◇ Interaction ◇

Up to now we have described simple processes in isolation. Although we have often assumed that our processes might be placed in some environment and expected to interact with it — for example, there should be a customer who will use the ticket machine — this environment has not been made explicit.

We will now see how to take two (or more) processes and force them to interact with each other. Interaction between two processes means that they simultaneously perform events; an event thus becomes a joint activity in which two (or more) processes may participate.

When placing processes in parallel so that they can interact, it is important to specify which events they are supposed to be interacting on, or sharing. This is where alphabets (interfaces) come into play.

If the interfaces of processes P and Q are A and B respectively, then the process

$$P \;{}_A\|_B\; Q$$

is a parallel combination of P and Q.

In this combination, P can only perform events in A, Q can only perform events in B, and any events in the intersection of A and B require synchronisation between P and Q.

The interface of P should contain at least all the events used in the definition of P, and similarly for the interface of Q.

Example: Consider processes representing a vending machine, and a customer:

$$\begin{aligned} \mathit{VM} &= \mathit{coin} \rightarrow (\mathit{choc} \rightarrow \mathit{Stop} \mid \mathit{toffee} \rightarrow \mathit{Stop}) \\ \mathit{CUST} &= \mathit{coin} \rightarrow \mathit{choc} \rightarrow \mathit{Stop} \end{aligned}$$

with $\alpha \mathit{VM} = \alpha \mathit{CUST} = \{\mathit{coin}, \mathit{choc}, \mathit{toffee}\} = A$.

The process $\mathit{VM} \;{}_A\|_A\; \mathit{CUST}$ models the interaction of the customer with the machine. How does it behave? Any event done by $\mathit{VM} \;{}_A\|_A\; \mathit{CUST}$ must be an event which is done simultaneously by both *VM* and *CUST*.

At the first step, both *VM* and *CUST* can do the event *coin*. We therefore expect $\mathit{VM} \;{}_A\|_A\; \mathit{CUST}$ to do *coin*. Subsequently, *VM* and *CUST* enter new states which continue to interact.

After the event *coin*, *VM* becomes

$$choc \rightarrow Stop \mid toffee \rightarrow Stop$$

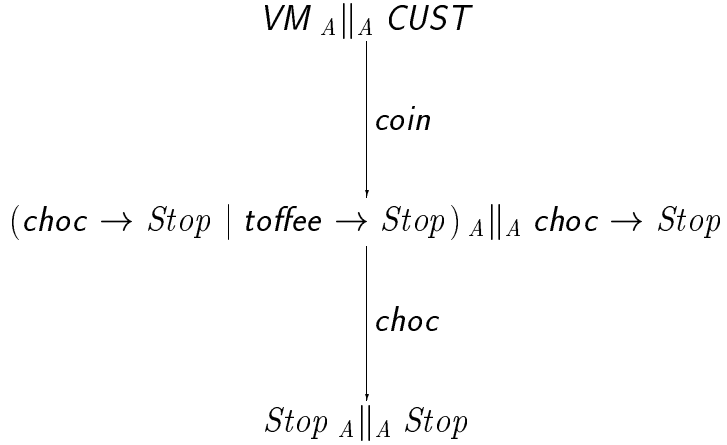
and *CUST* becomes

$$choc \rightarrow Stop.$$

Synchronisation is still required for all events, and therefore only *choc* can happen. The choice between *choc* and *toffee* in *VM* is resolved in favour of *choc*.

After the event *choc*, both processes become $Stop$, so the system becomes $Stop \;{}_A\|_A\; Stop$, which cannot do anything else.

We can draw a transition diagram for $VM \;{}_A\|_A\; CUST$.



In this example, both *VM* and *CUST* continued to the end of their potential behaviour. This may not happen in general: if we change the definition to

$$CUST = coin \rightarrow Stop$$

then after the event *coin* we get

$$(choc \rightarrow Stop \mid toffee \rightarrow Stop) \;{}_A\|_A\; Stop$$

and nothing further can happen. Although one of the processes could do either *choc* or *toffee*, both of these events require synchronisation with the other process; but because $Stop$ cannot do anything, synchronisation is not possible.

Example: Recall the definition of *STUDENT*:

$$\begin{aligned} STUDENT &= year1 \rightarrow (pass \rightarrow YEAR2 \\ &\quad \mid fail \rightarrow STUDENT) \\ YEAR2 &= year2 \rightarrow (pass \rightarrow YEAR3 \\ &\quad \mid fail \rightarrow YEAR2) \\ YEAR3 &= year3 \rightarrow (pass \rightarrow graduate \rightarrow Stop \\ &\quad \mid fail \rightarrow YEAR3) \end{aligned}$$

We will now explicitly state that the alphabet is

$$\begin{aligned} \alpha STUDENT = \{&year1, year2, year3, \\ &pass, fail, graduate\} \end{aligned}$$

which we will abbreviate to S.

Suppose that the student has a generous parent, who buys a present every time the student passes the exams.

$$PARENT = pass \rightarrow present \rightarrow PARENT$$

Again we explicitly define the alphabet:

$$\alpha PARENT = \{pass, present\} = P.$$

Notice that the event *pass* now has two different interpretations. For the student it means passing the exams, but for the parent it means seeing the student pass the exams.

We can now consider the parallel combination of the student and the parent:

$$STUDENT \ {}_S\|_P \ PARENT.$$

Synchronisation is required for the event *pass*, which is the only event in both alphabets. The other events can happen independently.

The behaviour of this system will be explored in the lab session.

◇ More Processes ◇

Any number of processes can be put in parallel, by using the $\|$ operator repeatedly.

Example: Suppose the student has a tutor who is annoyed by failure.

$$TUTOR = fail \rightarrow shout \rightarrow TUTOR$$

$$\alpha TUTOR = \{fail, shout\} = T$$

We can add the tutor to the system consisting of the student and the parent.

$$(STUDENT \ {}_S\|_P \ PARENT) \ {}_{S \cup P}\|_T \ TUTOR$$

As before, *pass* must be synchronised between *STUDENT* and *PARENT*. Also, *fail* (which is the only event in both $S \cup P$ and T) must be synchronised between $STUDENT \ {}_S\|_P \ PARENT$ and *TUTOR*.

We know that *fail* events come from *STUDENT* not *PARENT*, so in effect this means that *pass* must be synchronised between *STUDENT* and *PARENT*, and *fail* must be synchronised between *STUDENT* and *TUTOR*.

◇ More Synchronisation ◇

Some parallel combinations require some events to be synchronised between more than two processes.

Example: If a student completes the degree programme without failing at all, then the college awards a prize.

$$\begin{aligned} \mathit{COLLEGE} &= \mathit{fail} \rightarrow \mathit{Stop} \mid \mathit{pass} \rightarrow \mathit{C1} \\ \mathit{C1} &= \mathit{fail} \rightarrow \mathit{Stop} \mid \mathit{pass} \rightarrow \mathit{C2} \\ \mathit{C2} &= \mathit{fail} \rightarrow \mathit{Stop} \mid \mathit{pass} \rightarrow \mathit{prize} \rightarrow \mathit{Stop} \end{aligned}$$

$$\alpha\,\mathit{COLLEGE} = \{\mathit{pass}, \mathit{fail}, \mathit{prize}\} = C$$

Now we can consider combinations of *STUDENT* with any or all of *PARENT*, *TUTOR* and *COLLEGE*. If we combine everything:

$$((\mathit{STUDENT} \;{}_S\|_P\; \mathit{PARENT}) \;{}_{S\cup P}\|_T\; \mathit{TUTOR}) \;{}_{S\cup P\cup T}\|_C\; \mathit{COLLEGE}$$

then *pass* must be synchronised between *STUDENT*, *PARENT* and *COLLEGE*, and so on.

Consider the processes *PASS* ("passenger") and *TICKETS*, both with alphabet

$$A = \{\mathit{ashford}, \mathit{staines}, \mathit{feltham}, \mathit{ticket}, \mathit{pound}\}$$

defined by

$$\begin{aligned} \mathit{PASS} &= \mathit{ashford} \rightarrow \mathit{pound} \rightarrow \\ &\quad (\mathit{ticket} \rightarrow \mathit{PASS} \\ &\quad \mid \mathit{pound} \rightarrow \mathit{ticket} \rightarrow \mathit{PASS}) \\ &\quad \mid \mathit{feltham} \rightarrow \mathit{pound} \rightarrow \mathit{ticket} \rightarrow \mathit{Stop} \\ \mathit{TICKETS} &= \mathit{staines} \rightarrow \mathit{pound} \rightarrow \\ &\quad \mathit{ticket} \rightarrow \mathit{TICKETS} \\ &\quad \Box\ \mathit{ashford} \rightarrow \mathit{pound} \rightarrow \mathit{pound} \rightarrow \\ &\quad \mathit{ticket} \rightarrow \mathit{TICKETS} \end{aligned}$$

△ What is the behaviour of $\mathit{TICKETS} \;{}_A\|_A\; \mathit{PASS}$? Draw a transition diagram.

Given a transition diagram, it is possible to define a process, without using the parallel operator, which has the same transition diagram.

△ Do this for $\mathit{TICKETS} \;{}_A\|_A\; \mathit{PASS}$.