Capturing Infinity by Watching the Clock

Eric C.R. Hehner

There is a clock. When the clock starts, at each time unit, either it ticks, denoted τ , or it pings, denoted π , until the clock stops. Ticking and pinging are different, unequal actions.

$$\tau \neq \pi$$

Sequencing is denoted by a semicolon. For example:

$$\tau;\ \tau;\ \pi$$

specifies, or describes, the action of starting, then ticking twice, then pinging, then stopping.

A comma denotes a choice, or option. For example:

$$\tau;\ \tau;\ \pi\ ,\ \tau;\ \pi;\ \pi$$

specifies that the clock either ticks twice then pings, or ticks then pings twice. And

$$\tau;\ (\tau, \pi);\ \pi$$

specifies that the clock first ticks, then either it ticks or pings, then it pings. These are the same.

$$\tau;\ (\tau, \pi);\ \pi\ \ =\ \ \tau;\ \tau;\ \pi\ ,\ \tau;\ \pi;\ \pi$$

The brackets are needed because semicolon has precedence over (binds tighter than) comma. For future reference, the precedence of all operators we will be using is as follows:

$$*\ \text{ before }\ \bar{,}\ \text{ before }\ ;\ \text{ before }\ \text{‘}\ \text{ before }\ ,\ \text{ before }\ =\ \neq$$

An inverted comma, or apostrophe, denotes joint specification. For example:

$$(\tau;\ \tau;\ \pi\ ,\ \tau;\ \pi;\ \pi)\ \text{‘}\ (\tau;\ \tau;\ \pi\ ,\ \pi;\ \tau;\ \pi)\ =\ \tau;\ \tau;\ \pi$$

The specification $(\tau;\ \tau;\ \pi\ ,\ \tau;\ \pi;\ \pi)$ offers a choice between $(\tau;\ \tau;\ \pi)$ and $(\tau;\ \pi;\ \pi)$; to satisfy this specification, the clock can either $(\tau;\ \tau;\ \pi)$ or $(\tau;\ \pi;\ \pi)$. The specification $(\tau;\ \tau;\ \pi\ ,\ \pi;\ \tau;\ \pi)$ offers a different choice. The only sequence that satisfies both specifications is $(\tau;\ \tau;\ \pi)$.

The symbol $\bar{,}$ denotes choice removal. For example:

$$(\tau;\ \tau;\ \pi\ ,\ \tau;\ \pi;\ \pi)\ \bar{,}\ \tau;\ \tau;\ \pi\ \ =\ \ \tau;\ \pi;\ \pi$$

The specification $(\tau;\ \tau;\ \pi\ ,\ \tau;\ \pi;\ \pi)$ offers a choice between $(\tau;\ \tau;\ \pi)$ and $(\tau;\ \pi;\ \pi)$, and then the $(\tau;\ \tau;\ \pi)$ option is removed, leaving only the $(\tau;\ \pi;\ \pi)$ option.

Various laws apply. For examples, if S , T , and U are specifications,

$S, T\ =\ T, S$	symmetry (commutativity)
$(S, T), U\ =\ S, (T, U)$	associativity
$(S; T); U\ =\ S; (T; U)$	associativity
$(S, T); U\ =\ S;U\ ,\ T;U$	distributivity
$S; (T, U)\ =\ S;T\ ,\ S;U$	distributivity
$(S\bar{,}T)\bar{,}U\ =\ S\bar{,}(T, U)\ =\ (S\bar{,}T)\text{‘}(S\bar{,}U)$	union removal
$(S, T)\bar{,}U\ =\ S\bar{,}U\ ,\ T\bar{,}U$	union removal
$A\text{‘}(B\bar{,}C)\ =\ (A\text{‘}B)\bar{,}C\ =\ B\text{‘}(A\bar{,}C)$	intersection Removal

The identity for sequence is the empty sequence, which we denote nil .

$$S;\ nil\ =\ S\ =\ nil;\ S$$

The identity for choice, and a zero for sequence, is the empty choice, which we denote $null$.

$$S, null\ =\ S\ =\ null, S$$

$$S;\ null\ =\ null\ =\ null;\ S$$

Choice removal does not have a left identity. It has $null$ as right identity.

$$S\bar{,}null\ =\ S$$

Joint specification has (τ, π) as identity for single actions. That means

$$\tau ` (\tau, \pi) = \tau = (\tau, \pi) ` \tau$$
$$\pi ` (\tau, \pi) = \pi = (\tau, \pi) ` \pi$$

If n is a natural number and S is a specification, then $n*S$ specifies that S happens n times. For examples:

$$3*(2*\tau; \pi) = 3*(\tau; \tau; \pi) = \tau; \tau; \pi; \tau; \tau; \pi; \tau; \tau; \pi$$
$$0*S = nil$$
$$2*(\tau, \pi) = (\tau, \pi); (\tau, \pi) = \tau; \tau \ , \ \tau; \pi \ , \ \pi; \tau \ , \ \pi; \pi$$

Operator $*$ does not distribute over comma in its right operand. For example,

$$2*(\tau, \pi) \neq 2*\tau, 2*\pi = \tau; \tau \ , \ \pi; \pi$$

But $*$ does distribute over comma in its left operand.

$$(2, 3)*\tau = 2*\tau, 3*\tau = \tau;\tau \ , \ \tau;\tau;\tau$$

Let nat be the natural numbers. Informally,

$nat = 0, 1, 2,$ and so on

More formally, nat is the smallest solution of the equation

$$nat = 0, nat+1$$

Now define $nat*S$ as the smallest solution of

$$nat*S = nil, \ S; nat*S$$

Then $nat*S$ denotes all natural repetitions of S (all finite repetitions of S).

$nat*S = (0, 1, 2,$ and so on$)*S = 0*S \ , \ 1*S \ , \ 2*S \ ,$ and so on

$nat*\tau = nil \ , \ \tau \ , \ \tau;\tau \ , \ \tau;\tau;\tau \ , \ \tau;\tau;\tau;\tau \ ,$ and so on

So $nat*(\tau, \pi)$ is all finite length sequences.

The last notation we introduce is $\infty*S$ meaning an infinite repetition of S .

$\infty*S = S; S; S;$ and so on

More formally, $\infty*S$ is the smallest solution of

$$\infty*S = S; \infty*S$$

So $\infty*(\tau, \pi)$ is all infinite length sequences. And $(nat, \infty)*(\tau, \pi)$ is the universe of specifications, and that is the identity for joint specifications.

$$S ` (nat, \infty)*(\tau, \pi) = S = (nat, \infty)*(\tau, \pi) ` S$$

We are using $S=T$ to say that specifications S and T are in some way the same, and $\neq$ to say that specifications S and T are in some way different. The intent is as follows: $S=T$ if no observation can distinguish whether the clock is behaving according to S or T ; $S \neq T$ if some observation can distinguish whether the clock is behaving according to S or T . For example,

$$\tau; \tau \ , \ \tau; \pi \ , \ \pi; \tau \ , \ \pi; \pi \ \neq \ \tau; \tau \ , \ \pi; \pi$$

because if we observe $(\tau; \pi)$ then we know the clock is behaving in accordance with specification $(\tau; \tau \ , \ \tau; \pi \ , \ \pi; \tau \ , \ \pi; \pi)$ but not in accordance with specification $(\tau; \tau \ , \ \pi; \pi)$.

All observations occur at finite times, and therefore the difference between $nat*(\tau, \pi)$ and $(nat, \infty)*(\tau, \pi)$ is not observable. At any time, if you observe the end of a sequence, it is in $nat*(\tau, \pi)$, and therefore also in $(nat, \infty)*(\tau, \pi)$. If you observe that the sequence is not ended, it may be a longer finite sequence in $nat*(\tau, \pi)$, or an infinite sequence in $(nat, \infty)*(\tau, \pi)$. In general, for any specification S ,

$$nat*S = (nat, \infty)*S = nat*S, \infty*S$$

Unexpectedly, the infinite sequences $\infty*(\tau, \pi)$ are included in the finite sequences $nat*(\tau, \pi)$.

Anyone familiar with *a Practical Theory of Programming* (aPToP) or with Unified Algebra (UA) recognizes choice and sequencing as bunches and strings. The reason for a clock, rather than just a binary sequence, is that a clock is a physical object, and observation is a physical action, and observations happen over time. The equality used here, motivated by physical observation, differs from that in aPToP and UA.

In aPToP, the implementable specification $t<\infty \Rightarrow t'<\infty$ seems to say that if execution starts at a finite time, then it will end at a finite time. Nonetheless, it can be refined by an infinite loop.

$$t<\infty \Rightarrow t'<\infty \;\Leftarrow\; t:= t+1.\; t<\infty \Rightarrow t'<\infty$$

That is because we can never observe a computation failing to satisfy $t<\infty \Rightarrow t'<\infty$.

In UA, the reals are defined as the limits of all rational sequences. Although no rational sequence includes ∞ , some of them have limit ∞ . So the reals include ∞ .

In all of these cases, “finite but unbounded” captures “infinite”.