14.4 Property and Polychrony

14.4.1 Input_Time

$$
\begin{aligned}
Input\_Time &::= \textbf{list}(IO\_Time\_Spec)\\
IO\_Time\_Spec &::= TimeRange \times IO\_Reference\_Time\\
TimeRange &::= Time \times Time\\
Time &::= aadlinteger \times Time\_Unit
\end{aligned}
$$

1. If **Input_Time** contains only one value of **IO_Time_Spec**:

$$
\begin{aligned}
\text{Let } x \ = \ (\{x_1, x_2, x_3, x_4, x_5\}) &\in Input\_Time\\
\text{where } x_1, x_3 &\in aadlinteger\\
x_2, x_4 &\in Time\_Unit\\
x_5 &\in IO\_Reference\_Time
\end{aligned}
$$

$\textbf{PropertyTranslation}(x)$ =
$Between\{min\_offset, max\_offset\}(InEvent, ReferenceTime, unit1, unit2)$
where min_offset = $\textbf{PropertyTranslation}(x_1)$
max_offset = $\textbf{PropertyTranslation}(x_3)$
$unit1$ = $\textbf{PropertyTranslation}(x_2)$
$unit2$ = $\textbf{PropertyTranslation}(x_4)$
$ReferenceTime$ = $\textbf{PropertyTranslation}(x_5)$

```
process Between = {integer min_offset, max_offset}
(? event Controlled_Time, Reference_Time, unit1, unit2;)
(| event1 := when ((unit1 after Reference_Time) = min_offset)
 | event2 := when ((unit2 after Reference_Time) = max_offset)
 | IN_INTERVAL := INTER_IN_IN(event1, event2, Controlled_Time)
 | Controlled_Time ^= when IN_INTERVAL
 |)
where
  event event1, event2;
  boolean IN_INTERVAL;
end;

process INTER_IN_IN=
  (? event START, FINISH,S;
```

```
  ! IN_INTERVAL)
    pragmas
      Comment "Is S present between START and FINISH ?"
              " S on START is IN the interval, a S on FINISH is IN a
    end pragmas
 (| MEM ^= START ^+ FINISH ^+ S
  | MEM := START default not FINISH default ZMEM
  | ZMEM := (MEM$ init false)
  | IN_INTERVAL := (FINISH default MEM) when S
  |)
 where
   boolean MEM, ZMEM;
 end;
```

2. If **Input_Time** contains more than one values (two for example). This case is left for further work.

14.4.2 Output_Time

$$
\begin{aligned}
&Output\_Time \ ::= \ \textbf{list}(IO\_Time\_Spec)\\
&IO\_Time\_Spec \ ::= \ TimeRange \times IO\_Reference\_Time\\
&TimeRange \ ::= \ Time \times Time\\
&Time \ ::= \ aadlinteger \times Time\_Unit
\end{aligned}
$$

1. If **Output_Time** contains only one value of **IO_Time_Spec**:

$$
\begin{aligned}
&\text{Let } x \ = \ (\{x_1, x_2, x_3, x_4, x_5\}) \in Output\_Time\\
&\text{where } x_1, x_3 \in aadlinteger\\
&\qquad x_2, x_4 \in Time\_Unit\\
&\qquad x_5 \in IO\_Reference\_Time
\end{aligned}
$$

$\textbf{PropertyTranslation}(x)$ =

$Between\{min\_offset, max\_offset\}(OutEvent, ReferenceTime, unit1, unit2)$

where min_offset = $\textbf{PropertyTranslation}(x_1)$

max_offset = $\textbf{PropertyTranslation}(x_3)$

$unit1$ = $\textbf{PropertyTranslation}(x_2)$

$unit2$ = $\textbf{PropertyTranslation}(x_4)$

$ReferenceTime$ = $\textbf{PropertyTranslation}(x_5)$

2. If **Output_Time** contains more than one values (two for example). This case is left for further work.

14.4.3 Access_Time

$$Access\_Time ::= First \times Last$$
$$First ::= IO\_Time\_Spec$$
$$Last ::= IO\_Time\_Spec$$

Let $x = (\{x_1, x_2, x_3, x_4, x_5\}, \{x_6, x_7, x_8, x_9, x_{10}\}) \in Access\_Time$
where $x_1, x_3, x_6, x_8 \in aadlinteger$
$x_2, x_4, x_7, x_9 \in Time\_Unit$
$x_5, x_10 \in IO\_Reference\_Time$

$\mathbf{PropertyTranslation}(x) =$
$(|Between\{F\_min\_offset, F\_max\_offset\}$
$(FirstAccess, F\_ReferenceTime, F\_unit1, F\_unit2)$
$|Between\{L\_min\_offset, L\_max\_offset\}$
$(LastAccess, L\_ReferenceTime, L\_unit1, L\_unit2)|)$
where $F\_min\_offset = \mathbf{PropertyTranslation}(x_1)$
$F\_max\_offset = \mathbf{PropertyTranslation}(x_3)$
$F\_unit1 = \mathbf{PropertyTranslation}(x_2)$
$F\_unit2 = \mathbf{PropertyTranslation}(x_4)$
$F\_ReferenceTime = \mathbf{PropertyTranslation}(x_5)$
$L\_min\_offset = \mathbf{PropertyTranslation}(x_6)$
$L\_max\_offset = \mathbf{PropertyTranslation}(x_8)$
$L\_unit1 = \mathbf{PropertyTranslation}(x_7)$
$L\_unit2 = \mathbf{PropertyTranslation}(x_9)$
$L\_ReferenceTime = \mathbf{PropertyTranslation}(x_1)$

14.4.4 aadlinteger

Let $x \in aadlinteger$
$\mathbf{PropertyTranslation}(x) = V(x)$

14.4.5 Time_Unit

$Time\_Unit$ ::= $\{ps, ns, us, ms, sec, min, hr\}$
Let $x \in Time\_Unit$

$$\begin{aligned}
&\textbf{PropertyTranslation}(x) \ = \ \textbf{event}\ V(x) \\
\text{where } &ns\,\hat{}\, = \textbf{when}\ ((ps\ \textbf{after}\ ns) = 1000) \\
&us\,\hat{}\, = \textbf{when}\ ((ps\ \textbf{after}\ us) = 1000) \\
&ms\,\hat{}\, = \textbf{when}\ ((us\ \textbf{after}\ ms) = 1000) \\
&sec\,\hat{}\, = \textbf{when}\ ((ms\ \textbf{after}\ sec) = 60) \\
&min\,\hat{}\, = \textbf{when}\ ((sec\ \textbf{after}\ min) = 60) \\
&hr\,\hat{}\, = \textbf{when}\ ((min\ \textbf{after}\ hr) = 60)
\end{aligned}$$

14.4.6 IO_Reference_Time

$IO\_Reference\_Time$ ::= $\{Dispatch, Start, Completion, Deadline, NoIO\}$
Let $x \in IO\_Reference\_Time$

$$\textbf{PropertyTranslation}(x) \ = \ \textbf{event}\ V(x)$$

14.4.7 Queue_Size

$Queue\_Size$:= $aadlinteger$
Let $x \in Queue\_Size$, where $x \in aadlinteger$
$\textbf{PropertyTranslation}(x) \ = \ V(x)$

14.4.8 Dequeue_Items

$Dequeue\_Items$:= $aadlinteger$
Let $x \in Dequeue\_Items$, where $x \in aadlinteger$
$\textbf{PropertyTranslation}(x) \ = \ V(x)$

14.4.9 Fan_Out_Policy

A controler (Distributer) is needed to choose the recipients of an out port. (Figure 26.) For each connection, a Distributer is added to control that whether the output will be sent to this connection or not.

A set of Distributer are defined in a library: Distributer_Broadcast, Distributer_OnDemand,etc. Depending on the Fan_Out_Policy, a corresponding Distributer will be used.

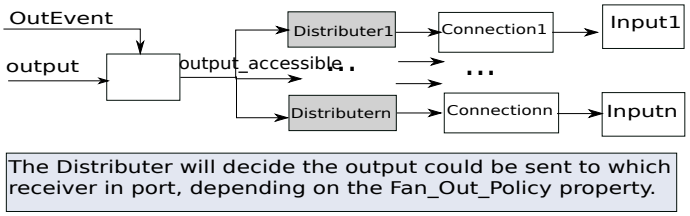

Figure 26: Fan_Out_Policy

```
process Distributer_Broadcast =
{integer Distributer_ID;}
(? i; ! o;)
  (|o := i|);

process Distributer_OnDemand =
{integer Distributer_ID;}
(? i; event demand; ! o;)
  (|o := i when demand |);

process Distributer_Selective =
{integer Distributer_ID;}
(? i; boolean select; ! o;)
(|o := i when select|);

process Selection =
{integer m;}
(? event trigger; ! boolean[m] select;)
(|...|)

process Distributer_RoundRobin =
{integer Distributer_ID;}
(? i; integer RR_ID; ! o;)
(|o := i when (RR_ID = Distributer_ID)
 |);

process RoundRobin =
{integer m;}
(? event trigger; ! integer RR_ID;)
(| RR_ID := (zID+1) modulo m when trigger default zID
 | zID := RR_ID$1 init 0
 |)
where
  integer zID;
```

```
end;
```

If it is set to be RoundRobin, an extra process RoundRobin is needed to calculate the sequence. Only the one whose ID equals to the sequence identifier will receive the output (Figure 27). (Similar for Selective: if it is set to be Selective, an extra process Selection is added. Only the one who is slected will receive the output.)

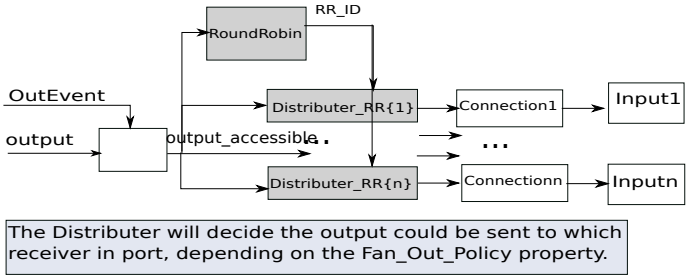

Figure 27: Fan_Out_Policy: RoundRobin

$Fan\_Out\_Policy ::= \{Broadcast, RoundRobin, Selective, OnDemand\}$

Let $x \in Fan\_Out\_Policy$, and y is the Out port,

c_k is a connection sourced from y

$$\textbf{PropertyTranslation}(x, y, c_k) =$$

$$\begin{cases} o := Distributer\_Broadcast\{k\}(i) & if\ V(x) = Broadcast \\ o := Distributer\_OnDemand\{k\}(i, demand) & if\ V(x) = OnDemand \\ o := Distributer\_Selective\{k\}(i, select) & if\ V(x) = Selective \\ o := Distributer\_RoundRobin\{k\}(i, RR\_ID) & if\ V(x) = RoundRobin \end{cases}$$

where : k is the identifier of the corresponding connectionc_k,

i is the output from the out port y,

$demand$ comes from the connecting in port,

$select$ is generated by a selection process,

RR_ID is calculated by a RoundRobin process.

15 Modes

(13)The modes subclause declares a state machine describing the dynamic mode switching behavior of modes. The states of the state machine represent the different modes and the transitions specify the event(s) that can trigger a mode switch to the destination mode. Only one mode alternative represents the current mode at any one time.