Calculus of Communicating Systems

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

Table of Contents

A sprinkle of linguistics

We will often face two linguistic concepts that every programmer ought to know

- syntax - determines whether a sentence is valid or not
- semantics - the meaning of valid sentences

Example (syntax)

The sentence (program) $\texttt{x} := \texttt{p}\,;\texttt{q}$ is forbidden by the syntactic rules of most programming languages

Example (semantics)

The sentence (program) $\texttt{x} := 1$ has the meaning "writes 1 in the memory address corresponding to $\texttt{x}$"

The need for semantics in programming

How can one prove that a program does what is supposed to do if its semantics (i.e. its meaning) is not established *a priori*?

Examples

What are the outputs of the following programs?

- $f()\{\texttt{print}\ a; \texttt{ret.}\ 1\};\ g()\{\texttt{print}\ b; \texttt{ret.}\ 0\};\ x := f() + g()$
- $x := 2\ ;(x := x + 1 \parallel x := 0)$
- $(a := 1\,;\texttt{print}\ b) \parallel (b := 1\,;\texttt{print}\ a)$

Transition systems as semantic providers

Transition systems are an ubiquitous mechanism for defining the semantics of programming languages

Following tradition, we will use them to define the semantics of a simple (but powerful !!) concurrent language —

and then base on this learning step to tackle Dijkstra's

Dining Philosophers Problem (*circa* 1965)

Table of Contents

Calculus of Communicating Systems

Syntax

$$P, Q ::= X \mid a.P \mid \sum_{i \in I} P_i \mid P \parallel Q \mid P \backslash L \mid \ldots$$

- X is a process name
- $a.P$ communicates via channel a and proceeds as P
- $\sum_{i \in I} P_i$ non-deterministic choice between processes P_i
- $P \parallel Q$ parallel composition between processes P and Q
- $P \backslash L$ makes channels in L private 'outside' of P

First Steps with CCS

Conventions

- $0 = \sum_{i \in \emptyset} P_i$ (denotes a terminating process)
- $\bar{a}$ denotes outgoing information via channel a
- τ denotes an invisible action

Examples (processes in CCS)

- $a.0 \parallel \bar{a}.0$ - two processes connected via channel a; information flows in one direction
- $a.\bar{b}.0 \parallel \bar{a}.b.0$ - info. flows in one direction via a and then in the inverse direction via b
- $(a.\bar{b}.0 \parallel \bar{a}.b.0) \backslash \{a, b\}$ - both channels a, b now private

First steps with CCS

Which of these expressions are valid sentences in CCS?

1. $a.b.P + Q$
2. $a + b$
3. $P.a$
4. $(P + Q).a$
5. $a.0 + b.0$
6. $P.Q$

CCS and Cyclic Behaviour

We now add the construct $\texttt{rec}\ X.\ P$ to the syntax of CCS – so that we can describe cyclic behaviour

Example

$\texttt{rec}\ X.\ a.b.X$ - receive communication through a and then through b; after that repeat protocol

Example (the coffee machine and the student)

$(\texttt{rec}\ X.\ coin.\overline{coffee}.X) \mid\mid (\texttt{rec}\ Y.\ \overline{coin}.coffee.\overline{wrk}.Y)$

Write down a coffee machine that fails to deliver coffee sometimes

Semantics of CCS

Every process yields a transition system according to the rules

$$\frac{}{a.P \xrightarrow{a} P}\ (\textbf{pr}) \qquad \frac{P_i \xrightarrow{a} Q}{\sum_{i\in I} P_i \xrightarrow{a} Q}\ (\textbf{ch}) \qquad \frac{P \xrightarrow{a} P'}{P\backslash L \xrightarrow{a} P'\backslash L}\ (\textbf{res})\, a, \overline{a} \notin L$$

$$\frac{P \xrightarrow{a} P'}{P \parallel Q \xrightarrow{a} P' \parallel Q}\ (\textbf{coml}) \qquad \frac{Q \xrightarrow{a} Q'}{P \parallel Q \xrightarrow{a} P \parallel Q'}\ (\textbf{comr})$$

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\overline{a}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}\ (\textbf{com}) \qquad \frac{P[\texttt{rec}\ X.\ P/X] \xrightarrow{a} P'}{\downarrow \texttt{rec}\ X.\ P \xrightarrow{a} P'}\ (\textbf{rec})$$

Substitution of X in P by $\texttt{rec}\ X.\ P$

First steps with CCS

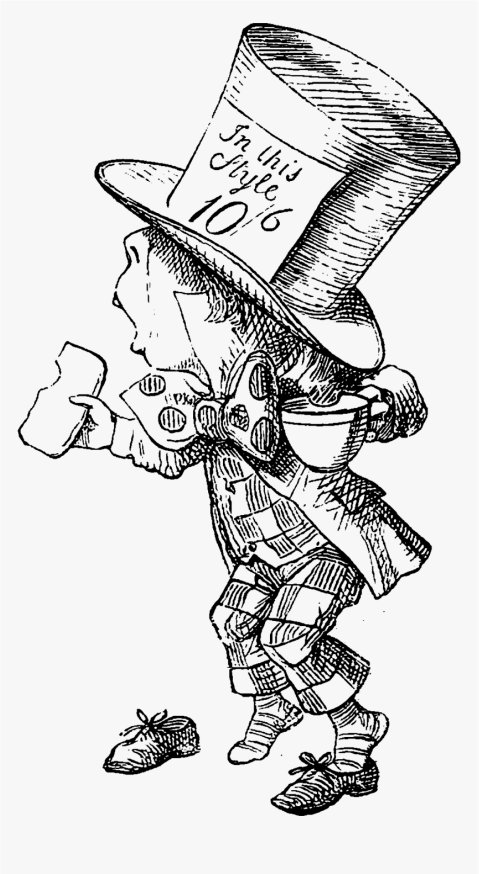
What are the semantics of the following processes?

1. $a.b.0$
2. $a.b.0 + c.d.0$
3. $a.b.0 \parallel c.d.0$
4. $\texttt{rec}\ X.\ a.b.X$

Table of Contents

CCS at Work

With the syntax and semantics of CCS now in place, we put on our working hats and start to formally analyse communication and synchronisation mechanisms



Starvation and Mutual Exclusion in CCS

We define three recursive processes

$$S = \mathtt{rec}\, X.\ \overline{start}.finish.X \qquad \text{(the semaphore)}$$

$$P_1 = \mathtt{rec}\, Y.\ start.a_1.b_1.\overline{finish}.Y \qquad \text{(process 1)}$$

$$P_2 = \mathtt{rec}\, Z.\ start.a_2.b_2.\overline{finish}.Z \qquad \text{(process 2)}$$

and then write down $(S \parallel P_1 \parallel P_2)\backslash\{start, finish\}$

Question: will we ever observe a sequence of actions $x_1 \ldots x_n \ldots$ such that $x_i = a_1$ and $x_{i+1} = a_2$?

think of a_i as writing on a critical region and of b_i as ending this process

Dining Philosophers Problem

Two philosophers sitting at the table in front of each other . . . thinking . . .

They will wish to eat and for that effect there are precisely two forks on the table, at their left and right-hand sides

When Philosopher 1 wishes to eat he picks the fork on his left and then the one on his right

Phil. 2 picks the fork on her left and then the fork on her right

Write down this system in CCS and discover whether it is possible that both philosophers can no longer eat

Going beyond the Dining Philosophers Problem ...

Detection of both deadlocks and livelocks *e.g.* in

- Driving systems
- Pacemakers
- at the LHC
- ...

See details at `https://www.mcrl2.org/web/index.html`

. . . and beyond CCS

Different extensions of CCS to the

- probabilistic
- quantum
- and timed

domains, among others

Stay tuned!


Robin Milner, *A calculus of communicating systems*, Springer, 1980.