Postgraduate Course: Process Algebra

Communicating Sequential Process

Chapter 2 Concurrency

Huibiao Zhu

Software Engineering Institute, East China Normal University

Interaction

(1) When two processes are brought together to evolve concurrently, the usual intention is that they will interact with each other. These interactions may be regarded as events that require simultaneous participation of both the processes involved.

(2) For the time being, let us confine attention to such events, and ignore all others. Thus we will assume that the alphabets of the two processes are the same. Consequently, each event that actually occurs must be a possible event in the independent behavior of each process separately.

(3) For example, a chocolate can be extracted from a vending machine only when its customer wants it and only when the vending machine is prepared to give it.

Definition of Interaction

If P and Q are processes with the same alphabet, we introduce the notation $P \parallel Q$ to denote the process which behaves like the system composed of processes P and Q interacting in lock-step synchronization as described above.

Examples

X1 A greedy customer of a vending machine is perfectly happy to obtain a toffee or even a chocolate without paying. However, if thwarted in these desires, he is reluctantly prepared to pay a coin, but then he insists on taking a chocolate

$$\begin{aligned} GRCUST = \quad & (toffee \to GRCUST \\ & |choc \to GRCUST \\ & |coin \to choc \to GRCUST) \end{aligned}$$

When this customer is brought together with the machine $VMCT$ (1.1.3 X3) his greed is frustrated, since the vending machine does not allow goods to be extracted before payment. Similarly, $VMCT$ never gives a toffee, because the customer never wants one after he has paid

$$(GRCUST || VMCT) = \mu X \bullet (coin \to choc \to X)$$

X2 A foolish customer wants a large biscuit, so he puts his coin in the vending machine VMC. He does not notice whether he has inserted a large coin or a small one; nevertheless, he is determined on a large biscuit

$$\begin{aligned} FOOLCUST = \quad & (in2p \rightarrow large \rightarrow FOOLCUST \\ & |in1p \rightarrow large \rightarrow FOOLCUST) \end{aligned}$$

Unfortunately, the vending machine is not prepared to yield a large biscuit for only a small coin

$$(FOOLCUST||VMC) = \mu X \bullet (in2p \rightarrow large \rightarrow X|in1p \rightarrow STOP)$$

Laws for Interaction

L1	$P \| Q = Q \| P$
L2	$P \| (Q \| R) = (P \| Q) \| R$
L3A	$P \| STOP_{\alpha P} = STOP_{\alpha P}$
L3B	$P \| RUN_{\alpha P} = P$
L4A	$(c \to P) \| (c \to Q) = (c \to (P \| Q))$
L4B	$(c \to P) \| (d \to Q) = STOP \quad \text{if } c \neq d$

L4

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

X1 Let $P = (a \to b \to P | b \to P)$

and $Q = (a \to (b \to Q | c \to Q))$

Then

$(P||Q)$

$= a \to ((b \to P) || (b \to Q | c \to Q))$ [by L4A]

$= a \to (b \to (P||Q))$ [by L4A]

$= \mu X \bullet (a \to b \to X)$ [since the recursion is guarded.]

Traces for Interaction

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

L2 $(P||Q)/s = (P/s)||(Q/s)$

Concurrency

(1) Concurrent is the extension of Interaction. Its two operands P and Q have different alphabets.

(2) Notation: $P \parallel Q$. where $\alpha(P \parallel Q) = \alpha(P) \cup \alpha(Q)$

(3) Execution Mechanism:

(a) When such processes are assembled to run concurrently, events that are in both their alphabets require simultaneous participation of both P and Q.

(b) Events in the alphabet of P but not in the alphabet of Q are of no concern to Q. Such events may occur independently of Q whenever P engages in them.

(c) Another similar to (b)

Examples of Concurrency

(1) Let $NOISYVM = \{coin, choc, clink, clunk, toffee\}$, where clink is the sound of a coin dropping into the moneybox of a noisy vending machine, and clunk is the sound made by the vending machine on completion of a transaction. The noisy vending machine has run out of toffee

$NOISYVM = (coin \rightarrow clink \rightarrow choc \rightarrow clunk \rightarrow NOISYVM)$

The customer of this machine definitely prefers toffee; the curse is what he utters when he fails to get it; he then has to take a chocolate instead

$\alpha CUST = \{coin, choc, curse, toffee\}$

$CUST = (coin \rightarrow (toffee \rightarrow CUST | curse \rightarrow choc \rightarrow CUST))$

The result of the concurrent activity of these two processes is

$(NOISYVM || CUST) =$

$$\begin{aligned}\mu X \bullet (coin \rightarrow (&clink \rightarrow curse \rightarrow choc \rightarrow clunk \rightarrow X \\ &| curse \rightarrow clink \rightarrow choc \rightarrow clunk \rightarrow X))\end{aligned}$$

Laws of Concurrency

L1,2 $||$ is symmetric and associative

L3A $P||STOP_{\alpha P} = STOP_{\alpha P}$

L3B $P||RUN_{\alpha P} = P$

Let $a \in (\alpha P - \alpha Q)$, $b \in (\alpha Q - \alpha P)$ and $\{c, d\} \subseteq (\alpha P \cap \alpha Q)$. The following laws show the way in which P engages alone in a, Q engages alone in b, but c and d require simultaneous participation of both P and Q.

L4A $(c \to P)||(c \to Q) = c \to (P||Q)$

L4B $(c \rightarrow P)||(d \rightarrow Q) = STOP \qquad \text{if} c \neq d$

L5A $(a \rightarrow P)||(c \rightarrow Q) = a \rightarrow (P||(c \rightarrow Q))$

L5B $(c \rightarrow P)||(b \rightarrow Q) = b \rightarrow ((c \rightarrow P)||Q)$

L6 $(a \rightarrow P)||(b \rightarrow Q) = (a \rightarrow (P||(b \rightarrow Q))|b \rightarrow ((a \rightarrow P)||Q))$

These laws can be generalised to deal with the general choice operator.

$$\begin{array}{lrl}
\textsf{L7} & \text{Let} & P = (x : A \rightarrow P(x)) \\
 & \text{and} & Q = (y : B \rightarrow Q(y)) \\
 & \text{Then} & (P || Q) = (z : C \rightarrow P' || Q') \\
 & \text{where} & C = (A \cap B) \cup (A - \alpha Q) \cup (B - \alpha P) \\
 & \text{and} & P' = P(z) \qquad \text{if } z \in A \\
 & & P' = P \qquad \text{otherwise} \\
 & \text{and} & Q' = Q(z) \qquad \text{if} z \in B \\
 & & Q' = Q \qquad \text{otherwise.}
\end{array}$$

These laws permit a process defined by concurrency to be redefined without that operator, as shown in the following example.

Example

X1 Let $\alpha P = \{a, c\}$ and $P = (a \to c \to P)$

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

$P||Q$

$= (a \to c \to P)||(c \to b \to Q)$ [by definition]

$= a \to ((c \to P)||(c \to b \to Q))$ [by L5A]

$= a \to c \to (P||(b \to Q))$ [by L4A...‡]

Also

$P||(b \to Q)$

分别表示第一个动作

$= (a \to \underline{(c \to P)||(b \to Q)}|b \to \underline{(P||Q)})$ [by L6]

$= (a \rightarrow b \rightarrow ((c \rightarrow P)||Q)|b \rightarrow (P||Q))$ [by L5B]

$= (a \rightarrow b \rightarrow c \rightarrow (P||(b \rightarrow Q))|$

$\qquad b \rightarrow a \rightarrow c \rightarrow (P||(b \rightarrow Q)))$ [by ‡above]

$= \mu X \bullet (a \rightarrow b \rightarrow c \rightarrow X|b \rightarrow a \rightarrow c \rightarrow X)$ [since this is guarded]

Therefore

$(P||Q) = (a \rightarrow c \rightarrow X(a \rightarrow b \rightarrow c \rightarrow X|b \rightarrow a \rightarrow c \rightarrow X))$ by ‡above

Example: The Dining Philosophers

(1) Describing the example:

In ancient times, a wealthy philanthropist endowed a College to accommodate five eminent philosophers. Each philosopher had a room in which he could engage in his professional activity of thinking; there was also a common dining room, furnished with a circular table, surrounded by five chairs, each labelled by the name of the philosopher who was to sit in it. The names of the philosophers were PHIL0, PHIL1, PHIL2, PHIL3, PHIL4, and they were disposed in this order anticlockwise around the table. To the left of each philosopher there was laid a golden fork, and in the centre stood a large bowl of spaghetti, which was constantly replenished.

A philosopher was expected to spend most of his time thinking; but when he felt hungry, he went to the dining room, sat down in his own chair, picked up his own fork on his left, and plunged it into the spaghetti. But such is the tangled nature of spaghetti that a second fork is required to carry it to the mouth. The philosopher therefore had also to pick up the fork on his right. When he was finished he would put down both his forks, get up from his chair, and continue thinking. Of course, a fork can be used by only one philosopher at a time. If the other philosopher wants it, he just has to wait until the fork is available again.

Diagram for Illustrating the Example

Please see page 57 (i.e., 79) of the book.

Alphabets of the Example

(1) For $PHIL_i$, its alphabet is defined:

$$\begin{aligned}\alpha PHIL_i = \{\, & i.sits\ down, i.gets\ up,\\ & i.picks\ up\ fork.i, i.picks\ up\ fork.(i \oplus 1),\\ & i.puts\ down\ fork.i, i.puts\ down\ fork.(i \oplus 1)\}\end{aligned}$$

(2) For the i-th fork, its alphabet is defined as:

$$\begin{aligned}\alpha FORK_i = \{\, & i.picks\ up\ fork.i, (i \ominus 1).picks\ up\ fork.i,\\ & i.puts\ down\ fork.i, (i \ominus 1).puts\ down\ fork.i\}\end{aligned}$$

Behavior of the Example

(1) Apart from thinking and eating which we have chosen to ignore, the life of each philosopher is described as the repetition of a cycle of six events

$$
\begin{aligned}
PHIL_i = (&i.sits\ down \rightarrow \\
&i.picks\ up\ fork.i \rightarrow \\
&i.picksupfork.(i \oplus 1) \rightarrow \\
&i.puts\ down\ fork.i \rightarrow \\
&i.puts\ down\ fork.(i \oplus 1) \rightarrow \\
&i.gets\ up \rightarrow PHIL_i)
\end{aligned}
$$

(2) The role of a fork is a simple one; it is repeatedly picked up and put down by one of its adjacent philosophers (the same one on both occasions)

$FORK_i = (i.picks\ up\ fork.i \rightarrow i.puts\ down\ fork.i \rightarrow FORK_i$
$\quad |(i \ominus 1).picks\ up\ fork.i \rightarrow (i \ominus 1).puts\ down\ fork.i \rightarrow$
$FORK_i)$

(3) The behaviour of the whole College is the concurrent combination of the behaviour of each of these components

$$\begin{aligned} PHILOS &= (PHIL_0||PHIL_1||PHIL_2||PHIL_3||PHIL_4) \\ FORKS &= (FORK_0||FORK_1||FORK_2||FORK_3||FORK_4) \\ COLLEGE &= PHILOS||FORKS \end{aligned}$$

Variation of the Example

(1) **Variation 1**

An interesting variation of this story allows the philosophers to pick up their two forks in either order, or put them down in either order. Consider the behaviour of each philosophers hand separately. Each hand is capable of picking up the relevant fork, but both hands are needed for sitting down and getting up

$$\begin{aligned}
\alpha LEFT_i &= \{i.picks\ up\ fork.i, i.puts\ down\ fork.i,\\
&\quad i.sits\ down, i.gets\ up\}\\
\alpha RIGHT_i &= \{i.picks\ up\ fork.(i \oplus 1), i.puts\ down\ fork.(i \oplus 1),\\
&\quad i.sits\ down, i.gets\ up\}\\
LEFT_i &= (i.sits\ down \rightarrow i.picks\ up\ fork.i \rightarrow\\
&\quad i.puts\ down\ fork.i \rightarrow i.gets\ up \rightarrow LEFT_i)\\
RIGHT_i &= (i.sits\ down \rightarrow i.picks\ up\ fork.(i \oplus 1) \rightarrow\\
&\quad i.puts\ down\ fork.(i \oplus 1) \rightarrow i.gets\ up \rightarrow RIGHT_i)\\
PHIL_i &= LEFT_i || RIGHT_i
\end{aligned}$$

从进程的alphabet角度去分析。

(2) **Variation 2**

In yet another variation of the story, each fork may be picked up and put down many times on each occasion that the philosopher sits down. Thus the behaviour of the hands is modified to contain an iteration, for example

$$\begin{aligned} LEFT_i = (&i.sits\ down \rightarrow \\ &\quad \mu X \bullet (i.picks\ up\ fork.i \rightarrow i.puts\ down\ fork.i \rightarrow X \\ &\qquad | i.gets\ up \rightarrow LEFT_i)) \end{aligned}$$

Deadlock Analysis

(1) Deadlock problem:

When a mathematical model had been constructed, it revealed a serious danger. Suppose all the philosophers get hungry at about the same time; they all sit down; they all pick up their own forks; and they all reach out for the other fork, which is not there. In this undignified situation, they will all inevitably starve.

(2) One solution to deadlock: Introducing a footman

The footman's alphabet was defined as:

$$\bigcup_{i=0}^{4}\{i.sits\ down, i.gets\ up\}$$

$$U = \bigcup_{i=0}^{4}\{i.gets\ up\} \qquad D = \bigcup_{i=0}^{4}\{i.sits\ down\}$$

This footman was given secret instructions never to allow more than four philosophers to be seated simultaneously. His behaviour is most simply defined by mutual recursion. Let

$FOOT_j$ defines the behaviour of the footman with j philosophers seated

$$FOOT_0 = (x : D \rightarrow FOOT_1)$$

$$FOOT_j = (x : D \rightarrow FOOT_{j+1} \mid y : U \rightarrow FOOT_{j-1})$$
$$\text{for } j \in \{1, 2, 3\}$$

$$FOOT_4 = (y : U \rightarrow FOOT_3)$$

A college free of deadlock is defined

$$NEWCOLLEGE = (COLLEGE || FOOT_0)$$

Proof of absence of deadlock for the example

In the original COLLEGE the risk of deadlock was far from obvious; the claim that NEWCOLLEGE is free from deadlock should therefore be proved with some care. What we must prove can be stated formally as

$$(NEWCOLLEGE/s) \neq STOP \quad \text{for all } s \in traces(NEWCOLLEGE)$$

Change of symbol

Let f be a one-one function (injection) which maps the alphabet of P onto a set of symbols A

$$f: \ \alpha P \rightarrow A$$

We define the process $f(P)$ as one which engages in the event $f(c)$ whenever P would have engaged in c. It follows that

$$\begin{aligned} \alpha f(P) &= f(\alpha P) \\ traces(f(P)) &= \{f^*(s) \mid s \in traces(P)\} \end{aligned}$$

Examples:

(1) After a few years, the price of everything goes up. To represent the effect of inflation, we define a function f by the following equations

$f(in2p) = in10p$ $\qquad$ $f(large) = large$

$f(in1p) = in5p$ $\qquad$ $f(small) = small$

$f(out1p) = out5p$

The new vending machine is

$$NEWVMC = f(VMC)$$

(2) We wish to connect two instances of COPYBIT (1.1.3 X7) in series, so that each bit output by the first is simultaneously input by the second. First, we need to change the names of the events used for internal communication; wearray therefore introduce two new events mid.0 and mid.1, and define the functions f and g to change the output of one process and the input of the other

$$f(out.0) = g(in.0) = mid.0$$

$$f(out.1) = g(in.1) = mid.1$$

此处应该是f函数。

$$(in.0) = in.0, f(in.1) = in.1$$

$$g(out.0) = out.0, g(out.1) = out.1$$

从2个进程的alphabet角度去理解

The answer we want is

$$CHAIN2 = f(COPYBIT) \parallel g(COPYBIT)$$

Note: For the illustration figure (Fig. 2.7), please see page 63.

Laws for Change of Symbol

Note: (1) $f(B) = \{f(x) \mid x \in B\}$, (2) f^{-1} is the inverse of f

(3) $f \circ g$ is the composition of f and g

After change of symbol, $STOP$ still performs no event from its changed alphabet

L1 $f(STOP_A) = STOP_{f(A)}$

In the case of a choice, the symbols offered for selection are changed, and the subsequent behaviour is similarly changed

L2 $f(x : B \to P(x)) = (y : f(B) \to f(P(f^{-1}(y))))$

Change of symbol simply distributes through parallel composition

L3 $f(P||Q) = f(P)||f(Q)$

Change of symbol distributes in a slightly more complex way over recursion, changing the alphabet in the appropriate way

L4 $f(\mu X : A \bullet F(X)) = (\mu Y : f(A) \bullet f(F(f^{-1}(Y))))$

The composition of two changes of symbol is defined by the composition of the two symbol-changing functions

L5 $f(g(P)) = (f \circ g)(P)$

The traces of a process after change of symbol are obtained simply by changing the individual symbols in every trace of the original process

L6 $traces(f(P)) = \{f^*(s) | s \in traces(P)\}$

The explanation of the next and final law is similar to that of L6.

L7 $f(P)/f^*(s) = f(P/s)$

Process labelling

A process P labelled by l is denoted by

$$l : P$$

It engages in the event $l.x$ whenever P would have engaged in x. The function required to define $l : P$ is

$$f_l(x) = l.x \qquad \text{for all } x \text{ in } \alpha P$$

and the definition of labelling is

$$l : P = f_l(P)$$

Example:

(1) A pair of vending machines standing side by side

$$(left : VMS)||(right : VMS)$$

The alphabets of the two processes are disjoint, and every event that occurs is labelled by the name of the machine on which it occurred.

Note: If we don't use process labelling, what will happen next?

$$VMS \parallel VMS = VMS \text{ (???)}$$

Specification

L1 If P **sat** $S(tr)$ and Q **sat** $T(tr)$

then $(P||Q)$ **sat** $(S(tr \restriction \alpha P) \ \wedge \ T(tr \restriction \alpha Q))$

Example:

page 71, 此程序原来在page 51出现过。

Let $\alpha P = \{a, c\}, \quad \alpha Q = \{b, c\}$

and $P = (a \rightarrow c \rightarrow P), \quad Q = (c \rightarrow b \rightarrow Q)$

We wish to prove that

$$(P||Q) \textbf{ sat } 0 \leq tr \downarrow a - tr \downarrow b \leq 2$$

The proof of 1.10.2 X1 can obviously be adapted to show that

$$P \textbf{ sat } (0 \leq tr \downarrow a - tr \downarrow c \leq 1)$$

and

$$Q \ \mathbf{sat} \ (0 \leq tr \downarrow c - tr \downarrow b \leq 1)$$

By L1 it follows that

$$\begin{aligned} &(P||Q) \\ &\quad \mathbf{sat} \ (0 \leq (tr \restriction \alpha P) \downarrow a - (tr \restriction \alpha P) \downarrow c \leq 1 \wedge \\ &\qquad 0 \leq (tr \restriction \alpha Q) \downarrow c - (tr \restriction \alpha Q) \downarrow b \leq 1) \\ &\quad \Rightarrow 0 \leq tr \downarrow a - tr \downarrow b \leq 2 \end{aligned}$$

请举例说明！

L2 If P and Q never stop and if $(\alpha P \cap \alpha Q)$ contains at most one event, then $(P||Q)$ never stops.

Example:

The process $(P||Q)$ defined in X1 will never stop, because

$$\alpha P \cap \alpha Q = c$$

The proof rule for change of symbol is:

L3 If P sat $S(tr)$, then $f(P)$ sat $S(f^{-1*}(tr))$

Mathematical theory of deterministic processes

In our description of processes, we have stated a large number of laws, and we have occasionally used them in proofs. The laws have been justified (if at all) by informal explanations of why we should expect and want them to be true.

Question:

- Are these laws in fact true?
- Are them consistent?
- Are them complete?
- Could one manage with fewer and simpler laws?

These are questions for which an answer must be sought in a deeper mathematical investigation.

The basic definitions

1. From pair (A, S) to a process

Consider now an arbitrary pair of sets (A, S) which satisfy these three laws (in 1.8.1, L6, L7, L8). This pair uniquely identifies a process P whose traces are S constructed according to the following definitions. Let

$$P^0 = \{x | \langle x \rangle \in S\}$$

and, for all x in P^0, let $P(x)$ be the process whose traces are

$$\{t | \langle x \rangle ^\frown t \in S\}$$

Then $\alpha P = A$ and $P = (x : P^0 \rightarrow P(x))$

2. From a process to pair (A, S)

Furthermore, the pair (A, S) can be recovered by the equations

$$A = \alpha P$$

$$S = traces(x : P^0 \rightarrow P(x))$$

3. The one-one correspondence between a process and the corresponding pair

Thus there is a one-one correspondence between each process P and the pairs of sets $(\alpha P, traces(P))$.

D0 A deterministic process is a pair

$$(A,\ S)$$

where A is any set of symbols and S is any subset of A^* which satisfies the two conditions

C0 $\langle\rangle \in S$

C1 $\forall s, t \bullet s^\frown t \in S \Rightarrow s \in S$

D1 $STOP_A = (A, \{\langle\rangle\})$

D2 $RUN_A = (A, A^*)$

D3 $(x : B \to (A, S(x))) = (A, \{\langle\rangle\} \cup \{\langle x\rangle s | x \in B \wedge s \in S(x)\})$

provided $B \subseteq A$

D4 $(A, S)/s = (A, \{t | (s \frown t) \in S\})$ provided $s \in S$

D5 $\mu X : A \bullet F(X) = (A, \bigcup_{n \geq 0} traces(F^n(STOP_A)))$

provided F is a guarded expression

D6

$(A, S)||(B, T) = (A \cup B, \{s | s \in (A \cup B)^* \wedge (s \restriction A) \in S \wedge (s \restriction B) \in T\})$

D7 $f(A, S) = (f(A), \{f^*(s) | s \in S\})$ provided f is one-one

Fixed point theory

1. Aim:

The purpose of this section is to give an outline of a proof of the fundamental theorem of recursion, that a recursively defined process (2.8.1 D5) is indeed a solution of the corresponding recursive equation, i.e.,

$$\mu X \bullet F(X) = F(\mu X \bullet F(X))$$

The treatment follows the fixed-point theory of Scott.

2. $\sqsubseteq$ relation:

An ordering relationship among processes

D1 $(A, S) \sqsubseteq (B, T) = (A = B \wedge S \subseteq T)$

3. Properties of $\sqsubseteq$ (partial order)

L1 $P \sqsubseteq P$

L2 $P \sqsubseteq Q \wedge Q \sqsubseteq P \Rightarrow P = Q$

L3 $P \sqsubseteq Q \wedge Q \sqsubseteq R) \Rightarrow P \sqsubseteq R$

4. Chain A chain in a partial order is an infinite sequence of elements

$$\{P_0, P_1, P_2, ...\}$$

such that

$$P_i \sqsubseteq P_{i+1} \qquad \text{for all } i$$

We define the limit (least upper bound) of such a chain

$$\sqcup_{i \geq 0} P_i = (\alpha P_0,\ \bigcup_{i \geq 0} traces(P_i))$$

5. Complete Partial Order A partial order is said to be complete if it has a least element, and all chains have a least upper bound. The set of all processes with a given alphabet A forms a complete partial order (c.p.o.), since it satisfies the laws

L4 $STOP_A \sqsubseteq P$ provided $\alpha P = A$

L5 $P_i \sqsubseteq \ \sqcup_{i \geq 0} P_i$

L6 $(\forall i \geq 0 \bullet P_i \sqsubseteq Q) \Rightarrow (\sqcup_{i \geq 0} P_i) \sqsubseteq Q$

Remark: Furthermore the definition of μ (2.8.1 D5) can be reformulated in terms of a limit

L7 $\mu X : A \bullet F(X) = \sqcup_{i \geq 0} F^i(STOP_A)$

6. Continuous Complete Partial Order A function F from one c.p.o. to another one (or the same one) is said to be continuous if it distributes over the limits of all chains, i.e.,

$$F(\sqcup_{i \geq 0} P_i) = \sqcup_{i \geq 0} F(P_i) \quad \text{if } \{P_i \mid i \geq 0\} \text{ is a chain}$$

Remark:

(1) All continuous functions are monotonic in the sense that

$$P \sqsubseteq Q \Rightarrow F(P) \sqsubseteq F(Q) \quad \text{for all } P \text{ and } Q$$

(2) A function G of several arguments is defined as continuous if it is continuous in each of its arguments separately (for examples, please see page 76).

(3) The composition of continuous function is also continuous.

All the operators (except /) defined in D3 to D7 are continuous in the sense defined above

L8 $(x : B \to (\sqcup_{i \geq 0} P_i(x))) = \sqcup_{i \geq 0}(x : B \to P_i(x))$

L9 $\mu X : A \bullet F(X, (\sqcup_{i \geq 0} P_i)) = \sqcup_{i \geq 0}\ \mu X : A \bullet F(X, P_i)$

provided F is continuous

L10 $(\sqcup_{i \leq 0} Pi) || Q = Q || (\sqcup_{i \geq 0} P_i) = \sqcup_{i \geq 0}(Q || P_i)$

L11 $f(\sqcup_{i \geq 0} P_i) = \sqcup_{i \geq 0} f(P_i)$

7. The Proof of basic fixed-point theorem

Consequently if $F(X)$ is any expression constructed solely in terms of these operators, it will be continuous in X . Now it is possible to prove the basic fixed-point theorem

$$\begin{array}{lll} & F(\mu X : A \bullet F(X)) & \{\text{Def of } \mu\} \\ = & F(\sqcup_{i \geq 0} F^i(STOP_A)) & \{\text{Continuity } F\} \\ = & \sqcup_{i \geq 0} F(F^i(STOP_A)) & \{\text{Def } F^{i+1}\} \\ = & \sqcup_{i \geq 1} F^i(STOP_A) & \{STOP_A \sqsubseteq F(STOP_A)\} \\ = & \sqcup_{i \geq 0} F^i(STOP_A) & \{\text{Def } \mu\} \\ = & \mu X : A \bullet F(X) & \end{array}$$

Unique solution for fixed point

1. Aim: In this section we treat more formally the reasoning given in Section 1.1.2 to show that an equation defining a process by guarded recursion has only one solution.

2. Definition of $P \upharpoonright n$

If P is a process and n is a natural number, we define $(P \upharpoonright n)$ as a process which behaves like P for its first n events, and then stops; more formally

$$(A, S) \upharpoonright n = (A, \{s | s \in S \wedge \#s \leq n\})$$

3. Properties of $P \upharpoonright n$

L1 $P \upharpoonright 0 = STOP$

L2 $P \upharpoonright n \sqsubseteq P \upharpoonright (n+1) \sqsubseteq P$

L3 $P = \sqcup_{n \geq 0} P \upharpoonright n$

L4 $\sqcup_{n \geq 0} P_n = \sqcup_{n \geq 0} (P_n \upharpoonright n)$

4. Constructive Function

Let F be a monotonic function from processes to processes. F is said to be constructive if

$$F(X) \upharpoonright (n+1) = F(X \upharpoonright n) \upharpoonright (n+1) \quad \text{for all } X$$

This means that the behaviour of $F(X)$ on its first $n+1$ steps is determined by the behaviour of X on its first n steps only; so if $s \neq \langle\rangle$

$$s \in traces(F(X)) \equiv s \in traces(F(X \upharpoonright (\#s - 1)))$$

5. Constructive Properties of Processes:

(a) Prefixing is the primary example of a constructive function, since

$$(c \to P) \restriction (n+1) = (c \to (P \restriction n)) \restriction (n+1)$$

(b) General choice is also constructive

$$(x : B \to P(x)) \restriction (n+1) = (x : B \to (P(x) \restriction n)) \restriction (n+1)$$

(c) The identity function I is not constructive, since

$$\begin{aligned} & I(c \to P) \restriction 1 \\ = \quad & c \to STOP \\ \neq \quad & STOP \\ = \quad & I((c \to P) \restriction 0) \restriction 1 \end{aligned}$$

6. Main Result: Let F be a constructive function. The equation

$$X = F(X)$$

has only one solution for X.

Proof Let X be an arbitrary solution. First by induction we prove the lemma that

$$X \restriction n = F^n(STOP) \restriction n$$

Base case

$$X \restriction 0 = STOP = STOP \restriction 0 = F^0(STOP) \restriction 0$$

Induction step

$$\begin{array}{rlr}
 & X \restriction (n+1) & \{\text{since } X = F(X)\} \\
= & F(X) \restriction (n+1) & \{F \text{ is constructive}\} \\
= & F(X \restriction n) \restriction (n+1) & \{\text{hypothesis}\} \\
= & F(F^n(STOP) \restriction n) \restriction (n+1) & \{F \text{ is constructuve}\} \\
= & F(F^n(STOP)) \restriction (n+1) & \{\text{Def of } F^n\} \\
= & F^{n+1}(STOP) \restriction (n+1) &
\end{array}$$

Now we go back to the main theorem

$$\begin{array}{lll} & X & \{\mathbf{L3}\} \\ = & \sqcup_{n\geq 0}(X \restriction n) & \{\text{just proved}\} \\ = & \sqcup_{n\geq 0}F^n(STOP) \restriction n & \{\mathbf{L4}\} \\ = & \sqcup_{n\geq 0}F^n(STOP) & \{2.8\ \mathbf{L7}\} \\ = & \mu X \bullet F(X) & \end{array}$$

Thus all solutions of $X = F(X)$ are equal to $\mu X \bullet F(X)$; or in other words, $\mu X \bullet F(X)$ is the only solution of the equation.

7. Nondestructive Function

A function G is said nondestructive if

$$G(P) \restriction n = G(X \restriction n) \restriction n \quad \text{for all } n$$

Example:

(1) Alphabet transformation is nondestructive in this sense, since

$$f(P) \restriction n = f(P \restriction n) \restriction n$$

(2) Identity function is nondestructive.

(3) Any monotonic function which is constructive is also nondestructive.

(4) But the after operator is destructive. For the reason, please see page 78.

8. Guarded Expression

Let E be an expression containing the process variable X . Then E is said to be guarded in X if every occurrence of X in E has a constructive function applied to it, and no destructive function.

9. Syntactically Defining Constructive Function

D1 Expressions constructed solely by means of the operators concurrency, symbol change, and general choice are said to be guard-preserving.

D2 An expression which does not contain X is said to be guarded in X.

D3 A general choice

$(x : B \to P(X, x))$ is guarded in X if $P(X, x)$ is

guard-preserving for all x.

D4 A symbol change $f(P(X))$ is guarded in X if $P(X)$ is guarded in X.

D5 A concurrent system $P(X)||Q(X)$ is guarded in X if both $P(X)$ and $Q(X)$ are guarded in X.

10. Main Result

L6 If E is guarded in X, then the equation

$$X = E$$

has a unique solution.