A second generation model: SL/Esterel

Some informal, driving ideas

- In SCCS/Meije (or in Moore machines) the behaviour of a process within an instant is determined by a big, fixed table. We want to be able to *program the actions* we take within an instant.
- We want to *share information* among several processes without introducing non-determinism. To this end, we consider *signals* which are messages that *persist* within an instant.
- At the end of the instant, we want to be able to detect the *absence of a signal*. This allows to move from a unary to a binary notation.

A simple calculus for the SL model

Write $s, s', \ldots$ for signals.

$$\text{Processes} \qquad P \quad ::= 0 \parallel \overline{s} \parallel s.P, K \parallel (P \mid P) \parallel \nu s\ P \parallel A(\mathbf{s})$$

$$\text{Continuations} \quad K \quad ::= P \parallel \mathsf{ite}\ s\ K\ K$$

- $s.P, K$ is the present statement of the SL model: run P if s is emitted and otherwise evaluate K at the *end of the instant*.
- K allows to select a continuation in the following instant according to an arbitrary boolean condition on signals.

Resuming at the following instant

We 'define':

$$\mathsf{pause}.K = \nu s\ (s.0, K) \qquad s \notin fn(K)$$

We wait till the end of the instant and then we evaluate K.

Waiting for a signal

- Suppose $fn(P) \cup \{s\} = \{\mathbf{s}\}$.
- We 'define':

$$\mathsf{await}\ s.P = A(\mathbf{s})$$

where:

$$A(\mathbf{s}) = s.P, A(\mathbf{s})$$

We wait an arbitrary number of instants for a signal.

Example: programming a NOR gate in SL

- Input signals: s_0, s_1.
- Output signal: s.
- At instant $i+1$, emit s iff neither s_0 nor s_1 were emitted at instant i.

$$
\begin{aligned}
N_0 &= \mathsf{pause}.(\ \mathsf{ite}\ s_0\ N_0\ (\mathsf{ite}\ s_1\ \ N_0\ N_1)\ ) \\
N_1 &= \overline{s} \mid N_0
\end{aligned}
$$

Remarks

- We can *program* the boolean function NOR rather than writing down its *truth table*.
- Several threads can *share the same signal*:

$$\overline{s} \mid s.P_1, K_1 \mid s.P_2, K_2$$

- We can react to the *absence of a signal* at the end of the instant and therefore we can regard a signal as a *binary information*.

Some historical remarks

- In the ESTEREL model it is actually possible to *react immediately* (rather than at the end of the instant) to the absence of a signal.
- This requires some semantic care, to avoid writing paradoxical programs such as:

  $$s.0, \overline{s}$$

  which are supposed to emit s when s is not there (cf. stabilization problems in synchronous circuits).
- It also requires some clever *compilation techniques* to determine whether a signal is not emitted. Infact these techniques seem specific to *finite state models*.

- SL is a *relaxation* of the ESTEREL model where the *absence of a signal* can only be detected at the end of the instant.
- If we forget about *name generation*, then the SL model essentially defines a kind of *monotonic Mealy machine*. Monotonic in the sense that output signals can only depend *positively* on input signals (within the same instant).
- The monotonicity restriction allows to *avoid the paradoxical programs* (monotonic boolean equations do nhave a least fixed point!).
- The SL model has a natural and *efficient implementation model* that works well for *general programs* (not just finite state machines).

- The ESTEREL/SL models were conceived in Sophia-Antipolis shortly after the *SCCS/Meije models* and in the *same research team*.
- In spite of this, there is *no* strong formal result on the possibility/impossibility of embedding one model into the other up to some reasonable equivalence.

Macro-scopic transitions

- Let I, O be finite sets of signals.
- The judgement:

$$P \overset{I/O}{\rightarrow} P'$$

  means: P receiving the signals I at the beginning of the instant, reaches the end of the instant, emits the signals O, and becomes the process P' in the following instant.

- A *derivative* of P is a process P' such that:

$$P \overset{I/O}{\rightarrow} \ldots \overset{I'/O'}{\rightarrow} P'$$

Two desirable properties

Reactivity Instant should end, *i.e.*, for all derivatives Q of P, for all inputs I, the process Q must reache the end of the instant (suspends).

Determinacy (assuming reactivity) For all derivatives Q of P, for all inputs I,

$$\frac{Q \overset{I/O_1}{\rightarrow} Q_1 \quad Q \overset{I/O_2}{\rightarrow} Q_2}{O_1 = O_2 \quad Q_1 \text{ equivalent to } Q_2}$$

Micro-scopic semantics

The *macro*-scopic semantics relies on a *micro*-scopic semantics that is built on three relations:

- A *labelled transition system* $\xrightarrow{\alpha}$ describing how a process can interact during the instant.
- A *suspension predicate* $\downarrow$ that holds when a process cannot perform any more internal transition.
- An *evaluation relation* $\mapsto$ describing how the process evolves at the end of the instant.

Labelled transition system

Actions

$$\alpha ::= \tau \parallel s \parallel \overline{s}$$

Special rules for signals

$$\frac{}{\overline{s} \xrightarrow{\overline{s}} \overline{s}} \qquad \frac{}{s.P, K \xrightarrow{s} (P \mid \overline{s})}$$

Standard rules (cf. CCS)

$$\frac{P_1 \xrightarrow{s} P_1' \quad P_2 \xrightarrow{\overline{s}} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} P_1' \mid P_2'} \qquad \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 \mid P_2 \xrightarrow{\alpha} P_1' \mid P_2}$$

$$\frac{P \xrightarrow{\alpha} P' \quad s \notin \alpha}{\nu s\ P \xrightarrow{\alpha} \nu s\ P'} \qquad \frac{A(\mathbf{x}) = P}{A(\mathbf{s}) \xrightarrow{\tau} [\mathbf{s}/\mathbf{x}]P}$$

Remarks

- Emission of a signal is *persistent*.
- When a signal is present, the continuation is guaranteed to see the signal as present. For instance,

$$s.(\overline{s}_1 \mid s.\overline{s}_2, 0), 0$$

either suspends on s or emits *both* s_1 *and* s_2.

- The input rule has the effect that:

$$\overline{s} \mid s.0, 0 \xrightarrow{\tau} \overline{s} \mid 0 \mid \overline{s}$$

- However this is not a problem, because emitting a signal *once* is the *same* as emitting it *twice*.

Suspension

- We write $P \downarrow$ if $\neg P \xrightarrow{\tau} \cdot$.
- By inspection of the semantics, one verifies that suspension means that every 'thread' is either *terminated* (0) or *waiting* for a signal that was not emitted.
- The (virtual) machine running a program has efficient means to *detect suspension* and to move the computation to the following instant.

End of the instant

The evaluation at the end of the instant is defined by a judgement

$$P \mapsto P'$$

which is defined compositionally using a more elaborate judgement

$$P \overset{E,V}{\longmapsto} P'$$

where E, V are sets of signals.

Intuitively, in $P \overset{E,V}{\longmapsto} P'$:

- E: *signals emitted* by P.
- V: signals that are *assumed to be emitted* at the end of the instant.

Thus we expect: $E \subseteq V$. At the end of the instant, we compute:

$$[\![K]\!]V = \begin{cases} P & \text{if } K = P \\ [\![K_1]\!]V & \text{if } K = \mathsf{ite}\ s\ K_1\ K_2, \quad s \in V \\ [\![K_2]\!]V & \text{if } K = \mathsf{ite}\ s\ K_1\ K_2, \quad s \notin V \end{cases}$$

Rules for the end of the instant

$$\frac{}{0 \overset{\emptyset,V}{\longmapsto} 0} \qquad \frac{s \in V}{\overline{s} \overset{\{s\},V}{\longmapsto} 0} \qquad \frac{s \notin V}{s.P, K \overset{\emptyset,V}{\longmapsto} [\![K]\!]V}$$

$$\frac{P_i \overset{E_i,V}{\longmapsto} P_i' \quad i = 1,2}{(P_1 \mid P_2) \overset{E_1 \cup E_2,V}{\longmapsto} (P_1' \mid P_2')}$$

$$\frac{P \overset{E,V}{\longmapsto} P' \quad \{s\} \cap E = \{s\} \cap V}{\nu s\ P \overset{E \setminus \{s\},V \setminus \{s\}}{\longmapsto} \nu s\ P'}$$

$$\frac{P \overset{E,V}{\longmapsto} P' \quad E = V}{P \mapsto P'}$$

Example

$$\nu s_1\ (s_1.0, (\mathsf{ite}\ s_2\ A()\ 0) \mid s_2) \mid \overline{s}_1 \mid \overline{s}_2 \quad \overset{\{s_1,s_2\},\{s_1,s_2\}}{\longmapsto} \quad \nu s_1 (A() \mid 0) \mid 0 \mid 0$$

NB Evaluation at the end of the instant is *deterministic*. If $P \downarrow$ then $\exists! P'\ (P \mapsto P')$.

From micro to macro transitions

- If $I = \{s_1, \ldots, s_n\}$ is a finite set of signals let $P_I = \overline{s}_1 \mid \cdots \mid \overline{s}_n$.
- Let $Out(P) = \{s \mid P \xrightarrow{\overline{s}} \cdot\}$.
- Define the I/O macro transitions as follows:

$$\frac{(P \mid P_I) \stackrel{\tau}{\Rightarrow} P'' \qquad P'' \downarrow \qquad Out(P'') = O \qquad P'' \mapsto P'}{P \stackrel{I/O}{\longrightarrow} P'}$$

Exercise

Define the rules for the lts and the end of the instant for the operators await and pause.

Coming next

A survey on:

1. Reactivity.
2. Expressivity.
3. Determinacy.
4. Compositional reasoning.
5. Data types extensions.

in the SL model.

Reactivity in the SL model

A simple static analysis that guarantees reactivity

- We assume the instruction pause is *explicitly* used in the program.
- We compute an *over-approximation* of the *control flow* of the system of equations.
- We check that within an instant it is *not possible to loop* through a thread identifier.

Call graph

If P is a process then $Call(P)$ is an *over-approximation* of the set of process identifiers that P may possibly call within the current instant:

$$
\begin{array}{ll}
Call(P) & = \mathsf{case}\ P \\
0 & : \emptyset \\
\mathsf{pause}.P & : \emptyset \\
B(\mathbf{a}) & : \{B\} \\
\overline{s} & : \emptyset \\
s.P, K & : Call(P) \\
\nu s\ P & : Call(P) \\
P_1 \mid P_2 & : Call(P_1) \cup Call(P_2)
\end{array}
$$

Given a system of equations:

$$\begin{aligned} A_1(\mathbf{a}_1) &= P_1 \\ \ldots & \\ A_n(\mathbf{a}_1) &= P_n \end{aligned}$$

build a (directed) *call graph* with *nodes* $\{A_1, \ldots, A_n\}$ and such that

$$(A_i, A_j) \text{ is an edge iff } A_j \in \mathit{Call}(A_i)$$

Proposition If the call graph has no loops then any process relying on the related system of equations is reactive.

Proof idea

- If the graph has no loops then we can define a *well-founded order* $>$ on thread identifiers such that $A > B$ whenever there is an edge from A to B in the call graph.
- A process is essentially a *multi-set* of threads:

$$\{\!|P_1, \ldots, P_n|\!\}$$

- Whenever we perform an internal reduction either we reduce the size of a P_i or we unfold a recursive equation $A_i(\mathbf{a}) = P_i$ and then we have:

$$Call(A_i) = \{\!|A_i|\!\} >_{mset} Call(P_i)$$

Exercise

Define a well-founded measure on processes that shows that all internal reductions terminate.

Expressivity of the SL model

Simulating push-down automata

- We want to write a SL program that simulates a *deterministic push-down automata*.
- This example serves three purposes:
  1. Some *non-trivial hacking* in SL.
  2. An opportunity for *reactivity analysis*.
  3. Suggests that the SL model is *Turing equivalent*.

- A *configuration* is a pair (q, w) where $q \in Q$ is a *state* and $w = S \cdots SZ$ is a *stack*.
- Possible transitions are:

$$\begin{array}{lll} (q, w) & \to (q', Sw) & \text{(increment)} \\ (q, Sw) & \to (q', w) & \text{(decrement)} \\ (q, Z) & \to (q', Z) & \text{(positive zero test)} \\ (q, Sw) & \to (q', Sw) & \text{(negative zero test)} \end{array}$$

- We assume that the automaton is *deterministic* and that we *check that the stack is not empty* before running a decrement instruction.

- Associate a *recursive equation* with each state/instruction:

$$q \quad = \overline{inc} \mid \mathsf{await}\ ack.\mathsf{pause}.q'$$

$$q \quad = \overline{dec} \mid \mathsf{await}\ ack.\mathsf{pause}.q'$$

$$q \quad = zero.(\mathsf{pause}.q'), q''$$

- With a configuration $(q, S \cdots SZ)$ associate the program:

$$\nu \mathbf{s}_0, \ldots \mathbf{s}_n \ (q(\mathbf{s}_0) \mid S(\mathbf{s}_0, \mathbf{s}_1) \mid \cdots \mid S(\mathbf{s}_{n-1}, \mathbf{s}_n) \mid Z(\mathbf{s}_n))$$

- Z is described by the equation:

$$Z(\mathbf{s}) \quad = \overline{zero} \mid inc.(\overline{ack} \mid \mathsf{pause}.\nu \mathbf{s}'(S(\mathbf{s}, \mathbf{s}') \mid Z(\mathbf{s}'))), Z(\mathbf{s})$$

- and S by the equations:

$$
\begin{aligned}
S(\mathbf{s},\mathbf{s}') &= S_{inc}(\mathbf{s},\mathbf{s}') \mid S_{dec}(\mathbf{s},\mathbf{s}') \\
S_{inc}(\mathbf{s},\mathbf{s}') &= inc.\mathsf{pause}.S^{+}(\mathbf{s},\mathbf{s}'),(\mathsf{ite}\ dec\ 0\ S_{inc}(\mathbf{s},\mathbf{s}')) \\
S_{dec}(\mathbf{s},\mathbf{s}') &= dec.\mathsf{pause}.S^{r}(\mathbf{s},\mathbf{s}'),(\mathsf{ite}\ inc\ 0\ S_{dec}(\mathbf{s},\mathbf{s}')) \\
S^{+}(\mathbf{s},\mathbf{s}') &= \overline{ack} \mid \nu\mathbf{s}''\ (S(\mathbf{s},\mathbf{s}'') \mid S(\mathbf{s}'',\mathbf{s}')) \\
S^{r}(\mathbf{s},\mathbf{s}') &= zero'.(\mathsf{pause}.(\overline{ack} \mid Z(\mathbf{s})),(\overline{dec'} \mid S^{l}(\mathbf{s},\mathbf{s}')) \\
S^{l}(\mathbf{s},\mathbf{s}') &= ack'.\mathsf{pause}.(\overline{ack} \mid S(\mathbf{s},\mathbf{s}')),S^{l}(\mathbf{s},\mathbf{s}')
\end{aligned}
$$

Some dynamics

Increment

$$SSZ \to S^{+}SZ \to SSSZ$$

Decrement

$$SSSZ \to S^{r}SSZ \to S^{l}S^{r}SZ \to S^{l}S^{l}S^{r}Z \to S^{l}S^{l}Z \to S^{l}SZ \to SSZ$$

Remarks

- In S we have two parallel threads: one waiting for the signal *inc* and the other for the signal *dec*. At the end of the instant, the first thread will abort if the signal *dec* has been emitted (and symmetrically).
- The call graph associated with the system of equations is *acyclic*. Hence the program is *reactive*.
- It is easy to adapt the program to simulate a *two counters machine* (name generation is essential here).

Exercise

Show that the presented encoding of push-down automata can be adapted to CCS.

Determinacy in the SL model

(Very) Strong confluence

Determinacy of macro transitions follows by a *strong confluence property* of micro transitions:

$$\frac{P \xrightarrow{\tau} P_1 \qquad P \xrightarrow{\tau} P_2}{P_1 = P_2 \text{ or } \exists\, Q \;\; (P_1 \xrightarrow{\tau} Q, P_2 \xrightarrow{\tau} Q)}$$

NB We close the diagram in *at most one step* and *up to α-renaming*.

Proof idea

- Internal reductions are due either to *unfolding* or to *synchronisation*.
- The only possibility for a *superposition* of the redexes is:

$$\overline{s} \mid s.P_1, K_1 \mid s.P_2, K_2$$

- And we exploit the fact that emission is *persistent*.

- Now consider $(P \mid P_I)$.
- By *reactivity*, all internal reduction sequences terminate.
- By reactivity, *strong confluence implies confluence*. Thus there is a unique process P'' such that

$$(P \mid P_I) \stackrel{\tau}{\Rightarrow} P'' \text{ and } P'' \downarrow$$

- Then the observable output is $Out(P'')$ and the continuation at the next instant is the unique P' such that $P'' \mapsto P'$ (remember that evaluation at the end of the instant is deterministic).

Exercise

Show a strong confluence property for the *labelled transition system*.

A compositional semantics for the SL model

Trace semantics

Following, our definition for CCS, we could define a *trace semantics* as:

$$tr(P) = \{(I_1/O_1) \cdots (I_n/O_n) \mid P \stackrel{I_1/O_1}{\rightarrow} \cdots \stackrel{I_n/O_n}{\rightarrow} .\}$$

Problems

- The definition is *not very manageable* because the transitions are complicated.
- It is not clear *how to prove congruence properties*.

A glimpse at a characterisation

It is possible to:

- Define a notion of *bisimulation* at the level of the micro transitions ($\xrightarrow{\alpha}$, $\downarrow$, $\mapsto$).
- Show that the notion of bisimulation is *preserved by the operators* and that can be characterised as a *contextual bisimulation*.
- Prove that *because of determinacy* bisimulation *collapses* with trace equivalence.

The definition of bisimulation (for reactive processes)

Assuming $P \; R \; Q$:

$$(L1) \qquad \frac{P \xrightarrow{\alpha} P' \quad \alpha = \tau \text{ or } \alpha = \overline{s}}{\exists\, Q' \;\; (Q \stackrel{\tau}{\Rightarrow} Q' \text{ and } P' \; R \; Q')}$$

$$(L2) \qquad \frac{P \xrightarrow{s} P'}{\begin{array}{c}\exists Q' \;\; (\; Q \stackrel{s}{\Rightarrow} Q' \text{ and } P' \; R \; Q') \text{ or} \\ (\; Q \stackrel{\tau}{\Rightarrow} Q' \text{ and } P' \; R \; (Q' \mid \overline{s})\;)\end{array}}$$

$$(L3) \qquad \frac{\begin{array}{c} S = \overline{s}_1 \mid \cdots \mid \overline{s}_n, \quad n \geq 0 \\ P' = (P \mid S)\downarrow, \text{ and } P' \mapsto P'' \end{array}}{\begin{array}{c}\exists Q', Q'' \;\; (\; (Q \mid S) \stackrel{\tau}{\Rightarrow} Q', \;\; Q'\downarrow, \;\; P' \; R \; Q', \\ Q' \mapsto Q'', \quad \text{and} \quad P'' \; R \; Q'' \;)\end{array}}$$

Remarks on the definition

(L1) Standard.

(L2) Inspired by the π-calculus with *asynchronous communication*.
No reason to distinguish $s.0, 0$ from 0.

(L3) Emitted values have an effect at the *end of the instant*.

Consider: $P = s_1.0, (\mathsf{ite}\ s_2\ \overline{s}_3, 0) \qquad Q = s_1.0, 0$

Then: $P \downarrow, \qquad Q \downarrow, \qquad P \mapsto 0, \qquad Q \mapsto 0$

However: $(P \mid \overline{s}_2)$ is not equivalent to $(Q \mid \overline{s}_2)$

Extending the SL model with data values

SL and its evolution

- The language with *pure* signals is *deterministic*.
- Reasonable extension to *(infinite) data domains*. The resulting language becomes *non-deterministic*.
- Efficient *implementation model*.
- Embedded in many *programming environments*: C, C++, Scheme, ML.
- Significant *applications*: event-driven control, data flow, GUI, simulations, web services, multiplayer games.

Some references

- Boussinot. Reactive C: an extension of C to program reactive systems. *Soft. Practice and Experience*, 1991.
- Mandel-Pouzet. Reactive ML, a reactive extension to ML. In *Proc. ACM PPDP*, 2005.

The $S\pi$-calculus: a *synchronous* π-calculus

Assume $v_1 \neq v_2$ are two distinct values and

$$
\begin{aligned}
P = \nu\ s_1, s_2\ (&\overline{s_1} v_1 \mid \overline{s_1} v_2 \mid \\
&s_1(x).\ (s_1(y).\ (s_2(z).\ A(x,y)\ , B(!s_1)) \\
&\qquad , 0) \\
&, 0)
\end{aligned}
$$

P is a π-calculus process if we forget about the else branches of the read instructions.

Spot the differences...

$P = \nu\ s_1, s_2\ (\overline{s_1}v_1 \mid \overline{s_1}v_2 \mid s_1(x).\ (s_1(y).\ (s_2(z).\ A(x,y)\ , B(!s_1)), 0), 0)$

- In π, P reduces to

$$P_1 = \nu s_1, s_2\ (s_2(z).0, A(\sigma(x), \sigma(y)), B(!s_1))$$

where $\sigma(x), \sigma(y) \in \{v_1, v_2\}$ and $\sigma(x) \neq \sigma(y)$.

- In $S\pi$, *signals persist within the instant* and P reduces to

$$P_2 = \nu s_1, s_2\ (\overline{s_1}v_1 \mid \overline{s_1}v_2 \mid (s_2(z).A(\sigma(x), \sigma(y)), B(!s_1)))$$

where $\sigma(x), \sigma(y) \in \{v_1, v_2\}$.

- In π, P_1 is now *deadlocked*.
- In $S\pi$, the *current instant ends* and we move to the following one

$$P_2 \mapsto P_2' = \nu s_1, s_2 \; B(\sigma(\ell))$$

  where $\sigma(\ell) \in \{[v_1; v_2], [v_2; v_1]\}$.
- Thus at the end of the instant, $!s_1$ becomes *a list of (distinct) values* emitted on s_1 during the instant.
- For this reason, $S\pi$ includes *lists has a primitive data structure*.

Internal choice

Non-determinism arises when emitting *distinct values* on the same signal:

Within the instant...

$$s(x).P, K \mid \overline{s}0 \mid \overline{s}1$$

with suitable encodings of 0 and 1.

...and at the end of the instant

$$\mathsf{pause}.A(!s) \mid \overline{s}0 \mid \overline{s}1$$

Results and/or Problems

1. (Feasible) Reactivity.
2. Determinacy.
3. Compositional Semantics.

Some references

- G. Berry and G. Gonthier, *The Esterel synchronous programming language*. Science of computer programming, 19, 1992.

  It introduces an imperative language to program reactive systems. The language can be compiled to finite automata. The semantics allows to react immediately to the absence of a signal. Static analysis is required to avoid 'causality problems'.

- F. Boussinot and R. De Simone, *The SL Synchronous Language*. IEEE Trans. on Software Engineering, 22, 1996.

  Relaxation of the Esterel model. It allows reaction to the absence of a signal only at the end of the instant.

- A., *The SL synchronous language, revisited. Journal of Logic and Algebraic Programming*, 70:121-150, 2007.

  A process calculus description of the SL model with pure signals.

- A., *A synchronous π-calculus. https://hal.ccsd.cnrs.fr/ccsd-00078319.* June 2006.

  The generalisation of the previous work to signals carrying data values.

Advertising (bis repetita)

- On January 8th, 15th, 22nd, 29th (Monday, last slot) there will be 4 lectures by Robin Milner on Bigraphs. Attendance is recommended. You can get 2 credits for this.
- No course on January 12th! Instead, attend a colloquium in Paris in memory of Gilles Kahn (see INRIA web page and register (freely) before 29/12).
- 3 years grant available in PPS to prepare a thesis on *Concurrency Theory with applications to Security*.