Distributed Esterel

A Direct Constructive Approach

Peter Wullinger

University of Bamberg

November 30th 2007

Outline

Parallel Synchronous Programming

Parallel Synchronous Programming

- Synchronous programming has many advantages:
  - Determinism
  - Reactive behaviour
  - *Relatively* simple programming model
- *But* even embedded realtime systems today are actually distributed
- Consider *fly by wire*, ...

⇒ We really want to keep synchronicity and ”simply” make it distributed.

Esterel

Short overview of Esterel

- Clock tick A clock tick divides time into a sequences of instants.
- Signals
  - Signals can be present or absent
  - Within a single instant: Signals cannot be disabled
- Control Flow Model
  - Control flows through program with each instant
- Transitions that are enabled through signal state
- Deterministic
- Semantic calculus for provable correctness

Constructive Semantics on the fly

An Esterel program in the constructive calculus consists of a set of **transitions** T:

Transitions

- **Transition:** Guards + Emissions
- **Grammar:** $(<G>?(,<G>?)*)(!\ <E>\ (,<E>)*)?$
- **Example:** $a+, b-!c$

G	$=$	$signal+?$	if signal in current round
G	$=$	$signal-?$	if **not** signal in current round
G	$=$	$pre(signal)+?$	if signal in last round
G	$=$	$pre(signal)-?$	if **not** signal in last round
E	$=$	$!signal$	emit signal.

Constructive Semantics on the fly

1 $input+?$ $pre(d0)-?$ $d0$
2 $input-?$ $pre(d0)+?$ $!d0$
3 $input+?$ $pre(d0)+?$ $!c0$
4 $c0+?$ $pre(d1)-?$ $!d1$
5 $c0-?$ $pre(d1)+?$ $!d1$
6 $c0+?$ $pre(d1)+?$ $!c1$
7 $c1+?$ $pre(d2)-?$ $!d2$
8 $c1-?$ $pre(d2)+?$ $!d2$
9 $c1+?$ $pre(d2)+?$ $!c3$

Must The set of signals that must be present

Cannot The set of signals that cannot be present

Fixed point point iteration

Minimal (w.r.t. number of present signals) fixed point

Constructive Semantics on the fly

$$
\begin{array}{llll}
1 & input+? & pre(d0)-? & d0 \\
2 & input-? & pre(d0)+? & !d0 \\
3 & input+? & pre(d0)+? & !c0 \\
4 & c0+? & pre(d1)-? & !d1 \\
5 & c0-? & pre(d1)+? & !d1 \\
6 & c0+? & pre(d1)+? & !c1 \\
7 & c1+? & pre(d2)-? & !d2 \\
8 & c1-? & pre(d2)+? & !d2 \\
9 & c1+? & pre(d2)+? & !c3
\end{array}
$$

- Start with
  $Must = \{input\}$

Constructive Semantics on the fly

1	$input+?$	$pre(d0)-?$	$d0$
2	$input-?$	$pre(d0)+?$	$!d0$
3	$input+?$	$pre(d0)+?$	$!c0$
4	$c0+?$	$pre(d1)-?$	$!d1$
5	$c0-?$	$pre(d1)+?$	$!d1$
6	$c0+?$	$pre(d1)+?$	$!c1$
7	$c1+?$	$pre(d2)-?$	$!d2$
8	$c1-?$	$pre(d2)+?$	$!d2$
9	$c1+?$	$pre(d2)+?$	$!c3$

- Start with
  $Must = \{input\}$
- Check Guards

Green	Present
Red	Absent
Black	Undecided

Constructive Semantics on the fly

1	$input+?$	$pre(d0)-?$	$d0$
2	$input-?$	$pre(d0)+?$	$!d0$
3	$input+?$	$pre(d0)+?$	$!c0$
4	$c0+?$	$pre(d1)-?$	$!d1$
5	$c0-?$	$pre(d1)+?$	$!d1$
6	$c0+?$	$pre(d1)+?$	$!c1$
7	$c1+?$	$pre(d2)-?$	$!d2$
8	$c1-?$	$pre(d2)+?$	$!d2$
9	$c1+?$	$pre(d2)+?$	$!c3$

- Start with $Must = \{input\}$
- Check Guards

Green	Present
Red	Absent
Black	Undecided

Undecided	Not all guard signals surely absent/present
Decided	All guard signals surely absent/present
Enabled	All guard signals match definition
Disabled	At least one guard signal does not match definition

Constructive Semantics on the fly

1	$input+?$	$pre(d0)-?$	$d0$
2	$input-?$	$pre(d0)+?$	$!d0$
3	$input+?$	$pre(d0)+?$	$!c0$
4	$c0+?$	$pre(d1)-?$	$!d1$
5	$c0-?$	$pre(d1)+?$	$!d1$
6	$c0+?$	$pre(d1)+?$	$!c1$
7	$c1+?$	$pre(d2)-?$	$!d2$
8	$c1-?$	$pre(d2)+?$	$!d2$
9	$c1+?$	$pre(d2)+?$	$!c3$

- Transition 1 is enabled
- **d0 must** be emitted.
- $d0 \in Must$

Constructive Semantics on the fly

1 $input+?$ $pre(d0)-?$ $d0$
2 $input-?$ $pre(d0)+?$ $!d0$
3 $input+?$ $pre(d0)+?$ $!c0$
4 $c0+?$ $pre(d1)-?$ $!d1$
5 $c0-?$ $pre(d1)+?$ $!d1$
6 $c0+?$ $pre(d1)+?$ $!c1$
7 $c1+?$ $pre(d2)-?$ $!d2$
8 $c1-?$ $pre(d2)+?$ $!d2$
9 $c1+?$ $pre(d2)+?$ $!c3$

- Transition 1 is enabled
- **d0 must** be emitted.
- $d0 \in Must$
- Transitions 2, 3, 5, 6, 8, 9 are disabled

Constructive Semantics on the fly

1 $input+?$ $pre(d0)-?$ $d0$
2 $input-?$ $pre(d0)+?$ $!d0$
3 $input+?$ $pre(d0)+?$ $!c0$
4 $c0+?$ $pre(d1)-?$ $!d1$
5 $c0-?$ $pre(d1)+?$ $!d1$
6 $c0+?$ $pre(d1)+?$ $!c1$
7 $c1+?$ $pre(d2)-?$ $!d2$
8 $c1-?$ $pre(d2)+?$ $!d2$
9 $c1+?$ $pre(d2)+?$ $!c3$

- Transition 1 is enabled
- **d0 must** be emitted.
- $d0 \in Must$
- Transitions 2, 3, 5, 6, 8, 9 are disabled

- What about Transitions 4 and 7?

Constructive Semantics on the fly

1	$input+?$	$pre(d0)-?$	$d0$
2	$input-?$	$pre(d0)+?$	$!d0$
3	$input+?$	$pre(d0)+?$	$!c0$
4	$c0+?$	$pre(d1)-?$	$!d1$
5	$c0-?$	$pre(d1)+?$	$!d1$
6	$c0+?$	$pre(d1)+?$	$!c1$
7	$c1+?$	$pre(d2)-?$	$!d2$
8	$c1-?$	$pre(d2)+?$	$!d2$
9	$c1+?$	$pre(d2)+?$	$!c3$

- Transition 1 is enabled
- **d0 must** be emitted.
- $d0 \in Must$
- Transitions 2, 3, 5, 6, 8, 9 are disabled

- What about Transitions 4 and 7?
- $c0$ **cannot be emitted**! $\rightarrow c0 \in Cannot$
- Transition 4 is disabled

Constructive Semantics on the fly

1 $input+?$ $pre(d0)-?$ $d0$
2 $input-?$ $pre(d0)+?$ $!d0$
3 $input+?$ $pre(d0)+?$ $!c0$
4 $c0+?$ $pre(d1)-?$ $!d1$
5 $c0-?$ $pre(d1)+?$ $!d1$
6 $c0+?$ $pre(d1)+?$ $!c1$
7 $c1+?$ $pre(d2)-?$ $!d2$
8 $c1-?$ $pre(d2)+?$ $!d2$
9 $c1+?$ $pre(d2)+?$ $!c3$

- Transition 1 is enabled
- **d0 must** be emitted.
- $d0 \in Must$
- Transitions 2, 3, 5, 6, 8, 9 are disabled

- What about Transitions 4 and 7?
- $c0$ **cannot be emitted**! $\rightarrow c0 \in Cannot$
- Transition 4 is disabled
- $c1$ **cannot be emitted**! $\rightarrow c1 \in Cannot$
- Transition 7 is disabled

Distributed Synchronous Programming

Problem

- Calculation methods are centralised.
- But even the binary counter is actually already a parallel system: *Signal transmission delay*

Distributed calculation

- Multiple approaches: Girault, Berry, Boussinot
- Most approaches: Distribute code according to some ruleset
- **Problem**: Non-constructiveness is hard to detect at runtime.

Distributed Constructive Semantics

Basic architecture

- Rulesets are distributed to nodes of a transition system:
  $T_i \subseteq T$ $\qquad\qquad$ $\bigcup_i T_i = T$
- Only local information is available

Goal

- Same behaviour as in the centralised case

Idea

- Must-Cannot is distributive

Algorithm draft

Algorithm draft

Step Calculate Must/Cannot locally.

- Broadcast-convergecast Must-Cannot-Sets
- New must set is union of all Must sets
- New cannot set is intersection of all Cannot sets

Term A round ends, if there are no changes any more

Distributed Calculation

A	(input)	$()$
B	$()$	$()$

- Distribute single digit only

$$T_A = \{ \quad \text{input+,pre(d0)-!d0} \}$$
$$T_B = \{ \quad \text{input-,pre(d0)+!d0,}$$
$$\quad \text{input+,pre(d0)+!c0} \}$$

Distributed Calculation

A $(input, d0, c0)$ $()$

B $(input, d0, c0)$ $()$

- Distribute single digit only

  $T_A = \{$ input+,pre(d0)-!d0 $\}$
  $T_B = \{$ input-,pre(d0)+!d0,
  input+,pre(d0)+!c0 $\}$

Distributed Calculation

$$
\begin{tikzcd}
A & (input, d0, c0) \arrow[dr, "E_A"] & () \\
B & (input, d0, c0) & ()
\end{tikzcd}
$$

- Distribute single digit only

$$
\begin{aligned}
T_A &= \{ \quad \text{input+,pre(d0)-!d0} \ \} \\
T_B &= \{ \quad \text{input-,pre(d0)+!d0,} \\
&\qquad\ \ \text{input+,pre(d0)+!c0} \ \}
\end{aligned}
$$

Distributed Calculation

$A \quad (input, d0, c0) \qquad \xrightarrow{E_B} \qquad ()$

$B \quad (input, d0, c0) \qquad (input, d0, c0)$

- Distribute single digit only

$$
\begin{aligned}
T_A &= \{ \quad \text{input+,pre(d0)-!d0 } \} \\
T_B &= \{ \quad \text{input-,pre(d0)+!d0,} \\
&\qquad\ \ \text{input+,pre(d0)+!c0 } \}
\end{aligned}
$$

Distributed Calculation

A $(input, d0, c0)$ $(input, d0, c0)$

B $(input, d0, c0)$ $(input, d0, c0)$

- Distribute single digit only

$$T_A = \{ \quad \text{input+,pre(d0)-!d0} \}$$
$$T_B = \{ \quad \text{input-,pre(d0)+!d0,}$$
$$\text{input+,pre(d0)+!c0} \}$$

Distributed Calculation

A $(input, d0, c0)$ $()$

B $(input, d0, c0)$ $()$

Combine

- Distribute single digit only

$T_A = \{$ input+,pre(d0)-!d0 $\}$
$T_B = \{$ input-,pre(d0)+!d0,
input+,pre(d0)+!c0 $\}$

Distributed Calculation Algorithm

Distributed Calculation Algorithm

- $\forall i E_k^i \leftarrow 0$
- $\forall i E_{k+1}^i \leftarrow Must(T_i, E_k)^+ \cup Cannot(T_i, E_k)^- \cup E_k$
- Broadcast/convergecast E_{k+1}^i
- Combine information
  $E_{k+1} = (\bigcup_i Must(T_i, E_k))^+ \cup (\bigcap_i Cannot(T_i, E_k))^- \cup E_k$
- Repeat until for some $n+1$: $E_{n+1} = E_n$

Properties of the simple algorithm

Properties

- **Good** Arbitrary distribution of transition set possible
- **Good** Provable same behaviour as centralised version
- **Bad** $M = O(|S|n^2)$ $T = O(|S|n)$
- **Bad** Waits for the slowest node, even if decision does not depend on that node.

Improvements – Early Local Termination

Barrier

- The broadcast-convergecast is a simple implementation of a BSP barrier.
- This is strong enough for our synchronisation needs
- But can we do with something weaker?

Local Early Termination

- If a node is decided (all local transitions decided)

$\Rightarrow$ Transmit a marker on outbound channels

$\Rightarrow$ Stop processing for the decided node

- If a marker is received on a channel:

$\Rightarrow$ Ignore that channel for the current round

Improvements (continued)

Non-Fully Connected Graphs

- Some channels never carry useful information
- If a node only ever emits messages that are not useful to the other endpoint
- Those channels can be left out

Improvements (continued 2)

Sequential composition

- $p; q$
- Require p to be decided, before evaluating q
- Transition p depends on another transition q

- Introduce *decision* signals
- Emit *decision* signal when a transition is decided
- Enable transition only if *decision* signal of all its preconditions are available

Conclusion

Distributed Esterel

- A simple distribution algorithm exists
- $O(n^2)$ unfortunately
- We can make some improvements
- Distributing circuits is more efficient

Implementation

- Java is not so good for this
- Communication system works well
- Debugging is very hard

Questions?

μ-steps

How is the output signal set calculated?

Procedure: Fixed point iteration.

- Calculate a converging sequence of signal sets
- *Must* and *Cannot* combined: *Environment E*

e.g. $Must = \{a\}, Cannot = \{b\} \rightarrow E = \{a+, b-\}$

- Sequence of environments $E_0, E_1, \ldots, E_\infty$

Esterel μ-Steps (continued 1)

Step Sequence

Step For every transition $t \in T$, calculate $must(t, E_k)$.

- For every transition t, calculate $cannot(t, E_k)$.

Esterel μ-Steps (continued 1)

Step Sequence

Step For every transition $t \in T$, calculate $must(t, E_k)$.

- For every transition t, calculate $cannot(t, E_k)$.
- $Must(T, E_k)$ is the union of all $must(t, E_k)$
- $Cannot(T, E_k)$ is the intersection of all $cannot(t, E_k)$

Esterel μ-Steps (continued 1)

Step Sequence

Step For every transition $t \in T$, calculate $must(t, E_k)$.

- For every transition t, calculate $cannot(t, E_k)$.
- $Must(T, E_k)$ is the union of all $must(t, E_k)$
- $Cannot(T, E_k)$ is the intersection of all $cannot(t, E_k)$
- Set E_{k+1} to E_k plus all signals in the must cannot sets marked with $+$ and $-$ accordingly:

$$E_{k+1} = must(T, E_k)^{+} \cup Cannot(T, E_k)^{-} \cup E_k$$

- Repeat until there are no more changes.

Esterel μ-Steps (continued 2)

$must()$

$$t = a{+}?(u) \quad must(t, E_k) = \begin{cases} must(u, E_k) & \text{if } a+ \in E_k \\ \emptyset & \text{otherwise} \end{cases}$$

$$t = a{-}?(u) \quad must(t, E_k) = \begin{cases} must(u, E_k) & \text{if } a- \in E_k \\ \emptyset & \text{otherwise} \end{cases}$$

$$t = !s \quad must(t, E_k) = s$$

Esterel μ-Steps (continued 3)

$cannot()$

$$t = a{+}?(u) \quad cannot(t, E_k) = \begin{cases} S & \text{if } a- \in E_k \\ cannot(u, E_k) & \text{otherwise} \end{cases}$$

$$t = a{-}?(u) \quad cannot(t, E_k), = \begin{cases} S & \text{if } a+ \in E_k \\ S \backslash cannot(t, E_k) & \text{otherwise} \end{cases}$$

$$t = !s \quad cannot(t, E_k) = S \backslash \{s\}$$

Esterel μ-Steps (continued 4)

Fixed Point Iteration

- *Must* and *Cannot* iterations converge
- The resulting sets are a *canonical fixed point* with regard to number of active signals.
- The fixed point in general may not be unique.
- Example:

  `c-(a+?(!b))`
- Under the environment $c+$ there are two fixed points points $\{c+, a+\}$, $\{c+, a-\}$.
- Only the last fixed point is canonical.

Esterel μ-Steps (continued 5)

Non-Constructive Programs

- We said *Must* and *Cannot* converge
- This is not true ...
- ... and even worse: the result of convergence may not be consistent

Ex a - ! b
b + ! a

- Analysis says: $Must_\infty = \{a, b\}$
- It also says: $Cannot_\infty = \{a, b\}$.
- This cannot be right! Such programs are called *non-constructive*.
- It happens when there is a non-resolvable cycle.