The Meaning and Implementation of SKIP in CSP

Thomas Gibson-Robinson and Michael Goldsmith

Department of Computer Science, University of Oxford

August 25, 2013

Introduction

CSP has long had a method of composing processes *sequentially*. In particular, the process $P\,;Q$ runs P until it *terminates* at which point Q is run.

There has been some debate over the correct termination semantics, with two main definitions:

- $\checkmark$*-as-Refusal* semantics, as developed by Hoare.
- $\checkmark$*-as-Signal*, as developed by Roscoe.

Defining Termination in CSP

Ω is the process that has terminated. It can perform no events.

SKIP is the process that terminates immediately. In CSP, termination is indicated using the event $\checkmark$ and thus SKIP is defined as $\checkmark \rightarrow \Omega$.

The operational semantics rules of the *sequential composition* operator ; are:

$$\frac{P \xrightarrow{a} P'}{P \,;\, Q \xrightarrow{a} P' \,;\, Q}\ a \in \Sigma \cup \{\tau\} \qquad \frac{P \xrightarrow{\checkmark} \Omega}{P \,;\, Q \xrightarrow{\tau} Q}$$

Termination and the Standard CSP Operators

We also need to define how the standard CSP operators respond to one of their arguments offering a $\checkmark$.

$\rightarrow$ and $\sqcap$ have no **on** arguments, so cannot terminate.

$[\![\cdot]\!]$, $\setminus \cdot$, $\Theta_\cdot$ and $\triangleright$ only have one **on** argument, so terminate when their argument does:

$$\frac{P \xrightarrow{\checkmark} \Omega}{P \setminus A \xrightarrow{\checkmark} \Omega}$$

Termination and the Standard CSP Operators

The more interesting case concerns operators that have more than one **on** argument.

- Operators that terminate *Independently* terminate when *either* of their arguments terminate. $\Box$ and $\triangle$ are defined as terminating Independently. Thus:

$$\frac{P \xrightarrow{\checkmark} \Omega}{P \Box Q \xrightarrow{\checkmark} \Omega} \qquad \frac{Q \xrightarrow{\checkmark} \Omega}{P \Box Q \xrightarrow{\checkmark} \Omega}$$

- Operators that *Synchronise* their termination terminate when *all* of their arguments terminate. All CSP parallel operators have Synchronising termination semantics. The operational semantics of operators with Synchronising termination semantics varies.

$\checkmark$-as-Refusal

This semantics treats $\checkmark$ as a standard visible event. This means that the process $SKIPChoice_a \mathrel{\widehat{=}} SKIP \mathbin{\Box} a \rightarrow STOP$ can either perform an a or a $\checkmark$ and the environment is free to choose.

Thus, the termination operational semantics of operators with Synchronising termination semantics can be defined as follows:

$$\frac{P \xrightarrow{\checkmark} \Omega \wedge Q \xrightarrow{\checkmark} \Omega}{P \mathbin{|||} Q \xrightarrow{\checkmark} \Omega}$$

$\checkmark$-as-Signal

Under the $\checkmark$-as-Signal semantics, $\checkmark$ is treated as a communication to the environment that cannot be refused. Thus, the termination operational semantics of operators with Synchronising termination semantics are as follows:

$$\frac{P \xrightarrow{\checkmark} \Omega}{P \mid\mid\mid Q \xrightarrow{\tau} \Omega \mid\mid\mid Q} \qquad \frac{Q \xrightarrow{\checkmark} \Omega}{P \mid\mid\mid Q \xrightarrow{\tau} P \mid\mid\mid \Omega} \qquad \frac{}{\Omega \mid\mid\mid \Omega \xrightarrow{\checkmark} \Omega}$$

The most important difference is in how the failures of processes are calculated.

Denotational Semantics

The failures of a process represent what a process is allowed to refuse having performed a certain sequence of events.

$$\mathcal{F}^r(P) \mathrel{\widehat{=}} \{(tr, X) \mid \exists\, Q \cdot P \xRightarrow{tr} Q \land X \subseteq \Sigma \cup \{\checkmark\} \land Q\ \mathbf{ref}\ X\}$$

where $Q\ \mathbf{ref}\ X$ iff Q is stable (i.e. $Q \stackrel{\tau}{\not\longrightarrow}$), and, $\forall\, x \in X \cdot Q \stackrel{x}{\not\longrightarrow}$.

$$\mathcal{F}^s(P) \mathrel{\widehat{=}} \mathcal{F}^r(P) \cup \{(tr, X) \mid P \xRightarrow{tr ^\frown \langle \checkmark \rangle} \Omega, X \subseteq \Sigma\}$$

Hence, for $SKIPChoice_a$ ($SKIP \,\Box\, a \to STOP$) with $\Sigma = \{a\}$:

$$\mathcal{F}^r(SKIPChoice_a) = \{(\langle\rangle, \{\}), (\langle a \rangle, \{a, \checkmark\}), (\langle \checkmark \rangle, \{a, \checkmark\})\}$$
$$\mathcal{F}^s(SKIPChoice_a) = \{(\langle\rangle, \{\}), (\langle a \rangle, \{a, \checkmark\}), (\langle \checkmark \rangle, \{a, \checkmark\})\}$$
$$\cup \{(\langle\rangle, \{a\})\}$$

Thus, under $\checkmark$-as-Signal, $SKIPChoice_a = a \to STOP \rhd SKIP$.

Simulating ✓-as-Signal

Consider $SKIPChoice_a \mid\mid\mid STOP$. Under ✓-as-Refusal this is equal to $a \to STOP$, but under ✓-as-Signal this is equal to $a \to STOP \triangleright STOP = a \to STOP \sqcap STOP$.

Simulating ✓-as-Signal

Consider $SKIPChoice_a \mid\mid\mid STOP$. Under ✓-as-Refusal this is equal to $a \rightarrow STOP$, but under ✓-as-Signal this is equal to $a \rightarrow STOP \rhd STOP = a \rightarrow STOP \sqcap STOP$.

Let τ_r be a fresh event and define $BSkip \mathrel{\widehat{=}} \tau_r \rightarrow \checkmark \rightarrow \Omega$.

We define the operational semantics of ; on τ_r by:

$$\frac{P \xrightarrow{\tau_r} P'}{P \mathbin{;} Q \xrightarrow{\tau} P' \mathbin{;} Q}$$

All other operators are defined as treating τ_r exactly like any other event in Σ. In particular, observe that:

$$(BSkip \mathrel{\Box} a \rightarrow STOP) \setminus \{\tau_r\} = a \rightarrow STOP \rhd SKIP.$$

Simulating ✓-as-Signal

We can define our simulation as:

$$
\begin{aligned}
Sig(SKIP) &\mathrel{\widehat=} BSkip \\
Sig(STOP) &\mathrel{\widehat=} STOP \\
Sig(a \rightarrow P) &\mathrel{\widehat=} a \rightarrow Sig(P) \\
Sig(P \mathbin{\Box} Q) &\mathrel{\widehat=} Sig(P) \mathbin{\Box} Sig(Q) \\
Sig(P \,;\, Q) &\mathrel{\widehat=} Sig(P) \,;\, Sig(Q) \\
Sig(P \mathbin{|||} Q) &\mathrel{\widehat=} (Sig(P) \,;\, BSkip) \underset{\{\tau_r\}}{\parallel} (Sig(Q) \,;\, BSkip)
\end{aligned}
$$

Theorem

$\mathcal{F}^s(P) = \mathcal{F}^r(Sig(P) \setminus \{\tau_r\})$.

Proof (!) by Example

$$
\begin{aligned}
&Sig(SKIPChoice_a \,|||\, STOP)\\
&= (a \to STOP \,\Box\, BSkip) \,;\, BSkip \underset{\{\tau_r\}}{\|} (STOP \,;\, BSkip)\\
&= (a \to STOP \,\Box\, BSkip) \,;\, BSkip \underset{\{\tau_r\}}{\|} STOP.
\end{aligned}
$$

The interesting bit concerns the left hand side:

$$
\begin{aligned}
&(a \to STOP \,\Box\, BSkip) \,;\, BSkip\\
&= a \to STOP \triangleright BSkip.
\end{aligned}
$$

Thus $Sig(SKIPChoice_a \,|||\, STOP) \setminus \{\tau_r\} = a \to STOP \triangleright STOP$.

Simulation Efficiency

FDR has a specialised representation of labelled-transition systems known as *high-level machines*.

For example, a high-level machine for $P \mid\mid\mid Q$ has rules:

$$\begin{array}{ll} (a, \_) \mapsto a & a \in \alpha P \\ (\_, a) \mapsto a & a \in \alpha Q \end{array}$$

The rules can also be organised into *formats*. For example, the rules for $P \mathbin{;} Q$ are divided into two formats. The first specifies how the transitions of P are promoted:

$$\begin{array}{ll} (a, \_) \mapsto a & a \in \alpha P, a \neq \checkmark \\ (\checkmark, \_) \mapsto \tau \wedge \text{move to format 2} & \end{array}$$

The second format simply has the rules:

$$\begin{array}{ll} (\_, a) \mapsto a & a \in \alpha Q \end{array}$$

Supercompilation

FDR also combines together the rules for high-level machines in a process known as *supercompilation*. For example, the process $(P \mathbin{|||} Q) \mathbin{|||} R$ is not represented as two high-level machines, but as one with the rules:

$$
\begin{aligned}
&(a, \_, \_) \mapsto a && a \in \alpha P \\
&\ldots
\end{aligned}
$$

However, this means that:

$$(P_1 \,;\, Q_1) \mathbin{|||} \ldots \mathbin{|||} (P_N \,;\, Q_N)$$

has 2^N formats.

Impact on the Simulation

Recall that $Sig(P \mathbin{|||} Q) = (Sig(P)\,;BSkip) \mathbin{|||} (Sig(Q)\,;BSkip)$ and thus the simulation of $P_1 \mathbin{|||} \ldots \mathbin{|||} P_N$ will have 2^N formats.

However, we only need to apply the simulation to processes that contain a choice between a $\checkmark$ and a visible event.

We can predict which processes contain a choice between a $\checkmark$ and a visible event by using a structural definition that identifies which processes can immediately perform a $\checkmark$.

Some care has to be taken in order to correctly consider processes such as $(a \to SKIP \setminus Y) \mathbin{\Box} b \to STOP$: this requires the simulation to be applied iff $a \in Y$.

Summary

- We have developed a way of simulating $\checkmark$-as Signal under the $\checkmark$-as Refusal semantics.
- We have developed a way of statically identifying which processes the simulation *has* to be applied to, in order to improve the performance of the simulation.