How is a Conditional Statement Implemented in a Timing or Micro-timing Machine?

Ellis D. Cooper April 13, 2015

Conditionals in computer programming languages formalize a severely emasculated version of Free Will. One may consider Will to be Choice after Deliberation on Situation. Alternatives for Choice may be Created during Deliberation based on the Situation. In computer programming, unlike Philosophy, alternative processes are fixed beforehand, and the one to perform is based on an algorithm with system status as input. In general, a process alters status, so the same opportunity to choose at a different time may lead to a different choice among its alternatives. If there are exactly two alternative processes, say A and B, and $P(x)$ is an algorithm with status input x with result either 0 or 1 (or "false" or "true", or "no" or "yes"), then a computer program statement such as

$$\text{IF } P(x) = 0 \text{ THEN } A \text{ ELSE } B \tag{1}$$

performs process A under the condition that $P(x) = 0$ and B otherwise. That is why it is called a "conditional statement."

Timing machines are diagrams composed of labeled dots $(a), (b), (c), \ldots$ called *states* connected by four types of arrows[1]

$$\begin{tikzcd}
(a) \arrow[r, Rightarrow] & (b) & \textit{timeout} \\
(a) \arrow[r, dotted] & (b) & \textit{signal} \\
(a) \arrow[r, two heads] & (b) & \textit{trigger} \\
(a) \arrow[r] & (b) & \textit{probability}
\end{tikzcd} \tag{2}$$

Status of a timing machine is determined by the unique active state in each part of the machine, and by the values of variables. These are not truly distinct ideas, since a variable $[v]$ is itself a timing machine (see p.47 of [Coo15]) which may be queried by a ? signal in response to which it delivers a value signal, $\bullet$:

$$\begin{tikzcd}
(a) \arrow[r, dotted, "?"] & {[v]} \arrow[r, dotted, "\bullet"] & (b)
\end{tikzcd} \tag{3}$$

[1]Notation here is a slight variant of that adopted in Chapter 2 of [Coo15].

The value returned by the query in (3) may be the input to a timing machine $\{P\}$ which has two states, one of which by design is guaranteed to time out, leading to either (a state in) timing machine $\{A\}$ or $\{B\}$. Thus, (4) is a timing machine implementation of (1).

$$\begin{tikzcd}
 & & & \{A\} \\
(a) \arrow[r, dotted, "?"] & {[v]} \arrow[r, dotted, "\bullet"] & \{P\} \arrow[ur, Rightarrow] \arrow[dr, Rightarrow] & \\
 & & & \{B\}
\end{tikzcd} \tag{4}$$

Micro-timing machines are diagrams of labeled dots called *nodes* connected only by signal and probability arrows, and $a \overset{\varepsilon}{\dashrightarrow} b \xrightarrow{P^a_b(c)} c$ means that node a has maximum time ε, and if a times out while b is active then b is triggered to c with probability $P^a_b(c)$.

In the micro-timing machine (5), $\{P\}$ is a micro-timing machine with nodes x, y, u, v such that either timeout of x may trigger y to A, or timeout of u may trigger v to B, all depending on the prior activity of a or b.

$$\begin{tikzcd}
a \arrow[dr, dotted] & & A \\
 & \{P\} \arrow[ur, "P^x_y(A)=1"'] \arrow[dr, "P^u_v(B)=1"] & \\
b \arrow[ur, dotted] & & B
\end{tikzcd} \tag{5}$$

This too implements (1), but only in a rudimentary way, given the expressive paucity of small micro-timing machines. More expressive timing machines may be "compiled down" to micro-timing machines, so this is a practical not a conceptual limitation.

References

[Coo15] Ellis D. Cooper. *Microlects of Mental Models.* Cognocity Press, 2015.