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

Control flow paradigms

- UML Statecharts
- Statemate
- Stateflow
- RSML
- Argos

Data flow paradigms

- VisualState
- Modecharts
- SyncCharts
- iCONNECT
- Syndex
- Lustre V7
- LabView
- Simulink
- SCADE

Statecharts (Harel)
Esterel (Berry, Gonthier)
Signal (Benveniste, LeGuernic)
Lustre (Caspi, Halbwachs)
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

\[
\begin{array}{cccccc}
 t_0 & t_1 & t_2 & t_3 & t_4 \\
 x & 2 & 2 & 2 & 502 & 0 & \ldots \\
y & T & F & T & T & F & \ldots \\
z & 0.48 & 0.49 & 0.50 & 0.51 & 0.52 & \ldots \\
m & \text{red} & \text{blue} & \text{green} & \text{red} & \text{blue} & \ldots \\
k & 1 & 4 & 4 & 4 & 2 & \ldots \\
\text{memory} & : & : & : & : & : & \text{causality}
\end{array}
\]
### Data Flow

<table>
<thead>
<tr>
<th>Data</th>
<th>t&lt;sub&gt;0&lt;/sub&gt;</th>
<th>t&lt;sub&gt;1&lt;/sub&gt;</th>
<th>t&lt;sub&gt;2&lt;/sub&gt;</th>
<th>t&lt;sub&gt;3&lt;/sub&gt;</th>
<th>t&lt;sub&gt;4&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>z</td>
<td>0.48</td>
<td>0.49</td>
<td>0.50</td>
<td>0.51</td>
<td>0.52</td>
</tr>
<tr>
<td>m</td>
<td>red</td>
<td>blue</td>
<td>green</td>
<td>red</td>
<td>blue</td>
</tr>
<tr>
<td>k</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

**Memory:** :

**Time:**

**Data Flow:**

**Causality:**
Q: How do we treat the cyclic DF dependencies?

A: Continuity Hypothesis, Kahn stream semantics!
Orthogonality in Time and Space

\[\begin{array}{cccccc}
 & t_0 & t_1 & t_2 & t_3 & t_4 \\
x & 2 & 2 & 2 & 502 & 0 & \cdots \\
y & \text{T} & \text{F} & \text{T} & \text{T} & \text{F} & \cdots \\
z & 0.48 & 0.49 & 0.50 & 0.51 & 0.52 & \cdots \\
m & \text{red} & \text{blue} & \text{green} & \text{red} & \text{blue} & \cdots \\
k & 1 & 4 & 4 & 4 & 2 & \cdots \\
\text{memory} & : & : & : & : & : & \text{causality}
\end{array}\]
State Flow

State flow

<table>
<thead>
<tr>
<th></th>
<th>$t_0$</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>2</td>
<td>2</td>
<td>502</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>$y$</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>...</td>
</tr>
<tr>
<td>$z$</td>
<td>0.49</td>
<td>0.50</td>
<td>0.51</td>
<td>0.52</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$m$</td>
<td>blue</td>
<td>green</td>
<td>red</td>
<td>blue</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$k$</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Memory

causality
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

\[ \text{REACT} := \]
\[
  t_1 \parallel b \land t_2 \parallel b \land t_3 \parallel c \land t_4 \parallel b \land t_5 \parallel a \land \\
  (s_{11} \land a \land \neg t_2) \parallel t_1 \land \\
  (s_{11} \land \neg a \land \neg t_1) \parallel t_2 \land \\
  (s_{31} \land \neg a \land b) \parallel t_3 \land \\
  (s_2 \land c) \parallel t_4 \land \\
  (s_{21} \land b \land \neg c \land \neg t_4) \parallel t_5
\]
Programming Synchronous Reactions

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

\[
\text{REACT} := \\
t_1 \supset b \land t_2 \supset b \land t_3 \supset c \land t_4 \supset b \land t_5 \supset a \land \\
(s_{11} \land a \land \neg t_2) \supset t_1 \land \\
(s_{11} \land \neg a \land \neg t_1) \supset t_2 \land \\
(s_{31} \land \neg a \land b) \supset t_3 \land \\
(s_2 \land c) \supset t_4 \land \\
(s_{21} \land b \land \neg c \land \neg t_4) \supset t_5
\]
Programming Synchronous Reactions

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

\[
\text{REACT} :=
\begin{align*}
t_1 & \supset b \land t_2 \supset b \land t_3 \supset c \land t_4 \supset b \land t_5 \supset a \\
(s_{11} \land a \land \neg t_2) & \supset t_1 \\
(s_{11} \land \neg a \land \neg t_1) & \supset t_2 \\
(s_{31} \land \neg a \land b) & \supset t_3 \\
(s_2 \land c) & \supset t_4 \\
(s_{21} \land b \land \neg c \land \neg t_4) & \supset t_5
\end{align*}
\]
In which sense does \texttt{REACT} describe an atomic macro step?

\textit{instantaneous reaction}

\begin{center}
\begin{tabular}{l}
\texttt{REACT} := \\
\qquad t_1 \lor b \land t_2 \lor b \land t_3 \lor c \land t_4 \lor b \land t_5 \lor a \land \\
\qquad (s_{11} \land a \land \neg t_2) \lor t_1 \land \\
\qquad (s_{11} \land \neg a \land \neg t_1) \lor t_2 \land \\
\qquad (s_{31} \land \neg a \land b) \lor t_3 \land \\
\qquad (s_{21} \land c) \lor t_4 \land \\
\qquad (s_{21} \land b \land \neg c \land \neg t_4) \lor t_5 \\
\end{tabular}
\end{center}
In which sense does REACT describe an atomic macro step?

\[ REACT := \]
\[ t_1 \supset b \land t_2 \supset b \land t_3 \supset c \land t_4 \supset b \land t_5 \supset a \land \]
\[ (s_{11} \land a \land \neg t_2) \supset t_1 \land \]
\[ (s_{11} \land \neg a \land t_2) \supset t_1 \land \]
\[ (s_{31} \land \neg a \land b) \supset t_2 \land \]
\[ (s_{21} \land c) \supset t_4 \land \]
\[ (s_{21} \land b \land \neg c \land \neg t_4) \supset t_5 \]

Instantaneous reaction

Cyclic dependencies? \( \Rightarrow \) Fixed-Points!
3. Causal Reaction = Fixed-Point?
Synchronous Reactive Component

Reactive component

\[ C = (\mathcal{S}, \mathcal{T}, \text{pos}, \text{neg}, \text{act}) \]

\( \mathcal{S}, \mathcal{T} \) atomic logical signals, logical transitions

\( \text{pos}, \text{neg}, \text{act} : \mathcal{T} \to 2^{\mathcal{S}} \) positive, negative triggers, actions

Response of \( C \)

\[ \text{action}(T) = \{ s | \exists t \in T. s \in \text{act}(t) \} \quad T \subseteq \mathcal{T} \]

\[ \text{enabled}(S) = \{ t | \text{pos}(t) \subseteq S \land \text{neg}(t) \subseteq \overline{S} \} \quad S \subseteq \mathcal{S} \]

\[ \text{AE}_C(S) = \text{action}(\text{enabled}(S)) \] „response function“

M. Mendler

tubs.CITY, Braunschweig, 1.7.2009
"A signal $s$ is present in an instant if and only if an `emit $s$' statement is executed in this instant."

**Logical Coherence [Berry]**

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.
Problem

The response function

\[ AE_C(S) = action(\{t \mid pos(t) \subseteq S \land neg(t) \subseteq \overline{S}\}) \]

is not monotonic!
Problem

The response function $\text{covariant}$ $\Rightarrow$ $\text{contravariant}$

$\text{AE}_C(S) = \text{action} \left( \{ t \mid \text{pos}(t) \subseteq S \land \text{neg}(t) \subseteq \overline{S} \} \right)$

is not monotonic!

- no unique (least) fixed points!
- compositionality and full-abstraction problems!
- different computation methods!
  $\rightarrow$ different notions of steps, instants, reactions ...
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?
Example

\[
\begin{align*}
    s_1 &= s_2 + x \\
    s_2 &= \overline{s_1} + s_3 \\
    s_3 &= \overline{x} \cdot s_2
\end{align*}
\]

Oscillation under up-bounded inertial delay scheduling

[Brzozowski & Seger]
Example

\[
\begin{align*}
  s_1 &= s_2 + x \\
  s_2 &= \overline{s_1} + s_3 \\
  s_3 &= \overline{x} \cdot s_2 
\end{align*}
\]

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!