◇ Input and Output ◇

So far we have treated all events in the same way, regardless of whether they are thought of as inputs or outputs. It is useful, however, to introduce separate notation for inputs and outputs.

We will use events of the form $c.v$ where c is the name of a *channel* and v is the *value* of a message passing along the channel. Each channel has a *type*, which is simply the set of possible values which can be transmitted along it. If the type of c is T, then the set of events associated with c is $\{c.t \mid t \in T\}$.

We can define two new forms of prefixing. The process $c!v \to P$ outputs the message v on the channel c and then behaves like P. We require $v \in T$, where T is the type of c. In fact, $c!v \to P = c.v \to P$ (using the ordinary prefix notation), but the $c!v$ notation emphasises the fact that c and v are viewed as a channel and a message.

The process $c?x : T \to P(x)$ is prepared to input any value x of type T, and then behave like $P(x)$. In the ordinary menu choice notation,

$$c?x : T \to P(x) =$$
$$y : \{c.z \mid z \in T\} \to P(message(y)),$$

where, if $y = c.z$, $message(y) = z$.

We can define input and output prefixes, using labelled transition rules, as follows.

$$\frac{}{(c!v \to P) \xrightarrow{c.v} P}$$

$$\frac{}{(c?x : T \to P(x)) \xrightarrow{c.v} P(v)}\ [v \in T]$$

Example:

$$COPYBIT = in?x : \{0,1\} \to out!x \to COPYBIT$$

$$COPY = in?x : \mathbb{N} \to out!x \to COPY$$

$$SQUARE = in?x : \mathbb{Z} \to out!(x * x) \to SQUARE$$

◇ Specifications ◇

Recall the definitions for the specification of the system consisting of the student and the college.

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

$$\begin{aligned}
COLLEGE &= fail \to CF \mid pass \to C1 \\
C1 &= fail \to CF \mid pass \to C2 \\
C2 &= fail \to CF \mid pass \to prize \to STOP \\
CF &= fail \to CF \;\Box\; pass \to CF
\end{aligned}$$

$$SYSTEM = STUDENT\ {}_S\|_C\ COLLEGE$$

Initially we defined

$$\begin{aligned}
SPECF &= pass \to SPECF \mid fail \to SPECF \\
SPEC &= pass \to SPEC1 \mid fail \to SPECF \\
SPEC1 &= pass \to SPEC2 \mid fail \to SPECF \\
SPEC2 &= pass \to prize \to STOP \mid fail \to SPECF
\end{aligned}$$

but the specification

$$SPECP \sqsubseteq_T SYSTEM$$

is not quite what we want, because it does not allow $SYSTEM$ to do $year1, year2, year3$ or $graduate$.

◇ The Correct Specification ◇

To allow for $year1, year2, year3$ and $graduate$ we defined

$$\begin{aligned}
EXTRA = \;& year1 \to EXTRA \\
\mid\;& year2 \to EXTRA \\
\mid\;& year3 \to EXTRA \\
\mid\;& graduate \to EXTRA
\end{aligned}$$

and then

$$SPEC = SPECP\ {}_{SP}\|_E\ EXTRA$$

where

$$\begin{aligned}
SP &= \{pass, fail, prize\} \\
E &= \{year1, year2, year3, graduate\}.
\end{aligned}$$

In general, to simplify the definition of processes such as $EXTRA$, we can define, for any set A of events, the process RUN_A.

$$RUN_A = x : A \to RUN_A$$

Then $EXTRA = RUN_E$, and $SPECF = RUN_{\{pass,fail\}}$.

◇ Hiding ◇

There is an alternative approach to this kind of specification. Instead of putting a process in parallel with the specification to generate the events which we don't care about, we can *hide* those events from the process being specified.

If we define

$NEWSYSTEM =$

$SYSTEM \setminus \{year1, year2, year3, graduate\}$

then the behaviour of $NEWSYSTEM$ is derived from that of $SYSTEM$ by making the listed events invisible. The traces of $NEWSYSTEM$ are the traces of $SYSTEM$ with these events removed.

Now we can simply write

$$SPEC \sqsubseteq_T NEWSYSTEM.$$

as the specification. $SPEC$ only involves the events which we are interested in, and the hiding in the definition of $NEWSYSTEM$ shows which events we are leaving out of the specification.

◇ Using Hiding ◇

Returning to the level crossing example, there is an alternative approach to specifying the desired behaviour. We can use hiding to avoid specifying the events which we don't care about. In this case, all we want to do is specify that $crash$ never occurs.

If we hide all the events except $crash$ from $SYSTEM$ (or $SAFE\_SYSTEM$) then all we need for the specification is a process which never does $crash$:

$$STOP \sqsubseteq_T SYSTEM \setminus (E_T \cup E_C \cup E_G)$$

◇ Defining Hiding ◇

The transition rules defining hiding are

$$\frac{P \xrightarrow{a} P'}{P \setminus A \xrightarrow{\tau} P' \setminus A}\ [a \in A]$$

$$\frac{P \xrightarrow{a} P'}{P \setminus A \xrightarrow{a} P' \setminus A}\ [a \notin A]$$

As we saw when using FDR, the hidden events are replaced by τ, representing "silent" or "internal" events. τ events are not normally included in traces, although as we have seen, FDR can show where in a trace the τ events occur. When we discuss traces, we will not include τ.