Notes and Exercises

Marco Peressotti

Week 38

The term *process algebra* is used in different meanings and often interchangeably with *process calculus*. First, the term "process" refers to the observable behaviour or dynamics of a system under scrutiny. Here a system can be anything that can be observed like the execution of a program, a chemical process, a physical machinery *etc.* To precisely describe system dynamics in its entirety is challenging at best. Instead, systems are modelled abstracting information that is not relevant to the current analysis or properties of interest. What is left is what is referred as observable. For instance, when discussing deadlock it is not relevant whether processes are awaiting for a message, a resource, or a memory location to hold a certain value. What is relevant instead is that they are waiting for events that will never occur. The observable events (or just observations) that a system exhibits are called its (observable) behaviour. In this sense, processes are sometimes also called *discrete event systems*. It is important to stress that what is relevant is not absolute but rather depends on the specific scenario and property modelled *e.g.* to know the possibility or the likelihood of deadlock.

The term "algebra" denotes an algebraic/axiomatic approach to system modelling. Intuitively this means to have operations and axioms (algebraic laws) for building processes, reasoning about them, and in general performing calculations. Additionally, operations and terms freely generated from them provide a syntax for system behaviours.

1 Exercises

1.1 Parallel composition and synchronisation

Fix a set $\mathbb{C}$ of channel names. Define a process as any (ground) term generated by the grammar

$$P, Q ::= \mathbf{0} \mid c.P \mid \overline{c}.P \mid P \parallel Q$$

where $c \in \mathbb{C}$.[1] Intuitively, the term $\mathbf{0}$ represents a terminated system that does not exhibit any event; terms $c.P$ and $\overline{c}.P$ represent processes that respectively send and receive over channel c before continuing as P; the term $P \parallel Q$ represents a process composed by two sub-processes P and Q running in parallel. This intuitive behaviour is formalised by means of a *reduction semantics i.e.* a relation $(\twoheadrightarrow)$ assigning to each process its possible evolutions. Additionally, algebraic

[1] That is any string obtained starting from P (or Q) and iteratively replacing any occurrence of symbols P and Q with any of the strings $\mathbf{0}$, $c.P$, $\overline{c}.P$, and $P \parallel Q$ until there are no occurrences left.

laws (*e.g.* commutativity of operators) are formalised in terms of a structural congruence ($\equiv$) *i.e.* an equivalence relation on processes that is preserved by all operations ($\parallel$ *etc.*). The reduction relation $\twoheadrightarrow$ and the structural congruence $\equiv$ are defined as the least relations closed under the following rules.

$$\frac{}{c.P \parallel \overline{c}.Q \twoheadrightarrow P \parallel Q}\lfloor\text{Com}\rceil \qquad \frac{P \twoheadrightarrow P'}{P \parallel Q \twoheadrightarrow P' \parallel Q}\lfloor\text{Par}\rceil$$

$$\frac{P \equiv Q \qquad Q \twoheadrightarrow Q' \qquad Q' \equiv P'}{P \twoheadrightarrow P'}\lfloor\text{Str}\rceil \qquad \frac{}{P \parallel Q \equiv Q \parallel P}\lfloor\text{PC}\rceil$$

$$\frac{}{(P \parallel Q) \parallel R \twoheadrightarrow P \parallel (Q \parallel R)}\lfloor\text{PA}\rceil$$

Rule $\lfloor$Com$\rceil$ formalise how two processes running in parallel synchronise by communicating over the channel c and reduce to the parallel composition of their respective continuations (P and Q). Rule $\lfloor$Par$\rceil$ formalise the fact that parallel processes can reduce independently. Rule $\lfloor$Str$\rceil$ allows derivation of reductions from structurally equivalent processes. Rules $\lfloor$PC$\rceil$ and $\lfloor$PA$\rceil$ assert that the operator $\parallel$ satisfy the algebraic laws of commutativity and associativity, respectively. Rules stating that $\equiv$ is a structural congruence are standard and usually omitted. For the sake of precision, these are listed below.

$$\frac{}{P \equiv P} \qquad \frac{P \equiv Q \qquad Q \equiv R}{P \equiv R} \qquad \frac{P \equiv Q}{c.P \equiv c.Q} \qquad \frac{P \equiv Q}{\overline{c}.P \equiv \overline{c}.Q}$$

$$\frac{P \equiv P' \qquad Q \equiv Q'}{P \parallel Q \equiv P' \parallel Q'}$$

1. Write all reductions of $\mathbf{0}$, $(c.a.\mathbf{0} \parallel \overline{c}.b.\mathbf{0})$, and $(c.a.\mathbf{0} \parallel \overline{c}.b.\mathbf{0} \parallel \overline{c}.c.\mathbf{0})$.

2. Show that the symmetric of rule $\lfloor$Par$\rceil$

   $$\frac{Q \twoheadrightarrow Q'}{P \parallel Q \twoheadrightarrow P \parallel Q'}\lfloor\text{Par}'\rceil$$

   is derivable *i.e.* show a proof of $P \parallel Q \twoheadrightarrow P \parallel Q'$ from $Q \twoheadrightarrow Q'$ without using rule $\lfloor$Par$'\rceil$.

3. Is there a process U with the property that

   $$P \twoheadrightarrow Q \iff U \parallel P \twoheadrightarrow Q$$

   for any P and Q?

4. If there is U as above then, U acts as a unit for the operator $\parallel$. State this fact in terms of algebraic laws and extend $\equiv$ accordingly.

5. What kind of algebraic structure (magma, monoid, group, ring, *etc.*) is $(Proc, \parallel, \mathbf{0})$?

1.2 Progress and termination

A process P is said "to admit a reduction" ("to reduce", for short) whenever there is a process Q (not necessarily distinct from P) such that $P \twoheadrightarrow Q$. Write $P \twoheadrightarrow$

or $P \not\rightarrow$ to denote whether P admits a reduction or not *i.e.*:

$$P \rightarrow \overset{\triangle}{\iff} \exists Q(P \rightarrow Q) \qquad P \not\rightarrow \overset{\triangle}{\iff} \nexists Q(P \rightarrow Q).$$

1. Is it appropriate to say that if $P \not\rightarrow$ then P is terminated?
2. Which processes among $\mathbf{0}$, $\mathbf{0} \parallel \mathbf{0}$, $c.\mathbf{0}$, $c.\mathbf{0} \parallel \overline{c}.\mathbf{0}$ reduces and which are terminated?
3. Define "terminated" and "deadlock" in this setting.

A process P may terminate iff it can reduce to a process that is terminated. Likewise for deadlock.

Write $\rightarrow^*$ for the transitive and reflective closure of the reduction relation $\rightarrow$ *i.e.* the smallest relation such that

$$\frac{}{P \rightarrow^* P} \qquad \frac{P \rightarrow^* Q \qquad Q \rightarrow R \qquad R \rightarrow^* S}{P \rightarrow^* S}$$

4. Formally define "P may terminate" and "P may reach a deadlock".

In the sequel various extensions of the syntax (signature) and semantics of CCS processes ($\rightarrow$ and $\equiv$) will be discussed. Given the crucial role played by $\mathbf{0}$, any extension is required to preserve its absence of reductions. Reworded, when adding new reduction or congruence rules to the definition of $\rightarrow$ and $\equiv$, always check that $\mathbf{0} \not\rightarrow$.

1.3 Restriction

Communication channels in CCS are global *i.e.* any process can access any channel. As a consequence, termination and absence of deadlocks are not preserved by parallel composition. For instance, $(a.\mathbf{0} \parallel \overline{a}.\mathbf{0})$ terminates whereas its parallel composition with $a.R$ does not. To this end, CCS is extended with a *restriction operator* that intuitively prevents any interaction on a restricted channel between its argument any other parallel process.

Formally, the syntax and semantics of CCS processes are extended adding the following production rule to the grammar:

$$P ::= \cdots \mid P \setminus L \qquad \text{for } L \subseteq \mathbb{C}$$

and the following derivation rule to the definition of the reduction relation $\rightarrow$:

$$\frac{P \rightarrow P'}{P \setminus L \rightarrow P' \setminus L} \lfloor \text{H0} \rceil$$

1. Derive all reductions for the process $(a.b \parallel \overline{a}.c) \setminus \{a\} \parallel \overline{a}.\overline{b}$.
2. Does rule $\lfloor \text{H0} \rceil$ capture the intuitive behaviour of the restriction operator?
3. Define a function $\text{gc}\colon \mathit{Proc} \rightarrow \wp(\mathbb{C})$ mapping each process to the set of channels it uses *e.g.* $\text{gc}(\overline{c}.c.\mathbf{0}) = \{c\}$, $\text{gc}((\overline{c}.\overline{a}.\mathbf{0}) \setminus \{c\}) = \{a\}$ (proceed by induction on the argument structure)

Consider the function gc: $Proc \to \wp(\mathbb{C})$ given Add to the definition $\equiv$ the following laws concerning redundant restrictions:

$$\frac{}{P \setminus \emptyset \equiv P}\lfloor \text{H1} \rceil \qquad \frac{L' \not\subseteq \text{gc}(P)}{P \setminus L \equiv P \setminus (L \cup L')}\lfloor \text{H2} \rceil$$

4. Is $\equiv$ a congruence?
5. Is any law for $\|$ violated?
6. Does $\mathbf{0}$ reduce?
7. Is $(-) \setminus L$ idempotent?
8. Does $(a.P) \setminus \{a\} \equiv \mathbf{0}$ hold? Is this law meaningful (think about termination and deadlocks).