SPECIFYING PROBLEM ONE USING THE 'FAILURE' SETS MODEL FOR CSP AND DERIVING CSP PROCESSES WHICH MEET THIS SPECIFICATION

A.W. Roscoe
Programming Research Group
University of Oxford

In this note we sketch how an abstract mathematical model can be used to specify the two-way channel. We see how theorems proved about the abstract specification suggest designs of processes which satisfy it. The model used can express safety and liveness properties and allows non-determinism. It does not deal with fairness however.

1. The Failure Sets Model

For a summary description of the failure sets model see the appendix to this paper.

The important points to note are:

(i) every process is represented by a set of pairs (s, δ), where s is a possible trace and δ is either

- $\uparrow$ representing divergence (non-termination), or
- X representing a set of symbols to which the process can refuse to respond.

(ii) The model is good for expressing correctness conditions because it -

- permits the banning of divergence
- allows possible traces to be specified (safety)
- allows the banning of refusal by the process (liveness).

If on operating some process A with current trace s we offer it a set X such that $(s, X) \notin A$, then we can guarantee that the process will eventually accept some element of X.

First, a notation for manipulating traces.

(i) If $s,t \in \Sigma^*$, we say $s \leq t$ if $\exists u.su = t$.

(ii) If $s \in \Sigma^*$, $X \subseteq \Sigma$, we say

$$s \upharpoonright X = \langle\rangle \text{ if } s = \langle\rangle$$
$$= t \upharpoonright X \text{ if } s = t\langle a\rangle \text{ and } a \notin X$$
$$= (t \upharpoonright X)\langle a\rangle \text{ if } s = t\langle a\rangle \text{ and } a \in X.$$

(iii) If a is some name used for symbols, and $b \in \Sigma$, then

$$\mathrm{strip}(a)(b) = c, \text{ if } b = ac$$
$$= b \text{ otherwise.}$$

(iv) If $s \in \Sigma^*$, then

$$\mathrm{strip}(a)(s) = \langle\rangle , \text{ if } s = \langle\rangle$$
$$= (\mathrm{strip}(a)(t)) \langle \mathrm{strip}(a)(b) \rangle ,$$
$$\text{if } s = tb, \text{ where } t \in \Sigma^*, \ b \in \Sigma \ .$$

(v) $s \downarrow a = \mathrm{strip}(a)(s \upharpoonright a.\Sigma)$

eg.: $\langle a.b,\ c.b,\ a.c\rangle \downarrow a = \langle b,c\rangle$

2. Problem 1: Two-way Channel with Disconnect

What is a channel?

- Informally it passes messages either way.
- 'dis' is a message, so its order is preserved, and it is transmitted.
- To be a reliable channel it must accept all input it is offered while empty (in either direction?).
- When it contains a message this must eventually be delivered to a destination which does nothing but wait.
- On delivering or receiving a dis at either end it must die at that end. (To die without transmitting a 'dis' to the environment seems unreasonable.)

Specification of a Channel (Abstract)

- The set of ordinary messages is T ($\mathrm{dis} \notin T$).
- The alphabet $\Sigma_{\mathrm{CHAN}(a,b)}$ of a channel, whose end ports are named a and b, is: $\{ a!t',\ a?t',\ b!t',\ b?t' \quad t' \in T \cup \{\mathrm{dis}\}\}$
- The predicate CHAN(a,b) is defined as follows:

$$\mathrm{CHAN}(a,b)(C) = [(s,\delta) \in C \Rightarrow \delta \neq \uparrow]$$
$$\& [(s,X) \in C \Rightarrow s \in (\Sigma_{\mathrm{CHAN}(a,b)})^*]$$
$$\& [s\downarrow a? \geqslant s\downarrow b! \ \& \ s\downarrow b? \geqslant s\downarrow a!]$$

....order of messages preserved

$$\& [(s\downarrow a = u\langle d\rangle v) \ \& \ d \in \{?\mathrm{dis},\ !\mathrm{dis}\} \Rightarrow v = \langle\rangle]$$
$$\& [(s\downarrow b = u\langle d\rangle v) \ \& \ d \in \{?\mathrm{dis},\ !\mathrm{dis}\} \Rightarrow v = \langle\rangle]$$

....neither end can do anything after a dis

$$\& \ \neg \mathrm{dis}(a)(s) \ \& \ (s\downarrow a? = s\downarrow b!) \Longrightarrow X \cap (a?T \cup \{a?\mathrm{dis}\}) = \emptyset$$
$$\& \ \neg \mathrm{dis}(b)(s) \ \& \ (s\downarrow b? = s\downarrow a!) \Longrightarrow X \cap (b?T \cup \{b?\mathrm{dis}\}) = \emptyset$$

....a non-disconnected end, all of whose communication has been received, must be able to transmit

$\& \neg dis(b)(s) \;\&\; (s\downarrow a? \geq (s\downarrow b!)\langle t'\rangle) \Longrightarrow b!t' \notin X$
$\& \neg dis(a)(s) \;\&\; (s\downarrow b? \geq (s\downarrow a!)\langle t'\rangle) \Longrightarrow a!t' \notin X$

....the channel can never refuse to output a message it contains

where $dis(\alpha)(s) \equiv s\upharpoonright\{\alpha!dis, \alpha?dis\} \neq \langle\,\rangle$.

A buffer may be defined in the same way:

$$
\begin{aligned}
BUFF(B) \;&\equiv \forall s (s,\uparrow) \notin B\\
\& (s,X)\in B \Longrightarrow\; & s \in (?T'\cup !T')^*\\
& \& \; s\downarrow ? \geq s\downarrow !\\
& \& (s\downarrow ? = s\downarrow ! \Longrightarrow X \cap ?T' = \emptyset)\\
& \& (s\downarrow ? \geq (s\downarrow !)\langle t\rangle) \Rightarrow !t \notin X
\end{aligned}
$$

where $T' = T\cup\{dis\}$.

BUFF(a,b) is the same except that ! is replaced by b!
and ? is replaced by a?.

Assorted theorems can be proved concerning channels and buffers:

eg. $CHAN(a,b)(C) \;\&\; CHAN(b,d)(C^*) \;\&\; d \neq a \Longrightarrow CHAN(a,d)(C \underset{b}{\ast} C^*)$
$BUFF(A) \;\&\; BUFF(B) \Longrightarrow BUFF(A \gg B)$
$BUFF(A) \;\&\; BUFF(A \gg B) \Longrightarrow BUFF(B)$
$BUFF(A \gg B) \;\&\; BUFF(B) \Longrightarrow BUFF(A)$

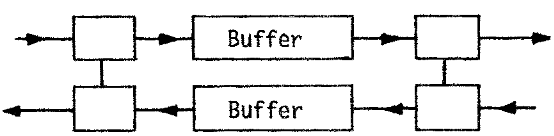
$a \neq e, \;\&\; b \neq d \;\&\; CHAN(a,b)(C_1) \;\&\; BUFF(b,d)(B_1)$
$\& \; BUFF(d,b)(B_2) \;\&\; CHAN(d,c)(C_2)$
$\Longrightarrow CHAN(a,e)(C_1 \underset{b}{\ast} (B_1 \parallel B_2) \underset{d}{\ast} C_2)$

Such theorems are easy, if tedious, to prove.

Examples of buffers:

$$
\begin{aligned}
B^1 &= ?x{:}T' \to !x \to B^1\\
B^n &= B^1 \gg \ldots\ldots \gg B^1 \quad \text{(n times)}\\
B^\infty &= ?x{:}T' \to (B^\infty \gg !x \to B^1) \quad \text{(unbounded)}\\
B^* &= ?x{:}T' \to (B^1 \gg !x \to B^*) \quad \text{(bounded but growing)}
\end{aligned}
$$

One obvious configuration for a channel is:



This can be realised:

$$\begin{aligned}
\mathrm{INS}(\alpha,\beta) = {} & (\alpha?x{:}T \rightarrow (\beta!x \rightarrow \mathrm{INS}(\alpha,\beta) \;\Box\; \alpha!dis \rightarrow \underline{abort}\;))\\
& \Box\; (\alpha?dis \rightarrow \beta!dis \rightarrow \underline{abort}\;)\\
& \Box\; (\alpha!dis \rightarrow \underline{abort}\;)\\
\mathrm{OUTS}(\alpha,\beta) = {} & (\beta?x{:}T \rightarrow (\alpha!x \rightarrow \mathrm{OUTS}(\alpha,\beta) \;\Box\; \alpha?dis \rightarrow \underline{abort}\;))\\
& \Box\; (\beta?dis \rightarrow (\alpha!dis \rightarrow \underline{abort} \;\;\Box\; \alpha?dis \rightarrow \underline{abort}\;))\\
& \Box\, (\alpha?dis \rightarrow \underline{abort}\;)
\end{aligned}$$

If B_1 is any (b,a) buffer and B_2 is any (a,b) buffer, then

$$a \neq b \Longrightarrow ((\mathrm{INS}(a,b) \,\|\, \mathrm{OUTS}(a,b)) \underset{b}{\gtrless} (B_1 \| B_2) \underset{a}{\gtrless} (\mathrm{INS}(b,a) \,\|\, \mathrm{OUTS}(b,a)))$$
satisfies CHAN(a,b).

Appendix - A summary of the failure sets model for CSP

For a fuller description of this model the reader should consult any of [1, 2, 3, 4]. The model is similar to, and makes the same basic postulates about processes as, the well-known 'traces' model. It is, however, able to make some important distinctions between processes not made by simple traces.

The agents $\alpha.\gamma$ NIL + $\alpha.\beta$ NIL and $\alpha.(\beta.$NIL + $\gamma.$NIL) would be identified over traces, as would $(\mu p.\alpha.p)\backslash\alpha$ and NIL. There are good reasons for wishing to avoid these identifications: the idea is to record not only the possible traces (ie. sequences of atomic actions) of a process but also its refusals (the sets which it can reject in a 'stable' state after some trace), and divergences (the occasions when it can become involved in an infinite sequence of internal actions and never give any answer to its environment).

A process is thus a set of pairs (s,δ), the first component being a trace and the second either X, a refusal set ($X \subseteq \Sigma$, the alphabet), or $\uparrow$ indicating divergence.

A process Q will be any subset of $\Sigma^* \times (\mathbb{P}(\Sigma) \cup \{\uparrow\})$ such that

1. $\mathrm{dom}(Q) = \{s \in \Sigma^* \mid \exists \delta.(s,\delta) \in Q\}$ is non-empty and prefix-closed
   $[\langle\rangle \in \mathrm{dom}(Q),\; st \in \mathrm{dom}(Q) \Longrightarrow s \in \mathrm{dom}(Q)]$
2. $(s,X) \in Q \;\&\; Y \subseteq X \Longrightarrow (s,Y) \in Q$
3. $(s,X) \in Q \;\&\; Y \cap (Q\ \underline{\mathrm{after}}\ s)^0 = \emptyset \Longrightarrow (s, X \cup Y) \in Q$

4. $(\forall \text{finite } Y \subseteq X.\ (s, Y) \in Q) \Longrightarrow (s, X) \in Q$

5. $(s, \Uparrow) \in Q \Longrightarrow (st, \delta) \in Q$

$Q\ \underline{\text{after}}\ s\ =\ \{(t, \delta) \mid (st, \delta) \in Q\}$, $\quad Q^0 = \{a \in \Sigma \mid \langle a \rangle \in \text{dom}Q\}$

Technical Notes

The space $\mathbb{M}$ of all processes is a complete semi-lattice under the reverse inclusion order $A \sqsubseteq B \Longleftrightarrow A \supseteq B$. This order is naturally interpreted as $A \sqsupseteq B \equiv$ B is more non-deterministic than A. The bottom or minimal element of $\mathbb{M}$ is $\Sigma^* \times (\mathbb{P}(\Sigma) \times \{\Uparrow\})$ (called CHAOS), one of whose many realisations is a process which can diverge immediately.

There is a natural map from boundedly non-deterministic synchronisation trees to $\mathbb{M}$, and a not-quite-so natural one from arbitrary synchronisation trees to $\mathbb{M}$. CSP can be given operational semantics which are in each case congruent to the abstract semantics given below.

The failure sets model gives a very expressive language for specification, since it regulates not only traces but also liveness (via divergence and refusals).

CSP can be given a semantics over $\mathbb{M}$:

$\underline{\text{abort}}\ =\ \{(\langle\rangle, X) \mid X \subseteq \Sigma\}$
the process which does nothing at all

$\underline{\text{skip}}\ =\ \{(\langle\rangle, X), (\langle\checkmark\rangle, Y) \mid \checkmark \notin X\}$
the process which immediately terminates successfully

$a \rightarrow A\ =\ \{(\langle\rangle, X) \mid a \notin X\} \cup \{(\langle a \rangle s, \delta) \mid (s, \delta) \in A\}$
communicates 'a' and then behaves like A

$a.x{:}T \rightarrow A(x)\ =\ \{(\langle\rangle, X) \mid a.T \cap X = \emptyset\} \cup \{(\langle a.b \rangle s, \delta) \mid (b \in T \ \&\ (s, \delta) \in A(b)\}$
inputs a value b named by 'a', then behaves like A(b).

$A \sqcap B\ =\ A \cup B$
behaves like A or B at the process' choice

$A \,\square\, B \;=\; \{(\langle\rangle, X \cap Y) \mid (\langle\rangle, X) \in A \;\&\; (\langle\rangle, Y) \in B\} \cup \{(s, \delta) \mid s \neq \langle\rangle \;\&\; (s, \delta) \in A \cup B\}$
$\quad \cup \{(s, \delta) \mid (s, \uparrow) \in A \cup B\}$

behaves like A or B giving the <u>environment</u> the choice of <u>first</u> steps.

$A;B \;=\; \{(s, X) \mid s \text{ does not contain } \surd, \text{ and } (s, X \cup \{\surd\}) \in A\}$
$\quad \cup \{(st, \delta) \mid s \text{ does not contain } \surd, \text{ and } (s, \uparrow) \in A\}$
$\quad \cup \{(st, \delta) \mid s \text{ does not contain } \surd, \; (s\langle\surd\rangle, \phi) \in A, \; (t, \delta) \in B\}$

behaves like A until it terminates successfully, then like B.

$A \backslash b \;=\; \{(s \backslash b, X) \mid (s, X \cup \{b\}) \in A\}$
$\quad \cup \{((s \backslash b)t, \delta) \mid (s, \uparrow) \in A\}$
$\quad \cup \{((s \backslash b)t, \delta) \mid \forall n \; (s \langle b \rangle^n, \phi) \in A\}$

where $\langle\rangle \backslash b = \langle\rangle$

$s\langle a \rangle \backslash b = (s \backslash b)\langle a \rangle \quad \text{if } a \neq b$

$\qquad\qquad = s \backslash b \qquad\qquad \text{if } a = b$

hides the event 'b' in A (note the divergence introduced by an infinite sequence of b's)

$A \backslash X \;=\; (\ldots(A \backslash b_1)\ldots.) \backslash b_n$, where $X = \{b_1 \ldots. b_n\}$, is any finite set.

$(A_X \|_Y B) \;=\; \{(s, (U \cap X) \cup (V \cap Y) \cup Z) \mid s \in (X \cup Y)^*$
$\quad \& \; (s \upharpoonright X, U) \in A \;\&\; (s \upharpoonright Y, V) \in B \;\&\; Z \cap (X \cup Y) = \phi\}$
$\quad \cup \{(st, \delta) \mid s \in (X \cup Y)^* \;\&\; s \upharpoonright X \in \mathrm{dom}A \;\&\; s \upharpoonright Y \in \mathrm{dom}B$
$\quad \& \; ((s \upharpoonright X, \uparrow) \in A \vee (s \upharpoonright Y, \uparrow) \in B)\}$

A, with alphabet X, operates in parallel with B which has alphabet Y.

$(A \| B)$ will be used as an abbreviation in the case where both A and B have as their alphabets the totality of symbols they can ever use.

$\backslash X$, $\|$ and easy alphabetical transformations can be used to derive operators.

$\gg$ which expects both arguments to have alphabet $?T \cup !T$, and connects the ! channel of its left-hand argument to the ? channel of its right-hand argument. Internal communication is hidden.

$\underset{a}{\times}$ which expects the intersection of the alphabets of its arguments to be $a?T \cup a!T$. The outputs ($a!$) of each argument are connected to the input ($a?$) of the other. Internal connection is hidden.

There are of course many theorems connecting these operators.

References

1. Hoare, C.A.R., Brookes, S.D., Roscoe, A.W. "A Theory of Communicating Sequential Processes". Oxford PRG monograph PRG-16 (1981) and JACM July 1984.

2. Brookes, S.D. Oxford D.Phil thesis, 1983 (SDB)

3. Roscoe, A.W. Oxford D.Phil thesis, 1982 (AWR)

4. Brookes, S.D., Roscoe, A.W. "An Improved Failure-Sets Model for Communicating Processes" To appear in Proceedings of NSF-SERC Seminar on concurrency, Springer-Verlag LNCS. Available as a Carnegie-Mellon Technical Report.

Note

The failures model has appeared in several forms. The original version [1,2,3] was deficient in its treatment of divergence and was improved in [2,3]. The version described in this note is the improved form from [2]; this differs in presentation from the "standard" improved form of [4] but is easily seen to be isomorphic to it.