Setting real-time CSP

Jim Davies

1 Introduction

The language and models of CSP have undergone a gradual evolution since the publication of the first CSP textbook—Hoare's *Communicating Sequential Processes* (Prentice-Hall, 1985). The forthcoming text on real-time CSP will provide for some degree of standardisation.

In parallel, we hope to provide a standard set of macros for setting documents which use CSP notation. This will allow users to exchange documents in electronic form, and will form part of the user interface to the language tools.

The macros are defined by a style file called `zed-csp.sty`. This should work with LaTeX 2_ε. Inquiries, suggestions, or complaints should be addressed to

`Jim.Davies@comlab.ox.ac.uk`.

Note that this is a fairly quick fix of the style to enable myself and others to use the improved facilities offered by the new version of LaTeX. It has not been rigorously tested, although it seems to work for me.

2 Symbols

We can divide the symbols used into three separate classes: symbols for the language itself, symbols used in the definition of the semantics, and symbols used in the specification language.

2.1 The language of real-time CSP

The operators of real-time CSP are set using macros of the same name. The macros for atomic operators begin with an uppercase letter; the same is true for those representing indexed versions of parallel and choice operators. All other macros are lowercase throughout. Some operators accept optional arguments, but no argument is compulsory.

When an operator with an optional argument appears within an optional argument, LaTeXmay require assistance if it is to parse the expression correctly. In these circumstances, we use an extra pair of braces to delimit the process expression: e.g.,

```
\Ftf[{P \parallel[A] Q}].
```

bottom	`\Bottom`	$\bot$
stop	`\Stop`	$Stop$
skip	`\Skip`	$Skip$
wait	`\Wait`	$Wait$
prefix	`\then`	$\rightarrow$
external choice	`\extchoice`	$\Box$
internal choice	`\intchoice`	$\sqcap$
hiding	`\hide`	$\setminus$
parallel	`\parallel[A][B]`	$\|[\, A \mid B \,]\|$
interleaving	`\interleave`	$\|\|\|$
sharing	`\parallel[C]`	$\|[\, C \,]\|$
recursion	`\mu X \spot P`	$\mu\, X \bullet P$
timeout	`\timeout[t]`	$\triangleright\{t\}$
transfer	`\transfer[t]`	$\swarrow\{t\}$
interrupt	`\interrupt`	$\triangle$
timer	`\at`	$@$
indexed external choice	`\Extchoice`	$\Box$
indexed internal choice	`\Intchoice`	$\bigsqcap$
indexed alphabet parallel	`\Parallel`	$\Vert$
indexed interleaving	`\Interleave`	$\Vert\Vert$

2.2 Parallel combinations

There are several ways to denote the parallel combination of two processes in CSP. Firstly, we can describe the set of events upon which they must cooperate: e.g., in the process

$$P \,|[\, C \,]|\, Q$$

components P and Q must cooperate upon every event from the shared set C. Alternatively, we can declare two alphabets

$$\begin{aligned} \alpha P &= A \\ \alpha Q &= B \end{aligned}$$

and write

$$P \parallel Q$$

to denote the parallel combination in which P and Q must cooperate upon every event in the intersection of their alphabets. Finally, we can add explicit alphabet information to the parallel operator: e.g.,

$$P \ |[\, A \mid B \,]| \ Q$$

is equivalent to the above parallel combination, given the values chosen for αP and αQ.

2.3 Delays and timers

We write $Wait\, t;\ P$ to denote the process which will delay for time t before behaving as P. The wait process is a delayed form of termination *Skip*: i.e.,

$$Wait\, 0 \quad = \quad Skip$$

To model a nondeterministic delay, we can use an internal choice operator indexed by a range of time values:

$$\sqcap_{t \in [t_1, t_2)} Wait\ t$$

A convenient abbreviation for this involves overloading the *Wait* operator: e.g.,

$$Wait[t_1, t_2)$$

abbreviates the above choice.

External events in a process description are performed in cooperation with the environment of that process. It is therefore quite likely that an external event will not occur as soon as the process is ready. The time elapsed between the offer of an event and its occurrence can influence future behaviour; the rest of the process description should be allowed to refer to this time.

Accordingly, real-time CSP includes a timer construct, or 'passage-of-time' operator. We write

$a @ t \rightarrow P$ `a \at t \then P`

to denote a process which is initially ready to engage in event a. The time variable t is assigned the relative time at which a occurs. This is the same as the elapsed time between control being passed to this process—at which point the offer of a is made—and the event a actually occuring.

A useful extension to this, which adds nothing to the expressivity of the language but can make for more intelligible process descriptions, is the offer timeout. We write

$a@t\{d\} \rightarrow P$ `a \at t \{ d \} \then P`

to denote a process which offers to perform a, and will store the time of occurrence in t, but will withdraw the offer if it has not been accepted by time d. (This form of timeout was suggested by Guy Leduc for his version of timed LOTOS.)

3 Mathematical language

The semantic models of CSP come with a great deal of notational baggage. We need to define operators to project information out of traces, refusals, and timed failures. There is also a specification language based upon the timed semantics, and the names used for the models themselves.

3.1 Logic, sets, and sequences

`\defs`	$\widehat{=}$	`\land`	$\land$	`\power`	$\mathbb{P}$
`\mu`	μ	`\lor`	$\lor$	`\finset`	$\mathbb{F}$
`\lambda`	λ	`\Land`	$\bigwedge$	`\cross`	$\times$
`\exists`	$\exists$	`\Lor`	$\bigvee$	`\union`	$\cup$
`\forall`	$\forall$	`\lnot`	$\neg$	`\inter`	$\cap$
`\spot`	$\bullet$	`\implies`	$\Rightarrow$	`\Union`	$\bigcup$
`\nat`	$\mathbb{N}$	`\iff`	$\Leftrightarrow$	`\Inter`	$\bigcap$
`\num`	$\mathbb{Z}$	`\upto`	$..$	`\dom`	dom
`\rat`	$\mathbb{Q}$	`\le`	$\leqslant$	`\ran`	ran
`\real`	$\mathbb{R}$	`\ge`	$\geqslant$	`\emptyset`	$\emptyset$
`\seq`	seq	`\project`	$\restriction$	`\set{x}`	$\{x\}$

3.2 Operators on traces

empty trace	`\nil`	$\langle\rangle$
trace	`\trace{e_1,e_2}`	$\langle e_1, e_2\rangle$
catenation of traces	`\cat`	$\frown$
count	`\cnt`	$\downarrow$
during	`\during`	$\upharpoonright$
tick event	`\tick`	$\checkmark$
subsequence	`\subseq`	$\preccurlyeq$
data values	`\data`	$\Downarrow$

3.3 Projection functions

begin	`\Begin`	begin
end	`\End`	end
head	`\Head`	head
first	`\First`	first
tail	`\Tail`	tail
front	`\Front`	front
last	`\Last`	last
times	`\Times`	times
events	`\Events`	events

times and events are projection functions from timed traces to sequences of times and sequences of events respectively. head and tail may be applied to any sequence. begin and end may be applied to timed traces and timed refusals. first is a synonym for head. front is the dual of tail. last is the dual of head.

To denote the set of events mentioned in a timed or untimed trace or refusal, we prefix the name of the object with α. For example, the set of events mentioned in the timed trace s would be written αs. Earlier version of real-time CSP did this using the σ operator to avoid confusion with process alphabets. Where there is scope for confusion, we suggest that this practice is continued.

3.4 Semantic functions, models, and spaces

In *Advanced CSP*, we use long names for the semantic functions:

semantics	`\Semantics`	*semantics*
traces	`\Traces`	*traces*
failures	`\Failures`	*failures*
timed failures	`\TimedFailures`	*timed failures*
divergences	`\Divergences`	*divergences*
infinites	`\Infinites`	*infinites*

Any semantic function macro can be given an optional argument. This will be set within semantic brackets: e.g., `\Traces[P]` yields *traces* $[\![P]\!]$. To obtain the semantic brackets alone, use the `\semb` macro; this takes a single compulsory argument. Alternatively, the macros `\leftsemb` and `\rightsemb` produce left and right semantic brackets respectively.

In theoretical papers, we often need to refer to several models, functions, and associated spaces. To make things easier on ourselves, we adopt short names for these mathematical objects, using subscripts to identify the model concerned. For example, the objects associated with the timed failures model are all subscripted with TF.

The models themselves have macros beginning `\M`:

traces	`\Mut`	$\mathcal{M}_{UT}$
failures	`\Muf`	$\mathcal{M}_{UF}$
failures-divergences	`\Mufd`	$\mathcal{M}_{UFD}$
timed failures	`\Mtf`	$\mathcal{M}_{TF}$
timed failures-stability	`\Mtfs`	$\mathcal{M}_{TFS}$
timed infinite	`\Mti`	$\mathcal{M}_{TI}$

The matching semantic functions use `\F` instead—e.g., `\Fut` for untimed traces—and the observation spaces use `\S`.

3.5 Refinement and satisfaction

The satisfaction notation employed in Hoare's *Communicating Sequential Processes* has been retained. We also have a refinement relation between processes, possibly indexed by the name of the model concerned.

The satisfaction relation is set as follows: `P \sat S` produces P **sat** S. The refinement relation is produced by `\lessdet` (or `refinedby`, a synonymbol).

3.6 Specifications

To capture timing constraints at the level of the semantic models, we use a number of specification 'macros'. These are set using LaTeX macros which begin with an 'm' (for macro) and

are then capitalised.

internal	`\mInternal`	internal
refuses	`\mRef`	ref
at	`\mAt`	at
live	`\mLive`	live
open	`\mOpen`	open
from	`\mFrom`	from
until	`\mUntil`	until
live from	`\mLiveFrom`	live from
open from	`\mOpenFrom`	open from
name of last	`\mNameOfLast`	name of last
before	`\mBefore`	before
after	`\mAfter`	after
time of last	`\mTimeOfLast`	time of last

4 Discussion

4.1 Dependencies

You must have the AMS fonts available, and the `amsfonts` installation must have been performed for LaTeX 2_ε. This requires the `mfnfss` package; it takes about twenty seconds.

4.2 CSP and Z

You may have problems if you try to use the `zed-csp` package with `fuzz` or any style package that uses the AMS fonts. The good news is that you shouldn't need them. All of the AMS symbols are defined in the `zed-csp` package, using the standard names.