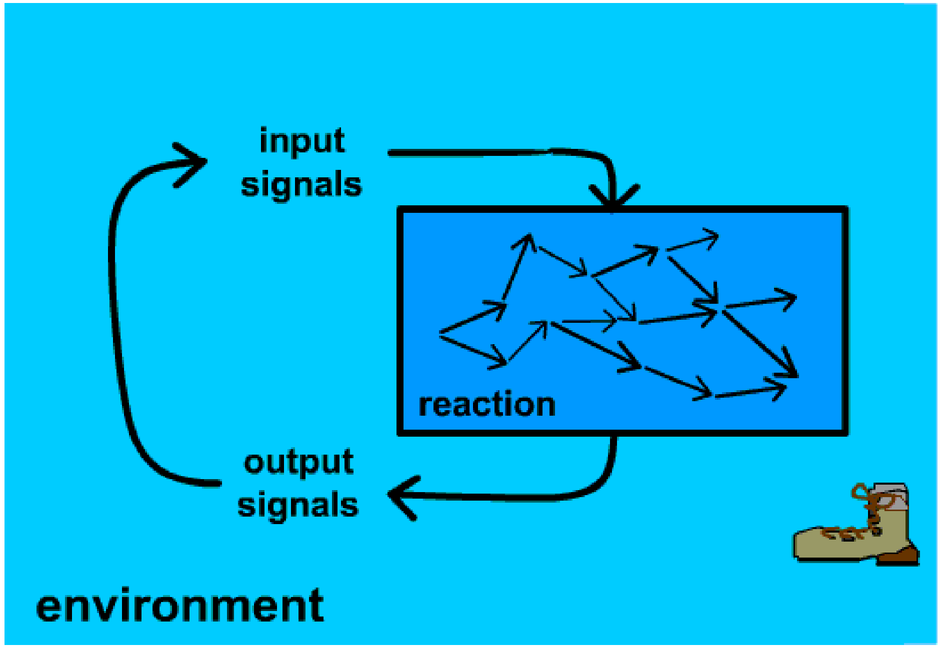


A: Synchrony Hypothesis, Fourman response semantics

2. The Synchrony Hypothesis

Synchrony Hypothesis

Environment view:
Reactions are
- atomic
- deterministic
- bounded



System view:
Reactions may be
- non-atomic
- non-deterministic
- unbounded

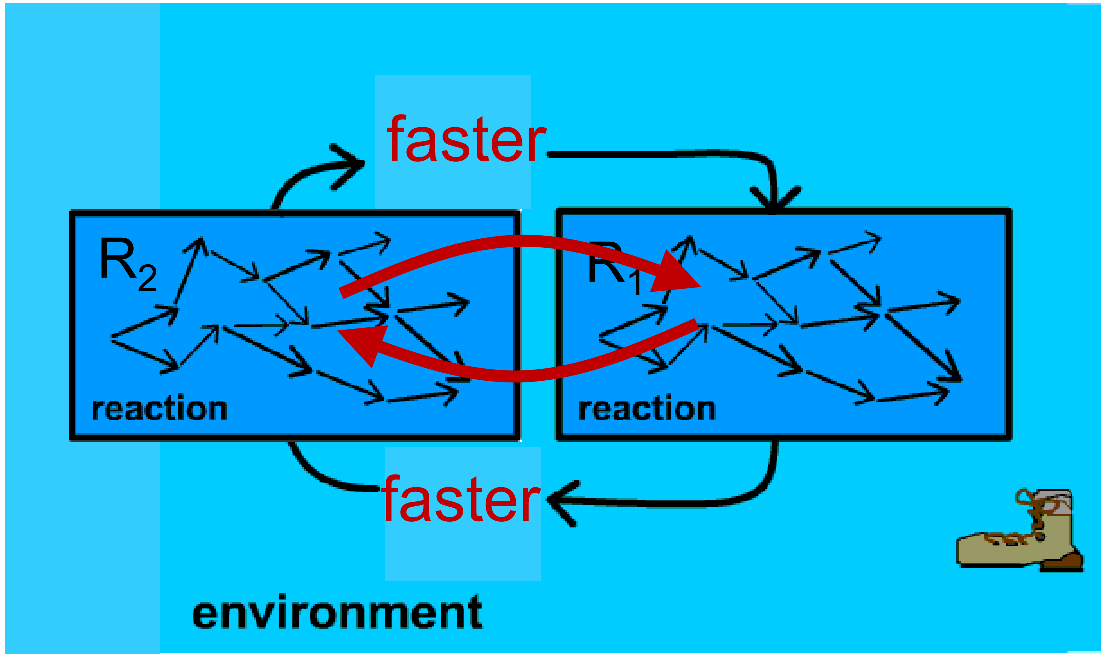
"A reactive system is faster than its environment, hence reactions can be considered atomic"

The Synchrony Paradox

Environment view:
Reactions are
- atomic
- deterministic
- bounded

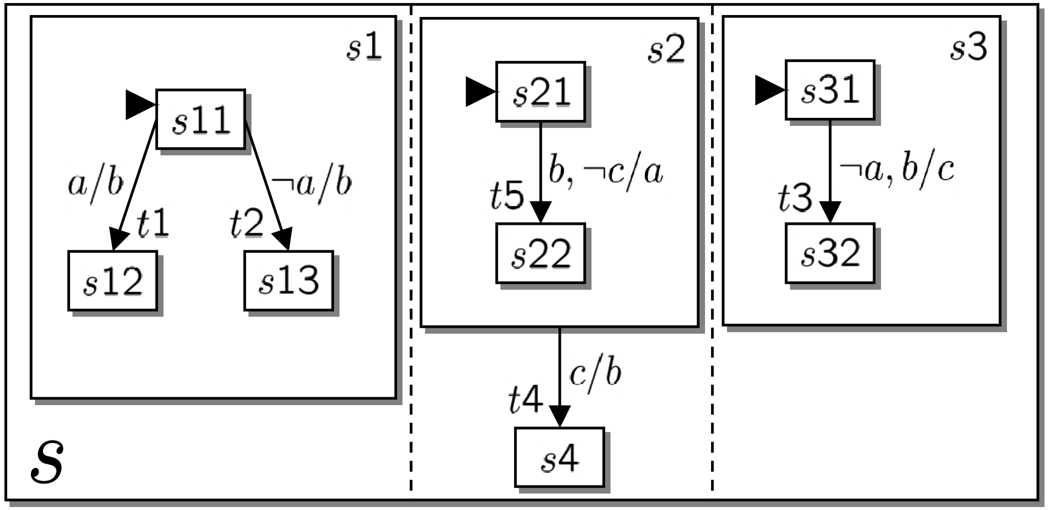
↕ ***Paradox ?***

System view:
Reactions may be
- non-atomic
- non-deterministic
- unbounded



"A reactive system is faster than its environment, hence reactions can be considered atomic"

Programming Synchronous Reactions



- logical transitions
- conjunctions = parallelism
- negations code choices, priorities and hierarchy

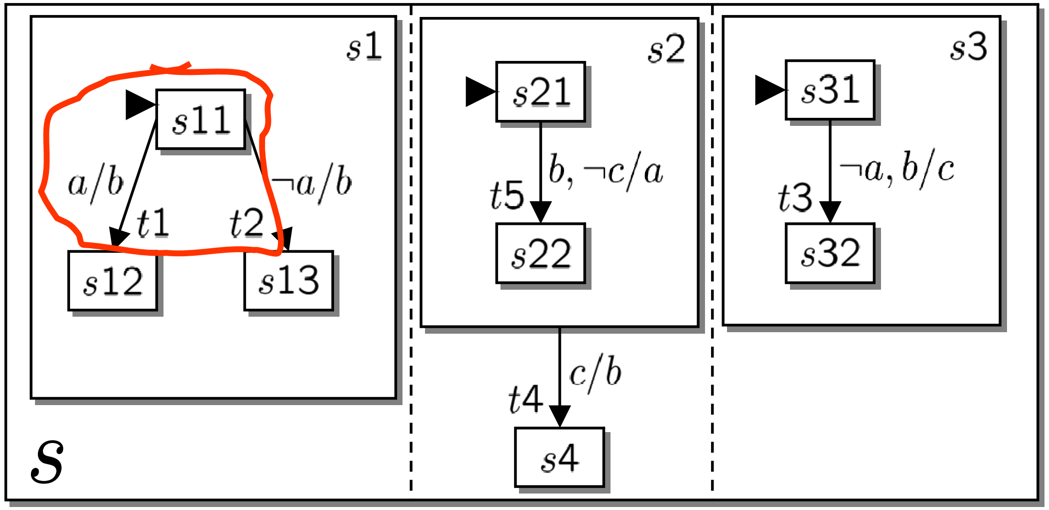
$$
\begin{aligned}
&REACT := \\
&\quad t1 \supset b \wedge t2 \supset b \wedge t3 \supset c \wedge t4 \supset b \wedge t5 \supset a \wedge \\
&\quad (s11 \wedge a \wedge \neg t2) \supset t1 \wedge \\
&\quad (s11 \wedge \neg a \wedge \neg t1) \supset t2 \wedge \\
&\quad (s31 \wedge \neg a \wedge b) \supset t3 \wedge \\
&\quad (s2 \wedge c) \supset t4 \wedge \\
&\quad (s21 \wedge b \wedge \neg c \wedge \neg t4) \supset t5
\end{aligned}
$$

parallelism

logical transitions

choice, priority

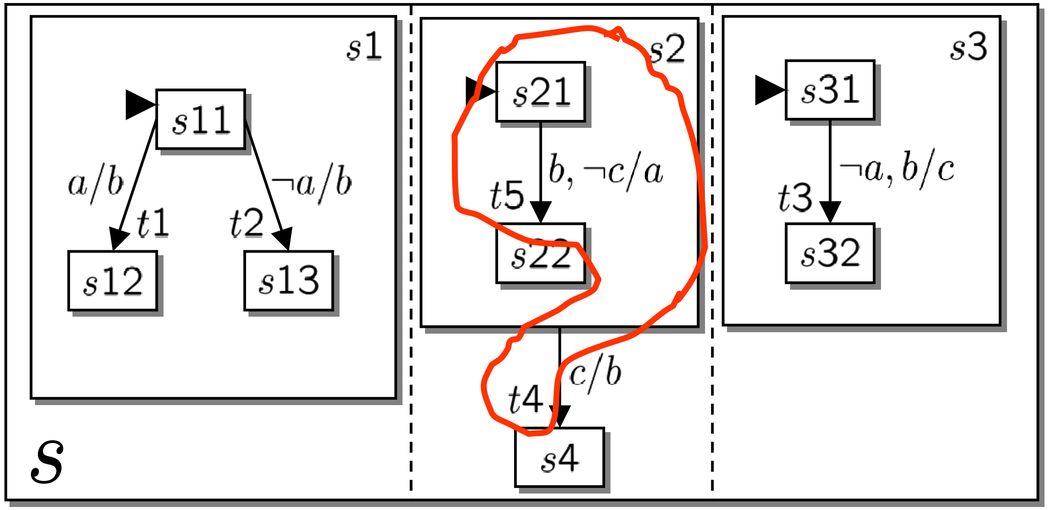
Programming Synchronous Reactions



- logical transitions
- conjunctions = parallelism
- negations code choices, priorities and hierarchy

$$
\begin{aligned}
&REACT := \\
&\quad t1 \supset b \wedge t2 \supset b \wedge t3 \supset c \wedge t4 \supset b \wedge t5 \supset a \wedge \\
&\quad (s11 \wedge a \wedge \neg t2) \supset t1 \wedge \\
&\quad (s11 \wedge \neg a \wedge \neg t1) \supset t2 \wedge \\
&\quad (s31 \wedge \neg a \wedge b) \supset t3 \wedge \\
&\quad (s2 \wedge c) \supset t4 \wedge \\
&\quad (s21 \wedge b \wedge \neg c \wedge \neg t4) \supset t5
\end{aligned}
$$

Programming Synchronous Reactions



- logical transitions
- conjunctions = parallelism
- negations code choices, priorities and hierarchy

$$
\begin{aligned}
REACT := \\
& t1 \supset b \wedge t2 \supset b \wedge t3 \supset c \wedge t4 \supset b \wedge t5 \supset a \wedge \\
& (s11 \wedge a \wedge \neg t2) \supset t1 \wedge \\
& (s11 \wedge \neg a \wedge \neg t1) \supset t2 \wedge \\
& (s31 \wedge \neg a \wedge b) \supset t3 \wedge \\
& (s2 \wedge c) \supset t4 \wedge \\
& (s21 \wedge b \wedge \neg c \wedge \neg t4) \supset t5
\end{aligned}
$$

Synchronous Abstraction

In which sense does **REACT** describe an atomic macro step ?

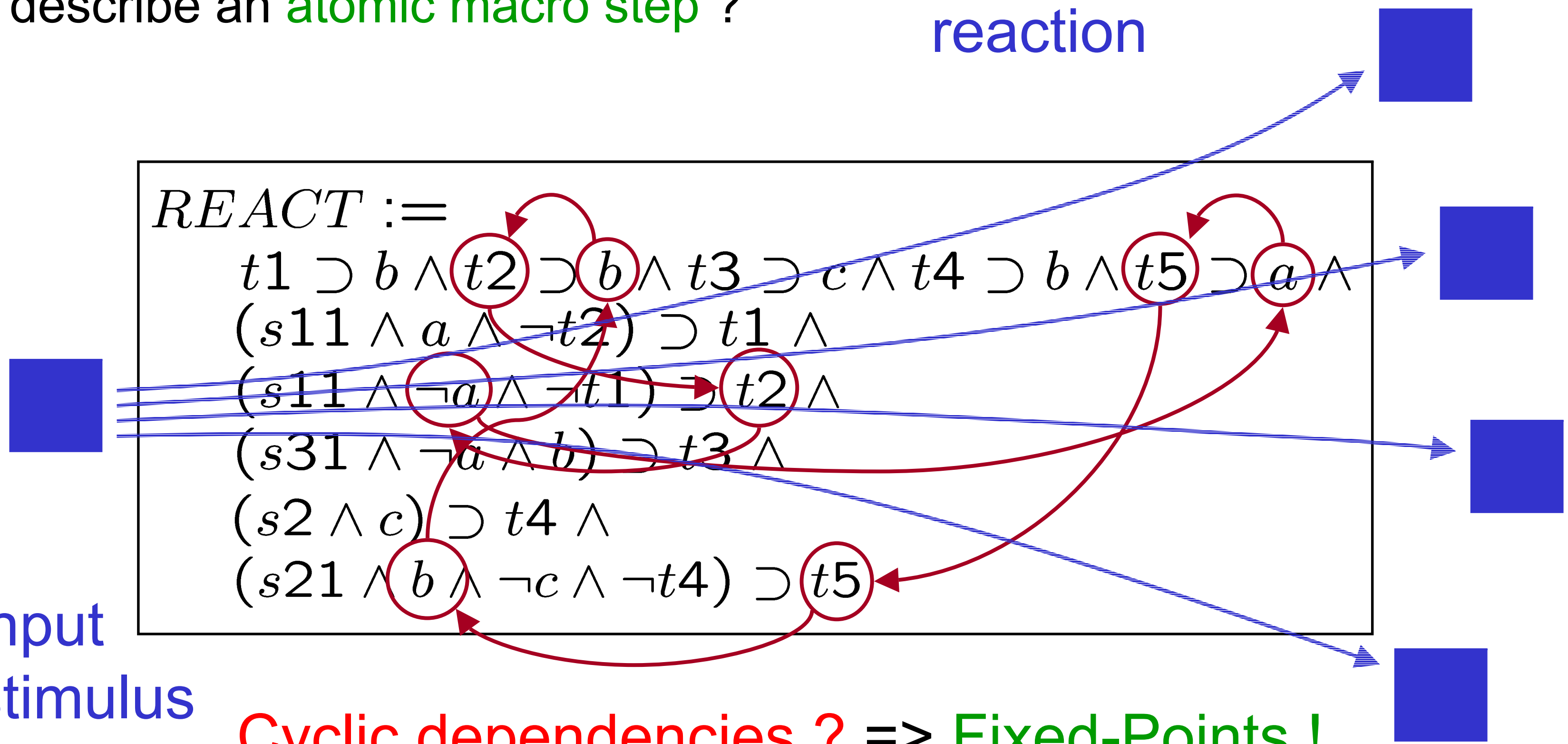
instantaneous reaction

$$
\begin{aligned}
&REACT := \\
&\quad t1 \supset b \wedge t2 \supset b \wedge t3 \supset c \wedge t4 \supset b \wedge t5 \supset a \wedge \\
&\quad (s11 \wedge a \wedge \neg t2) \supset t1 \wedge \\
&\quad (s11 \wedge \neg a \wedge \neg t1) \supset t2 \wedge \\
&\quad (s31 \wedge \neg a \wedge b) \supset t3 \wedge \\
&\quad (s2 \wedge c) \supset t4 \wedge \\
&\quad (s21 \wedge b \wedge \neg c \wedge \neg t4) \supset t5
\end{aligned}
$$

input stimulus

Synchronous Abstraction

In which sense does **REACT** describe an atomic macro step ?



$$
\begin{aligned}
REACT :=\;& t1 \supset b \wedge t2 \supset b \wedge t3 \supset c \wedge t4 \supset b \wedge t5 \supset a \wedge \\
& (s11 \wedge a \wedge \neg t2) \supset t1 \wedge \\
& (s11 \wedge \neg a \wedge \neg t1) \supset t2 \wedge \\
& (s31 \wedge \neg a \wedge b) \supset t3 \wedge \\
& (s2 \wedge c) \supset t4 \wedge \\
& (s21 \wedge b \wedge \neg c \wedge \neg t4) \supset t5
\end{aligned}
$$

Cyclic dependencies ? => Fixed-Points !

3. Causal Reaction = Fixed-Point ?

Synchronous Reactive Component

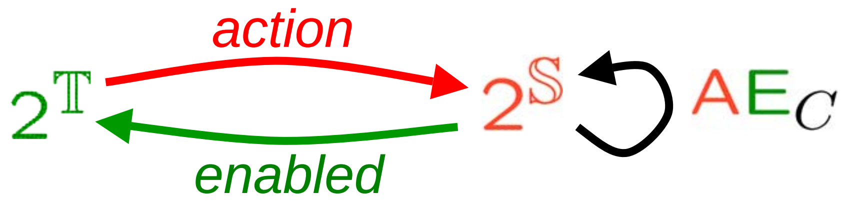
Reactive component

$$C = (\mathbb{S}, \mathbb{T}, pos, neg, act)$$

$\mathbb{S}$,$\mathbb{T}$ atomic logical signals, logical transitions

$pos, neg, act : \mathbb{T} \to 2^{\mathbb{S}}$ positive, negative triggers, actions

Response of C



$$action(T) = \{s \mid \exists t \in T.\, s \in act(t)\} \qquad T \subseteq \mathbb{T}$$

$$enabled(S) = \{t \mid pos(t) \subseteq S \wedge neg(t) \subseteq \overline{S}\} \qquad S \subseteq \mathbb{S}$$

$$\mathsf{AE}_C(S) = action(enabled(S))$$ „response function“

Reaction = Fixed-Point ?

Logical Coherence [Berry]

"A signal s is present in an instant if and only if an `emit s' statement is executed in this instant."

Logical Coherence & Reactiveness

- A response *S* is logically coherent iff *S* is a fixed-point of AE_C, i.e., $S = AE_C(S)$.
- C is logically reactive iff it in every activation state and environment, AE_C has a fixed-point.

Causal Response = Unique Fixed Point ?

Problem

The response function

$$\mathsf{AE}_C(S) = action(\{t \mid pos(t) \subseteq S \wedge neg(t) \subseteq \overline{S}\})$$

is not monotonic !

Causal Response = Unique Fixed Point ?

Problem

The response function

covariant → S ; *contravariant* → $\overline{S}$

$$\mathsf{AE}_C(S) = action(\{t \mid pos(t) \subseteq S \wedge neg(t) \subseteq \overline{S}\})$$

is not monotonic !

- no unique (least) fixed points !
- compositionality and full-abstraction problems !
- different computation methods !

  $\rightarrow$ different notions of steps, instants, reactions ...

Example

$$\begin{aligned} s_1 &= s_2 + x \\ s_2 &= \overline{s_1} + s_3 \\ s_3 &= \overline{x} \cdot s_2 \end{aligned} \quad \approx \quad \begin{aligned} s_1 &= 1 \\ s_2 &= \overline{x} \\ s_3 &= \overline{x} \end{aligned}$$

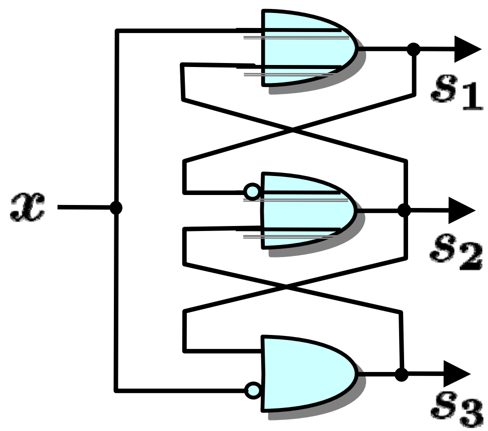
For all inputs there is a unique stationary Boolean fixed point. Thus, the system is logically reactive.

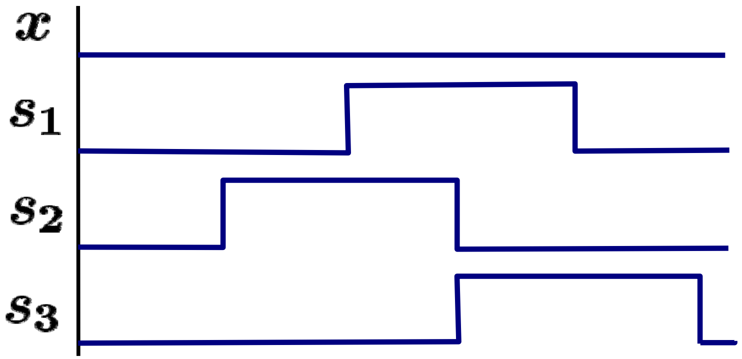
We can compile & execute Boolean solution atomically !

But what if we are compiling for a component-based and distributed architecture ?

$$s_1 = s_2 + x$$
$$s_2 = \overline{s_1} + s_3$$
$$s_3 = \overline{x} \cdot s_2$$

≉

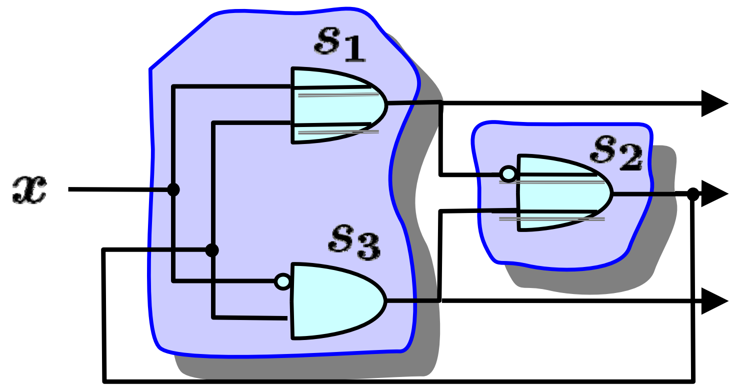




Oscillation under up-bounded inertial delay scheduling *[Brzozowski & Seger]*

Example

$$s_1 = s_2 + x$$
$$s_2 = \overline{s_1} + s_3$$
$$s_3 = \overline{x} \cdot s_2$$



Oscillation can be avoided if we

- schedule s_1, s_3 with higher priority than s_2 or
- implement s_1, s_3 atomically, as a 2in/2out block.

Then, whenever s_2 is executed, we maintain the invariant

$$s_2 = \overline{s_1} + s_3 = \overline{x}$$

4. What is in a Step ? Notions of Causality

What is in a Step ? - A Profusion of Options

1 Avoid Negations

- only positive triggers [Modecharts´94, Argos´89]

2 Modify Semantics of Negation

- give up global consistency [Huizing&al.´88, Modecharts´96]
- add consistency as implicit trigger
  [Maggiolo-Schettini &al.´96, Lüttgen &al.´99]

3 Give up Synchrony Hypothesis (no abstraction)

- all signals delayed[Statemate´90,VHDL,RSML´95,PretC´09]
- negative triggers delayed [Saraswat TCCP´94,
  Boussinot & deSimone SL´95, Boussinot FunLoft'07]

What is in a Step ? - A Profusion of Options

4 Conflict-avoiding Schedules

- only accept stratifiable (statically schedulable) programs [Normal Logic Programming]
- sequential schedule (endochrony) [Benveniste &al.´00]
- NRSA „no reaction to signal absence“ (weak endochrony, concurrent input reading) [Butucaru, Caillaud´06]

What is in a Step ? - A Profusion of Options

5 Self-scheduled Run-time (explicit absence, dual rail)

- non-deterministic speculation on absence
  [Pnueli & Shalev´91; Boussinot‘s „basic semantics“ ‘98]

What is in a Step ? - A Profusion of Options

5 Self-scheduled Run-time (explicit absence, dual rail)

- non-deterministic speculation on absence
  [Pnueli & Shalev´91; Boussinot‘s „basic semantics“ ‘98]

  „Feel free to assume the absence of a signal as long as it is consistent to do so; if necessary, backtrack!“

  - fully-abstract, compositional intuitionistic Kripke semantics [Lüttgen & Mendler´01]
  - game-theoretic “lazy“ fixed-points [Aguado & Mendler´05]

What is in a Step ? - A Profusion of Options

5 Self-scheduled Run-time (explicit absence, dual rail)

- non-deterministic speculation on absence
  [Pnueli & Shalev´91; Boussinot‘s „basic semantics“ ‘98]

- constructiveness = “computed“ absence [Berry´00]

What is in a Step ? - A Profusion of Options

5 Self-scheduled Run-time (explicit absence, dual rail)

- non-deterministic speculation on absence
  [Pnueli & Shalev´91; Boussinot‘s „basic semantics“ ‘98]

- constructiveness = “computed“ absence [Berry´00]

  “Accept the absence of a signal only under computable evidence that it may not occur later“

  - game-theoretic “eager“ fixed-points [Aguado & Mendler´05]
  - delay-insensitivity = non-inertial delay
    = constructive modal logic [Mendler & Shiple & Berry´07]

- SugarCubes [Boussinot ´98] (Esterel v3,v4,v5,v6,v7)

- & many other hardware approaches
  - speed-independence, semi-modularity, distributivity, ...

5. The Synchrony Hypothesis Thesis

Outlook

Thesis 1

There are as many notions of constructive causality as there are scheduling/run-time models

Thesis 2

Synchronous reaction requires intensional semantics:

classical Boolean logic
$\Rightarrow$ constructive logic (e.g., Heyting algebra)

least and greatest fixed points
$\Rightarrow$ general game-theoretic fixed points

Thank You !