Postgraduate Course: Process Algebra

Communicating Sequential Process

Chapter 3 Nondeterminism

Huibiao Zhu

Software Engineering Institute, East China Normal University

Introduction

(1) Processes are called **deterministic**, because whenever there is more than one event possible, the choice between them is determined externally by the environment of the process.
It is determined either in the sense that the environment can actually make the choice, or in the weaker sense that the environment can observe which choice has been made at the very moment of the choice.

(2) There is nothing mysterious about this kind of **nondeterminism**: it arises from a deliberate decision to ignore the factor which influence the selection.

(3) We need to introduce the concept of **nondeterminism**.

Nondeterministic or

1. Definition: If P and Q are processes, then we introduce the notation $P \sqcap Q$ (P or Q) to denote a process which behaves either like P or like Q, where the selection between them is made arbitrarily, without the knowledge of control of the external environment.

$$\alpha(P \sqcap Q) = \alpha P = \alpha Q$$

Examples:

(1) A change-giving machine which always gives the right change in one of two combinations

$$CH5D = (in5p \rightarrow ((out1p \rightarrow out1p \rightarrow out1p \rightarrow out2p \rightarrow CH5D) \sqcap (out2p \rightarrow out1p \rightarrow out2p \rightarrow CH5D)))$$

Notes:

(1) $\sqcap$ is not intended as a useful operator for implementing a process. It would be very foolish to build both P and Q, put them in a black bag, make an arbitrary choice between them, and then throw the other one away!

(2) The main advantage of nondeterminism is in specifying a process. A process specified as $(P \sqcap Q)$ can be implemented either by building P or by building Q.

2. Laws

L1 $P \sqcap P = P$ (idempotence)

L2 $P \sqcap Q = Q \sqcap P$ (symmetry)

L3 $P \sqcap (Q \sqcap R) = (P \sqcap Q) \sqcap R$ (associativity)

Several distributed laws

L4 $x \to (P \sqcap Q) = (x \to P) \sqcap (x \to Q)$

L5 $(x : B \to (P(x) \sqcap Q(x))) = (x : B \to P(x)) \sqcap (x : B \to Q(x))$

L6 $P||(Q \sqcap R) = (P||Q) \sqcap (P||R)$

L7 $(P \sqcap Q)||R = (P||R) \sqcap (Q||R)$

L8 $f(P \sqcap Q) = f(P) \sqcap f(Q)$

Remark: The recursion operator is not distributive, except in the trivial case where the operands of $\sqcap$ are identical. For example,

$$P = \mu X \bullet ((a \to X) \sqcap (b \to X))$$

$$Q = (\mu X \bullet (a \to X)) \sqcap (\mu X \bullet (b \to X))$$

(a) P can make an independent choice between a and b on each iteration, so its traces include $\langle a, b, b, a, b \rangle$

(b) Q must make a choice between always doing a and always doing b, so its traces do not include the one displayed above. However, P may choose always to do a or always to do b, so

$$traces(Q) \subseteq traces(P)$$

Note:

In view of laws L1 to L3 it is useful to introduce a multiple-choice operator. Let S be a finite nonempty set

$$S = i, j, ..., k$$

Then we define

$$\sqcap_{x:S} P(x) = P(i) \sqcap P(j) \sqcap ... \sqcap P(k)$$

$\sqcap_{x:S}$ is meaningless when S is either empty or infinite.

3. Traces

L1 $traces(P \sqcap Q) = traces(P) \cup traces(Q)$

L2
$$\begin{aligned} (P \sqcap Q)/s &= Q/s && \text{if } s \in (traces(Q) - traces(P)) \\ &= P/s && \text{if } s \in (traces(P) - traces(Q)) \\ &= (P/s) \sqcap (Q/s) && \text{if } s \in (traces(P) \cap traces(Q)) \end{aligned}$$

General choice

1. Introduction

(a) The environment of $(P \sqcap Q)$ has no control or even knowledge of the choice that is made between P and Q.

(b) Another operation $(P \Box Q)$ is introduced, for which the environment can control which of P and Q will be selected, provided that this control is exercised on the very first action.

$$\alpha(P \Box Q) = \alpha P = \alpha Q$$

(c) Intuitive understanding of $(P \Box Q)$:

(c.1) If this action is not a possible first action of P, Q will be selected.

(c.2) If Q cannot engage initially in the action, P will be selected.

(c.3) If the first action is possible for both P and Q, the choice between them is nondeterministic.

Analysis:

(1) In the case that no initial event of P is also possible for Q, the general choice operator is the same as the $|$ operator, which has been used hitherto to represent choice between different events

$$(c \to P \Box d \to Q) = (c \to P | d \to Q) \qquad \text{if } c \neq d.$$

(2) However if the initial events are the same, $(P \Box Q)$ degenerates to nondeterministic choice $(c \to P \Box c \to Q) = (c \to P \sqcap c \to Q)$

2. Laws

The algebraic laws for $\Box$ are similar to those for $\sqcap$, and for the same reasons.

L1-L3 $\Box$ is idempotent, symmetric, and associative.

L4 $P \Box STOP = P$

L5
$$(x : A \to P(x)) \Box (y : B \to Q(y)) =$$
$$(z : (A \cup B) \to$$
$$\text{if } z \in (A - B) \text{ then}$$
$$P(z)$$
$$\text{else if } z \in (B - A) \text{ then}$$
$$Q(z)$$
$$\text{else if } z \in (A \cap B) \text{ then}$$
$$(P(z) \sqcap Q(z)))$$

L6 $P \Box (Q \sqcap R) = (P \Box Q) \sqcap (P \Box R)$

L7 $P \Box (Q \sqcap R) = (P \sqcap Q) \Box (P \sqcap R)$

3. Traces

Every trace of $(P \Box Q)$ must be a trace of P or a trace of Q, and conversely

L1 $traces(P \Box Q) = traces(P) \cup traces(Q)$

The next law is slightly different from the corresponding law for $\sqcap$.

L2
$$\begin{aligned}(P \Box Q)/s &= P/s && \text{if } s \in traces(P) - traces(Q)\\ &= Q/s && \text{if } s \in traces(Q) - traces(P)\\ &= (P/s) \sqcap (Q/s) && \text{if } s \in traces(P) \cap traces(Q)\end{aligned}$$

Refusals

1. Introduction

Why do we need to introduce *refusals*?

Aim: How can we distinguish between $(P \sqcap Q)$ and $(P \Box Q)$? They cannot be distinguished by their traces.

Solution: It is possible to put them in an environment in which $(P \sqcap Q)$ can deadlock at its first step, but $(P \Box Q)$ cannot. For example let $x \neq y$ and

$$P = (x \to P),\ Q = (y \to Q),\ \alpha P = \alpha Q = \{x, y\}$$

Then

$$(P \Box Q)||P = (x \to P) = P$$

but

$$(P \sqcap Q)||P = (P||P) \sqcap (Q||P) = P \sqcap STOP$$

2. Definition of Refusals

In general, let X be a set of events which are offered initially by the environment of a process P, which in this context we take to have the same alphabet as P. If it is possible for P to deadlock on its first step when placed in this environment, we say that X is a refusal of P. The set of all such refusals of P is denoted as $refusals(P)$

Note: Intuitive understanding of $refusals(P)$?

3. The difference between deterministic and nondeterministic processes

(a) P is deterministic $\Rightarrow$ ($X \in refusals(P) \equiv (X \cap P^0 = \{\})$)

(b) P is deterministic $\equiv$

$$\forall s : traces(P) \bullet (X \in refusals(P/s) \equiv (X \cap (P/s)^0 = \{\}))$$

(c) A nondeterministic process is one that does not enjoy this property.

3. Laws

L1 $refusals(STOP_A) =$ (including A itself)

L2 $refusals(c \to P) = \{X | X \subseteq (\alpha P - \{c\})\}$

L3 $refusals(x : B \to P(x)) = \{X | X \subseteq (\alpha P - B)\}$

If P can refuse X , so will $P \sqcap Q$ if P is selected. Similarly every refusal of Q is also a possible refusal of $(P \sqcap Q)$. These are its only refusals.

L4 $refusals(P \sqcap Q) = refusals(P) \cup refusals(Q)$

A converse argument applies to $(P \Box Q)$. If X is not a refusal of P, then P cannot refuse X , and neither can $(P \Box Q)$. Similarly, if X is not a refusal of Q, then it is not a refusal of $(P \Box Q)$. However, if both P and Q can refuse X , so can $(P \Box Q)$

L5 $refusals(P \Box Q) = refusals(P) \cap refusals(Q)$

If P can refuse X and Q can refuse Y , then their combination $(P||Q)$ can refuse all events refused by P as well as all events refused by Q, i.e., it can refuse the union of the two sets X and Y.

L6 $refusals(P||Q) = \{X \cup Y | X \in refusals(P) \ \wedge \ Y \in refusals(Q)\}$

L7 $refusals(f(P)) = \{f(x) | X \in refusals(P)\}$

Some general laws:

L8 $X \in refusals(P)) \Rightarrow X \subseteq \alpha P$

L9 $\{\} \in refusals(P)$

L10 $(X \cup Y) \in refusals(P)) \Rightarrow X \in refusals(P)$

L11 $X \in refusals(P) \Rightarrow (X \cup \{x\}) \in refusals(P) \ \vee \ \langle x \rangle \in traces(P)$

Concealment

1. Definition:

If C is a finite set of events to be concealed in this way, then $P \setminus C$ is a process which behaves like P, except that each occurrence of any event in C is concealed. Clearly it is our intention that
$\alpha(P \setminus C) = (\alpha P) - C$

Example:

(1) A noisy vending machine (2.3 X1) can be placed in a soundproof box

$$NOISYVM \setminus \{clink, clunk\}$$

Its unexercised capability of dispensing toffee can also be removed from its alphabet, without affecting its actual behaviour. The resulting process is equal to the simple vending machine

$$VMS = NOISYVM \setminus \{clink, clunk, toffee\}$$

(2) Let $\alpha P = \{a, c\}$ $\qquad P = (a \to c \to P)$

$\alpha Q = \{b, c\}$ $\qquad Q = (c \to b \to Q)$.

The action c in the alphabet of both P and Q is now regarded as an internal action, to be concealed

$$(P||Q) \setminus \{c\} = (a \to c \to \mu X \bullet (a \to b \to c \to X \mid b \to a \to c \to X)) \setminus \{c\}$$

$$= a \to \mu X \bullet (a \to b \to X \mid b \to a \to X)$$

2. Laws

L1 $P \setminus \{\} = P$

L2 $(P \setminus B)\ C = P \setminus (B \cup C)$

L3 $(P \sqcap Q)\ C = (P \setminus C) \sqcap (Q \setminus C)$

L4 $STOP_{A \setminus C} = STOP_{A - C}$

L5 $(x \to P) \setminus C = x \to (P \setminus C) \qquad \text{if } x \notin C$

$\qquad\qquad\qquad\qquad = P \setminus C \qquad\qquad\quad \text{if } x \in C$

L6 If $\alpha P \cap \alpha Q \cap C = \{\}$, then $(P || Q) \setminus C = (P \setminus C) || (Q \setminus C)$

L7 $f(P \setminus C) = f(P) \setminus f(C)$

L8 If $B \cap C = \{\}$, then $(x : B \to P(x)) \setminus C = (x : B \to (P(x) \setminus C))$

L9 If $B \subseteq C$, and B is finite and not empty, then

$$(x : B \to P(x)) \setminus C = \sqcap_{x \in B}(P(x) \setminus C)$$

Remark: In the intermediate case, when some of the initial events are concealed and some are not, the situation is rather more complicated. Consider the process

$$(c \to P | d \to Q) \setminus C, \quad \text{where } c \in C,\ d \notin C$$
$$= (P \setminus C) \sqcap ((P \setminus C) \Box (d \to (Q \setminus C)))$$

Similar reasoning justifies the more general law

L10 If $C \cap B$ is finite and non-empty, then

$$(x : B \to P(x)) \setminus C$$
$$= Q \sqcap (Q \Box (x : (B - C) \to P(x))), \text{ where } Q = \sqcap_{x \in B \cap C} P(x) \setminus C$$

Remark: Note that $\backslash C$ does not distribute backwards through $\Box$. A counterexample is

$$
\begin{array}{lll}
 & (c \to STOP \Box d \to STOP) \setminus \{c\} & \{\mathbf{L10}\} \\
= & STOP \sqcap (STOP \Box (d \to STOP)) & \{3.3\mathbf{L4}\} \\
= & STOP \sqcap (d \to STOP) & \\
\neq & d \to STOP & \\
= & STOP \Box (d \to STOP) & \\
= & ((c \to STOP) \setminus \{c\}) \Box ((d \to STOP) \setminus \{c\}) &
\end{array}
$$

3. Alphabet Extension

We can also define an operation which extends the alphabet of a process P by inclusion of symbols of a set B

$$\alpha(P_{+B}) = \alpha P \cup B$$

$$P_{+B} = (P || STOP_B) \quad \text{provided } B \cap \alpha P = \{\}$$

L11 $traces(P_{+B}) = traces(P)$

L12 $P_{+B} \setminus B = P$

Remark: Concealment may raise divergence.

$$(\mu X : A \bullet (c \to X)) \setminus \{c\}$$
$$= \mu X : (A - \{c\}) \bullet ((c \to X_{+\{c\}}) \setminus \{c\})$$
$$= \mu X : (A - \{c\}) \bullet X \qquad \{\text{by } L12, L5\}$$

4. Traces

L1 $traces(P \setminus C) = \{t \upharpoonright (\alpha P - C) \mid t \in traces(P)\}$

provided that $\forall s : traces(P) \bullet \neg diverges(P/s, C)$

Remark: The condition $diverges(P, C)$ means that P diverges immediately on concealment of C.

$$diverges(P, C) = \forall n \bullet \exists s : traces(P) \cap C^* \bullet \#s > n$$

L2 $(P \setminus C)/s = (\sqcap_{t \in T} P/t) \setminus C$

where $T = traces(P) \cap \{t | t \upharpoonright (\alpha P - C) = s\}$

provided that T is finite and $s \in traces(P \setminus C)$

Interleaving

1. Introduction:

It is sometimes useful to join processes with the same alphabet to operate concurrently without directly interacting or synchronizing with each other.

(a) If one of the processes cannot engage in the action, then it must have been the other one;

(b) But if both processes could have engaged in the same action, the choice between them is nondeterministic.

$P|||Q$ P interleave Q

and its alphabet is defined by the usual stipulation

$$\alpha(P|||Q) = \alpha P = \alpha Q$$

2. Examples:

(1) A vending machine that will accept up to two coins before dispensing up to two chocolates (1.1.3 X6)

$$(VMS|||VMS) = VMS2$$

(2) A footman made from four lackeys, each serving only one philosopher at a time (see Section 2.5.4).

3. Laws

L1-L3 $|||$ is associative, symmetric, and distributes through $\sqcap$.

L4 $P|||STOP = P$

L5 $P|||RUN = RUN$ provided P does not diverge

L6 $(x \to P)|||(y \to Q) = (x \to (P|||(y \to Q))\Box y \to ((x \to P)|||Q))$

L7 If $P = (x : A \to P(x))$ and $Q = (y : B \to Q(y))$, then

$$P|||Q = (x : A \to (P(x)|||Q)\Box y : B \to (P|||Q(y)))$$

Remark:

Note that $|||$ does not distribute through $\Box$. This is shown by the counterexample (where $b \neq c$)

$$\begin{aligned}
&((a \to STOP)|||(b \to Q\Box c \to R))/\langle a\rangle \\
&= (b \to Q\Box c \to R) \\
&\neq ((b \to Q) \sqcap (c \to R)) \\
&= ((a \to STOP\Box b \to Q)|||(a \to STOP\Box c \to R))/\langle a\rangle
\end{aligned}$$

L6 and L7 state that it is the environment which chooses between the initial events offered by the operands of $|||$. Nondeterminism arises only when the chosen event is possible for both operands.

Example: Let Let $R = (a \to b \to R)$, then

$$
\begin{aligned}
& (R|||R) && \{\text{L6}\} \\
= \; & (a \to ((b \to R)|||R) \Box a \to (R|||(b \to R))) && \\
= \; & a \to ((b \to R)|||R) \sqcap (R|||(b \to R)) && \{\text{L2}\} \\
= \; & a \to ((b \to R)|||R) &&
\end{aligned}
$$

Also

$$
\begin{aligned}
& (b \to R)|||R && \{\text{L6}\} \\
= \; & (a \to ((b \to R)|||(b \to R)) \Box b \to (R|||R)) && \{\text{as shown above}\} \\
= \; & (a \to (b \to ((b \to R)|||R)) && \{\text{The recursion} \\
& \quad \Box b \to (a \to ((b \to R)|||R))) && \text{is guarded}\} \\
= \; & \mu X \bullet (a \to b \to X \Box b \to a \to X) &&
\end{aligned}
$$

4. Traces and refusals

A trace of $(P|||Q)$ is an arbitrary interleaving of a trace from P with a trace from Q.

L1 $traces(P|||Q) =$

$$\{s|\exists t : traces(P) \bullet \exists u : traces(Q) \bullet s \ \ interleaves\ (t,u)\}$$

$(P|||Q)$ can engage in any initial action possible for either P or Q; and it can therefore refuse only those sets which are refused by both P and Q

L2 $refusals(P|||Q) = refusals(P \Box Q)$

The behaviour of $(P|||Q)$ after engaging in the events of the trace s is defined by the rather elaborate formula

L3 $(P|||Q)/s = \sqcap_{(t,u)\in T}(P/t)|||(Q/u)$

where

$$T = \{(t,u)|t \in traces(P) \wedge u \in traces(Q) \wedge s\ interleaves\ (t,u)\}$$

Specification

1. Introduction

Refusal sets are introduced as one of the important indirectly observable aspects of the behaviour. In specifying a process, we therefore need to describe the desired properties of its refusal sets as well as its traces.

$$P \textbf{ sat } S(tr, ref)$$

its meaning:

$$\forall tr, ref \bullet tr \in traces(P) \wedge ref \in refusals(P/tr) \Rightarrow S(tr, ref)$$

2. Examples:

(1) When a vending machine has ingested more coins than it has dispensed chocolates, the customer specifies that it must not refuse to dispense a chocolate

$$FAIR = (tr \downarrow choc < tr \downarrow coin) \Rightarrow choc \notin ref)$$

It is implicitly understood that every trace tr and every refusal ref of the specified process at all times should satisfy this specification.

(2) When a vending machine has given out as many chocolates as have been paid for, the owner specifies that it must not refuse a further coin

$$PROFIT1 = (tr \downarrow choc = tr \downarrow coin \Rightarrow coin \notin ref)$$

(3) A simple vending machine should satisfy the combined specification

$$NEWVMSPEC = FAIR \wedge PROFIT \wedge (tr \downarrow choc \leq tr \downarrow coin)$$

(4) If desired, one may place a limit on the balance of coins which may be accepted in a row

$$ATMOST2 = (tr \downarrow coin - tr \downarrow choc \leq 2)$$

(5) If desired, one can insist that the machine accept at least two coins in a row whenever the customer offers them

$$ATLEAST2 = (tr \downarrow coin - tr \downarrow choc < 2 \Rightarrow coin \notin ref$$

(6) The process $STOP$ refuses every event in its alphabet. The following predicate specifies that a process with alphabet A will never stop

$$NONSTOP = (ref \neq A)$$

Remark: How to describe a process satisfying $NEWVMSPEC$ will never stop?

$$NEWVMSPEC \Rightarrow ref \neq \{coin, choc\}$$

2. Proofs

L1 If P **sat** S and Q **sat** T,

then $(P \sqcap Q)$ **sat** $(S \vee T)$

L2A $STOP_A$ **sat** $(tr = \langle\rangle \wedge ref \subseteq A)$

Note: This law is identical to the one for deterministic process.

$$STOP \textbf{ sat } tr = \langle\rangle$$

L2B If P **sat** $S(tr)$

then $(c \rightarrow P)$ **sat** $((tr = \langle\rangle \wedge c \notin ref) \vee (tr_0 = c \wedge S(tr')))$

L2 If $\forall x : B \bullet P(x)$ **sat** $S(tr, x)$

then $(x : B \rightarrow P(x))$ **sat**

$$((tr = \langle\rangle \wedge (B \cap ref = \{\})\ \vee\ (tr_0 \in B \wedge S(tr', tr_0))))$$

L3 If P **sat** $S(tr, ref)$ and Q **sat** $T(tr, ref)$

and neither P nor Q diverges

then $(P||Q)$ **sat**

$(\exists X, Y, ref \bullet ref = (X \cup Y) \wedge S(tr \upharpoonright \alpha P, X) \wedge T(tr \upharpoonright \alpha Q, Y))$

L4 If P **sat** $S(tr, ref)$

then $f(P)$ **sat** $S(f^{-1*}(tr), f^{-1}(ref))$ provided f is one-one.

L5 If P **sat** $S(tr, ref)$ and Q **sat** $T(tr, ref)$

and neither P nor Q diverges

then $(P \Box Q)$ **sat** (if $tr = \langle\rangle$ then $(S \wedge T)$ else $(S \vee T)$)

L6 If P **sat** $S(tr)$ and Q **sat** $T(tr)$

and neither P nor Q diverges

then $(P|||Q)$ **sat** $(\exists s, t \bullet (tr$ interleaves $(s, t) \wedge S(s) \wedge T(t)))$

L7 If P **sat** $(NODIV \wedge S(tr, ref))$

then $(P \setminus C)$ **sat** $\exists s \bullet tr = s \upharpoonright (\alpha P - C) \wedge S(tr, ref \cup C)$

where $NODIV$ states that the number of hidden symbols that can occur is bounded by some function of the non-hidden symbols that have occurred

$$NODIV = \#(tr \upharpoonright C) \leq f(tr \upharpoonright (\alpha P - C))$$

where f is some total function from traces to natural numbers.

L8 If $S(0)$ and $(X$ **sat** $S(n))) \Rightarrow F(X)$ **sat** $S(n+1))$

then $(\mu X \bullet F(X))$ **sat** $(\forall n \bullet S(n))$

Divergence

1. Introduction:

(a) Why do we have the restriction of guardedness when defining equations previous? (*for the aim of achieving a single solution*)

(b) What does the infinite recursion mean?

$$\mu X \bullet X$$

(c) The introduction of concealment means that an apparently guarded recursion is not constructive. For example, consider the equation

$$X = c \to (X \setminus \{c\})_{+\{c\}}$$

This has as solutions both $(c \to STOP)$ and $(c \to a \to STOP)$

(d) How can we give a (possibly nondeterministic) meaning to every expression of the form $\mu X \bullet F(X)$? where F can be non-guarded.

2. Laws

L1 $P \sqcap CHAOS = CHAOS$

Remark: A function of processes that yields $CHAOS$ if any of its arguments is $CHAOS$ is said to be strict.

L2 The following operations are strict

$$/s, \quad ||, \quad f\ , \quad \Box, \quad \backslash C, \quad |||, \quad \text{and} \quad \mu X$$

L3 $CHAOS \neq (a \rightarrow CHAOS)$

L4 $traces(CHAOS_A) = A^*$

L5 $refusals(CHAOS_A) =$ all subsets of A.

3. Divergence

A divergence of a process is defined as any trace of the process after which the process behaves chaotically. The set of all divergences is defined

$$divergences(P) = \{s \mid s \in traces(P) \wedge (P/s) = CHAOS_{\alpha P}\}$$

L1 $divergences(P) \subseteq traces(P)$

L2 $s \in divergences(P) \wedge t \in (\alpha P)^* \Rightarrow (s^\frown t) \in divergences(P)$

L3 $s \in divergences(P) \wedge X \subseteq \alpha P) \Rightarrow X \in refusal(P/s)$

L4 $divergences(STOP) = \{\}$

L5 $divergences(CHAOS_A) = A^*$

L6 $divergences(x : B \rightarrow P(x))$

$$= \{\langle x \rangle^\frown s \mid x \in B \wedge s \in divergences(P(x))\}$$

Any divergence of P is also a divergence of $(P \sqcap Q)$ and of $(P \Box Q)$

L7 $divergences(P \sqcap Q) = divergences(P \Box Q)$

$$= divergences(P) \cup divergences(Q)$$

Since $||$ is strict, a divergence of $(P||Q)$ starts with a trace of the nondivergent activity of both P and Q, which leads to divergence of either P or of Q (or of both)

L8 $divergences(P||Q) =$

$$\{s^\frown t \mid t \in (\alpha P \cup \alpha Q)^* \wedge$$
$$((s \upharpoonright \alpha P \in divergences(P) \wedge s \upharpoonright \alpha Q \in traces(Q)) \ \vee$$
$$s \upharpoonright \alpha P \in traces(P) \wedge s \upharpoonright \alpha Q \in divergences(Q))\}$$

A similar explanation applies to $|||$

L9 $divergences(P|||Q) =$

$$\{u \mid \exists s, t \bullet u \text{ interleaves } (s, t) \wedge$$
$$((s \in divergences(P) \wedge t \in traces(Q)) \ \vee$$
$$(s \in traces(P) \wedge t \in divergences(Q)))\}$$

Divergences of a process resulting from concealment include traces derived from the original divergences, plus those resulting from the attempt to conceal an infinite sequence of symbols

L10 $divergences(P \setminus C) =$

$$\{(s \upharpoonright (\alpha P - C))^\frown t \mid$$
$$t \in (\alpha P - C)^* \wedge$$
$$(s \in divergences(P) \ \vee$$
$$(\forall n \bullet \exists u \in C^* \bullet \#u > n \wedge (s^\frown u) \in traces(P)))\}$$

A process defined by symbol change diverges only when its argument diverges

L11 $divergences(f(P)) = \{f^*(s) \mid s \in divergences(P)\}$

provided f is one-one.

Mathematical theory of non-deterministic processes

1. Introduction

(a) From the concept to refusals to the concept of failures

$$failures(P) = \{(s, X) \mid s \in traces(P) \wedge X \in refusals(P/s)\}$$

(b) How to calculate traces and refusals from failures?

$$\begin{aligned} traces(P) &= \{s \mid \exists X \bullet (s, X) \in failures(P)\} \\ &= domain(failures(P)) \end{aligned}$$

$$refusals(P) = \{X | (\langle\rangle, X) \in failures(P)\}$$

2. Mathematical model

D0 A process is a triple (A, F, D), where:

A is any set of symbols (for simplicity finite),

F is a relation between A^* and PA,

D is a subset of A^*

provided that they satisfy the following conditions

C1 $(\langle\rangle, \{\}) \in F$

C2 $(s^\frown t, X) \in F \Rightarrow (s, \{\}) \in F$

C3 $(s, Y) \in F \wedge X \subseteq Y \Rightarrow (s, X) \in F$

C4 $(s, X) \in F \wedge x \in A \Rightarrow (s, X \cup \{x\}) \in F \vee (s^\frown\langle x\rangle, \{\}) \in F$

C5 $D \subseteq domain(F)$

C6 $s \in D \wedge t \in A^* \Rightarrow s^\frown t \in D$

C7 $s \in D \wedge X \subseteq A \Rightarrow (s, X) \in F$

4. Mathematical Definition of Processes

$CHAOS_A$ is the worst process. This is the largest process with alphabet A.

D1 $CHAOS_A = (A,\ (A^* \times PA),\ A^*)$

$STOP_A$ never does anything, can refuse everything, and has no divergences.

D2 $STOP_A = (A, \{\langle\rangle\} \times PA, \{\})$

The the nondeterministic choice of two processes, the resulting process can fail or diverge in all cases that either of its two operands can do so.

D3 $(A, F1, D1) \sqcap (A, F2, D2) = (A,\ F1 \cup F2,\ D1 \cup D2)$

The following are for the consideration of alphabets.

D4 If $\alpha P(x) = A$ for all x and $B \subseteq A$
then $\alpha(x : B \to P(x)) = A$

D5 $\alpha(P||Q) = (\alpha P \cup \alpha Q)$

D6 $\alpha(f(P)) = f(\alpha(P))$

D7 $\alpha(P \Box Q) = \alpha(P|||Q) = \alpha P$ provided $\alpha P = \alpha Q$

D8 $\alpha(P \setminus C) = \alpha P - C$

The following are the failures considerations of processes.

D9 $failures(x : B \to P(x)) =$

$$\{(\langle\rangle, X) \mid X \subseteq (\alpha P - B)\} \cup$$
$$\{(\langle x\rangle^\frown s, X) \mid x \in B \wedge (s, X) \in failures(P(x))\}$$

D10 $failures(P||Q) =$

$$\{(s, X \cup Y) | s \in (\alpha P \cup \alpha Q)^* \wedge (s \upharpoonright \alpha P, X) \in failures(P) \wedge (s \upharpoonright \alpha Q, Y) \in failures(Q)\} \cup$$
$$\{(s, X) | s \in divergences(P||Q)\}$$

D11 $failures(f(P)) = \{(f^*(s), f(X)) \mid (s, X) \in failures(P)\}$

D12 $failures(P \Box Q) =$

$$\{(s, X) \mid (s = \langle\rangle \wedge (s, X) \in failures(P) \cap failures(Q)) \ \vee$$
$$(s \neq \langle\rangle \wedge (s, X) \in failures(P) \cup failures(Q))\} \ \cup$$
$$\{(s, X) \mid s \in divergences(P \Box Q)\}$$

D13 $failures(P|||Q) =$

$$\{(s, X) \mid \exists t, u \bullet s \ interleave(t, u) \wedge (t, X) \in failures(P)$$
$$\wedge (u, X) \in failures(Q) \ \} \cup$$
$$\{(s, X) \mid s \in divergences(P|||Q)\}$$

D14 $failures(P \setminus C) =$

$$\{(s \upharpoonright (\alpha P - C), \ X) \mid (s, X \cup C) \in failures(P)\} \cup$$
$$\{(s, X) \mid s \in divergences(P \setminus C)\}$$

5. The Definition of Fixed Point

$P \sqsubseteq Q$ now means that Q is equal to P or better in the sense that it is less likely to diverge and less likely to fail.

D15 $(A, F1, D1) \sqsubseteq (A, F2, D2) \equiv (F2 \subseteq F1 \wedge D2 \subseteq D1)$

$CHAOS$ can do anything at any time, and can refuse to do anything at any time. True to its name, it is the least predictable and controllable of all processes; or in short the worst

L1 $CHAOS \sqsubseteq P$

For a chain of processes, the limit operation defined in terms of the intersections of descending chains of failures and divergences

D16 $\sqcup_{n \geq 0}(A, F_n, D_n) = (A, \cap_{n \geq 0} F_n, \cap_{n \geq 0} D_n)$

provided $(\forall n \geq 0 \bullet F_{n+1} \subseteq F_n \wedge D_{n+1} \subseteq D_n)$

The μ operation is defined in the same way as for deterministic processes (2.8.2 L7). It requires that $CHAOS$ be used in place of $STOP$

D17 $\mu X : A \bullet F(X) = \sqcup_{i \geq 0} F^i(CHAOS_A)$