Towards a Synchronous Game Semantics

Mohamed N. Menaa
&
Dan Ghica

University of Birmingham

GaLoP
28 March 2009

* (Work in progress)
The Perfectly Synchronous Concurrency Model

Based on the *synchronous hypothesis*: concurrent processes can compute and communicate in *zero time* (on a level of abstraction).

Synchronous Languages

Computation proceeds in a sequence of atomic macro-steps (rounds) within which micro-steps are considered *simultaneous*, cyclically:

1. read the inputs
2. compute
3. produce the outputs
1 – Game Semantics is Asynchronous
Concurrent Game Semantics

Game semantics of Concurrent Algol [GM07]

- Language constants interpreted by *saturated strategies*
  - record all sequential observations of parallel interactions.

Definition
\( \sigma : A \) is saturated iff

1. If \( s_0.m_1.m_2.s_1 \in \sigma \) and \( \lambda_A(m_1) = \lambda_A(m_2) \) then \( s_0.m_2.m_1.s_1 \in \sigma \)

2. If \( s_0.p.o.s_1 \in \sigma \) and \( s_0.o.p.s_1 \in P_A \) then \( s_0.o.p.s_1 \in \sigma \)
Saturated strategies capture the intuition that in a concurrent (asynchronous) setting, some of the ordering of events in a play is arbitrary:

- Arbitrary delays on communication channels.

\[ m \parallel m' \leadsto m.m', m'.m \]
True Concurrency

In some execution models (e.g. clocked digital hardware), concurrent events are truly simultaneous.

\[ o_1 \parallel o_2 \leadsto \langle o_1, o_2 \rangle \]
2 – Synchronous Interpretations of Asynchronous Primitives
I/O Simultaneity

\[ \text{seq : com}_1 \times \text{com}_2 \Rightarrow \text{com}_3 \]

\[ R_3.R_1.D_1.R_2.D_2.D_3 \]
I/O Simultaneity

\[ \text{seq : } \text{com}_1 \times \text{com}_2 \Rightarrow \text{com}_3 \]

\[ \langle R_3, R_1 \rangle . \langle D_1, R_2 \rangle . \langle D_2, D_3 \rangle \]

In a synchronous setting:
Round Abstraction

- Given an output variable $x$ on an asynchronous module $P$, $\text{next } x$ for $P$ is the module obtained by collapsing all computational steps occurring between two changes in $x$ into a single computational step [AH99].
- Use a variant where every output in a round marker, to systematically derive synchronous strategies for primitive that have an asynchronous definitions.

Round generation

- if $s_1.o.p.s_2 \in \sigma$ then $s_1.\langle o, p \rangle.s_2 \in RA(\sigma)$
- if $s_1.p_1.p_2.s_2 \in \sigma$ then $s_1.\langle p_1, p_2 \rangle.s_2 \in RA(\sigma)$
I/O Simultaneity

\[ \text{[seq : com}_1 \times \text{com}_2 \Rightarrow \text{com}_3]\]

\[ R_3 \rightarrow \text{D}_3 \rightarrow \]
\[ \left\langle R_1, D_1 \right\rangle \]
\[ \left\langle D_2, D_3 \right\rangle \]

In a synchronous setting:
\[ \langle R_3, R_1 \rangle . \langle D_1, R_2 \rangle . \langle D_2, D_3 \rangle \]
\text{round}
O/I Simultaneity

\[
[seq : com_1 \times com_2 \Rightarrow com_3]
\]

\[
\langle R_3, R_1 \rangle \cdot \langle D_1, R_2 \rangle \cdot \langle D_2, D_3 \rangle \\
\langle R_3, R_1, D_1, R_2 \rangle \cdot \langle D_2, D_3 \rangle
\]
O/I Simultaneity

\[
\begin{align*}
\text{seq} : & \ \text{com}_1 \times \text{com}_2 \Rightarrow \text{com}_3 \\
\end{align*}
\]

\[
\langle R_3, R_1 \rangle \cdot \langle D_1, R_2 \rangle \cdot \langle D_2, D_3 \rangle \\
\langle R_3, R_1, D_1, R_2 \rangle \cdot \langle D_2, D_3 \rangle \\
\langle R_3, R_1 \rangle \cdot \langle D_1, R_2, D_2, D_3 \rangle \\
\]
O/I Simultaneity

\[ \text{seq} : \text{com}_1 \times \text{com}_2 \Rightarrow \text{com}_3 \]

\[ \langle R_3, R_1 \rangle \cdot \langle D_1, R_2 \rangle \cdot \langle D_2, D_3 \rangle \]
\[ \langle R_3, R_1, D_1, R_2 \rangle \cdot \langle D_2, D_3 \rangle \]
\[ \langle R_3, R_1 \rangle \cdot \langle D_1, R_2, D_2, D_3 \rangle \]
\[ \langle R_3, R_1, D_1, R_2, D_2, D_3 \rangle \]
Round Abstraction

▶ Given an output variable $x$ on an asynchronous module $P$, $\text{next } x$ for $P$ is the module obtained by collapsing all computational steps occurring between two changes in $x$ into a single computational step [AH99].

▶ Use a similar concept to systematically derive synchronous strategies for primitive that have an asynchronous definitions

Round generation

▶ if $s_1.o.p.s_2 \in \sigma$ then $s_1.\langle o, p \rangle.s_2 \in RA(\sigma)$

▶ if $s_1.p_1.p_2.s_2 \in \sigma$ then $s_1.\langle p_1, p_2 \rangle.s_2 \in RA(\sigma)$

Instant feedback

▶ if $s_1.p.o.s_2 \in RA(\sigma)$ then $s_1.\langle p, o \rangle.s_2 \in RA(\sigma)$

▶ if $s_1.o_1.o_2.s_2 \in RA(\sigma)$ then $s_1.\langle o_1, o_2 \rangle.s_2 \in RA(\sigma)$
Strategy Derivation Through Round Abstraction

\[ [\text{if} : (exp_1 \times com_2 \times com_3) \rightarrow com_4] \]

\[
\begin{align*}
\langle R_4, Q_1 \rangle.\langle T_1, R_2 \rangle.\langle D_2, D_4 \rangle & \xrightarrow{\text{RA}} \langle R_4, Q_1 \rangle.\langle T_1, R_2 \rangle.\langle D_2, D_4 \rangle \\
& \quad \langle R_4, Q_1, T_1, R_2 \rangle.\langle D_2, D_4 \rangle \\
& \quad \langle R_4, Q_1 \rangle.\langle T_1, R_2, D_2, D_4 \rangle \\
& \quad \langle R_4, Q_1, T_1, R_2, D_2, D_4 \rangle \\
& \quad \langle R_4, Q_1, T_1, R_2, D_2, D_4 \rangle
\end{align*}
\]
Strategy Derivation Through Round Abstraction

\[ [\text{if} : (\text{exp}_1 \times \text{com}_2 \times \text{com}_3) \rightarrow \text{com}_4] \]

\[ R_4.Q_1.F_1.R_3.D_3.D_4 \xrightarrow{RA} \langle R_4, Q_1 \rangle.\langle F_1, R_3 \rangle.\langle D_3, D_4 \rangle \]

\[ \langle R_4, Q_1, F_1, R_3 \rangle.\langle D_3, D_4 \rangle \]

\[ \langle R_4, Q_1 \rangle.\langle F_1, R_3, D_3, D_4 \rangle \]

\[ \langle R_4, Q_1, F_1, R_3, D_3, D_4 \rangle \]
3 – Synchronous Interpretations of Synchronous Primitives
Synchronous Primitives

Strategies for synchronous primitives can be formulated.
Synchronous Primitives

Strategies for synchronous primitives can be formulated.

Esterel [BMR83]
Programs typically consist of several processes composed in parallel and synchronising using signals.

- Processes: sequential threads of execution.
- Signals: broadcast events of Boolean nature.
Strategies for synchronous primitives can be formulated.

**Esterel [BMR83]**

Programs typically consist of several processes composed in parallel and synchronising using signals.

- Processes: sequential threads of execution.
- Signals: broadcast events of Boolean nature.

**Some candidates (from Esterel)**

- `pause`
- `p || q`
- `emit S`
- `present S then p else q end`
- `await S`
- `suspend p when S`
Synchronous Primitives

- ReactiveML [MP05] extends ML with such synchronous primitives by adding entities that are orthogonal to the type system.
  - Processes.
  - Signals.
Synchronous Primitives

- ReactiveML [MP05] extends ML with such synchronous primitives by adding entities that are orthogonal to the type system.
  - Processes \(\rightarrow\) strategies.
  - Signals \(\rightarrow\) moves.
ReactiveML [MP05] extends ML with such synchronous primitives by adding entities that are orthogonal to the type system.

- Processes $\rightarrow$ strategies.
- Signals $\rightarrow$ moves.
- Use start and end of computation as *signals*. 
The Semantics of `await`

```
trap T in
  loop
  pause;
  present S then exit T else nothing end
end
```
The Semantics of \texttt{await}

\begin{verbatim}
 trap \ T \ in 
  loop 
   pause; 
   present \ S \ then \ exit \ T \ else \ nothing \ end 
end
\end{verbatim}

- Variant: await the start of a command.
- A semantic version of a pointcut in Aspect-oriented Programming.
The Semantics of `await`

```
trap T in
  loop
  pause;
  present S then exit T else nothing end
end
```

- Variant: await the start of a command.
- A semantic version of a pointcut in Aspect-oriented Programming.

```
await: com ⇒ com
        ^  
        ^
        r^0
        r^p
        d
```
The Semantics of `await`

```
await: \( \text{com}_1 \Rightarrow \text{com}_2 \times \text{com}_3 \)
```
The Semantics of `await`

\[
\text{await: } com_1 \Rightarrow com_2 \times com_3
\]

\[
\langle R2, R1 \rangle, \langle D1, R2 \rangle
\]

\[
\langle R2, R1, D1, R2 \rangle
\]
The Semantics of \texttt{await}

\begin{align*}
\text{await}: \quad & \text{com}_1 \Rightarrow \text{com}_2 \times \text{com}_3 \\
& \langle \text{R}_2, \text{R}_1 \rangle, \langle \text{D}_1, \text{R}_2 \rangle \\
& \langle \text{R}_2, \text{R}_1, \text{D}_1, \text{R}_2 \rangle \\
& \langle \text{R}_3 \rangle
\end{align*}
The Semantics of \texttt{await}

\[ \text{await: } com_1 \Rightarrow com_2 \times com_3 \]

\[ \langle R2, R1 \rangle, \langle D1, R2 \rangle \]

\[ \langle R2, R1, D1, R2 \rangle \]

\[ R3, \langle R2, R1, D3 \rangle \]
The Semantics of `await`

$$\text{await: } \text{com}_1 \Rightarrow \text{com}_2 \times \text{com}_3$$

$$\langle R_2, R_1 \rangle.\langle D_1, R_2 \rangle$$
$$\langle R_2, R_1, D_1, R_2 \rangle$$

$$R_3.\langle R_2, R_1, D_3 \rangle.\langle D_1, D_2 \rangle$$
The Semantics of \texttt{await}

\[\text{await: } \text{com}_1 \Rightarrow \text{com}_2 \times \text{com}_3\]

\[
\langle R_2, R_1 \rangle, \langle D_1, R_2 \rangle \\
\langle R_2, R_1, D_1, R_2 \rangle \\
R_3, \langle R_2, R_1, D_3 \rangle, \langle D_1, D_2 \rangle \\
R_3, \langle R_2, R_1, D_3, D_1, D_2 \rangle \\
\langle R_3, R_2, R_1, D_3, D_1, D_2 \rangle \\
\langle R_3, R_2, R_1, D_3, D_1, D_2 \rangle
\]
4 – Categorical Structure
Synchronous Traces

Plays represented using *synchronous traces*.

**Definition**

A trace $t \in U$, where $U$ is an arbitrary set of traces over a set of labels $L$, is a triple $\langle E, \preceq_E, \lambda : E \rightarrow L \rangle$ where

- $E$ is a set of events,
- $\preceq_E$ is a total preorder between events signifying *temporal precedence*. The equivalence relation $\approx_E$, which means the *simultaneous occurrence* of two events, is defined as:

$$\forall a, b \in E \bullet a \preceq_E b \land b \preceq_E a \iff a \approx_E b$$

- $\lambda$ is a function mapping events to labels in a set $L$. 
Category

- Objects: sets of labels.
- Morphisms: sets of synchronous traces between sets of labels.
Definition

$U : A \rightarrow B$ and $V : B \rightarrow C$ are two arbitrary sets of synchronous traces. Their composition is a set of traces $U; V : A \rightarrow C$ defined as:

$$U; V = \{ t' \in \Theta_{A+C} \mid \exists t \in \Theta_{A+B+C} \bullet$$

$$\text{out}_{A+B}^{A+B+C}(t) \in U \land$$

$$\text{out}_{B+C}^{A+B+C}(t) \in V \land$$

$$t' = \text{out}_{A+C}^{A+B+C}(t) \}$$
Identity

**Definition**

\[ ID_A = \{ \langle E, \preceq_E, \lambda : E \to A + A \rangle \mid \exists k \in \mathbb{N} \cdot E \cong^e \{1, 2, \ldots, 2k\}, \]
\[ \forall i < 2k \cdot e(i) \preceq_E e(i + 1) \land \]
\[ (i \text{ is odd } \Rightarrow e(i) \approx_E e(i + 1)) \land \]
\[ (out_{A_1 + A_2}^{A_1 + A_2} \circ \lambda \circ e)(i) = (out_{A_2}^{A_1 + A_2} \circ \lambda \circ e)(i + 1) \} \]
Definition
A tensor is a bifunctor $\otimes : S \times S \to S$ defined as

- On objects: $A \otimes B = A + B$.
- On morphisms: $U : A \to B$, $V : C \to D$

$$U \otimes V = \{ t \in \Theta_{A+B+C+D} \mid \text{out}_{A+B}(t) \in U \land \text{out}_{C+D}(t) \in V \}$$
Definition

The arrow is a functor $\Rightarrow: S^{op} \times S \to S$ with the same definitions as $\otimes$. In a polarised setting, its definitions are:

- On objects: $A \Rightarrow B = B + A^*$
- On morphisms: $U \Rightarrow V = V \otimes U^*$

where * reverses the I/O polarities of labels.
Evaluation

Definition
Eval is a morphism $\text{eval}_{A,B} : A \otimes (A \Rightarrow B) \to B$ that satisfies the following universal property: for every morphism $f : A \otimes X \to B$ in $S$ there exists a unique morphism $h : X \to A \Rightarrow B$ such that $f = \text{eval}_{A,B} \circ (\text{ID}_A \otimes h)$. It is defined as:

$$\text{eval}_{A,B} = \{ t \in \Theta_{A_1+A_2+B_1+B_2} \mid \text{out}_{A_1+A_2}(t) \in \text{ID}_{A_1+A_2} \land \text{out}_{B_1+B_2}(t) \in \text{ID}_{B_1+B_2} \}$$
∀f : A ⊗ X → B, !∃h : X → A ⇒ B such that:

\[ f = eval_{A,B} \circ (id_A \otimes h) \]
\[ \forall f : A \otimes X \rightarrow B, \exists h : X \rightarrow A \Rightarrow B \text{ such that:} \]

\[ f = \text{eval}_{A,B} \circ (id_A \otimes h) \]

(Compact) Closed monoidal category
Outlook

- Closed monoidal category provides the right structural properties.
- Extend it with Cartesian product.
- Definability as a test for the choice of primitives.
Outlook

- Closed monoidal category provides the right structural properties.
- Extend it with Cartesian product.
- Definability as a test for the choice of primitives.

THANKS!

