A MATHEMATICAL THEORY OF COMMUNICATING PROCESSES

by A.W. ROSCOE,

ST. EDMUND HALL

Thesis submitted for the degree of D.Phil., Trinity 1982

Contents

Introduction

This thesis is an examination of part of the mathematical theory of communicating processes. These are studied through the medium of C.A.R.Hoare's language C.S.P. (Communicating Sequential Processes). Our main theme is the use of mathematical models to produce correctness proofs for processes, though several mathematical sidelines are also developed. The first seven chapters are broadly concerned with the construction and use of two mathematical models for C.S.P., these being the well-known "traces" model and a related model which is capable of dealing with non-deterministic processes in an adequate way.

In chapter one we meet the version of C.S.P. which is used throughout the thesis. We also meet the first of the two models; this is a model which can adequately cope only with deterministic processes. It has the advantage however that its simple structure allows us to develop the techniques which we later apply to the more general model. We define various operators over the model which represent ways of constructing and combining processes. These are used to produce a formal semantics for the language. Various results are quoted which show how one can to a large extent ignore the distinction between processes defined by the formal semantics and objects defined purely by operators on the model.

Chapter two is an investigation of a class of proof rules which can be applied to the model. These are all derived from the basic notion of "recursion induction". We find that there several different pairs of conditions, which if satisfied by recursive definitions and the predicates we wish to prove of them, guarantee the validity of the proof rules we devize.

Chapter three is a digression into topology. We study the spaces of allowable predicates generated by the previous chapter and the topologies which these induce on the space of processes. We are able to explain many of the results of chapter two and to generalize several of them.

In chapter four we meet the second and more general model for communicating processes, which is used throughout chapters five, six and seven. This is able to represent non-deterministic processes in a fairly convincing way. This chapter is not an extensive introduction to this model, something which can be found elsewhere ( ),( ), but is rather an examination of two questions which arise from the mathematical foundations of the model. The second of these is especially interesting, since we discover that the well-definedness of our model is a powerful set-theoretic tool, equivalent to the compactness theorem for arbitrary propositional languages.

In chapter five we see how the work of chapter two can be transferred to the non-deterministic model with only a few alterations. We see that many total correctness problems translate naturally into the system we have developed. As an example we take the predicate "is a buffer" and develop a calculus for proving it true of processes. We begin our study of operators defined by parallelism and hiding by discovering the close connections between a "pipe" operator and the space of buffers. Recursions via the pipe operator are investigated and we are able to prove several of them to be correct (buffers).

In chapter six we deal with an operator which models the behaviour of a process operating in parallel with a second, "slave" process, all communications between the two being hidden. We discover by means of examples that this operator is a very useful recursive tool for describing potentially infinite trees of processes. We develop a calculus for proving correct processes which are recursively defined using it. We discover an unfortunate possible type of behaviour in infinite networks and discover methods by which we can prove it absent.

Chapter seven deals with other parallel/hiding combinators in less detail, and as an example shows how to construct a rectangular array of processes. It also deals with the problem of proving a network free from deadlock. We find that hiding has little to do with deadlock, and can often be ignored when studying it. Deadlock is analysed in cases where it is known to be absent from all small portions of a network. This work is shown to have special significance in cases where networks are constructed without loops, though we are also able to develop various methods for applying it to more general networks.

As a result of the work done in chapters 1-7 we are equipped with a powerful calculus for proving properties of the values assigned to programs in our mathematical models. What the eighth and final chapter seeks to do is to provide a framework for the application of this calculus to the systems which we are attempting to model. We therefore develop a calculus for relating models to the systems which they attempt to model. We see how predicates transfer between systems, and how operators over a model may be held to reliably reflect their implementations in the real world. We construct a plausible though abstract space of "real" processes by which to judge our models for C.S.P. We find that there are several different ways in which we might interpret each of these. We find that in several interpretations some of our definitions of operators do not appear to be readily implementable. Finally we see with the benefit of hindsight what alterations might be made to our models and operators to make them have as many desirable properties as possible.

Much of the mathematics used in this thesis is rather abstract, especially in chapters three and four. The author hopes however that its use is justified by the results achieved, and that it does not obscure too much the techniques developed for the proof of programs.

Acknowledgements

The author would like to thank each of the following, who have all made large or small contributions to this work, either by suggestions or through their lectures.

C.A.R. Hoare
S.D. Brookes
D.S. Scott
J.E. Stoy
P.J. Collins
R. Milner
D.M.R. Park
C.B. Jones
W. Rounds
S.R. Blamey

In addition he would like to thank Miss J. Scheenen for reading the text and pointing out his many typing mistakes.

Finally he would like to thank the Science Research Council and St. Edmund Hall, Oxford for their financial support during the preparation of this thesis.

Chapter 1 :-

An Introduction to CSP and its Deterministic Model

This chapter outlines the version of CSP which is used in this thesis. We also meet the model for deterministic processes upon which chapters 2 and 3 are based, and use it to give a semantics for CSP. This chapter is to a large extent a summary of some of the work of C.A.R.Hoare, in whose papers (e.g. (1),(2)) the reader can find further explanation and motivation for CSP as well as many simple illustrative examples.

A process is to be thought of as an entity which communicates with its environment (possibly other processes) in some alphabet Σ of atomic communications or "events". Communication can take place only with the co-operation of all participants, and is symmetric in that there is no "sender" and no "receiver" of an event. At each point in the history of a process there is clearly a sequence of elements of Σ which is what it has communicated with its environment; this is known as a <u>trace</u> of the process.

We will assume that Σ contains basic symbols such as "a", or "true", or "isempty" and compound or named symbols such as "a.true", "b.a", "?a" or "!a" (we conventionally drop the infix dot with the names "?" and "!" which will usually represent input and output respectively). In addition Σ contains a special symbol "$\checkmark$" which a process communicates to indicate successful termination, and for which $a.\checkmark = \checkmark$.

We denote the set of all strings of symbols by Σ^*. The empty string (containing no symbols) we denote by "$\langle\rangle$". The string containing symbols a,b,...,z (in that order) we denote by $\langle ab...z\rangle$ and the concatenation of strings v & w we denote by vw.
Usually a,b,c,... will represent elements of Σ; v,w,s,t,... elements of Σ^* and x,y,z,... variable elements of Σ. A,B,C,.. will be processes and S,T,.. subsets of Σ. Σ^- will denote $\Sigma-\{\checkmark\}$. X* denotes the set of finite strings of elements of X.

Let us suppose that A is a process. We will postulate that at each point in A's history there is a set X of events in which it is willing to co-operate, and will do so (picking one at random) if the environment is also able to execute any of the elements of X. This set will vary with time dependent on both the visible actions of A (namely its communications) and any internal progress which it makes. It is clearly possible that, depending on internal choices which are invisible to the environment, A might behave in one of several different ways after the same visible behaviour. Processes which behave in this way are known as non-deterministic. We will later (in chapter 4) meet models which are sufficiently expressive to deal with this type of behaviour, but for the moment we will only attempt to deal with deterministic processes. The criterion for A to be deterministic is that there be some function F from tr(A) (the traces of A, $\subseteq \Sigma^*$) to $\mathcal{P}(\Sigma)$ such that for every $s \in tr(A)$ the following two conditions hold.

a) Whenever s is the trace of A then $X \subseteq F(s)$.

b) Whenever s remains the trace indefinitely then for each $a \in F(s)$ & time t there is some $t' > t$ at which $a \in X$.

This means that whenever $s \in tr(A)$ then a sufficiently patient experimenter will be able to achieve it, so long as his current trace is some prefix of s.

It is possible to define a partial order on Σ^* by $v \leqslant w$ if $\exists s.\ w = vs$, or equivalently if v is a prefix of w. If $w \neq v$ we say $v < w$.

The model we choose for deterministic processes is sets of traces (subject to the conditions below), intending to identify a deterministic process with its set of traces. For the moment denote by $\overline{A}$ the representation of A in the model. Define $B \subseteq \Sigma^*$ to be an element of P, the space of deterministic processes, if it satisfies the conditions:

1.1 a) B is non-empty

b) B is prefix closed (i.e. $w \in B$ & $v < w \Rightarrow v \in B$)

c) $w\langle\surd\rangle v \in B \Rightarrow v = \langle\rangle$

Every "real" process A satisfies these conditions, since:
a) A can at least do nothing (so $\langle\rangle \in \bar{A}$).
b) If $v < w$ and A has performed trace w there must have been some earlier time when it had performed trace v.
c) A cannot do anything after it has terminated.

The strength of this model applied to deterministic processes is that, given a deterministic process A such that $s \in \bar{A}$, then the environment, by applying any set Y s.t. $Y \cap \{a \mid s\langle a\rangle \in \bar{A}\} \neq \emptyset$ to A (when its trace is s), can be sure of eventually getting some response (in Y). This allows us to express many notions of "total correctness" of a process solely by reference to its value in the model P. For example define <u>deadlock</u> as the state that a process is in when it (a) has not terminated successfully and (b) will never be able to communicate with its environment again. We can express the predicate "is free from deadlock", meaning that deadlock can never arise in a process, in terms of the model:

$$D\text{-}F(A) \equiv (s \in \bar{A} \Rightarrow ((\exists t.\ s=t\langle\surd\rangle) \vee (\exists a.\ s\langle a\rangle \in \bar{A})))\ .$$

From now on we will largely be studying the model P in itself, rather than its relation to "real" processes. Thus from now on A,B,C,... will denote elements of P.

1.2 <u>Lemma</u>

a) With the set inclusion order "$\subseteq$" P is a complete lattice with minimal element $\{\langle\rangle\}$ and maximal element $\{w, w\langle\surd\rangle \mid w \in (\Sigma^{-})^{*}\}$.
b) The finite elements of P (in the lattice-theoretic sense) are just the processes with finitely many elements.

The proof of a) follows immediately from the fact that if $\emptyset \neq X \subseteq P$ then $\bigcup X \in P$ and $\bigcap X \in P$. The proof of b) follows from the fact that if $A \in P$ and $F \subseteq A$ is finite (not necessarily $\in P$) then there is some finite size $B \in P$ s.t. $F \subseteq B \subseteq A$.

We will now start to introduce the language and its interpretation (semantics) within P. We first meet a few basic processes:

1.3 <u>abort</u> $= \{\langle\rangle\}$ the process which can do nothing

1.4 <u>skip</u> $= \{\langle\rangle, \langle\surd\rangle\}$ the process which immediately terminates successfully.

1.5 $\underline{run} = \{w, w\langle\surd\rangle \mid w \in (\Sigma^{-})^{*}\}$, the process which has the ability to do anything at all.

Note that abort corresponds to the minimal element of P, and run to the maximal one. Recall that our interpretation of elements of P is that the environment (external or another process) is at each stage given a choice of what it would like to do next. This choice, on the first step, is between the initials of a process, those elements of which are possible on the first step. We write this set as $A^{O} = \{a \in \Sigma \mid \langle a\rangle \in A\}$. Note that the strength of a process, in terms of the partial order, is measured by the amount of choice it offers.

For any $w \in A$ $(A \in P)$ we can define the process which A becomes immediately after it has executed the trace w.

1.6 A $\underline{after}$ $w = \{v \mid wv \in A\}$

Note that for any process $A \in P$ we have A $\underline{after}\langle\rangle = A$, and $\underline{run}$ $\underline{after}$ $w = \underline{run}$ for every $w \in (\Sigma^{-})^{*}$.

Before giving a formal semantics for the rest of our language, we will give an informal explanation and motivation for each of its constructs. These constructs take the form of operators which combine one or more processes together to form more complex ones.

By "a → A" we will mean the process which communicates "a" on its first step and then acts like A. This construct will be the most basic building block of our language. Along the same lines "x:T → A" will represent the process which inputs some value ("b" say) from the set T of unnamed symbols, and substitutes this value for the variable "x" within A. "a.x:T → A" will be the same except that we will expect its input to be named by "a".

The process "a.A" will be the same as A except that all events are given the (possibly additional) name "a".

The process "A□B" will give the environment the choice of all the communications offered by A and B.

The sequential composition of A and B, in which B takes over whenever A terminates successfully, is written A;B.

By $(A_{X}\|_{Y}B)$ we will mean the process which consists of A working in parallel with B. All communications of A will be in X, all those of B in Y and all communications which lie in $X \cap Y$ must be co-operated in (executed simultaneously) by both A and B.

By "A/X" we will mean the process formed by <u>hiding</u> all occurrences of elements of the set X from the environment, making them into internal actions which happen without the control of the environment.

Finally we will include recursion, for which we include variables representing processes in our syntax, on the understanding that all occurrences of these will be bound by some recursion.

We will now define some of these operators formally.

1.7 $a \to A = \{\langle\rangle\} \cup \{\langle a\rangle w \mid w \in A\}$ (for $a \in \Sigma^-$, $A \in P$)

1.8 $A \,\square\, B = A \cup B$ (for $A, B \in P$)

1.9 $A;B = (A \cap (\Sigma^-)^*) \cup \{wv \mid w\langle\surd\rangle \in A \ \&\ v \in B\}$

1.10 $(A_X\|_Y B) = \{w \mid (w\restriction(X \cup Y) = w) \ \&\ w\restriction X \in A \ \&\ w\restriction Y \in B\} \cap \{w, w\langle\surd\rangle \mid w \in (\Sigma^-)^*\}$

where $A, B \in P$, $X, Y \subseteq \Sigma$

the restriction operator $\restriction X$ is defined

$$\langle\rangle\restriction X = \langle\rangle \;, \quad \langle a\rangle w\restriction X = w\restriction X \ \text{ if } a \notin X$$
$$= \langle a\rangle (w\restriction X) \ \text{ if } a \in X$$

1.11 $A/X = \{w\restriction(\Sigma - X) \mid w \in A\}$

1.12 $a.A = \{a.w \mid w \in A\}$

where the operator "a." is defined on Σ^* by

$$a.\langle\rangle = \langle\rangle, \quad a.(\langle c\rangle w) = \langle a.c\rangle (a.w)$$

We will not be able to define the operators involving variables properly until later. The following is a list of easily proved identities involving the above.

1.13 <u>Theorem</u>

a) $A \,\square\, B = B \,\square\, A$ ($\square$ is commutative)

b) $A \,\square\, (B \,\square\, C) = (A \,\square\, B) \,\square\, C$ ($\square$ is associative)

c) $(A;B);C = A;(B;C)$ (; is associative)

d) $(A \,\square\, B);C = (A;C) \,\square\, (B;C)$ (; distributes to the left over $\square$)

e) $A;(B \,\square\, C) = (A;B) \,\square\, (A;C)$ (; distributes to the right over $\square$)

f) $(a \to B);C = a \to (B;C)$

g) $(A_X\|_Y B) = (B_Y\|_X A)$ ($\|$ is commutative)

h) $(A_X\|_{Y \cup Z}(B_Y\|_Z C)) = ((A_X\|_Y B)_{X \cup Y}\|_Z C)$ ($\|$ is associative)

i) $A \,\square\, A = A$ ($\square$ is idempotent)

j) $(A_X \|_X A) = A$ provided $A \subseteq X^*$

k) $(a \to A)/X = A/X$ if $a \in X$

$= a \to A/X$ if $a \notin X$

l) $(A/X_Y \|_Z B) = (A_{X \cup Y} \|_Z B)/X$ provided $X \cap Z = \emptyset$

m) $(A;B)/X = (A/X);(B/X)$ provided $\checkmark \notin X$

These identities will be used frequently and informally throughout the rest of this chapter and in chapters 2 and 3.

1.14 <u>Theorem</u>

Each of the operators defined in 1.7 - 1.12 represents a monotonic and continuous function of P or $P \times P$. This function is also reverse continuous except in the case /X when X is infinite. (Reverse continuous means continuous on the lattice with reverse order.)

To prove monotonicity it is sufficient to show (for each operator f) that it is monotonic in each parameter.

To prove continuity it is sufficient to show that each operator represents a continuous function of each of its parameters. That is, $f(\bigcup D) = \bigcup_{X \in D} f(X)$ for each nonempty directed set $D \subseteq P$. A function is thus reverse continuous if it satisfies $f(\bigcap D) = \bigcap_{X \in D} f(X)$ for each nonempty reverse directed set $D \subseteq P$.

Each of the cases of the theorem is fairly easily checked. The failure of reverse continuity for infinite hiding is illustrated by the following example.

1.15 <u>Example</u>

We will suppose $a \notin N$ (the natural numbers) and that $\{a\} \cup N \subseteq \Sigma$. For $n \in N$ define the process $A_n = \{\langle\rangle, \langle m\rangle, \langle ma\rangle \mid m \geq n\}$. It is easy to see that $D = \{A_n \mid n \in N\}$ is a reverse directed set with $\bigcap D =$ <u>abort</u> but that $A_n/N = a \to$ <u>abort</u> for each $n \in N$. We thus have $\bigcap_{A \in D} (A/N) = a \to$ <u>abort</u> $\neq (\bigcap D)/N =$ <u>abort</u>.

The difficult properties of infinite hiding will appear again in a much more fundamental way when we consider the non-deterministic model in chapter 4. The advantages of having operators which are reverse continuous will be seen in 2.46 et seq.

The monotonicity and continuity of the operators will be very important in defining recursion.

It is necessary here to strike a note of caution. We have happily given definitions in the deterministic model P to each of the operators we have described informally. Inspection of each of the definitions 1.7 - 1.12 will show that these reflect to the greatest possible extent the intentions we set out, in that the resulting sets of traces seem to be the best possible. This still leaves the question of whether it is reasonable to expect an implementor to produce deterministic processes as the result of applying each of the operators to deterministic processes. There are in fact several cases where this does not seem reasonable, in that the "natural" result of an operator is not in general deterministic.

The first and most obvious of these cases is with the hiding operator "/X". Consider for example the process written "$(a \to b \to \underline{skip}) \,\Box\, (c \to d \to \underline{skip})/\{a,c\}$" (= A, say). (The elements a,b,c,d of Σ are assumed to be distinct.) It is easy to prove the relation $A = (b \to \underline{skip}) \,\Box\, (d \to \underline{skip})$. This latter process is one which is invariably willing (given time) to execute either "b" or "d" on its first step, at the environment's choice. It is thus necessary in a correct implementation of the syntax of A that we should take account of its behaviour after both "hidden a" and "hidden c". This varies from what one might consider natural, namely that the process would either carry out hidden "a" or "c" but not both and that the actions of the process would then be independent of the consequences of executing the other one. To implement the hiding operator correctly with respect to definition 1.11 and produce a deterministic process it would in general be necessary to carry out an infinite amount of backtracking. The three examples below help to illustrate other aspects of this problem.

a) $(a \to b \to \underline{skip}) \,\Box\, (c \to \underline{abort})/\{a,c\} = b \to \underline{skip}$

b) $(a \to b \to \underline{abort}) \,\Box\, (b \to \underline{skip})/\{a\} = b \to \underline{skip}$

c) $A \,\Box\, (b - \underline{skip})/\{a\} = b \to \underline{skip}$
   where $A = \{\langle\rangle, \langle a\rangle, \langle aa\rangle, \langle aaa\rangle, \langle aaaa\rangle, \ldots\}$

The following three features of a process A give rise to implementation problems in A/X.

a) The choice at any stage between several elements of X. (Illustrated in (a) above and also in the example in the text.)

b) The choice between elements and non-elements of X at any stage. (Illustrated in (b) above.)

c) The possibility of infinite sequences of elements of X. (Illustrated in (c) above.)

While we are using the deterministic model we will only use the hiding operator in writing processes where these three conditions are avoided, so that our processes are reasonable to implement deterministically.

Another operator which gives rise to similar problems is the alternative composition operator "$\square$". This operator is intended to give the environment the choice of the first step symbols of the two processes, the combination then acting like the chosen process. The problem arises when the two processes have first step choices in common. The correct operation of the definition 1.8 would require the operation in parallel of the two processes until any time when one of the processes cannot continue with the executed trace. (An alternative is to back-track when one of the processes is unable to continue.) This is clearly inefficient, for there is the possibility of an exponential growth in effort with the number of "$\square$"s encountered. There is also the problem (which arises from both the suggested methods of implementation) of detecting just when a process cannot execute a symbol. This would almost certainly be tantamount to solving the halting problem; and in any case it is possible to find processes which do nothing for any given finite time before finally becoming able to communicate (e.g. a process with a very large number of hidden symbols to execute at the start). It would in fact probably be possible to implement "$\square$" deterministically by combining the two approaches above, running the two processes alternately and trying to make the one (if either) which falls behind catch up. This is unfortunately even less efficient than was expected for the implementation above, for there is now the <u>certainty</u> of exponential growth. Because of these difficulties we must make it a rule that in writing expressions which we expect to be implementable we should only use "□" when we can be certain that the initials of the two processes which we are combining are disjoint. This can be achieved (for example) by ensuring that the two processes have distinct guards, so that the combination looks like (a → A) □ (b → B) where a ≠ b.

The last problem of determinacy arises from sequential composition (;). It occurs when it is possible for the first process either to terminate successfully or to do something else ("a", say). If the second process has the ability to communicate "a" on its first step then we have a problem of a very similar nature to the one which arose with "□". To implement 1.9 correctly it would be necessary to run both processes in parallel until it could be detected that only the first or second process has the ability to continue with the trace. This problem of ambiguity gives rise to exactly the same difficulties as arose with "□" above. The rule which avoids this problem is to avoid creating processes which make successful termination anything other than the sole option at any point in their history at which it is possible.

The following illustrate the possible difficulties with the implementation of "□" and ";".

a) (a → A) □ (a → B)

   Here it is necessary to determine the result of executing both the left and right "a"s to correctly implement this process.

b) (skip □ (a → b → skip)); (a → b → a → b → skip)

   There are two ways in which this process can execute any of the traces ⟨a⟩, ⟨ab⟩, ⟨aba⟩, ⟨abab⟩. If the second process were prepared to carry out an infinite sequence of "a"s and "b"s this conflict would never be resolved.

In 1.19 we will meet a more interesting example of a process written without regard for these rules. (For its construction we will need recursion.)

We will find that the model introduced in chapter 4 gives more natural semantics to each of the above operators, avoiding the necessity of imposing strict conditions upon a process to ensure that implementation is possible. Indeed we will find that the only points at which the semantics in the two models differ fundamentally are those where the difficulties described above occur, and also in the case of ill-defined recursion (which we have yet to meet). This is true in the sense that (except in these cases) if we combine the representations in the more advanced model of one or more deterministic processes the result will be a deterministic process congruent to the result we would have obtained in the deterministic model P.

We must now introduce the various operators involving the use of variables representing processes and elements of Σ. Unfortunately the informal approach we have adopted so far, namely the introduction of our language purely as a collection of operators on P, seems to become inadequate here. We will thus recast our previous definitions and introduce the new ones within the framework of a formal syntax and semantics for C.S.P.

We first meet the various syntactic domains we will need.

1.16 Syntactic Domains

$A \in Exp$	(the C.S.P. expressions)
$a \in \Sigma^-$	(alphabet)
$B \in PV$	(process variables)
$B' \in PV'$	(process variable calls)
$x \in AV$	(alphabet variables)
$T \in IA$	(expressions for sets of unnamed symbols)
$X \in GA$	(expressions for sets of general symbols)
$U \in BA$	(basic subsets of Σ)

$A ::= \underline{skip} \mid \underline{abort} \mid a \to A \mid x \to A \mid a.x \to A \mid x{:}T \to A \mid a.x{:}T \to A \mid$

$A \,\square\, A \mid A;A \mid (A_X \|_X A) \mid A/X \mid a.A \mid B' \mid$

$rec_i(B_1,\ldots,B_k).(A_1,\ldots,B_k) \mid$ "infinite mutual recursion"

(the syntax for infinite mutual recursion will be found later, in 1.18)

$B' ::= B \mid B_g$ (explanation of B_g will be found in the definition of infinite mutual recursion)

$T ::= \emptyset \mid \{a\} \mid \{x\} \mid U \mid T \cup T \mid T \cap T \mid T-T \mid \ldots$

$X ::= \emptyset \mid \{a\} \mid \{x\} \mid U \mid X \cup X \mid X \cap X \mid X-X \mid a.X \mid \ldots$

We will assume that we are initially endowed with the sets Σ, PV, AV and BA and that they satisfy the condition that all the objects created from the above syntax are distinct if they have distinct parse trees.

We can only expect elements of Exp to represent elements of P if they do not depend on the value of any variable. (One would expect such expressions to represent functions of such variables.) One might expect an expression to depend on the value of a variable only if there were some occurrence of it within the expression which was not bound to some construct which assigned it a value. For example we would expect "a → x → skip" to depend on "x" but not "x:T → x → skip". It is easy to construct formal definitions of the notions of free and bound variables in an expression. Informally an occurrence of process variable B will be bound in $A \in Exp$ if and only if there is a syntactic subcomponent of A which contains the occurrence and has the form $rec_j(\ldots B \ldots).(A_1,\ldots,A_k)$ or the equivalent in infinite mutual recursion. Alphabet variables will be bound by the constructs "x:T → A" and "a.x:T → A".

1.17 Definition

a) Alphabet variables

Suppose $x \in AV$. It is easy to construct a formal recursive definition of the phrase "x occurs in X" (for $X \in GA$) meaning that the variable "x" occurs in the syntax of "X". The same can be done for "x occurs in T" ($T \in IA$) and "x occurs in B'" ($B' \in PV'$).

Given these definitions one can construct a formal definition of the terms free and bound variables.

- x is neither free nor bound in skip or abort.
- x occurs free (bound) in a → A if it occurs free (bound) in A.
- x occurs free (bound) in a.A if it occurs free (bound) in A.
- x occurs free in x → A and a.x → A.
- x occurs free in y → A and a.y → A ($x \neq y$) if it occurs free in A.
- x occurs bound in y → A and a.y → A if it occurs bound in A.
- x occurs bound in x:T → A and a.x:T → A.
- x occurs free in x:T → A and a.x:T → A if x occurs in T.
- x occurs free in y:T → A and a.y:T → A ($y \neq x$) if x occurs free in A or if x occurs in T.

x occurs bound in $y{:}T \to A$ and $a.y{:}T \to A$ ($y \neq x$) if x occurs bound in A.

x occurs free (bound) in $A \square C$ and $A;C$ if it occurs free (bound) in either A or C.

x occurs free (bound) in $rec_j(B_1,\ldots,B_k).(A_1,\ldots,A_k)$ if it occurs free (bound) in any A_i.

(similar clause for infinite mutual recursion)

x occurs free in A/X if x occurs free in A or x occurs in X.

x occurs bound in A/X if it occurs bound in A.

x occurs free in $(A_X \| _Y C)$ if x occurs free in A or C or if x occurs in X or Y.

x occurs bound in $(A_X \| _Y C)$ if it occurs bound in A or C.

x does not occur bound in B'.

x occurs free in B' if x occurs in B'.

b) Process variables

The definition of free and bound occurrences of process variables is very much the same as the above. The critical clauses are the following.

B occurs bound in $rec_j(\ldots B \ldots).(A_1,\ldots,A_k)$

B occurs free in $rec_j(B_1,\ldots,B_k).(A_1,\ldots,A_k)$ if it occurs free in some of $A_1,\ldots,A_k$ and $B \neq B_1$ & ... & $B \neq B_k$.

B occurs free in B.

B occurs free in B_g if $B \in rge(g)$ (see 1.18).

The clause for infinite mutual recursion will be found in 1.18.

We can now define Proc, the space of process expressions, to be the set of those elements of Exp which contain no free variables of either kind. We will expect to be able to construct a semantics for Proc within P.

In order to be able to assign a value to general elements of Exp we will need a state σ which will be a mapping from variables to their values ($\sigma \in S = (AV \to \Sigma^{\neg}) \times (PV \to P)$).

Our semantic functions will be the following.

1.18 Semantic Functions

$\mathcal{E}$: Exp $\to$ S $\to$ P

$\mathcal{V}$: GA $\to$ S $\to$ $\mathcal{P}(\Sigma)$

$\mathcal{U}$: IA $\to$ S $\to$ $\mathcal{P}(\Sigma)$

$\mathcal{S}$: PV' $\to$ S $\to$ P

The definitions of $\mathcal{U}$ and $\mathcal{V}$ are fairly obvious and are omitted. The definition of $\mathcal{S}$ is delayed till the section on infinite mutual recursion.

The definition of $\mathcal{E}$ is as follows. Where a definition appears circular it is because we are using the versions of our operators which were defined (over P) in 1.3 - 1.12.

a) $\mathcal{E}[\![\underline{\text{skip}}]\!]\sigma = \underline{\text{skip}}$

b) $\mathcal{E}[\![\underline{\text{abort}}]\!]\sigma = \underline{\text{abort}}$

c) $\mathcal{E}[\![a \to A]\!]\sigma = a \to \mathcal{E}[\![A]\!]\sigma$

d) $\mathcal{E}[\![x \to A]\!]\sigma = \sigma[\![x]\!] \to \mathcal{E}[\![A]\!]\sigma$

e) $\mathcal{E}[\![a.x \to A]\!]\sigma = a.(\sigma[\![x]\!]) \to \mathcal{E}[\![A]\!]\sigma$

f) $\mathcal{E}[\![x{:}T \to A]\!]\sigma = \bigcup\{a \to \mathcal{E}[\![A]\!]\sigma[a/x] \mid a \in \mathcal{U}[\![T]\!]\sigma\} \cup \{\langle\rangle\}$
(last term required in case $\mathcal{U}[\![T]\!]\sigma$ is empty)

g) $\mathcal{E}[\![a.x{:}T \to A]\!]\sigma = \bigcup\{a.b \to \mathcal{E}[\![A]\!]\sigma[b/x] \mid b \in \mathcal{U}[\![T]\!]\sigma\} \cup \{\langle\rangle\}$

h) $\mathcal{E}[\![A \,\square\, C]\!]\sigma = \mathcal{E}[\![A]\!]\sigma \,\square\, \mathcal{E}[\![C]\!]\sigma$

i) $\mathcal{E}[\![A;C]\!]\sigma = \mathcal{E}[\![A]\!]\sigma ; \mathcal{E}[\![C]\!]\sigma$

j) $\mathcal{E}[\![(A_X \|_Y C)]\!]\sigma = (\mathcal{E}[\![A]\!]\sigma_{X^*} \|_{Y^*} \mathcal{E}[\![C]\!]\sigma)$, where $X^* = \mathcal{V}[\![X]\!]\sigma$ & $Y^* = \mathcal{V}[\![Y]\!]\sigma$

k) $\mathcal{E}[\![A/X]\!]\sigma = (\mathcal{E}[\![A]\!]\sigma)/(\mathcal{V}[\![X]\!]\sigma)$

l) $\mathcal{E}[\![a.A]\!]\sigma = a.(\mathcal{E}[\![A]\!]\sigma)$

m) $\mathcal{E}[\![B']\!]\sigma = \mathcal{S}[\![B']\!]\sigma$

n) $\mathcal{E}[\![\text{rec}_j(B_1,\ldots,B_k).(A_1,\ldots,A_k)]\!]\sigma = (\bigcup_{n=0}^{\infty} H^n(\{\langle\rangle\}^k))_j$
where $H: P^k \to P^k$ is the function defined
$H(\underline{C}) = (\mathcal{E}[\![A_1]\!]\sigma^*, \ldots, \mathcal{E}[\![A_k]\!]\sigma^*)$ $\quad (\sigma^* = \sigma[C_1/B_1]\ldots[C_k/B_k])$

In the above $\sigma[a/x]$ represents the state which is identical to σ except that it maps x to a. $\sigma[C/B]$ is the state identical to σ except for mapping B to C.

o) Suppose that Λ is some set of indices and that we have an injective mapping from Λ to PV, written B_λ $(\lambda \in \Lambda)$. Suppose further that $(\Gamma_1,\ldots,\Gamma_k)$ is a partition of Λ. Then we allow recursive definitions of the form:

B_g, where $\lambda \in \Gamma_1 \Rightarrow B_\lambda \Leftarrow A_1$
....
....
$\lambda \in \Gamma_k \Rightarrow B_\lambda \Leftarrow A_k$

It is assumed that the constants ($\in \Sigma^-$) used in the construction of the A_i may depend on λ, and that the calls of the B_χ have the form $B_{g'}$ for some function g' of λ and any free alphabet variable(s).

We will assume that these functions carry with them a syntactic check on their ranges, to enable accurate determination of which process variables are free in an expression of the form B_g. Usually the functions associated with the above recursion will have ranges $\subseteq \Lambda$, though sometimes a call may range over several nested recursions.

We will assume no definite syntax for the space of functions "g" making calls of process variables, since it clearly is heavily dependant on the natures of Λ and Σ. We will however assume that there is some procedure for determining which alphabet variables are free in any B_g, and also that there is some procedure π for evaluating any g in any state. Typical functions will be (i) constant functions; (ii) identity functions on Λ and (if $\Sigma \subseteq \Lambda$) on Σ; and (iii) (if $\Lambda = N$) "+1", "-1" etc. An alphabet variable will occur bound in the above recursion if it occurs bound in any of the A_i, and free if it either occurs free in any of the A_i or in B_g. All the elements of $\{B_\xi | \xi \in \Lambda\}$ occur bound together with any other process variables occurring bound in any A_i. A process variable occurs free if it is not in $\{B_\xi | \xi \in \Lambda\}$ and it either lies in the range of g or occurs free in any A_i.

We can now define the function $\mathfrak{S}$ which evaluates procedure calls: $\mathfrak{S}[\![B]\!]\sigma = \sigma[\![B]\!]$; $\mathfrak{S}[\![B_g]\!]\sigma = \sigma[\![\pi(g)\sigma]\!]$.

Finally we can define $\mathcal{E}[\![B_g$, where.....$]\!]\sigma$

$= \sigma[\![\pi(g)\sigma]\!]$ if $\pi(g)\sigma \notin \{B_\xi | \xi \in \Lambda\}$;

$= (\bigcup_{n=0}^{\infty} H^n(\{\langle\rangle\}^\Lambda))_\xi$ if $\pi(g)\sigma = B_\xi$ $(\xi \in \Lambda)$

where P^Λ is the function space $\Lambda \to P$ (which is a complete lattice because P is); and $H: P^\Lambda \to P^\Lambda$ is defined:

$H(\underline{C})_\xi = \mathcal{E}[\![A_i^\xi]\!]\sigma^*$, where $\xi \in \Gamma_i$, $\sigma^* = \sigma[\underline{C}/\underline{B}]$ is the state identical to σ except for mapping B_ξ to C_ξ for each $\xi \in \Lambda$, and A_i^ξ is the expression obtained by substituting "ξ" for the parameter "λ" in A_i.

Note that to be fully rigorous we should be rather more careful in our treatment of the parameters λ used in this type of recursion. If we had been more pedantic we could have introduced a third class of variables representing these parameters; these would have their free occurrences in expressions representing elements of Σ and in B_gs, and be bound by the recursive definitions of the "schema" kind (1.18(o)).

1.19 Examples

(i) An integer register

Suppose that $\{iszero, up, down\} \cup set.N \subseteq \Sigma$ $(N = \{0,1,2,..\})$. Take Λ (the set of indices) to be $N \cup \{u\} (u \notin N)$. An initially undefined register is represented by

$$R_u, \text{ where } \quad R_u \Leftarrow set.x{:}N \rightarrow R_x$$
$$R_0 \Leftarrow (iszero \rightarrow R_0) \,\square\, (up \rightarrow R_1) \,\square\, (set.y{:}N \rightarrow R_y)$$
$$x \in N - \{0\} \Rightarrow R_x \Leftarrow (down \rightarrow R_{x-1}) \,\square\, (up \rightarrow R_{1+x}) \,\square\, (set.y{:}N \rightarrow R_y) \quad .$$

(ii) A stack

Suppose that T is some set of basic symbols, that $?T \cup !T \subseteq \Sigma$, and $T^* = \Lambda$. An initially empty stack of type T is represented by

$$S_{\langle\rangle}, \text{ where } \quad S_{\langle\rangle} \Leftarrow ?x{:}T \rightarrow S_{\langle x \rangle}$$
$$w \in T^* - \{\langle\rangle\} \Rightarrow S_w \Leftarrow (?x{:}T \rightarrow S_{\langle x \rangle w}) \,\square\, (!y \rightarrow S_v) \qquad \text{where } w = \langle y \rangle v \quad .$$

(iii) Palindromes

Suppose that $\{a, b, \surd\} \subseteq \Sigma$, then the following process is prepared to terminate successfully if and only if its current string is a palindrome of "a"s and "b"s.

$$P \Leftarrow \underline{skip} \,\square\, (a \rightarrow \underline{skip}) \,\square\, (b \rightarrow \underline{skip}) \,\square\, (a \rightarrow P; (a \rightarrow \underline{skip})) \,\square\, (b \rightarrow P; (b \rightarrow \underline{skip}))$$

Note that none of the above examples is completely strict in its use of recursive syntax, though it is perfectly clear in each case what the correct syntax is. From here on we will habitually use abbreviations such as the above, on the understanding that we could translate into "correct" syntax if challenged.

Note how the palindromes example breaks two of the conventions designed to make a process implementable in a reasonable way. The difficulties in this case seem to have more to do with the nature of the problem than with "bad programming" since it is possible for some strings to be initial segments of palindromes in several essentially different ways. For example $\langle ababa \rangle$, in addition to being a palindrome itself, is an initial segment of $\langle ababab a \rangle$, $\langle abababab a \rangle$ and all the longer ones of which $\langle ababa \rangle$ is less than half.

Many more examples will appear later, especially in chapters two, five and six.

The notation used in conjunction with the above formal semantics for CSP is rather cumbersome to use in practice. In future we will habitually identify the syntax of a process with its value in our model. The following is a sequence of easily proved results which justify this identification (in the case of expressions with no free variables of any type).

1.20 Lemma

(i) If $A \in$ Exp, $a \in \Sigma^-$ and x is an alphabet variable which does not occur free in A then $\mathcal{E}[\![A]\!]\sigma = \mathcal{E}[\![A]\!]\sigma[a/x]$ for every state σ.

(ii) If $A \in$ Exp, $\underline{C} \in P^\Lambda$ and $\underline{B} \in PV^\Lambda$ is a vector none of whose components occurs free in A then $\mathcal{E}[\![A]\!]\sigma = \mathcal{E}[\![A]\!]\sigma[\underline{C}/\underline{B}]$ for every state σ.

1.21 Corollary

If $A \in$ Exp has no free variables then $\mathcal{E}[\![A]\!]\sigma = \mathcal{E}[\![A]\!]\rho$ for all states σ and ρ.

If A_1 and A_2 are elements of Exp let us say that $A_1 \equiv A_2$ if $\mathcal{E}[\![A_1]\!]\sigma = \mathcal{E}[\![A_2]\!]\sigma$ for all states σ.

1.22 Lemma

Clauses a - i of 1.13 remain true if elements of Exp are substituted for elements of P (etc.) and $\equiv$ is substituted for = throughout. Clauses j - m remain true if the stated conditions are true for all states (e.g. clause l remains true provided $\mathcal{V}[\![X]\!]\sigma \cap \mathcal{V}[\![Z]\!]\sigma = \emptyset$ for all σ).

Additionally:

n) $x \rightarrow (B;C) \equiv (x \rightarrow B);C$, $a.x \rightarrow (B;C) \equiv (x \rightarrow B);C$

o) $x{:}T \rightarrow (A;C) \equiv (x{:}T \rightarrow B);C$ if x does not occur free in C

p) $a.x{:}T \rightarrow (A;C) \equiv (a.x{:}T \rightarrow B);C$ " " " " " "

1.23 Lemma

If $A_1, A_2, C \in$ Exp, $A_1 \equiv A_2$ and A_1 is a syntactic subcomponent of C, then the result of substituting A_2 for A_1 in C (C', say) satisfies $C' \equiv C$.

1.24 Lemma

If $A \in \text{Exp}$, $\sigma \in S$ and $\underline{B} \in PV^{\wedge}$ then the function $H: P^{\wedge} \to P^{\wedge}$ defined $H(\underline{C}) = \mathcal{E}[\![A]\!]\sigma[\underline{C}/\underline{B}]$ is monotonic and continuous.

1.25 Corollary

Each recursively defined process satisfies its defining equation in the following senses:

a) $\mathcal{E}[\![\text{rec}_j(B_1,\ldots,B_k).(A_1,\ldots,A_k)]\!]\sigma = \mathcal{E}[\![A_j]\!]\sigma[C_1/B_1]\ldots[C_k/B_k]$
where $C_r = \mathcal{E}[\![\text{rec}_r(B_1,\ldots,B_k).(A_1,\ldots,A_k)]\!]\sigma$.

b) In the recursion

$$\begin{array}{l} \lambda \in \Gamma_1 \Rightarrow B_\lambda \Leftarrow A_1 \\ \cdots\cdots \\ \cdots\cdots \\ \lambda \in \Gamma_k \Rightarrow B_\lambda \Leftarrow A_k \end{array}$$

($\{\Gamma_1,\ldots,\Gamma_k\}$ a partition of some indexing set Λ)

we have $\mathcal{E}[\![B_g$, where.....$]\!]\sigma = \mathcal{E}[\![A_i^\kappa]\!]\sigma[\underline{C}/\underline{B}]$ if $\pi(g)\sigma = B_\kappa \;\&\; \kappa \in \Gamma_i$ where $C_\mu = \mathcal{E}[\![B_\mu$, where.....$]\!]\sigma$ and A_i^κ is the element of Exp obtained by substituting the value "κ" into the parameter "λ".

1.26 Lemma

If all free occurrences of $B_1,\ldots,B_k$ have the form of simple calls of process variables (i.e. "B" rather than "B_g") and if A_j has no bound occurrence of any variable which occurs free in any of the A_i then

$$\text{rec}_j(B_1,\ldots,B_k).(A_1,\ldots,A_k) \equiv A_j[C_1'/B_1]\ldots[C_k'/B_k]$$

where $C_r' = \text{rec}_r(B_1,\ldots,B_k).(A_1,\ldots,A_k)$ and syntactic substitution for (free occurrences of) simple calls of process variables is defined in the obvious way.

1.27 Lemma

If in the recursion of 1.25(b) we have

(i) for all states σ, $\pi(g)\sigma \in \{B_\mu | \mu \in \Gamma_i\}$;

(ii) A_i contains no bound occurrence of any variable which occurs free in any of $A_1,\ldots,A_k$;

(iii) the only occurrences of B_μ ($\mu \in \Lambda$) in A_j are free and of the form B_h for some h s.t. $\text{rge}(h) \subseteq \Lambda$;

then 'B_g, where....' $\equiv A_i^*$, where A_i^* is the expression obtained by substituting 'B_h, where... (usual clauses)' for all calls of process variables s.t. $\text{rge}(h) \subseteq \Lambda$.

The conditions on the form of recursive calls within processes could be relaxed somewhat in 1.26 and 1.27, but this would be at the expense of simplicity and the extra cases included would very rarely arise in practice.

From here on we will identify elements of Exp with no free variables with their unique (by 1.21) counterparts in P. For this type of expression we thus identify "≡" (equivalence over Exp) with actual equality over P. We are enabled by 1.22 - 1.27 to manipulate these expressions and their subcomponents with a great deal of freedom. Subject to a few conditions on the use of variables we can treat the components of an expression very much as though they were elements of P, and recursive definitions very much like fixed point equations over P and its product spaces. We can thus practically forget the formal semantic definition of our language when doing manipulations, confident that we could fully justify all of them if challenged.

There is a clear ambivalence in our attitude to constructs in our language, which is brought about by the identification of elements of Exp and elements of P. One the one hand we can think of an expression as a formal piece of syntax which can only be given a value by reference to a semantic function and a state. On the other hand we can think of it as being an operator dependent on its free variables; there being a class of results (1.20 - 1.27) showing these two approaches to be congruent. Although from here on it is the second of these two ideas which will dominate, there are several important uses for the more formal approach. The most obvious of these is that it enables us to define precisely what we mean by any construct, however involved.

A second reason is that there are other, similar languages which are almost impossible to define without "states" and some kind of formal denotational semantics. The language thich we have adopted is very rich in its potential for parallelism and recursion, but lacks such features as assignment to variables (other than by communication with other processes) and conditional statements. The choice of language made in this chapter largely results from the fact that most of the work in the subsequent chapters chapters is chiefly concerned with parallelism, recursion, and the connection between the two. The introduction of further constructs such as those mentioned above would have tended to intr duce extra cases into our arguments (many of which are complex enough already). It is not difficult, either in the deterministic model P or the nondeterministic model we meet in chapter four, to devize semantics for languages with more conventional constructs (e.g. assignment, "<u>if</u>, <u>then</u>, <u>else</u>", less rich recursion) by simple adaptation of the formal semantics we have met in this chapter. Having done this one can prove congruences between the languages and thereby apply results obtained about one to the other.

1.28 <u>Example</u>

In the language we use assignment to variables is often modelled by elaborate subscripting in mutual recursion, and conditional statements by "$\square$". For examples of the former see 1.19 (i, ii); for examples of the latter see 6.9. The following example represents a process which inputs a succession of symbols from some set T and after each one outputs the largest one it has received so far. (We assume that T is endowed with some total order $<$, and that the syntax for IA is extended to encompass the usage below.)

$$A, \text{ where } \quad A \Leftarrow ?x{:}T \rightarrow !x \rightarrow B_x$$
$$x \in T \Rightarrow B_x \Leftarrow (?y{:}\{y \mid y > x\} \rightarrow !y \rightarrow B_y)$$
$$\square\, (?y{:}\{y \mid y \leqslant x\} \rightarrow !x \rightarrow B_x)$$

<u>Additional combinators</u>

From time to time we will want to use combinators additional to those which we have already met. These will be defined over P on the understanding that if the syntax introduced in 1.16 were extended to include them then the semantic function $\mathcal{E}$ would be extended in the natural way: if op is a (unary) operator over P and <u>op</u> is the syntactic object intended to model it then $\mathcal{E}[\![\underline{op}(A)]\!]\sigma = op(\mathcal{E}[\![A]\!]\sigma)$. A few operators are introduced in this way below.

(i) If $f{:}\Sigma - \Sigma \cup \{*\}$ is a function such that $f^{-1}(\surd) \subseteq \{\surd\}$ then one can extend f to Σ^* by the rule:

$$f(\langle\rangle) = \langle\rangle;\ f(w\langle a\rangle) = f(w) \text{ if } f(a) = *;$$
$$f(w\langle a\rangle) = f(w)\langle f(a)\rangle \text{ otherwise .}$$

One can now define an operator on P by

$$f^{-1}(A) = \{w \in \underline{run} \mid f(w) \in A\} .$$

For example if $X \subseteq \Sigma$ and f_X is the function defined

$f(a) = a$ if $a \in X$; $f(a) = *$ otherwise

then $f_X^{-1}(A)$ is the process which behaves like A in the set X and is prepared to do anything outside X; one can think of $f_X^{-1}(A)$ as "A ignoring $(\Sigma - X)$". Note that $(A_X \|_Y B) = (X \cup Y)^* \cap f_X^{-1}(A) \cap f_Y^{-1}(B)$.

(ii) If f is any function of the above type it can be extended to an operator over P directly:

$f(A) = \{f(w) \mid w \in A\}$.

For example if "a" is any name used for elements of Σ we can define a function strip(a) on Σ as follows:

$\text{strip}(a)(c) = c$ if $c \notin a.\Sigma$

$= d$ if $c = a.d$ for some $d \in \Sigma$.

Regarded as an operator over P strip(a) is the operator which removes the name "a" from any of its operand's communications which have it. This operator can be combined with others to construct some useful combinators. For example if T is some set of unnamed symbols and $T \cup ?T \cup !T \subseteq \Sigma$ we can define a "pipe" operator "$\gg$" which is intended to take two processes which communicate in $?T \cup !T$, join the inputs of one to the outputs of the other, and hide the resulting internal communications:

$A \gg B = (\text{strip}!(A)_{T \cup ?T} \|_{T \cup !T} \text{strip}?(B))/T$.

This and similar operators will be studied in some detail in the later chapters.

The two types of operators described above (f and f^{-1}) are known as alphabet transformers. Note that for all allowable functions f we have f^{-1}(run) = run and f(abort) = abort.

Chapter 2 :- Recursion Induction in the Deterministic Model

Recursion induction is the proof rule:

> If in a recursive definition of a process we can prove some property true of the process by assuming it true of all recursive calls, then we can infer the property true of the process.

In the model described in chapter 1 the above rule can be formalised thus:

2.1 In the (mutual) recursive definition $\underset{\sim}{A} = F(\underset{\sim}{A})$, where $\underset{\sim}{A}$ is a vector of processes in $P^\wedge$ and $F:P^\wedge \rightarrow P^\wedge$, if R is a predicate on $P^\wedge$ such that $\forall \underset{\sim}{B}.\ R(\underset{\sim}{B}) \Rightarrow R(F(\underset{\sim}{B}))$ then we can infer that R(A) holds (where $\underset{\sim}{A} = fix(F)$).

In this chapter we will examine the validity of this rule, give examples of its use and show how it can be extended in several directions. The rule as it stands is not universally valid. Below are some counter-examples which will motivate conditions upon F and R to ensure validity.

2.2 The rule 2.1 is completely without the base case usually found in inductive proofs. It is therefore possible to "prove" the predicate "false" true of any process whatsoever.

2.3 Of the process $A \Leftarrow A$ we could prove any predicate.

2.4 Of the process $A \Leftarrow A \,\square\, B$ (B some fixed process) we could prove "A = C" for any process $C \supseteq B$.

2.5 Of the process $A = a \rightarrow A$ we could prove the predicate "every trace of A can be extended to include a "b" ".

In 2.2 it is clearly the predicate which is at fault. Any predicate which is not equivalent to "false" is <u>satisfiable</u> in the sense $\exists \underset{\sim}{A}.\ R(\underset{\sim}{A})$. It is this condition that will represent the base case of our inductions. In 2.3 it is clearly the recursion which is at fault, and under the assumption that "=C" is a valid predicate this must also be the case in 2.4. In 2.5 the recursion is very straightforward and well-defined so we must assume that the predicate (with its implicit existential quantifier) is at fault.

There are several possible pairs of conditions on predicates and functions which, along with satisfiability, can be shown to make the rule 2.1 valid.

We firstly examine the most commonly useful of these. Define the notion of <u>restriction</u> of a deterministic process to strings of length n or less as follows:

2.6 $A\upharpoonright n = \{w\in A \mid |w|\leq n\}$

This can be extended to vectors of processes in P^{Λ} by pointwise restriction:

2.7 $(\underset{\sim}{A}\upharpoonright n)_{\lambda} = (A_{\lambda})\upharpoonright n$ for $\underset{\sim}{A}\in P^{\Lambda}$

These definitions give rise to some obvious results:

2.8 <u>Lemma</u>

a) $A \dot{\in} P \Rightarrow A\upharpoonright o = \underline{abort}$

b) $\underset{\sim}{A} \in P^{\Lambda} \Rightarrow (\underset{\sim}{A}\upharpoonright n)\upharpoonright m = \underset{\sim}{A}\upharpoonright \min(n,m)$

c) $(a \to A)\upharpoonright n+1 = a \to (A\upharpoonright n)$ if $a\in\Sigma$, $A\in P$

d) $\underset{\sim}{A}\in P^{\Lambda} \Rightarrow A_{\lambda} = \bigcup_{n=0}^{\infty}((\underset{\sim}{A}\upharpoonright n)_{\lambda})$ if $\lambda\in\Lambda$

Suppose $F:P^{\Lambda}\to P^{\Lambda}$, define F to be <u>constructive</u> if it satisfies

2.9 $\forall n\in N.\forall \underset{\sim}{A}\in P^{\Lambda}.\ F(\underset{\sim}{A})\upharpoonright n+1 = F(\underset{\sim}{A}\upharpoonright n)\upharpoonright n+1$

and <u>non-destructive</u> if it satisfies

2.10 $\forall n\in N.\forall \underset{\sim}{A}\in P^{\Lambda}.\ F(\underset{\sim}{A})\upharpoonright n = F(\underset{\sim}{A}\upharpoonright n)\upharpoonright n$

Informally these definitions mean that by looking at the result of applying F to the n-step behaviour of $\underset{\sim}{A}$ we can deduce the n-step behaviour of $F(\underset{\sim}{A})$ in the case of a non-destructive F and the n+1 step behaviour in the case of a constructive F. From the point of view of recursion a constructive F would seem to be a desirable thing, since then one can always guarantee to be able to deduce the behaviour of fix(F) up to any finite time from only finitely many iterations.

2.11 <u>Lemma</u>

a) If $F:P^{\Lambda}\to P^{\Lambda}$ and $G:P^{\Lambda}\to P^{\Lambda}$ are both non-destructive then so are FoG and GoF.

b) If a function is constructive then it is non-destructive.

c) If $F:P^{\Lambda}\to P^{\Lambda}$ is constructive and $G:P^{\Lambda}\to P^{\Lambda}$ is non-destructive then FoG and GoF are both constructive.

d) If $F:P^{\Lambda}\to P^{\Lambda}$ is constructive then it has a unique fixed point.

proof

a) $F(G(\underset{\sim}{A}))\upharpoonright n = F(G(\underset{\sim}{A})\upharpoonright n)\upharpoonright n$ (by nondestructiveness of F)

$\qquad = F(G(\underset{\sim}{A}\upharpoonright n)\upharpoonright n)\upharpoonright n$ (" " " " " G)

$\qquad = F(G(\underset{\sim}{A}\upharpoonright n))\upharpoonright n$ (" " " " " F)

b) $F(\underset{\sim}{A})\upharpoonright n = (F(\underset{\sim}{A})\upharpoonright n+1)\upharpoonright n$ (by 2.8(b))

$\qquad = (F(\underset{\sim}{A}\upharpoonright n)\upharpoonright n+1)\upharpoonright n$

$\qquad = F(\underset{\sim}{A}\upharpoonright n)\upharpoonright n$

c) $F(G(\underset{\sim}{A}))\upharpoonright n+1 = F(G(\underset{\sim}{A})\upharpoonright n)\upharpoonright n+1$

$\qquad = F(G(\underset{\sim}{A}\upharpoonright n)\upharpoonright n)\upharpoonright n+1$

$\qquad = F(G(\underset{\sim}{A}\upharpoonright n))\upharpoonright n+1$

GoF is very similar to this.

d) Suppose F has two distinct fixed points $\underset{\sim}{A}$ & $\underset{\sim}{B}$. Then by 2.8 (a) & (d) we have $\underset{\sim}{A}\upharpoonright o = \underset{\sim}{B}\upharpoonright o$ and $\exists n.\ \underset{\sim}{A}\upharpoonright n \neq \underset{\sim}{B}\upharpoonright n$. Hence there is some m such that $\underset{\sim}{A}\upharpoonright m = \underset{\sim}{B}\upharpoonright m$ but $\underset{\sim}{A}\upharpoonright m+1 \neq \underset{\sim}{B}\upharpoonright m+1$.

But then $\underset{\sim}{A}\upharpoonright m+1 = F(\underset{\sim}{A})\upharpoonright m+1$ (as $\underset{\sim}{A} = F(\underset{\sim}{A})$)

$\qquad = F(\underset{\sim}{A}\upharpoonright m)\upharpoonright m+1$ (as F is constructive)

$\qquad = F(\underset{\sim}{B}\upharpoonright m)\upharpoonright m+1$ ($\underset{\sim}{B}\upharpoonright m = \underset{\sim}{A}\upharpoonright m$)

$\qquad = F(\underset{\sim}{B})\upharpoonright m+1$ (as F is constructive)

$\qquad = \underset{\sim}{B}\upharpoonright m+1$ contradicting assumption

Thus F must only have one fixed point (it has at least one by Tarski).

Now suppose that $R:\hat{P} \rightarrow \{true, false\}$ is a predicate. Define R to be <u>continuous</u> if it satisfies

2.12 $\forall \underset{\sim}{A}.(\neg R(\underset{\sim}{A}) \Rightarrow \exists n.(\forall \underset{\sim}{B}.(\underset{\sim}{B}\upharpoonright n = \underset{\sim}{A}\upharpoonright n) \Rightarrow \neg R(\underset{\sim}{B})))$

or equivalently

2.13 $\forall \underset{\sim}{A}.(\forall n.\exists \underset{\sim}{B}.(\underset{\sim}{A}\upharpoonright n = \underset{\sim}{B}\upharpoonright n)\ \&\ R(\underset{\sim}{B})) \Rightarrow R(\underset{\sim}{A})$

(Later this will sometimes be termed <u>weak continuity</u> to contrast with another notion which will be introduced at the end of this chapter.)

Informally a predicate is continuous if, for every $\underset{\sim}{B}$ such that $R(\underset{\sim}{B})$ does not hold we can be sure of this after some finite time. Observe that the predicate of 2.5 does not satisfy this as at any finite time the "b"s might be "just around the corner".

2.14 Theorem

If $F:P^{\wedge} \to P^{\wedge}$ is a constructive function and if R is a continuous satisfiable predicate then rule 2.1 is valid.

$$\text{(i.e. } (\forall \underset{\sim}{B}.\ R(\underset{\sim}{B}) \Rightarrow R(F(\underset{\sim}{B}))) \Rightarrow R(fix(F)) \text{)}$$

proof

Since R is satisfiable there is some $\underset{\sim}{B}$ such that $R(\underset{\sim}{B})$ and (by 2.8 (a)) $\underset{\sim}{B}\restriction o = \underset{\sim}{A}\restriction o$ (where $\underset{\sim}{A} = fix(F)$). Claim that for all n we have $R(F^n(\underset{\sim}{B}))$ & $(F^n(\underset{\sim}{B})\restriction n = \underset{\sim}{A}\restriction n)$. This is true when n=0, so assume it true for n.

$$
\begin{aligned}
\text{Then}\quad \underset{\sim}{A}\restriction n{+}1 &= F(\underset{\sim}{A})\restriction n{+}1 && \text{(as } F(\underset{\sim}{A}) = \underset{\sim}{A}) \\
&= F(\underset{\sim}{A}\restriction n)\restriction n{+}1 && \text{(as F is constructive)} \\
&= F(F^n(\underset{\sim}{B})\restriction n)\restriction n{+}1 && \text{(by induction)} \\
&= F(F^n(\underset{\sim}{B}))\restriction n{+}1 && \text{(as F is constructive)} \\
&= F^{n+1}(\underset{\sim}{B})\restriction n{+}1 && \text{as desired}
\end{aligned}
$$

also $R(F^n(\underset{\sim}{B})) \Rightarrow R(F(F^n(\underset{\sim}{B})))$ completing induction

Hence the above holds for all n. Thus by 2.13 (since R is continuous) we must have $R(\underset{\sim}{A})$, as desired.

This pair of conditions is useful since they normally hold of well-defined recursions and predicates we wish to prove of them and also because there are some simple rules for checking that they hold.

2.15 Lemma

The following are all non-destructive functions of their variables: $a \to A$, $A \square B$, $A;B$, $(A_X \|_Y B)$, $a.A$.

The following are constructive functions of A:

$a \to A$

$B;A$ when $\surd \notin B^o$

$(B_X \|_Y A)$ when $Y \subseteq X$ and $B^o \cap Y = \emptyset$

The proofs of all these results are elementary and follow directly from the definitions of the operators.

$$
\begin{aligned}
\text{For example } (B;A)\restriction n &= \{w \mid \neg\exists v.w{=}v\surd \ \&\ w{\in}B \ \&\ |w|{\le}n\} \\
&\quad \cup \{wv \mid w\surd{\in}B \ \&\ v{\in}A \ \&\ v \ne \langle\rangle \ \&\ |wv|\le n\} \\
&\quad \text{(in line 2 the case } v{=}\langle\rangle \text{ is included in line 1)} \\
&\subseteq \{w \mid \neg\exists v.w{=}v\surd \ \&\ w{\in}B\restriction n\} \cup \{wv \mid w\surd \in B\restriction n \ \&\ v{\in}A\restriction n\} \\
&\subseteq (B\restriction n);(A\restriction n) \\
\Rightarrow B;A\restriction n &\subseteq ((B\restriction n);(A\restriction n))\restriction n \quad (\supseteq \text{ follows by monotonicity})
\end{aligned}
$$

We can now combine this lemma with 2.11 a&c to give the following result:

2.16 Theorem

Suppose that the function $F:\hat{P} \to \hat{P}$ is such that each component of $F(\underset{\sim}{A})$ is a syntactic expression involving only expressions independent of $\underset{\sim}{A}$ (such as skip and abort), "A_μ"s and the combinators $a \to B$, $a?x:T \to B(x)$, $?x:T \to B(x)$, $B;C$, $B \,\square\, C$, $a.B$ and $(B_X \|_Y C)$. Then provided that every recursive call of an A_λ is guarded directly (as in $a \to A_\lambda$) or indirectly (as in $a \to (A_\lambda \square B)$) the function F is constructive.

The hiding operator A/X is not non-destructive so we must be careful when we use it in a recursive definition. There are important cases where the $(A \| a::B)$ and $(A \gg B)$ operators are constructive and non-destructive. These will be examined later on.

2.17 Examples

The following recursions are all constructive:

a) $A \Leftarrow (a \to A) \,\square\, (b \to \text{skip})$

b) $B \Leftarrow (?t \to T) \,\square\, (?f \to F)$

$T \Leftarrow (?t \to T) \,\square\, (?f \to F) \,\square\, (t \to T)$

$F \Leftarrow (?t \to T) \,\square\, (?f \to F) \,\square\, (f \to F)$

c) $A \Leftarrow ((a \to (A \,\square\, (b \to \text{skip})))_X \|_Y A) \,\square\, b \to A$

where $X = \{a, b, \surd\}$, $Y = \{b, \surd\}$

None of the following is constructive:

d) $A \Leftarrow (A \,\square\, a \to \text{skip})$

e) $A \Leftarrow a.A \,\square\, a \to A$

f) $A \Leftarrow (a \to A) \,\square\, (B;A)$

$B \Leftarrow (a \to \text{skip}) \,\square\, A$

We will come back to the case of iterated recursions (i.e the body of a recursive definition including a recursion) later.

We now turn our attention to the classification of continuous predicates. The following is a list of the classes of predicates which can easily be shown to be continuous. Some of these classes can clearly be derived from others. In the next chapter we will examine the continuous predicates as a topology of the space P which will yield more insight on them.

2.18 <u>Theorem</u>

The following predicates are all (weakly) continuous:

$R(\underset{\sim}{A}) =$ (i) $A_\lambda = B$

(ii) $A_\lambda = A_\mu$

(iii) $A_\lambda \subseteq B$

(iv) $A_\lambda \supseteq B$

(v) $A_\lambda \subseteq A_\mu$

(vi) $A_\lambda \neq B$ if there is an upper bound on $\{|w| \mid w \in B\}$

(vii) A_λ is deadlock-free

(viii) $w \in A_\lambda \Rightarrow p(w)$ where p is any predicate on Σ^*

(ix) $w \in A_\lambda \Rightarrow p_w((A_\lambda \text{ \underline{after} } w)^0)$ if p_w are predicates on $\mathcal{P}(\Sigma)$

(x) $R_1(F(\underset{\sim}{A}))$ if $\exists g: N \to N$ s.t. $\forall \underset{\sim}{B}. \forall n. F(\underset{\sim}{B}) \restriction n = F(\underset{\sim}{B} \restriction g(n)) \restriction n$

(F monotonic $P^\Lambda \to P^\Gamma$ and R_1 predicate on P^Γ)

(xi) $F(\underset{\sim}{A})_\lambda \subseteq B$ for any continuous $F: P^\Lambda \to P^\Gamma$

(xii) $\bigwedge_{\gamma \in \Gamma} R_\gamma(\underset{\sim}{A})$ for any set Γ

(xiii) $R_1(\underset{\sim}{A}) \vee R_2(\underset{\sim}{A})$

where B is any constant process and R_1, R_2 & R_γ are all continuous predicates.

<u>example proofs</u>

(i) If $(A_\lambda \neq B)$ then either $\exists w.\ w \in A_\lambda$ & $w \notin B$

or $\exists w.\ w \notin A_\lambda$ & $w \in B$

In either case if we put $n = |w|$ then $C_\lambda \restriction n = A_\lambda \restriction n \Rightarrow C_\lambda \neq B$

so $\underset{\sim}{C} \in P^\Lambda \Rightarrow \underset{\sim}{C} \restriction n = \underset{\sim}{A} \restriction n \Rightarrow \neg(C_\lambda = B)$

(vi) Observe that $C \in P \Rightarrow (C = B \Leftrightarrow C \restriction n+1 = B)$ where n is the upper bound upon $\{|w| \mid w \in B\}$, since either $C \restriction n \neq B$ or C contains some string of length n+1 if $C \neq B$.

Thus $A_\lambda = B \Rightarrow \forall C. ((C \restriction n+1 = (A_\lambda) \restriction n+1) \Rightarrow (C = B))$

(x) $\neg R_1(F(\underset{\sim}{A})) \Rightarrow \exists n.\ \underset{\sim}{B} \restriction n = F(\underset{\sim}{A}) \restriction n \Rightarrow \neg R(\underset{\sim}{B})$

$\Rightarrow F(\underset{\sim}{B}) \restriction n = F(\underset{\sim}{A}) \restriction n \Rightarrow \neg R(F(\underset{\sim}{B}))$

but $\underset{\sim}{B} \restriction g(n) = \underset{\sim}{A} \restriction g(n) \Rightarrow F(\underset{\sim}{B}) \restriction n = F(\underset{\sim}{A}) \restriction n$

so $\underset{\sim}{B} \restriction g(n) = \underset{\sim}{A} \restriction g(n) \Rightarrow \neg R(F(\underset{\sim}{B}))$

The satisfiability of predicates is very often obvious. The examples (i) - (viii) on the previous page fall into this category (provided, in case (viii) that $p(\langle\rangle)$ holds, though if not the predicate is not satisfiable as every process contains $\langle\rangle$). Also (xiii) is satisfiable if one of its components is, and (xii) is if each of its components is and they all refer to different components of $\underline{A}$. A method for getting round this last requirement (that all the R_γ should be independent) will be indicated later. (xi) will be satisfied by $\underline{\text{abort}}^{\wedge}$ if it is satisfiable. The cases (ix) and (x) are more difficult and it is necessary to examine their individual instances. Often when it is not easy to prove a given predicate of these forms satisfiable it is possible to break it down into two or more sections and prove each of these to be true of fix(G) (where G is the function of the recursion). For example to prove $F(fix(G)) = B$ it suffices to prove $F(fix(G)) \supseteq B$ and $F(fix(G)) \subseteq B$.

It is now possible to give some provably valid applications of rule 2.1.

2.19 Example (C.A.R. Hoare)

We attempt to model an integer counter in two ways. An integer counter will be expected to communicate in the alphabet {up, down, iszero}. It will always be non-negative, so when it has value zero it will not accept a "down" instruction, and it will only communicate "iszero" when it has value zero. The instruction "up" will increase its value by one, the instruction "down" will reduce its value by one and "iszero" will not affect its value.

Our first attempt is by mutual recursion (with $\Lambda = \mathbb{N}$)

$$COUNT_0 \Leftarrow (iszero \rightarrow COUNT_0) \,\Box\, (up \rightarrow COUNT_1)$$

$$COUNT_{n+1} \Leftarrow (down \rightarrow COUNT_n) \,\Box\, (up \rightarrow COUNT_{n+2})$$

Of these processes in isolation it is possible to prove several properties without too much difficulty, notably that each $COUNT_n$ is deadlock-free and that $w \in COUNT_n \Rightarrow ups(w)+n \geq downs(w)$ (where ups and downs have the obvious interpretation). This second property is just the conjunction of independent cases of 2.18 (viii), as the first is of cases of (vii).

For the second attempt we will first construct a process which terminates successfully after inputting one more "down" than "up"s.

$$POS \Leftarrow (down \rightarrow \underline{skip}) \mathbin{[]} (up \rightarrow POS;POS)$$

(If POS inputs "down" on its first step it terminates, but if it inputs an "up" it must subsequently input two more "down"s than "up"s.)

We can now use this process to define an integer counter with initial value zero.

$$ZERO \Leftarrow (iszero \rightarrow ZERO) \mathbin{[]} (up \rightarrow POS;ZERO)$$

(ZERO will accept "iszero" and remain unaltered, or accept "up" and become ZERO again after inputting one more "down" than "up"s.)

It is an immediate consequence of 2.16 that all the recursions defined above (COUNT, POS & ZERO) are constructive. Also if we define $C_o = ZERO$, $C_{n+1} = POS;ZERO$ then the predicate $R(\underset{\sim}{A}) = \forall n.\ A_n = C_n$ is continuous and satisfiable (by 2.18 et seq). Claim that COUNT satisfies R. To prove this it is now sufficient to prove $R(\underset{\sim}{A}) \Rightarrow R(F(\underset{\sim}{A}))$ where F is the function of the COUNT recursion.

Suppose $R(\underset{\sim}{A})$

Then

$$\begin{aligned} F(\underset{\sim}{A})_o &= (iszero \rightarrow A_o) \mathbin{[]} (up \rightarrow A_1) \\ &= (iszero \rightarrow ZERO) \mathbin{[]} (up \rightarrow POS;ZERO) \quad \text{by assumption} \\ &= ZERO = C_o \quad \text{by definition of ZERO} \\ F(\underset{\sim}{A})_{n+1} &= (down \rightarrow A_n) \mathbin{[]} (up \rightarrow A_{n+2}) \\ &= (down \rightarrow skip;C_n) \mathbin{[]} (up \rightarrow POS;POS;C_n) \quad \text{by assumption} \\ &= ((down \rightarrow skip) \mathbin{[]} (up \rightarrow POS;POS));C_n \\ &= POS;C_n \quad \text{by definition of POS} \\ &= C_{n+1} \end{aligned}$$

This completes the proof. Thus in particular we have proved that $ZERO = COUNT_o$, so the two processes we have defined are essentially equivalent.

2.20 Example: Buffers

Define a buffer to be a process which accepts input from some set T and later outputs it in the same order. We can express this condition formally as: B is a buffer if $B \subseteq (?T \cup !T)^*$ and $w \in B \Rightarrow ins(w) \geq outs(w)$, where $ins(w) = strip?(w \restriction ?T)$ and

$outs(w) = strip^{!}(w \restriction {!}T)$. This condition, being of type (viii) only represents a form of partial correctness (it is satisfied by abort). If we were in addition to demand that B be free from deadlock this would eliminate abort but not other pathological cases like $B \Leftarrow ?a \rightarrow B$ which can only input "a"s and can never output at all. One condition which seems to be satisfactory in all respects is the following:

$$\begin{aligned} w \in B \;\Rightarrow\; & w \in (?T \cup T)^{*} \;\&\; ins(w) \geqslant outs(w) && \text{(i)} \\ & \&\; ins(w) = outs(w) \Rightarrow (B \text{ after } w)^{o} = ?T && \text{(ii)} \\ & \&\; ins(w) > outs(w) \Rightarrow (B \text{ after } w)^{o} \cap {!}T \neq \emptyset && \text{(iii)} \end{aligned}$$

The interpretation of line (ii) is that an empty buffer must be prepared to accept any input, and line (iii) means that a non-empty buffer must be prepared to output. The fact that this output is correct is implied by line (i).

That the above predicate is satisfiable is by no means obvious. However each of the lines individually is ((i) & (iii) by abort and (ii) by $?x{:}T \rightarrow$ abort) so if for any given process B we can prove all three independently we will have established that they are simultaneously satisfiable. The process we choose to do this job for us is the one-place buffer

$$B \Leftarrow ?x{:}T \rightarrow ({!}x \rightarrow B) \quad .$$

The function associated with this recursion can be written

$$F(A) = \{\langle\rangle\} \cup \{\langle ?x\rangle \mid x \in T\} \cup \{\langle ?x\,!x\rangle w \mid w \in A\}$$

Since each of the three predicates is continuous and satisfiable and this recursion is constructive, to prove them true of B it will suffice to show $\forall A.\ R(A) \Rightarrow R(F(A))$ for each R.

Suppose (i) holds of A $\quad w \in F(A) \Rightarrow w = \langle\rangle \quad (\Rightarrow ins(w) = outs(w)\ )$

$\quad$ or $w = \langle ?x\rangle$ for some $x \in T$

$\quad\quad (\Rightarrow ins(w) > outs(w)\ )$

$\quad$ or $w = \langle ?x\,!x\rangle v$ for some $x \in T$, $v \in A$

$\quad\quad (\Rightarrow ins(w) = \langle x\rangle ins(v)$

$\quad\quad\quad \geqslant \langle x\rangle outs(v)$

$\quad\quad\quad \geqslant outs(w)\ )$

Thus (i) holds of F(A).

Suppose (ii) holds of A. Then $w \in F(A)$ & $(ins(w)=outs(w))$

$\Rightarrow$ (a) $w=\langle\rangle$ or (b) $w=\langle ?x!x\rangle v$ where $x \in T$, $v \in A$ & $ins(v)=outs(v)$

(a) $\Rightarrow$ $(F(A)\ \underline{after}\ w)^{o} = ?T$

(b) $\Rightarrow$ $(F(A)\ \underline{after}\ w)^{o} = (A\ \underline{after}\ v)^{o} = ?T$

as desired

Thus (ii) holds of F(A).

Suppose (iii) holds of A. Then $w \in F(A)$ & $(ins(w)>outs(w))$

$\Rightarrow$ (a) $w=\langle ?x\rangle$ for some $x \in T$

or (b) $w=\langle ?x!x\rangle v$ for some $x \in T$, $v \in A$ s.t. $ins(v)>outs(v)$

(a) $\Rightarrow$ $!x \in (F(A)\ \underline{after}\ w)^{o}$ since $\langle ?x!x\rangle \in F(A)$ for all $x \in T$

(b) $\Rightarrow$ $(F(A)\ \underline{after}\ w)^{o} = (A\ \underline{after}\ v)^{o}$

$\Rightarrow$ $(F(A)\ \underline{after}\ w)^{o} \cap !T \neq \emptyset$ as A satisfies (iii)

This completes the proof that B is a buffer. Therefore when in future we want to use the predicate "is a buffer" (in the definition on the previous page) we will be entitled to assume that it is satisfiable.

<u>Extensions of our rule</u>

Firstly it is possible to weaken the definition of constructive function to F non-destructive and

2.21 $\forall \underset{\sim}{A}.\forall n.\exists m.(F^{m}(\underset{\sim}{A})\restriction n+1 = F^{m}(\underset{\sim}{A}\restriction n)\restriction n+1)$

Under this revised condition it is possible to prove the validity of 2.1 with exactly the same conditions upon the predicate as before. The proof of this is very similar to that of 2.14. This result does not get us much further in dealing with single recursions, but can sometimes be of use in mutual recursions.

e.g. $A \Leftarrow (a \rightarrow A) \square (b \rightarrow B) \square (c \rightarrow \underline{skip})$

$B \Leftarrow A;B$

this is not constructive in the sense of 2.9, but is in the sense of 2.21 with m=2 for all $\underset{\sim}{A}$,n

A more useful technique is (for mutual recursions) the concept of constructiveness relative to a partial order on Λ, the set of indices of the recursion. Define a partial order to be well founded if every subset of it contains some minimal elements (or equivalently if it contains no infinite descending chains). Suppose $<$ is a well founded partial order.

A function will be <u>constructive relative to</u> $<$ if it is non-destructive and at any point is constructive in all variables except those strictly less than the point in question.
If $F : P^\Lambda \to P^\Lambda$ this condition can be formally expressed:

2.22 $\forall \underset{\sim}{A}.\forall n.\forall \lambda.(F(\underset{\sim}{A})_\lambda)\lceil n+1 = (F(\underset{\sim}{A}^*)_\lambda)\lceil n+1$
where $A^*_\mu = A_\mu \lceil n+1$ if $\mu < \lambda$
$= A_\mu \lceil n$ if $\neg(\mu < \lambda)$.

A function will be <u>non-destructive relative to</u> $<$ if

2.23 $\forall \underset{\sim}{A}.\forall n.\forall \lambda.(F(\underset{\sim}{A})_\lambda)\lceil n+1 = (F(\underset{\sim}{A}^*)_\lambda)\lceil n+1$
where $A^*_\mu = A_\mu \lceil n+1$ if $\mu \leq \lambda$
$= A_\mu \lceil n$ if $\neg(\mu \leq \lambda)$

Notice that to be non-destructive a function must not, in some sense, work against the partial order by pulling information from high points to low points.

2.24 Suppose $\Lambda = \{0,1\}$ with partial order $0<1$
Then the function $F(A_0, A_1) = (a \to (A_0 \,\square\, A_1)\ ,\ b.A_0)$
is constructive,
the function $F(A_0, A_1) = ((a \to A_1) \,\square\, A_0\ ,\ A_1)$
is non-destructive,
but the function $F(A_0, A_1) = (A_1, A_0)$ is neither.

It is possible to prove a calculus of these definitions in very much the same manner as for the old kind:

2.25 <u>Theorem</u>
If $F, G : P^\Lambda \to P^\Lambda$ are functions and $<$ is a well-founded partial order on Λ then (relative to $<$)

a) If F & G are both non-destructive then so is FoG .
b) If F is constructive then it is non-destructive.
c) If F is constructive and G is non-destructive then both FoG and GoF are constructive.
d) If F is constructive then it has a unique fixed point.

<u>proof</u>

a) $F(G(\underset{\sim}{A}))_\lambda \lceil n+1 \supseteq F(G(\underset{\sim}{A}^*))_\lambda \lceil n+1$ by monotonicity.
$F(G(\underset{\sim}{A}))_\lambda \lceil n+1 = F(\underset{\sim}{B}^*)_\lambda \lceil n+1$ where $B^*_\mu = G(\underset{\sim}{A})_\mu \lceil n$ if $\neg(\mu \leq \lambda)$
$= G(\underset{\sim}{A}\lceil n)_\mu \lceil n$
$\subseteq G(\underset{\sim}{A}^*)_\mu \lceil n$ as $\underset{\sim}{A}^* \supseteq \underset{\sim}{A}\lceil n$
$\subseteq G(\underset{\sim}{A}^*)_\mu$

$$B^*_\mu = G(\underset{\sim}{A})_\mu \restriction n+1 \quad \text{if } \mu \leq \lambda$$
$$= G(\underset{\sim}{A}')_\mu \restriction n+1 \quad \text{where } A'_\gamma = A_\gamma \restriction n \quad \text{if } \neg(\gamma \leq \mu)$$
$$= A_\gamma \restriction n+1 \text{ if } \gamma \leq \mu$$
$$\text{but } \gamma \leq \mu \Rightarrow \gamma \leq \lambda \text{ so } \underset{\sim}{A}' \subseteq \underset{\sim}{A}^*$$
$$\Rightarrow \quad B^*_\mu \subseteq G(\underset{\sim}{A}^*)_\mu \restriction n+1 \subseteq G(\underset{\sim}{A}^*)_\mu$$

Thus $\underset{\sim}{B}^* \subseteq G(\underset{\sim}{A}^*)$

so $F(\underset{\sim}{B}^*)_\lambda \restriction n+1 \subseteq F(G(\underset{\sim}{A}^*))_\lambda \restriction n+1$

which completes the proof of part a)

b) This follows by monotonicity since the $\underset{\sim}{A}^*$ of 2.23 is greater than that of 2.22.

c) The proofs of both parts of this are almost identical to that of part a).

d) Suppose F has two distinct fixed points $\underset{\sim}{A}$ & $\underset{\sim}{B}$. As in the proof of 2.11 there must be some n such that $\underset{\sim}{A} \restriction n = \underset{\sim}{B} \restriction n$ but $\underset{\sim}{A} \restriction n+1 \neq \underset{\sim}{B} \restriction n+1$. Now $\underset{\sim}{A} \restriction n+1 \neq \underset{\sim}{B} \restriction n+1$ implies there must be some λs such that $(A_\lambda) \restriction n+1 \neq (B_\lambda) \restriction n+1$. Thus there is at least one λ which is minimal with respect to this property. For such a λ we must have:

$$(A_\lambda) \restriction n+1 = (F(\underset{\sim}{A})_\lambda) \restriction n+1 \qquad \text{(as } \underset{\sim}{A} \text{ is a fixed point of F)}$$
$$= (F(\underset{\sim}{A}^*)_\lambda) \restriction n+1 \quad \text{where } A^*_\mu = A_\mu \restriction n \quad \text{if } \neg(\mu < \lambda)$$
$$(= B_\mu \restriction n \quad \text{as } A \restriction n = B \restriction n)$$
$$= A_\mu \restriction n+1 \text{ if } \mu < \lambda$$
$$(= B_\mu \restriction n+1 \text{ as } \lambda \text{ is minimal})$$

[The above remarks show that $\underset{\sim}{A}^* = \underset{\sim}{B}^*$, the corresponding restriction of $\underset{\sim}{B}$.]

$$= (F(\underset{\sim}{B}^*)_\lambda) \restriction n+1$$
$$= (F(\underset{\sim}{B})_\lambda) \restriction n+1 \qquad \text{(as F is constructive)}$$
$$= (B_\lambda) \restriction n+1 \qquad \text{(as } \underset{\sim}{B} \text{ is a fixed point of F)}$$

which contradicts our choice of λ. Thus F must only have a single fixed point.

Note that in the case of Λ finite (or any other case where the partial order has a bound on the lengths of its ascending chains) this condition is implied by the earlier extension, for then by recursing a fixed number of times it is possible to refer only to $\underset{\sim}{A} \restriction n$ when calculating each $F^m(\underset{\sim}{A})_\mu \restriction n+1$.

For a general well founded partial order it is necessary to restrict the types of predicates which may be used in proofs.

2.26 Example

Let $\Lambda = N$, then the recursion

$$A_0 \Leftarrow a \rightarrow \underline{skip}$$

$$A_{n+1} \Leftarrow A_n$$

is constructive relative to the standard order on N, and the predicate $R(\underset{\sim}{A}) = \exists n.\ A_n = \underline{abort}$ is continuous under definition 2.12 and is clearly satisfiable. But $R(\underset{\sim}{A}) \not\Rightarrow R(F(\underset{\sim}{A}))$ since $A_n = \underline{abort} \Rightarrow F(\underset{\sim}{A})_{n+1} = \underline{abort}$ and R does not hold of fix(F). Thus rule 2.1 is not justified in this case.

2.27 Theorem

Suppose F is constructive relative to the well founded partial order $<$ on Λ and the predicate R on P^{Λ} is satisfiable. Then rule 2.1 is justified provided R has the form $\forall K.R_K(\underset{\sim}{A})$ $(K \in \mathcal{K}, \text{say})$ where each R_K is continuous and is independent of all components of $\underset{\sim}{A}$ except for a finite set $\Gamma_K \subseteq \Lambda$, and all the Γ_K are disjoint.

(i.e. each R_K is a predicate on finitely many components of $\underset{\sim}{A}$ and no component occurs in more than one R_K.)

proof (technical)

As in the proof of 2.14 set $\underset{\sim}{A} = \text{fix}(F)$. Since R is satisfiable there must be some $\underset{\sim}{B}$ s.t. $\underset{\sim}{B}\lceil o = \underset{\sim}{A}\lceil o$ and $R(\underset{\sim}{B})$, and since R is continuous, if we assume $\neg R(\underset{\sim}{A})$ there must be some n s.t. $\exists \underset{\sim}{B}^*.\ \underset{\sim}{B}^*\lceil n = \underset{\sim}{A}\lceil n \ \&\ R(\underset{\sim}{B}^*)$ and $\forall \underset{\sim}{B}.\ \underset{\sim}{B}\lceil n{+}1 = \underset{\sim}{A}\lceil n{+}1 \Rightarrow \neg R(\underset{\sim}{B})$.

If $\Gamma \subseteq \Lambda$ is non-empty, denote the set of its least elements by $\mu(\Gamma)$. Define a map from the ordinal numbers to $\mathcal{P}(\Lambda)$ as follows:

$$f(0) = \emptyset$$

$$f(\alpha+1) = \mu(\Lambda - f(\alpha)) \cup f(\alpha)$$

$$f(\alpha) = \bigcup_{\beta<\alpha} f(\beta) \quad \text{if } \alpha \text{ is a limit ordinal}$$

This map is clearly monotonic, and is strictly monotonic unless $\Lambda = f(\alpha)$ for some α. But a strictly monotonic function is one-one, so under the assumption that $\forall\alpha.\ f(\alpha) \neq \Lambda$ we have produced a 1-1 function from On to the set $\mathcal{P}(\Lambda)$. This is impossible by Hartog's theorem (see for example Enderton p195) so we must have $\Lambda = f(\alpha^*)$ for some α^*.

Claim that each $f(\alpha)$ $(\alpha \leq \alpha^*)$ satisfies $\lambda \in f(\alpha)$ & $\mu < \lambda \Rightarrow \mu \in f(\alpha)$.

This true if $\alpha = 0$.

Assume true of α and that $\lambda \in f(\alpha+1)$ & $\mu < \lambda$

if $\lambda \in f(\alpha)$ then we must have $\mu \in f(\alpha)$ by assumption

if $\lambda \in \mu(\Lambda - f(\alpha))$ then $\mu \notin f(\alpha)$ would contradict the minimality of λ in $(\Lambda - f(\alpha))$

hence in either case $\mu \in f(\alpha) \subseteq f(\alpha+1)$, so the result is true of $\alpha+1$.

If α is a limit ordinal then if we assume the result true of all $\beta < \alpha$ the result for α follows immediately since if $\mu < \lambda$ then

$\lambda \in f(\alpha) \Rightarrow \exists \beta$ s.t. $\lambda \in f(\beta)$ & $\beta < \alpha$

$\Rightarrow \mu \in f(\beta) \subseteq f(\alpha)$

Thus by transfinite induction the result holds of all $\alpha \leq \alpha^*$.

Claim next that for each $\alpha \leq \alpha^*$ there is some $\underset{\sim}{B} \in P^{\Lambda}$ satisfying

$$\underset{\sim}{B}\restriction n = \underset{\sim}{A}\restriction n \ \ \& \lambda \in f(\alpha) \Rightarrow B_\lambda \restriction n+1 = A_\lambda \restriction n+1 \ \ \& \ R(\underset{\sim}{B})$$

This is true for $\alpha = 0$ (put $\underset{\sim}{B} = \underset{\sim}{B}^*$)

Suppose $\underset{\sim}{B}$ satisfies the above for α, claim $F(\underset{\sim}{B})$ satisfies it for $\alpha+1$.

Firstly $F(\underset{\sim}{B})\restriction n = F(\underset{\sim}{B}\restriction n)\restriction n = F(\underset{\sim}{A}\restriction n)\restriction n = F(\underset{\sim}{A})\restriction n = \underset{\sim}{A}\restriction n$

and also $R(F(\underset{\sim}{B}))$ holds since $R(\underset{\sim}{B})$ does.

Now $\lambda \in f(\alpha+1) \Rightarrow (F(\underset{\sim}{B})_\lambda)\restriction n+1 = (F(\underset{\sim}{C})_\lambda)\restriction n+1$

where $C_\mu = B_\mu \restriction n+1$ if $\mu < \lambda$

$C_\mu = B_\mu \restriction n$ if $\neg(\mu < \lambda)$

But $B_\mu \restriction n = A_\mu \restriction n$ by assumption

and we established above that $\mu < \lambda \in f(\alpha+1)$ implies $\mu \in f(\alpha)$, so $\mu < \lambda \Rightarrow B_\mu \restriction n+1 = A_\mu \restriction n+1$.

Thus $C_\mu = A_\mu \restriction n+1$ if $\mu < \lambda$

$= A_\mu \restriction n$ if $\neg(\mu < \lambda)$

Hence $(F(\underset{\sim}{B})_\lambda)\restriction n+1 = (F(\underset{\sim}{A})_\lambda)\restriction n+1$ as F is constructive

$= A_\lambda \restriction n+1$

Finally suppose α is a limit ordinal and that for each $\beta < \alpha$ there is some $\underset{\sim}{B}^\beta$ satisfying the above. For each $\kappa \in K$ we can write the set Γ_κ as $\{\lambda_1, \lambda_2, \ldots, \lambda_r, \lambda_{r+1}, \ldots, \lambda_s\}$ where $\lambda_1 - \lambda_r \in f(\alpha)$ and $\lambda_{r+1} - \lambda_s \notin f(\alpha)$

As there are only finitely many "λ_j"s in $f(\alpha)$ there must be some $\beta < \alpha$ s.t. $f(\beta) \cap \Gamma_\kappa = f(\alpha) \cap \Gamma_\kappa$. Put $B_{\lambda_j} = (B^\beta)_{\lambda_j}$ for $j = 1 - s$.
If we do this for every $\kappa \in K$ there will be no clashes in the B_λ's so derived since the Γ_κ are disjoint.

Formally define $\underset{\sim}{B}$ as follows:

$$B_\lambda = (B^\beta)_\lambda \quad \text{if } \exists\kappa.\ \lambda\in\Gamma_\kappa \ (\beta \text{ dependent on } \kappa)$$
$$= A_\lambda \quad \text{otherwise}\ .$$

Each R_κ holds of the $\underset{\sim}{B}$ so defined since it holds of $\underset{\sim}{B}^\beta$ and is independent of all $\{B_\lambda | \lambda\in\Gamma_\kappa\}$. Also $\underset{\sim}{B}\lceil n = \underset{\sim}{A}\lceil n$ since this is true of each $\underset{\sim}{B}^\beta$ by assumption. Finally if $\lambda\in f(\alpha)$ then

$$\exists\kappa.\lambda\in\Gamma_\kappa \quad \Rightarrow B_\lambda = (B^\beta)_\lambda$$
$$\Rightarrow B_\lambda\lceil n{+}1 = (B^\beta)_\lambda\lceil n{+}1$$
$$= A_\lambda\lceil n{+}1 \quad \text{as } \lambda\in f(\beta) \text{ and } B^\beta \text{ satisfies our hypothesis for } \beta.$$

otherwise $B_\lambda = A_\lambda$

$$\Rightarrow B_\lambda\lceil n{+}1 = A_\lambda\lceil n{+}1$$

This completes the proof that the above $\underset{\sim}{B}$ satisfies the condition for α.

Hence by transfinite induction on $\alpha\leq\alpha^*$ there must be some $\underset{\sim}{B}$ corresponding to α^*. But then this $\underset{\sim}{B}$ satisfies $R(\underset{\sim}{B})$ and $\underset{\sim}{B}\lceil n{+}1 = \underset{\sim}{A}\lceil n{+}1$ (since $\Lambda = f(\alpha^*)$) which contradicts the assumption that $\forall\underset{\sim}{B}.\ \underset{\sim}{B}\lceil n{+}1 = \underset{\sim}{A}\lceil n{+}1 \Rightarrow \neg R(\underset{\sim}{B})$.

This result admits rather easier proofs in less general cases, such as predicates of the form $\forall\lambda.R_\lambda(A_\lambda)$ and also when each $\lambda\in\Lambda$ has some limit (dependent on λ) in N to the lengths of chains ascending to it.

Below is a simple example involving a finite partial order. See 2.3 for a more general example.

2.28 <u>Example</u> (after David Park)

Define four processes by mutual recursion:

$$A_0 \Leftarrow (a \rightarrow A_1) \,\square\, (b \rightarrow A_2)$$
$$A_1 \Leftarrow (a \rightarrow A_3) \,\square\, A_0$$
$$A_2 \Leftarrow A_0 \,\square\, (b \rightarrow A_3)$$
$$A_3 \Leftarrow A_2 \,\square\, A_1$$

It is easily verified that these equations define a recursion which is constructive relative to the standdard order on $\{0,1,2,3\}$. The predicate $\forall i.C_i = B$, where B is defined $B \Leftarrow (a \rightarrow B) \,\square\, (b \rightarrow B)$ is clearly satisfiable and of the correct form, so the rule is justified in this case.

Assume $R(\underset{\sim}{C})$

$$
\begin{aligned}
\text{Then}\quad F(\underset{\sim}{C})_0 &= (a \rightarrow C_1) \,\square\, (b \rightarrow C_2) \\
&= (a \rightarrow B) \,\square\, (b \rightarrow B) && \text{by assumption} \\
&= B && \text{by definition of } B \\
F(\underset{\sim}{C})_1 &= (a \rightarrow C_3) \,\square\, C_0 \\
&= (a \rightarrow B) \,\square\, B && \text{by assumption} \\
&= (a \rightarrow B) \,\square\, ((a \rightarrow B) \,\square\, (b \rightarrow B)) && \text{by definition of } B \\
&= (a \rightarrow B) \,\square\, (b \rightarrow B) && \text{by properties of } \square \\
&= B && \text{by definition of } B
\end{aligned}
$$

(case 2 is almost identical to case 1)

$$
\begin{aligned}
F(\underset{\sim}{C})_3 &= C_1 \,\square\, C_2 \\
&= B \,\square\, B && \text{by assumption} \\
&= B
\end{aligned}
$$

Hence $R(F(\underset{\sim}{C}))$ holds, so we can infer $\forall i. A_i = B$.

The next extension to our rule will allow us, amongst other things, to deal with "partial predicates" which are independent of one or more components of the mutual recursion.

2.29 <u>Theorem</u>

Suppose F is a constructive function (possibly relative to some partial order) and H is non-destructive. ($F,H : P^{\wedge} \rightarrow P^{\wedge}$)
Suppose further that $H(fix(F)) = fix(F)$, then

(i) HoF & FoH are both constructive

(ii) $fix(HoF) = fix(FoH) = fix(F)$

(iii) If R is continuous (or satisfies the conditions of 2.27 in the $<$ case) and satisfiable then
$(\forall \underline{A}.\ R(\underline{A}) \Rightarrow R(H(F(\underline{A})))) \Rightarrow R(fix(F))$ and
$(\forall \underline{A}.\ R(\underline{A}) \Rightarrow R(F(H(\underline{A})))) \Rightarrow R(fix(F))$

<u>proof</u>

(i) is just a restatement of 2.11(c) and 2.25(c)

(ii) We have $H(F(fix(F))) = F(H(fix(F))) = fix(F)$ by assumption, so fix(F) is <u>a</u> fixed point of HoF and of FoH. But by (i) and 2.11(d) or 2.25(d) these functions have <u>unique</u> fixed points, so the result follows.

(iii) This follows since, by (ii), anything which is true of fix(HoF) or fix(FoH) is also true of fix(F)

This result has the following application to the partial predicates described on the last page:

2.30 Theorem

Suppose $F: P^{\Lambda} \to P^{\Lambda}$ is a constructive function and that R_1 & R_2 are two predicates on P^{Λ} which satisfy the following conditions:

a) R_1 is continuous and satisfiable and is independent of some subset Γ of Λ.

b) R_2 has the form $R(\underset{\sim}{A}) \equiv ((\underset{\sim}{A} \upharpoonright \Gamma) = H(\underset{\sim}{A}))$ for some non-destructive function $H: P^{\Lambda} \to P^{\Gamma}$ which is constructive in its Γ-component (in a sense to be made clear below).

c) $R_2(fix(F))$ holds.

d) $\forall \underset{\sim}{A}.(R_1(\underset{\sim}{A}) \ \& \ R_2(\underset{\sim}{A})) \Rightarrow R_1(F(\underset{\sim}{A}))$

Then $R_1(fix(F))$ holds.

proof

Write every element of P^{Λ} in the form $(\underset{\sim}{A}, \underset{\sim}{B})$, where $\underset{\sim}{A} \in P^{\Lambda - \Gamma}$ & $\underset{\sim}{B} \in P^{\Gamma}$. A function will now be constructive in its Γ-component if it satisfies $\forall \underset{\sim}{A}.\forall \underset{\sim}{B}.\forall n.H(\underset{\sim}{A}, \underset{\sim}{B}) \upharpoonright n+1 = H(\underset{\sim}{A}, \underset{\sim}{B} \upharpoonright n) \upharpoonright n+1$.

For any $\underset{\sim}{A} \in P^{\Lambda - \Gamma}$ we can now define a function $G_{\underset{\sim}{A}}: P^{\Gamma} \to P^{\Gamma}$ by $G(\underset{\sim}{B}) = H(\underset{\sim}{A}, \underset{\sim}{B})$. This $G_{\underset{\sim}{A}}$ is constructive and so has a unique fixed point. We can write $H^*(\underset{\sim}{A}, \underset{\sim}{B}) = (\underset{\sim}{A}, fix(G_{\underset{\sim}{A}}))$. H^* is a non-destructive function, as will be shown in 2.36 and since each $G_{\underset{\sim}{A}}$ has a unique fixed point we must have $H^*(fix(F)) = fix(F)$ (condition (c) above implies that $fix(F) \upharpoonright \Gamma$ is a fixed point of $G_{\underset{\sim}{C}}$ where $\underset{\sim}{C} = fix(F) \upharpoonright (\Lambda - \Gamma)$).

Now apply 2.29 with F as above, $H = H^*$ and $R = R_1$ (if the FoH case is used) or $R = R_1$ & R_2 (if the HoF case is used). Note that H^* does not alter the truth of R_1 since it is the identity on all components upon which R_1 depends.

Informally this result allows us to make certain additional assumptions about recursive calls which lie outside the domain of the predicate we are trying to prove, though we must be able to justify these assumptions independently.

In the example we will see shortly R_2 will have the form $\gamma \in \Gamma \Rightarrow A_\gamma = fix(F)_\gamma$ (so that H is a constant function).

Other suitable "H"s for use in 2.29 are F^k, $H(A) = A \sqcap B$ for any $B \subseteq fix(F)$ and the device below for taking fixpoints one at a time.

2.31 If $F:P^{\Lambda}\to P^{\Lambda}$ is a constructive function defining a mutual recursion and $\Lambda=\Gamma\cup\Delta$ is a partition of the indexing set then we can write every element of P^{Λ} in the form $(\underset{\sim}{A},\underset{\sim}{B})$, where $\underset{\sim}{A}\in P^{\Gamma}$ & $\underset{\sim}{B}\in P^{\Delta}$ and there are some constructive $G:P^{\Lambda}\to P^{\Gamma}$ & $K:P^{\Lambda}\to P^{\Delta}$ s.t. $F(\underset{\sim}{A},\underset{\sim}{B}) = (G(\underset{\sim}{A},\underset{\sim}{B}),K(\underset{\sim}{A},\underset{\sim}{B}))$. A suitable H to use with 2.29 is $H(\underset{\sim}{A},\underset{\sim}{B}) = (\underset{\sim}{A},fix(\lambda\underset{\sim}{B}.K(\underset{\sim}{A},\underset{\sim}{B})))$. This could be useful if one wanted to do induction on the variables of a recursion one at a time. This topic of iterated fixed points will be treated in more detail later.

2.32 Example (illustrating 2.27 & 2.30)

For some $T\subseteq\Sigma$ take $\Lambda = T^{*}-\{\langle\rangle\}$.

We seek to model "stacks" which terminate successfully as soon as they first become empty.

Define $S_{\langle a\rangle} \Leftarrow (!a \to \underline{skip}) \,[]\, (?b{:}T \to S_{\langle ba\rangle})$ $\quad a\in T$

$S_{\langle a\rangle w} \Leftarrow (!a \to S_{w}) \,[]\, (?b{:}T \to S_{\langle ba\rangle w})$ $\quad a\in T,\ w\in\Lambda$

A second version of this:

$Q_{\langle a\rangle} \Leftarrow (!a \to \underline{skip}) \,[]\, (?b{:}T \to Q_{\langle ba\rangle})$ $\quad a\in T$

$Q_{w\langle a\rangle} \Leftarrow Q_{w};Q_{\langle a\rangle}$

Theorem: $w\in\Lambda \Rightarrow S_{w} = Q_{w}$

In proving this we first show that $a\in T$ & $w\in\Lambda \Rightarrow S_{w\langle a\rangle} = S_{w};S_{\langle a\rangle}$.

This is done using 2.30, the partial predicate used being $R_{1}(\underset{\sim}{A}) \equiv (\forall a\in T.\forall w\in\Lambda.A_{w\langle a\rangle} = S_{w};S_{\langle a\rangle})$ (this is independent of $A_{\langle a\rangle}, a\in T$). The secondary predicate we use is $R_{2}(\underset{\sim}{A}) \equiv (\forall a\in T.\ A_{\langle a\rangle} = S_{\langle a\rangle})$ which arises from the constant $H(\underset{\sim}{A})_{\langle a\rangle} = S_{\langle a\rangle}\ (a\in T)$. By construction the $\underset{\sim}{S}$-recursion is constructive, R_{1} is satisfiable and continuous and $R_{2}(\underset{\sim}{S})$ holds. Thus it only remains to show clause (d) of 2.30.

Assume $R_{1}(\underset{\sim}{A})$ & $R_{2}(\underset{\sim}{A})$

Then $F(\underset{\sim}{A})_{\langle ab\rangle} = (!a \to A_{\langle b\rangle}) \,[]\, (?c{:}T \to A_{\langle cab\rangle})$ $\quad a,b\in T$

$= (!a \to S_{\langle b\rangle}) \,[]\, (?c{:}T \to S_{\langle ca\rangle};S_{\langle b\rangle})$ by assumption

$=((!a \to \underline{skip}) \,[]\, (?c{:}T \to S_{\langle ca\rangle}));S_{\langle b\rangle}$

$= S_{\langle a\rangle};S_{\langle b\rangle}$ as desired (by definition of $\underset{\sim}{S}$)

$F(\underset{\sim}{A})_{\langle b\rangle w\langle a\rangle} = (!b \to A_{w\langle a\rangle}) \,[]\, (?c{:}T \to A_{\langle cb\rangle w\langle a\rangle})$ $\quad (a,b\in T,\ w\in\Lambda)$

$= (!b \to S_{w};S_{\langle a\rangle}) \,[]\, (?c{:}T \to S_{\langle cb\rangle w};S_{\langle a\rangle})$ by assumption

$= ((!b \to S_{w}) \,[]\, (?c{:}T \to S_{\langle cb\rangle w}));S_{\langle a\rangle}$

$= S_{\langle b\rangle w};S_{\langle a\rangle}$ as desired

This establishes $R_{1}(F(\underset{\sim}{A}))$, so we can infer $R_{1}(\underset{\sim}{S})$ by 2.30 .

Having established this result, we can now prove $\forall w. Q_w = S_w$ by induction on $\underset{\sim}{Q}$. Observe that the $\underset{\sim}{Q}$-recursion is constructive relative to the partial order induced by the length of $w \in \Lambda$. Let R' be the predicate $\forall w. A_w = S_w$. This is clearly satisfiable and is allowable in terms of 2.27.

Suppose $R'(\underset{\sim}{A})$, then if F is the function of the $\underset{\sim}{Q}$-recursion

$$F(\underset{\sim}{A})_{\langle a \rangle} = (!a \to \underline{skip}) \,[\!]\, (?b{:}T \to A_{\langle ba \rangle})$$
$$= (!a \to \underline{skip}) \,[\!]\, (?b{:}T \to S_{\langle ba \rangle}) \quad \text{by assumption}$$
$$= S_a \quad \text{by definition of } \underset{\sim}{S}$$
$$F(\underset{\sim}{A})_{w\langle a \rangle} = A_w ; A_{\langle a \rangle}$$
$$= S_w ; S_{\langle a \rangle} \quad \text{by assumption}$$
$$= S_{w\langle a \rangle} \quad \text{by earlier result}$$

This establishes $R'(F(\underset{\sim}{A}))$, so we can infer $R'(\underset{\sim}{Q})$ as desired.

The following two extensions are also valid, their proofs are omitted in this model but are given in chapter 5 for the non-deterministic model.

2.33 Suppose $A \in \mathcal{P}(P^\Lambda)$, define $A \restriction n = \{\underset{\sim}{B} \restriction n \mid \underset{\sim}{B} \in A\}$

Define a function $F: \mathcal{P}(P^\Lambda) \to \mathcal{P}(P^\Lambda)$ to be constructive if it satisfies $\forall A. \forall n. F(A) \restriction n{+}1 = F(A \restriction n) \restriction n{+}1$.

Suppose the predicate R on P^Λ is satisfiable and continuous.

Then $(\forall A'. (\forall \underset{\sim}{B}. (B \in A' \Rightarrow R(\underset{\sim}{B}))) \Rightarrow (\forall B. (\underset{\sim}{B} \in F(A') \Rightarrow R(\underset{\sim}{B})))) \,\&\, (A \subseteq F(A))$ implies $(\forall \underset{\sim}{B}. (\underset{\sim}{B} \in A \Rightarrow R(\underset{\sim}{B})))$.

This result is chiefly of use in proving general theorems about processes, for example the fundamental buffer theorem of chapter 5 (which is also true in this deterministic model).

2.34 Suppose $R_1 \ldots R_n$ are predicates which are individually satisfiable and all continuous but which may not have been shown to be simultaneously satisfiable. Suppose $F: P^\Lambda \to P^\Lambda$ is a constructive function which can be written as $F_i^* \circ D_n$ for $i = 1,2 \ldots n$, where $D_n: P^\Lambda \to (P^\Lambda)^n$ and $F_i^*: (P^\Lambda)^n \to P^\Lambda$, $D_n(\underset{\sim}{A}) = (\underset{\sim}{A}, \underset{\sim}{A}, \ldots, \underset{\sim}{A})$.

Then if $\forall (\underset{\sim}{A}_1 \ldots \underset{\sim}{A}_n). R_1(\underset{\sim}{A}_1) \,\&\, \ldots \,\&\, R_n(\underset{\sim}{A}_n) \Rightarrow R_1(F_1^*(\underset{\sim}{A}^*)) \,\&\ldots\&\, R_n(F_n^*(\underset{\sim}{A}^*))$ (where $\underset{\sim}{A}^* = (\underset{\sim}{A}_1, \ldots, \underset{\sim}{A}_n)$) we can infer $R_1(fix(F)) \&\ldots\& R_n(fix(F))$.

Informally this rule represents

uctive hypotheses for each recursive call of a process. The method of defining F_i^* depends on which assumption we wish to make of each call in proving $R_i(F(\underset{\sim}{A}))$.

Iterated recursions

As has been indicated earlier it is possible to use subsiduary recursions within the body of a recursive definition.

e.g. $A = (a \rightarrow \underline{skip}) \Box (b \rightarrow rec\ B.(b \rightarrow A; B))$

It is possible to analyse this type of construction in several ways. The first of these is the well-known result below which relates mutual recursions to iterated fixed points.

2.35 If $\Lambda = \Gamma \cup \Delta$ is a partition of Λ then as in 2.31 we can write elements of P^{Λ} and $P^{\Lambda} \rightarrow P^{\Lambda}$ as pairs $(\underset{\sim}{A}, \underset{\sim}{B})$ and (G,K) respectively. It can quite easily be shown that

$$fix(G,K) = (fix(\lambda \underset{\sim}{A}.G(\underset{\sim}{A}, fix(\lambda \underset{\sim}{B}.K(\underset{\sim}{A},\underset{\sim}{B})))), fix(\lambda \underset{\sim}{B}.K(fix(\lambda \underset{\sim}{A}.G(\underset{\sim}{A},\underset{\sim}{B})),\underset{\sim}{B})))$$

This gives a method for converting mutual recursions into iterated ones and vice-versa. Thus the example above is equal to A, where

$$A \Leftarrow (a \rightarrow \underline{skip}) \Box (b \rightarrow B)$$
$$B \Leftarrow b \rightarrow A; B$$

It is also possible to analyse recursions with free process variables (such as $rec\ B.(b \rightarrow A;B)$ above), to decide if they are constructive, non-destructive etc. The need to do this has already arisen in 2.30 & 2.31.

2.36 Theorem (in the same notation as 2.35)

Suppose $F:(P^{\Gamma} \times P^{\Delta}) \rightarrow P^{\Gamma}$ is continuous and is non-destructive in its Γ component. Then the function $G(\underset{\sim}{B}) = fix(\lambda \underset{\sim}{A}.F(\underset{\sim}{A},\underset{\sim}{B}))$ is non-destructive if F is non-destructive in its Δ component and constructive if F is in its Δ component.

proof

We have the identity $fix(H) = \bigcup_{n=0}^{\infty} H^n(\bot)$ for any continuous function H on a complete lattice of which $\bot$ is the minimal element.

Thus $G(\underset{\sim}{B}) = \bigcup_{n=0}^{\infty} K^n(\underset{\sim}{\bot})$ where $K(\underset{\sim}{A}) = F(\underset{\sim}{A},\underset{\sim}{B})$ and $\gamma \in \Gamma \Rightarrow \bot_{\gamma} = \underline{abort}$

We prove first the non-destructive case.

It is sufficient to show for arbitrary $\underset{\sim}{B}$ & n that $G(\underset{\sim}{B}\restriction n)\restriction n = G(\underset{\sim}{B})\restriction n$.

Claim that $\forall m. K^m(\underset{\sim}{\bot})\restriction n = L^m(\underset{\sim}{\bot})\restriction n$ where $L(\underset{\sim}{A}) = F(\underset{\sim}{A},\underset{\sim}{B}\restriction n)$

This is true when m=0 (both sides $= \underset{\sim}{\bot}$) so for induction assume true of m.

Then $K^{m+1}(\underset{\sim}{\perp})\lceil n = F(K^m(\underset{\sim}{\perp}),\underset{\sim}{B})\lceil n$

$= F(K^m(\underset{\sim}{\perp})\lceil n, \underset{\sim}{B}\lceil n)\lceil n$ as F is non-destructive

$= F(L^m(\underset{\sim}{\perp})\lceil n, \underset{\sim}{B}\lceil n)\lceil n$ by assumption

$= F(L^m(\underset{\sim}{\perp}), \underset{\sim}{B}\lceil n)\lceil n$ as F is non-destructive

$= L^{m+1}(\underset{\sim}{\perp})\lceil n$ as desired

Hence $K^m(\underset{\sim}{\perp})\lceil n = L^m(\underset{\sim}{\perp})\lceil n$ for all m.

Thus $G(\underset{\sim}{B})\lceil n = (\bigcup_{m=0}^{\infty} K^m(\underset{\sim}{\perp}))\lceil n$

$= (\bigcup_{m=0}^{\infty} (K^m(\underset{\sim}{\perp})\lceil n))$ as $\lceil n$ is continuous

$= (\bigcup_{m=0}^{\infty} (L^m(\underset{\sim}{\perp})\lceil n))$

$= (\bigcup_{m=0}^{\infty} L^m(\underset{\sim}{\perp}))\lceil n$

$= G(B\lceil n)\lceil n$

The proof of the constructive case is very similar (merely change some of the "n"s into "n+1"s).

It is now possible to strengthen 2.16 to include the possibility of iterated recursions.

2.37 <u>Theorem</u>

Suppose that the function $F:\hat{P}\to\hat{P}$ is such that each component of $F(\underset{\sim}{A})$ is a syntactic expression involving only expressions independent of all process variables, process variables, the combinators $a \to B$, $a?x:T \to B(x)$, $?x:T \to B(x)$, $B;C$, $B \square C$, $a.B$ and $(B_X \| _Y C)$ and iterated recursions which bind all instances of process variables which are not A_γ^s. Then provided that every (free) recursive call of an A_γ is guarded directly (e.g. rec $B.(a \to A_\gamma;B)$) or indirectly (e.g. $a \to$ rec $B.(A_\gamma;B)$) the function F is constructive.

The proof of this is a straightforward structural induction using 2.11, 2.15, 2.36.

Operators involving hiding in their definition

The following two operators are both very useful in defining processes by recursion:

a) The Master/Slave operator $(A \| a::B)$ $(A,B \in P)$ is intended to model process A working in parallel with process B and using it as a slave. All communications with the slave will be in some $T \subseteq \Sigma$ which does not include $\surd$, and may either take the form "t" $(t \in T)$, "?t" $(t \in T)$ or "!t" $(t \in T)$. The last two of these will represent input and output respectively. The inputs of the slave B will be connected to the "a"outputs of A (all communications with the slave by A will be labelled "a") and the outputs of B will be connected to the "a"inputs of A. To avoid confusion we will insist that T, ?T and !T are all disjoint (T will be an inplicit parameter of this operator, as it will be of the next). Finally all communications of B must be with A and all communications with the slave are hidden.

$(A \| a::B) = ((A_{\Sigma}\|_{a.\Gamma} a.(\text{swap}!?(B))/a.\Gamma)$

where $\Gamma = T \cup ?T \cup !T$ and swap is defined (on Σ, Σ^* and P):

$c \in \Sigma \Rightarrow$ $\text{swapab}(c) = c$ if $c \notin (a.T \cup b.T)$

$= b.d$ if $c = a.d$ for some $d \in T$

$= a.d$ if $c = b.d$ for some $d \in T$

swapab(w) $(w \in \Sigma^*)$ is the natural elementwise extension of the definition on Σ.

swapab(A) $(A \in P)$ is the natural elementwise extension of the definition on Σ^*.

b) The pipe operator $(A \gg B)$ is intended to model the behaviour of process A accepting input from the environment, processing it in some way, passing its output to the inputs of B down a hidden channel, and B using this input to produce outputs to the environment. This is effected by transforming all outputs of A (in !T) to T and all the inputs of B (in ?T) to T, thus identifying them. All internal communication is then hidden by hiding T. Because of this device, for the operator to be meaningful, it is important that A and B cannot themselves use T for their communications.

$(A \gg B) = (\text{strip}!(A)_{T \cup ?T}\|_{T \cup !T}\text{strip}?(B))/T$,

where strip is defined in a similar way to swap:

$c \in \Sigma \Rightarrow \text{stripa}(c) = c$ if $c \notin a.T$, $= d$ if $c = a.d$ for some $d \in T$ etc.

As was mentioned earlier the hiding operator A/X is not non-destructive. Thus the two derived operators $A \gg B$ and $(A \| a::B)$ are not obviously non-destructive either. This is unfortunate since both are useful tools in defining recursive processes.

e.g. $B^{\infty} \Leftarrow ?x:T \rightarrow (B^{\infty} \gg (!x \rightarrow B))$ (where $B \Leftarrow ?x:T \rightarrow !x \rightarrow B$)

represents an infinite capacity buffer

$S \Leftarrow (X \| a::S)$

where $X \Leftarrow ?x:T \rightarrow Y_x$

$Y_x \Leftarrow (!x \rightarrow Z) \,\square\, (?y:T \rightarrow a!x \rightarrow Y_y)$

$Z \Leftarrow (a?x:T \rightarrow Y_x) \,\square\, (?x:T \rightarrow Y_x)$

represents an infinite stack

We deal first with the master/slave operator $(A \| a::B)$. It is only possible to treat this as a function of its second variable, as it is certain (if $B \neq$ abort) to be "destructive" in A. It is not too difficult to see that it is not always constructive in B (e.g. $A = a!x \rightarrow a!x \rightarrow !x \rightarrow$ skip) but sometimes is constructive (e.g. if A never communicates with its slave $(A \| a::B) = A$ is a constant function of B). It turns out that the fundamental issue is the respective numbers of communications A makes with its environment and its slave. If A must at all times have made at least as many external as internal communications then $(A \| a::B)$ is non-destructive and if A must at all times (except initially) have made more external than internal communications then $(A \| a::B)$ is constructive. This is formally expressed in the following result:

2.38 Theorem

For $n \in Z$ (positive and negative integers) define the predicate

$C^a_n(A) = \forall w.\ w \in A \Rightarrow |w \lceil \Gamma| - |w \lceil a.\Gamma| \geq \min(n, |w \lceil \Gamma \cup a.\Gamma|)$

($\Gamma \subseteq \Sigma$ is assumed to be the alphabet of "atomic" communications, so that $\Sigma = \{\surd\} \cup \Gamma \cup \bigcup\{a.\Gamma | a \in N\}$ where N is the set of process names and $\forall a.\ \Gamma \cap a.\Gamma = \emptyset$.)

Note that in any process which satisfies C^a_n every trace must contain at least n+1 "Γ"s before the first "$a.\Gamma$".

a) $C^a_n(A)$ is a continuous predicate

b) $C^a_k(A) \Rightarrow \forall n. \forall B. (A \| a::B) \lceil n+k+1 = (A \| a::B \lceil n) \lceil n+k+1$

c) $C^a_k(A) \Rightarrow C^a_k(A \| b::B)$

proof

a) This follows immediately from 2.18(viii)

b) Define the notation "$(u,v) \Leftrightarrow w$ in $(A \| a::B)$" to mean
$u \in A$, $v \in B$, $u \upharpoonright \Sigma - a.\Gamma = w$ & $swap?!(v) = strip\, a\, (u \upharpoonright a.\Gamma)$
so that $w \in (A \| a::B) \Leftrightarrow \exists u. \exists v. ((u,v) \Leftrightarrow w$ in $(A \| a::B))$.

Suppose $w \in (A \| a::B) \upharpoonright n+k+1$. If $w = \langle\rangle$ then certainly $w \in (A \| a::B \upharpoonright n)$
so suppose $w = w'\langle b \rangle$. There must be some u of minimal length
and corresponding v s.t. $(u,v) \Leftrightarrow w$ in $(A \| a::B)$.

By construction u has the form $u'\langle b \rangle$ for some u' and (by $C^a_k(A)$)
we have $|u' \upharpoonright \Gamma| - |u' \upharpoonright a.\Gamma| \geqslant \min(k, |u' \upharpoonright \Gamma \cup a.\Gamma|)$

$= |u' \upharpoonright (\Sigma - a.\Gamma)| - |u' \upharpoonright a.\Gamma| \geqslant \min(k, |u'|)$

$= |u \upharpoonright (\Sigma - a.\Gamma)| - |u \upharpoonright a.\Gamma| \geqslant \min(k+1, |u|)$ (add one to both sides)

so either $|u| < k+1$ and $|u \upharpoonright (\Sigma - a.\Gamma)| - |u \upharpoonright a.\Gamma| \geqslant |u|$

$\Rightarrow u \upharpoonright a.\Gamma = \langle\rangle$ & $u = w$

$\Rightarrow (u, \langle\rangle) \Leftrightarrow w$ in $(A \| a::B \upharpoonright n)$ (as certainly $\langle\rangle \in B \upharpoonright n$)

or $|u| \geqslant k+1$ and $|u \upharpoonright (\Sigma - a.\Gamma)| - |u \upharpoonright a.\Gamma| \geqslant k+1$

$= |w| - |v| \geqslant k+1$

$= |v| \leqslant |w| - (k+1)$

$= |v| \leqslant (n+k+1) - (k+1)$ as $|w| \leqslant n+k+1$

Thus in either case $w \in (A \| a::B \upharpoonright n)$

Hence $(A \| a::B) \upharpoonright n+k+1 \subseteq (A \| a::B \upharpoonright n) \upharpoonright n+k+1$, the reverse inclusion following by monotonicity.

c) Suppose $w \in (A \| b::B)$ and $C^a_k(A)$ where $b \neq a$ (the case $b = a$ is trivial). Then there are some u,v s.t. $(u,v) \Leftrightarrow w$ in $(A \| b::B)$.
But then $w \upharpoonright \Gamma = u \upharpoonright \Gamma$, $w \upharpoonright a.\Gamma = u \upharpoonright a.\Gamma$ and $w \upharpoonright \Gamma \cup a.\Gamma = u \upharpoonright \Gamma \cup a.\Gamma$
so $|w \upharpoonright \Gamma| - |w \upharpoonright a.\Gamma| = |u \upharpoonright \Gamma| - |u \upharpoonright a.\Gamma| \geqslant \min(k, |u \upharpoonright \Gamma \cup a.\Gamma|)$

$\min(k, |w \upharpoonright \Gamma \cup a.\Gamma|)$ as desired

Methods for the determining of these conditions will be given in chapter 6.

2.39 Examples

It is possible to model ZERO (= $COUNT_0$) (of 2.19) using $(A \| a::B)$.

$Z \Leftarrow (X \| a::Z)$

where $X \Leftarrow (iszero \rightarrow X) \,[\,]\, (up \rightarrow Y)$

$Y \Leftarrow (up \rightarrow a.up \rightarrow Y)$

$[\,]\,(down \rightarrow (a.down \rightarrow Y)$

$[\,]\,(a.iszero \rightarrow X)$)

It is an easy induction to show that X satisfies C_1^a and Y satisfies C_0^a (use of 2.1 on their joint definition). Thus the recursion $Z \Leftarrow (X \| a::Z)$ is constructive.

Claim that $COUNT_0 = (X \| a::COUNT_0)$

and that $COUNT_{n+1} = (Y \| a::COUNT_n) \quad n \in N$

This can be proved by induction on the definition of COUNT. The predicate R on P^N defined

$$R(\underset{\sim}{C}) \equiv C_0 = (X \| a::COUNT_0) \;\&\; \forall n. C_{n+1} = (Y \| a::COUNT_n)$$

is clearly satisfiable and continuous. Attempt to prove it true of COUNT .

Assume $R(\underset{\sim}{C})$ and that F is the (constructive) function of the COUNT recursion.

Then
$$\begin{aligned}(X \| a::COUNT_0) &= ((iszero \rightarrow X) \,\square\, (up \rightarrow Y) \| a::COUNT_0) \\ &= iszero \rightarrow (X \| a::COUNT_0) \\ &\quad \square\, up \rightarrow (Y \| a::COUNT_0) \\ &= iszero \rightarrow C_0 \\ &\quad \square\, up \rightarrow C_1 \qquad \text{by assumption} \\ &= F(\underset{\sim}{C})_0 \qquad \text{as desired}\end{aligned}$$

$$\begin{aligned}(Y \| a::COUNT_0) &= ((up \rightarrow a.up \rightarrow Y) \,\square\, (down \rightarrow Y^*) \| a::COUNT_0) \\ &\quad (\text{where } Y^* = (a.iszero \rightarrow X) \,\square\, (a.down \rightarrow Y)\;) \\ &= up \rightarrow (a.up \rightarrow Y \| a::COUNT_0) \\ &\quad \square\, down \rightarrow (Y^* \| a::COUNT_0) \\ &= up \rightarrow (a.up \rightarrow Y \| a::(up \rightarrow COUNT_1 \,\square\, iszero \rightarrow COUNT_0)) \\ &\quad \square\, down \rightarrow (Y^* \| a::(iszero \rightarrow COUNT_0 \,\square\, up \rightarrow COUNT_1)) \\ &= up \rightarrow (Y \| a::COUNT_1) \\ &\quad \square\, down \rightarrow (X \| a::COUNT_0) \\ &= up \rightarrow C_2 \\ &\quad \square\, down \rightarrow C_0 \qquad \text{by assumption} \\ &= F(\underset{\sim}{C})_1 \qquad \text{as desired}\end{aligned}$$

The final proof of $(Y \| a::COUNT_{n+1}) = F(\underset{\sim}{C})_{n+2}$ is very similar and is ommitted. The manipulations of the $(A \| a::B)$ operator used above are formally justified in chapter 6. We have thus established $R(F(\underset{\sim}{C}))$, so we can infer R(COUNT).

It is now simple to show $Z = COUNT_0$, for if S is the predicate " $= COUNT_0$" we have
$$\begin{aligned}S(U) &\Rightarrow (X \| a::U) \\ &= (X \| a::COUNT_0) = COUNT_0 \\ &\Rightarrow S(X \| a::U)\end{aligned}$$

and the result follows since the Z-recursion is constructive.

The intuition behind this definition of Z is that on receiving an "up" or a "down" other than the first up or last down Z passes it on to its slave, which is a copy of itself. It knows when its slave has been brought back to zero by the slave's willingness to communicate "iszero". One might try some alternative definitions, such as on every "up" or "down" other than the first sending it twice to the slave. This is what is intended in the example below.

$$Z^* \Leftarrow (X \,\|\, a::Z^*)$$
$$\text{where } X^* \Leftarrow (up \rightarrow Y^*) \,\square\, (iszero \rightarrow X^*)$$
$$Y^* \Leftarrow (up \rightarrow a.up \rightarrow a.up \rightarrow Y^*)$$
$$\square\,(down \rightarrow (a.iszero \rightarrow X^*)$$
$$\square\,(a.down \rightarrow a.down \rightarrow Y^*) \quad)$$

It is indeed possible to show (as in the previous example) that $COUNT_o$ is a fixpoint of this recursion. One would use the earlier proof as a model for showing $(X^* \| a::COUNT_o) = COUNT_o$ and $\forall n.(Y^* \| a::COUNT_{2n}) = COUNT_{2n+2}$. But this X^* does not satisfy C^a_o (nor does it satisfy C^a_z for any $z \in Z$) so we are not justified in thinking that this recursion is constructive. This is brought out by the fact that it does have several other fixed points:

e.g. $Z_1 \Leftarrow (iszero \rightarrow Z_1) \,\square\, (up \rightarrow (down \rightarrow Z_1) \,\square\, (up \rightarrow MANY) \,)$

where $MANY \Leftarrow (up \rightarrow MANY) \,\square\, (down \rightarrow MANY)$

(This Z_1 can be regarded as only understanding clearly the ideas "zero" and "one" after which it gets confused.)

$Z_2 \Leftarrow (iszero \rightarrow Z_2) \,\square\, (up \rightarrow (down \rightarrow Z_2) \,\square\, (up \rightarrow \underline{abort} \,\square\, down \rightarrow \underline{abort}))$

which is the minimal fixed point.

The conditions for the $A \gg B$ operator to be non-destructive in either of its variables are intuitively very similar to those for $(A \| a::B)$. This operator will be treated at some length in the non-deterministic model (chapter 5) so we just state the conditions here.

For $w \in \Sigma^*$ let ins(w) and outs(w) have the same definitions as in 2.20.

2.40 Theorem

Suppose that $C \subseteq (?T \cup !T)^*$, then

a) If $w \in C \Rightarrow |outs(w)| \geqslant |ins(w)|$ then $(A \gg C)$ is a non-destructive function of A.

b) If $w \in C = |outs(w)| \leqslant |ins(w)|$ then $(C \gg A)$ is a non-destructive function of A.

The proof of this for the non-deterministic model will be found in 5.30.

By this result it can be shown without too much trouble that if C is any buffer (in either the strong or the weak sense of 2.20) then so is the process recursively defined

$A \Leftarrow ?x:T \rightarrow (C \gg (!x \rightarrow A))$

as any buffer plainly satisfies condition (b) above. There wi will be several worked examples similar to this in chapter 5.

Alternative conditions for the validity of 2.1

As has been indicated earlier, there are other pairs of conditions on functions and predicates which guarantee validity of 2.1 along with satisfiablity. Firstly we can vary our interpretation of the restriction operator $\lceil n$. The only thing we assume about this operator in the proof of 2.16 is that for all $\underset{\sim}{A},\underset{\sim}{B}\in P^{\wedge}$ we have $A\lceil o = B\lceil o$. Thus if for any class of operators $\{\lceil n \mid n\in N\}$ we define continuity of predicates and constructiveness of functions we can prove a theorem which corresponds to 2.16 provided that the above condition holds. Exactly how useful this result is will clearly depend on the resulting classes of continuous predicates and constructive functions, and the difficulty of proving a given recursion to be constructive. Provided that we wish equality to be continuous we must insist $(\forall n.\underset{\sim}{A}\lceil n = \underset{\sim}{B}\lceil n) \Rightarrow \underset{\sim}{A} = \underset{\sim}{B}$ and to give any "meaning" to constructive functions we must require that $(\underset{\sim}{A}\lceil n)\lceil m = \underset{\sim}{A}\lceil \min(n,m)$. Examples of such constructions are given below:

We could turn the old definition upside down and make

$A\lceil n = \{w\in A \mid |w| < n\} \cup \{wv \mid w\in A \ \& \ |w| = n\}$

or restrict attention to a subset of Σ

$A\lceil n = \{w\in A \mid |w\lceil\Gamma| < n\}$

or nest allowable symbols

$A\lceil n = \{w\in A \mid w\in \Gamma_n^*\}$ where $\Gamma_0 = \emptyset$, $\Gamma_n \subseteq \Gamma_{n+1}$ and $\bigcup_n \Gamma_n = \Sigma$

Lastly we examine a theory which is in several ways more elegant than that which has gone before. In some ways it represents a highest common factor (or greatest lower bound) of the theories which involve a restriction operator and its study therefore gives us insight into these. It is rarely, however, that we will wish to apply it directly to a problem since it usually turns out that the easiest method is through one of the less abstract theories. Therefore it is stated here only in terms of single recursion since this makes the notation rather easier and the proofs easier to understand. Also (unlike the earlier conditions) it does not seem to generalise naturally to the non-deterministic model, because it relies critically on the existence of a top element.

We firstly observe that any reasonable (under the above conditions) restriction operator will give rise to a definition of constructiveness which implies unique fixed points (like 2.11).

In many of the proofs we have performed using 2.1 it would have been sufficient to know that the recursive equations involved had a unique fixed point (for example in the cases where we showed that another process satisfied the equation, so the two must be equal). It is possible to formulate a condition on predicates which, along with UFP (unique fixed point) guarantees the validity of 2.1 . We have already shown that a constructive function satisfies UFP, so we would expect this condition to be at least as strong as the old one. That the condition needs to be strictly stronger is demonstrated by the following example

2.41 Of the recursion $A \Leftarrow (a.A_{\Sigma} \| _{\Sigma} a.A)$ (which has UFP abort)
we have $(B \neq \underline{abort} \wedge B \neq \underline{skip}) \Rightarrow (F(B) \neq \underline{abort} \wedge F(B) \neq \underline{skip})$
so by applying 2.1 we could deduce $A \neq \underline{abort}$

A predicate was weakly continuous if its truth of a sequence of processes implied the truth of its limit (2.13). We only looked, however, at a specific sort of convergence of processes. It is possible to define a more general sort of convergence.

2.42 Suppose $\langle A_i \mid i \in N \rangle$ is a sequence of processes

Define $\limsup(A_i) = \bigcap_{i=0}^{\infty} (\bigcup_{j=i}^{\infty} (A_j))$

and $\liminf(A_i) = \bigcup_{j=0}^{\infty} (\bigcap_{i=j}^{\infty} (A_i))$

(these correspond to the usual notions in real analysis)

Say that a sequence $\langle A_i \rangle$ is <u>convergent</u> to limit B iff $\limsup(A_i) = \liminf(A_i) = B$ (and that then $\lim(A_i) = B$).

$\text{Limsup}(A_i)$ is the set of traces which are contained in infinitely many A_i and $\liminf(A_i)$ is the set of traces contained in all but finitely many A_i.

2.43 <u>Lemma</u>

a) If $\langle A_i \rangle$ is a sequence of processes then both $\limsup(A_i)$ and $\liminf(A_i)$ are processes.

b) $\limsup(A_i) \supseteq \liminf(A_i)$

c) If $\forall i.\ B_i \restriction i = A \restriction i$ then the B_i converge to A .

<u>proof</u>

a) follows since the space of processes is closed under both infinite intersections and infinite unions.

b) Clearly $i \leq j \Rightarrow \bigcup_{k=i}^{\infty}(A_k) \supseteq \bigcap_{k=j}^{\infty}(A_k)$

$$\Rightarrow \bigcup_{k=i}^{\infty}(A_k) \supseteq \bigcup_{j=i}^{\infty}(\bigcap_{k=j}^{\infty}(A_k)) = \bigcup_{j=o}^{\infty}(\bigcap_{k=j}^{\infty}(A_k))$$

$$\Rightarrow \bigcap_{i=o}^{\infty}(\bigcup_{k=i}^{\infty}(A_k)) \supseteq \bigcup_{j=o}^{\infty}(\bigcap_{k=j}^{\infty}(A_k)) \quad \text{as desired}$$

c) $w \in A \ \& \ k \geq w \Rightarrow w \in B_k$ (by definition of $A \lceil k$)

$\Rightarrow w \in \bigcup_{k=i}^{\infty}(B_k)$ for each i

$\Rightarrow w \in \mathrm{limsup}(B_k)$

Hence $\mathrm{limsup}(B_k) \subseteq A$ (*)

Also $w \in \mathrm{liminf}(B_k) \Rightarrow \exists i.\ w \in \bigcap_{j=i}^{\infty}(B_j)$

$\Rightarrow w \in B_k$ where $k = \max(i, |w|)$

$\Rightarrow w \in B_k \lceil k$

$\Rightarrow w \in A \lceil k$

$\Rightarrow w \in A$

Hence $A \subseteq \mathrm{liminf}(B_k)$ (+)

But now (*) & (+) & (b) give the desired result.

Now define a predicate to be <u>strongly continuous</u> if it satisfies:

2.44 For every convergent sequence $\langle A_i \rangle$, $\forall i.R(A_i) \Rightarrow R(\lim(A_i))$

2.45 <u>Theorem</u>

a) If a predicate is strongly continuous then it is weakly continuous.

b) If Σ is finite and a predicate is weakly continuous then it is strongly continuous.

<u>proof</u>

a) Suppose R is strongly continuous and that A_i is a sequence of processes satisfying $\forall i.R(A_i)$ and $\forall i.(A_i)\lceil i = B\lceil i$

Then by 2.43(c) $\langle A_i \rangle$ is a convergent sequence with limit B so R(B) follows by the strong continuity of R.

b) Suppose R is weakly continuous and Σ is finite.

Then if $\langle A_i \rangle$ is any convergent sequence with limit B s.t. $\forall i.R(A_i)$ we have:

$w \in B \Rightarrow \exists j.\ i \geq j \Rightarrow w \in A_j$ (as w $\mathrm{liminf}(A_j)$)

$w \notin B \Rightarrow \exists j.\ i \geq j \Rightarrow w \notin A_j$ (as w $\mathrm{liminf}(A_j)$)

Since Σ is finite, for any fixed n there is only a finite number of elements of Σ^* with length n or less.

For each of these "w"s we can thus find a j_w s.t.

$$i \geq j_w \Rightarrow (w \in B \Leftrightarrow w \in A_i)$$

Hence if $j_n = \max\{j_w \mid |w| \leq n\}$ we have

$$i \geq j_n \Rightarrow A_i \lceil n = B \lceil n$$

and in particular we thus have $(A_{j_n})\lceil n = B\lceil n$ & $R(A_{j_n})$

Since we can carry out this procedure for each n these A_{j_n}s satisfy the left hand side of 2.13, so we can deduce R(B) by the weak continuity of R.

2.46 Theorem

The following predicates are all strongly continuous:

$R(A) \equiv$	(i)	$A = B$	
	(ii)	$A \subseteq B$	
	(iii)	$A \supseteq B$	
	(iv)	$w \in A \Rightarrow p(w)$	where p is any predicate on Σ^*
	(v)	$F(A) \subseteq B$	if F is continuous
	(vi)	$F(A) \supseteq B$	if F is rev-continuous
	(vii)	$R_1(F(A))$	if F is doubly continuous
	(viii)	$\forall \gamma . R_\gamma(A)$	$(\gamma \in \Gamma)$
	(ix)	$R_1(A) \vee R_2(A)$	,

where B is any constant process and R_1, R_2 & R_γs are all strongly continuous, and a function is doubly continuous if it is both continuous and rev-continuous.

example proofs

(i) If $\langle A_i \rangle$ is any convergent sequence s.t. $\forall i . R(A_i)$ then $\forall i . A_i = B$. So certainly $\lim(A_i) = B$.

(v) Suppose $\langle A_i \rangle$ is a convergent sequence s.t. $\forall i . R(A_i)$.
Set $A = \lim(A_i)$ and suppose E is any finite process $E \subseteq A$.
For each $w \in E$ there is some j_w s.t. $i \geq j \Rightarrow w \in A_i$ (as $A = \liminf(A_i)$)
Set $j_E = \max\{j_w \mid w \in E\}$, we then have $E \subseteq A_{j_E}$.
But then $F(E) \subseteq F(A_{j_E}) \subseteq B$ and by continuity of F we have

$$F(A) = \bigcup_{\substack{E \subseteq A \\ \text{finite}}} F(E) \subseteq B$$ as desired.

(vii) The proof of this will be given later.

(ix) Suppose $\langle A_i \rangle$ is any convergent sequence s.t. $\forall i . R_1(A_i) \vee R_2(A_i)$. Then there is either an infinite subsequence satisfying R_1 or one satisfying R_2. It is easy to show that if $\langle B_i \rangle$ is a subsequence of $\langle A_i \rangle$ then $\liminf(A_i) \subseteq \liminf(B_i) \subseteq \limsup(B_i) \subseteq \limsup(A_i)$. Thus when $\langle A_i \rangle$ is convergent so must be $\langle B_i \rangle$ with the same limit.

Hence we either have an infinite subsequence (with the same limit) which satisfies R_1 ($\Rightarrow R_1(\lim(A_i))$ by continuity of R_1) or the same for R_2. We hence have $R_1(\lim(A_i)) \vee R_2(\lim(A_i))$ as desired.

2.47 Theorem

Suppose $F:P \to P$ is doubly continuous and has a unique fixed point and that R is a satisfiable and strongly continuous predicate. Then rule 2.1 is valid.

i.e $(\forall A.R(A) \Rightarrow R(F(A))) \Rightarrow R(fix(F))$

proof

Let A_o be such that $R(A_o)$ (by satisfiability)

Then $\bot \subseteq A_o \subseteq \top$ where $\bot$ = abort and $\top$ = run

Claim that for each n we have $R(A_n)$ and $F^n(\bot) \subseteq A_n \subseteq F^n(\top)$, where $A_n = F^n(A_o)$.

This is true for n=o by the above.

Suppose it true for n.

Then $R(A_{n+1})[= R(F(A_n))]$ follows by supposition on R&F.

Also we then have $F^n(\bot) \subseteq A_n \subseteq F^n(\top)$

$$\Rightarrow F^{n+1}(\bot) \subseteq F(A_n) \subseteq F^{n+1}(\top) \text{ by monotonicity}$$

$$\Rightarrow F^{n+1}(\bot) \subseteq A_{n+1} \subseteq F^{n+1}(\top) \text{ as desired.}$$

Since F is doubly continuous and has a unique fixed point we have

$$\bigcup_{n=o}^{\infty} F^n(\bot) = fix(F) = \bigcap_{n=o}^{\infty} F^n(\top) .$$

As monotonic sequences both $F^n(\bot)$ and $F^n(\top)$ are convergent and have limit fix(F).

We therefore have $fix(F) = \limsup(F^n(\bot)) \subseteq \limsup(A_n)$ (by (*))

and $fix(F) = \liminf(F^n(\top)) \supseteq \liminf(A_n)$

thus $fix(F) \subseteq \liminf(A_n) \subseteq \limsup(A_n) \subseteq fix(F)$

so $\langle A_n \rangle$ converges to fix(F)

The result then follows by the strong continuity of R.

We now examine the way in which this theory fits in with the others we have already seen.

2.48 Lemma

Suppose $\langle A_n \rangle$ is a convergent sequence and that the function $F:P \to P$ is doubly continuous. Then the sequence $\langle F(A_n) \rangle$ is convergent with limit $F(\lim(A_n))$.

proof

By monotonicity we have for any $k \in N$

$$i \geqslant k \Rightarrow A_i \subseteq \bigcup_{j=k}^{\infty}(A_j) \quad \& \quad A_i \supseteq \bigcap_{j=k}^{\infty}(A_j)$$

$$\Rightarrow F(A_i) \subseteq F(\bigcup_{j=k}^{\infty}(A_j)) \ \& \ F(A_i) \supseteq F(\bigcap_{j=k}^{\infty}(A_j))$$

$$\Rightarrow \bigcup_{j=k}^{\infty}(F(A_j)) \subseteq F(\bigcup_{j=k}^{\infty}(A_j)) \ \& \ \bigcap_{j=k}^{\infty}(F(A_j)) \supseteq F(\bigcap_{j=k}^{\infty}(A_j))$$

Thus $\limsup(F(A_j)) = \bigcap_{k=o}^{\infty}(\bigcup_{j=k}^{\infty}(F(A_j)))$

$\subseteq \bigcap_{k=o}^{\infty}(F(\bigcup_{j=k}^{\infty}(A_j)))$

$\subseteq F(\bigcap_{k=o}^{\infty}(\bigcup_{j=k}^{\infty}(A_j)))$ by rev-continuity of F

$\subseteq F(\lim(A_n))$ as $\langle A_n \rangle$ is convergent

Also $\liminf(F(A_j)) = \bigcup_{k=o}^{\infty}(\bigcap_{j=k}^{\infty}(F(A_j)))$

$\supseteq \bigcup_{k=o}^{\infty}(F(\bigcap_{j=k}^{\infty}(A_j)))$ by the above

$\supseteq F(\bigcup_{k=o}^{\infty}(\bigcap_{j=k}^{\infty}(A_j)))$ by continuity of F

$\supseteq F(\lim(A_n))$

We have thus shown $\limsup(F(A_n)) \subseteq F(\lim(A_n)) \subseteq \liminf(F(A_n))$ so the desired result follows by 2.43(b).

One immediate corollary to this result is the delayed proof of 2.46(vii).

2.49 Define a class of restriction operators to be <u>normal</u> if it each of its members is doubly continuous and:

a) $\forall A. \forall B.\ A\restriction o = B\restriction o$

b) $\forall A. \forall n.\ m. (A\restriction n)\restriction m = A\restriction \min(m,n)$

c) $\forall w. \exists n.\ A.\ w \in A \Leftrightarrow w \in A\restriction n$

Note that each of the examples on page   is normal.

2.50 <u>Theorem</u>

If a predicate is strongly continuous then it is continuous relative to every normal class of restriction operators.

<u>proof</u>

Suppose $\{\restriction n \mid n \in N\}$ is such a class, $A \in P$ and $\langle A_i \rangle$ is a sequence s.t. $\forall i. A_i \restriction i = A \restriction i \ \& \ R(A_i)$ for some strongly continuous predicate R. It will clearly suffice to show that $\langle A_i \rangle$ is a convergent sequence with limit A, for then we have R(A) by continuity of R.

Suppose $w \in \limsup(A_i)$ then by 2.49 (b)&(c) there is some n_w s.t. $\forall B.\ w \in B\lceil m \Leftrightarrow w \in B$ if $m \geqslant n_w$

Now $w \in \bigcap_{i=o}^{\infty}(\bigcup_{j=i}^{\infty}(A_j)) \Rightarrow w \in \bigcup_{j=n_w}^{\infty}(A_j)$

$\Rightarrow \exists m \geqslant n_w\ w \in A_m$

$\Rightarrow w \in A_m\lceil m$

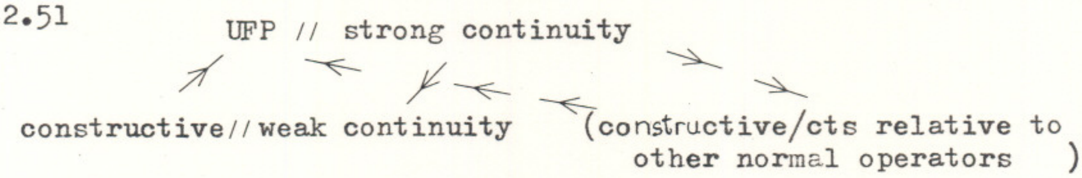
$\Rightarrow w \in A\lceil m \quad (\text{as } A_m\lceil m = A\lceil m)$

$\Rightarrow w \in A$

Hence $\limsup(A_i) \subseteq A$ and in a similar way we can prove that $A \supseteq \liminf(A_i)$ which gives us the desired result by 2.43(b).

This result is just a generalization of 2.45(a) (it would be necessary to impose stricter conditions upon the class of restriction operators to generalise 2.45(b)). It does however give us a ready-made class of predicates (2.46) to use with any normal class of restriction operators. We can draw the following diagram of conditions for validity of 2.1 showing their relative strengths.

2.51



Note that this assumes the double continuity of the function.

Postscript: Countable alphabets

It is possible to drop the condition of double continuity in 2.47 if we restrict our attention to countably infinite or finite alphabets. This has applications to the hiding operator, which as we have seen is not always reverse continuous. The fundamental fact used is that in either of these cases Σ^* is countable.

2.52 Theorem

If Σ is countable then every infinite sequence $\langle A_i \rangle$ of P has a convergent subsequence $\langle A_{n_i} \rangle$ where $i<j \Rightarrow n_i<n_j$.

proof

Enumerate Σ^* as $\{w_1, w_2, \ldots\ldots, w_i, \ldots\}$.

Claim that for each j we can find a sequence $\langle A_{n_{ji}} \rangle$ such that $n_{ji} < n_{j,i+1}$ and $n_{ji} \leq n_{j+1,i}$ and $j'<j \Rightarrow$ each n_{ji} is an $n_{j'i'}$ and $\forall i.\ A_{n_{ji}} \cap \{w_1, \ldots\ldots, w_{j-1}\} = A_{n_{j1}} \cap \{w_1, \ldots\ldots, w_{j-1}\}$

Suppose we have constructed such sequences for each $j'<j$.

If j=o we can simply set $n_{ji} = i$.

If $j=j'+1$ then either there is an infinite subsequence of $A_{n_{j'i}}$ s.t. each contains $w_{j'}$, or one s.t. each does not contain $w_{j'}$. Put n_{ji} equal to the ith $n_{j'i}$ of this subsequence.

It is easy to check that the sequence $A_{n_{ji}}$ thus formed satisfies the required conditions.

Thus the required sequence exists for each j.

Now set $m_i = n_{ii}$. We have $m_i < m_{i+1}$ by construction.

We know that $\limsup(A_{m_i}) \supseteq \liminf(A_{m_i})$ by 2.43(b) so suppose $w \in \limsup(A_{m_i})$. $w = w_j$ for some j, and as $w_j \in \limsup(A_m)$ we must have $w_j \in \bigcup_{k=j+1}^{\infty}(A_m)$. Thus there is some $k>j$ s.t. $w_j \in A_{m_k}$. But by construction each A_{m_k} $(k>j)$ is a $A_{n_{j+1,i}}$, and these have constant intersection with $\{w_1, \ldots\ldots, w_j\}$ so if $\forall k>j.\ w_j \in A_{m_k}$.

Thus $w_j \in \bigcap_{k=j+1}^{\infty}(A_{m_k})$

$\Rightarrow w_j \in \liminf(A_{m_k})$

Hence $\liminf(A_{m_i}) = \limsup(A_{m_i})$ and so $\langle A_{m_i} \rangle$ is the required convergent subsequence.

Given this result we can now prove the stronger version of 2.47 for countable alphabets.

2.53 Theorem

If Σ is countable and $F:P \to P$ is monotonic and if R is a strongly continuous satisfiable predicate then

$$(\forall A.(R(A) \Rightarrow R(F(A)))) \Rightarrow R(fix(F)) \qquad (*)$$

if F has a unique fixed point.

proof

We can define $F^{\alpha}(\bot)$ and $F^{\alpha}(\top)$ for arbitrary ordinal α as follows

$$F^{0}(\bot) = \bot \qquad F^{0}(\top) = \top$$
$$F^{\alpha+1}(\bot) = F(F^{\alpha}(\bot)) \quad F^{\alpha+1}(\top) = F(F^{\alpha}(\top))$$
$$F^{\mu}(\bot) = \bigcup_{\sigma<\mu} F^{\sigma}(\bot) \quad F^{\mu}(\top) = \bigcap_{\sigma<\mu} F^{\sigma}(\top) \quad \text{if } \mu \text{ is a limit ordinal.}$$

There must be some countable ordinal κ s.t.

$$F^{\kappa}(\bot) = F^{\kappa}(\top) = fix(F)$$

The fact that κ is countable can be derived in a similar way to 4.13.

We can now prove by transfinite induction on $\rho \leqslant \kappa$ that

$$\forall \rho.\ \exists A_{\rho}.\ R(A_{\rho})\ \&\ F^{\rho}(\bot) \subseteq A_{\rho} \subseteq F^{\rho}(\top)$$

This is true for ρ=o since R is satisfiable.

Suppose true for ρ

then $F^{\rho}(\bot) \subseteq A_{\rho} \subseteq F^{\rho}(\top)\ \&\ R(A_{\rho})$

$\Rightarrow F(F^{\rho}(\bot)) \subseteq F(A_{\rho}) \subseteq F(F^{\rho}(\top))\ \&\ R(F(A_{\rho}))$ by monotonicity and (*)

$\Rightarrow F^{\rho+1}(\bot) \subseteq A_{\rho+1} \subseteq F^{\rho+1}(\top)\ \&\ R(A_{\rho+1})$ where $A_{\rho+1} = F(A_{\rho})$ as desired.

Suppose true of all $\beta<\rho$ and that ρ is a limit ordinal.

Because ρ is countable there must be some ascending sequence $\langle \beta_i | i \in N \rangle$ of ordinals less than ρ s.t. $\bigcup_{i=1}^{\infty} \beta_i = \rho$.

Put $B_i = A_{\beta_i}$ This is an infinite sequence which satisfies $\forall i.R(B_i)$ and by 2.52 has an infinite convergent subsequence $\langle B_{n_i} \rangle$ s.t. $\forall i.n_i \geqslant i$. We then have $\forall i.F^{\beta_{n_i}}(\bot) \subseteq B_{n_i} \subseteq F^{\beta_{n_i}}(\top)$.

$\Rightarrow \forall i.F^{\beta_i}(\bot) \subseteq B_n \subseteq F^{\beta_i}(\top)$ as $\beta_i \leqslant \beta_{n_i}$

$\Rightarrow F^{\rho}(\bot) = \limsup(F^{\beta_i}(\bot)) \subseteq \lim(B_{n_i}) \subseteq \limsup(F^{\beta_i}(\top)) = F^{\rho}(\top)$

$R(\lim(B_{n_i}))$ holds by strong continuity of R, so we can put $A_{\rho} = \lim(B_{n_i})$.

This establishes the desired result for all $\rho \leqslant \kappa$.
Hence in particular it holds for κ, and so there is some A s.t. R(A) and $\text{fix}(F) = F^{\kappa}(\top) \supseteq A \supseteq F^{\kappa}(\bot) = \text{fix}(F)$.
Thus A = fix(F), which completes the proof that R(fix(F)) holds.

This result is <u>not</u> true when Σ is uncountable, as is demonstrated by the following example.

If Σ is uncountable there is some (uncountable) initial ordinal λ equinumerous with it; suppose $\{a_\kappa \mid \kappa \in \lambda\}$ is any enumeration of $\Sigma - \{\checkmark\}$ by λ. Define $F: P \to P$ by $F(A) = \{\langle\rangle, \langle\checkmark\rangle, \langle a_\kappa\rangle w \mid \forall \nu \in \kappa. \langle a_\nu\rangle \in A\}$. It is easy to see that F is monotonic but not continuous, and that F has unique fixed point <u>run</u>. Now consider the predicate R defined $R(A) =$ "A^{o} is countable"; R is plainly satisfiable and it is easy to show that $R(A) \Rightarrow R(F(A))$ for all $A \in P$. That R is strongly continuous follows from the fact that a countable union of countable sets is countable, so that whenever $\langle A_i \rangle$ is any sequence of processes s.t. $R(A_i)$ for each i we have that $(\text{limsup}(A_i))^{o}$ is countable also. However, R(<u>run</u>) plainly does not hold, demonstrating that the rule is not valid in this case.

In the appendix to the next chapter we will see that if we strengthen still further our definition of continuity of predicates 2.51 does hold for general alphabets. In addition we will see there that over uncountable alphabets neither strong continuity nor the even stronger continuity can be represented as continuity relative to any normal class of restriction operators. Such a class of operators does exist for countable alphabets, though:

2.52 <u>Theorem</u>

If Σ is countable then strong continuity is equivalent to continuity relative to the following normal class of restriction operators:

$A \restriction i = \{w_j \mid j \leqslant i \ \& \ w_j \in A\}$

where $\{w_0, w_1, \dots\}$ is any fixed enumeration of Σ^* with the property that $w_i < w_j \Rightarrow i < j$ (such enumerations are easy to construct).

<u>proof</u>

It is very easy to show that the above is indeed a normal class of restriction operators. By 2.50 we know that every predicate which is strongly continuous is continuous relative to any normal class of restriction operators, so it will be sufficient to show that continuity relative to the above class implies strong continuity. To this end let us suppose that R is a predicate continuous relative to our class of operators, and that $\langle A_i \rangle$ is a sequence of processes which converges to A and such that $R(A_j)$ holds for each $j \in N$.

Since $\limsup(A_i) = \liminf(A_i) = A$ it is easy to see that for all $i \in N$ there exists some minimal $k(i)$ with the property that $m \geqslant k(i) \Rightarrow ((w_i \in A_m) \Leftrightarrow (w_i \in A))$. Define $r(i) = \max\{k(j) \mid j \leqslant i\}$: for each $s \geqslant r(i)$ we have $A_s \restriction i = A \restriction i$ by definition of $\restriction i$.

Hence for each i there exists some B s.t. $B \restriction i = A \restriction i$ and $R(B)$; thus $R(A)$ follows by continuity of R relative to this class.

Thus $\{A \mid R(A)\}$ is closed under the limits of its convergent sequences, so R is strongly continuous as claimed.

Chapter 3 :- Continuous Predicates as a Topology

It is possible to define a topology on the space of processes P (or one of the product spaces P^{Λ}) in which the closed sets are identified with the continuous predicates. This can be done for each of the types of continuity we have met so far. In any particular case we will have:

3.1 $X \in \mathfrak{C}$ (the class of closed sets)
$\Leftrightarrow R(A) \equiv$ "$A \in X$" is a continuous predicate.

That these are the closed sets of a well-defined topology follows because (a) both "true" and "false" are always continuous and so both Ø and the whole of P (or P^{Λ}) are closed and (b) the class of continuous predicates is closed under arbitrary conjunctions and finite disjunctions (and so the closed sets are closed under arbitrary intersections and finite unions). (These results for general restriction operators follow in much the same ways as 2.18(xii)&(xiii) and 2.46(viii)&(ix).)

From here on we will consider only topologies of P (which correspond with predicates of a single variable). This is simply for the sake of clarity, and it is possible to extend many of the results obtained to the general space P^{Λ}.

We will first examine the topology arising from a metric defined relative to an arbitrary normal class of restriction operators. We will then show that this is the same topology which arises from definition 3.1 (for the same class of operators). Later we will see the implications of these results for the special cases of weak continuity (2.12) and strong continuity (2.44).

Suppose that $\{\lceil n \mid n \in N\}$ is a normal class of restriction operators (2.49). Define a metric on P as follows:

3.2 $$d(A,A) = 0$$
$$A \neq B \Rightarrow d(A,B) = \frac{1}{n}\ , \quad \text{where } n \text{ is minimal w.r.t. } A\lceil n \neq B\lceil n\ .$$

(That such an n exists when $A \neq B$ is easily shown from 2.49 .)

This is a well-defined metric, for (i) clearly $d(A,B) = d(B,A)$ for all A and B, (ii) $d(A,B) \geq 0$ and $d(A,B) = 0 \Rightarrow A=B$,

(iii) for all A,B&C if $A\restriction n \neq C\restriction n$ then clearly either $A\restriction n \neq B\restriction n$ or $B\restriction n \neq C\restriction n$ so that $d(A,C) \leq \max(d(A,B), d(B,C))$.

This strong type of metric (with "max" instead of the usual triangle inequality $d(A,C) \leq d(A,B) + d(B,C)$) has several interesting properties. Recall the following definitions (which are valid in an arbitrary metric space (P,d)).

3.3(i) An <u>open ball</u> is a set of the form $\{B \mid d(A,B) < a\}$ (denoted by $\mathbf{B}_A^a$) for any $a > 0$ and $A \in P$.

(ii) An <u>open set</u> is any union of open balls.

(iii) A <u>closed set</u> is any set whose complement is open.

(iv) (P,d) is said to have <u>dimension zero</u> if for every $A \in P$ and $\delta > 0$ there is some set $\mathbf{B}$ which contains A, is both open and closed (<u>clopen</u>) and has the property that $C \in \mathbf{B} \Rightarrow d(A,C) < \delta$.

3.4 <u>Theorem</u>

Suppose that (P,d) is a metric space which satisfies the condition $d(A,C) \leq \max(d,(A,B), d(B,C))$ for all $A,B,C \in P$. Then each of the following holds of (P,d).

(i) If $\mathbf{B}_A^a$ and $\mathbf{B}_C^c$ are two open balls such that $a \leq c$, then either they are disjoint or $\mathbf{B}_A^a \subseteq \mathbf{B}_C^c$.

(ii) $\{\bigcup C \mid C$ is a finite set of open balls$\}$ is a basis for the metric topology which is closed under finite intersection.

(iii) The open balls are closed sets.

(iv) (P,d) has dimension zero.

<u>proof</u>

(i) Suppose $\mathbf{B}_A^a \cap \mathbf{B}_C^c \neq \emptyset$, then $\exists D \in \mathbf{B}_A^a \cap \mathbf{B}_C^c$.
We then have that $d(A,D) \leq a$ and $d(C,D) \leq c$, so $d(A,C) \leq \max(a,c)$.
Hence $A \in \mathbf{B}_C^c$.
Now suppose that $D \in \mathbf{B}_A^a$, then
$$d(C,D) \leq \max(d(C,A), d(A,D)) \leq \max(c,a) = c.$$
Thus $D \in \mathbf{B}_C^c$, and so $\mathbf{B}_A^a \subseteq \mathbf{B}_C^c$ as desired.

(ii) This is trivially a basis (as the set of open balls is one). It is closed under finite intersections since by (i) the intersection of two open balls is either an open ball or $\emptyset$ (which is the union of the empty set of balls).

(iii) Part (i) shows that all the distinct a-balls (for any fixed a) are disjoint. This together with the fact that

$C \in \mathbf{B}^a_C$ gives us that for any $A \in P$ we have $\mathbf{B}^a_A \cap \mathbf{U} = \emptyset$, where $\mathbf{U}$ is the open set $\bigcup_{C \notin \mathbf{B}^a_A} \mathbf{B}^a_C$. Clearly also $\mathbf{B}^a_A \cup \mathbf{U} = P$, so we have $\overline{\mathbf{U}} = \mathbf{B}^a_A$ and so $\mathbf{B}^a_A$ is the complement of an open set as desired.

(iv) By part (iii) it is sufficient to take a sufficiently small open ball about a point $A \in P$, since these are now known to be clopen.

Recall the definition of a Cauchy sequence over a metric space:

3.5 If (P,d) is a metric space and $\langle A_i \mid i \in N \rangle$ is a sequence of points in P then $\langle A_i \mid i \in N \rangle$ is said to be a Cauchy sequence if $\forall \delta > 0.\ \exists n \in N.\ \forall r,s \geq n.\ d(A_r, A_s) < \delta$.

Note that under the conditions of 3.4 a sequence is a Cauchy sequence if it satisfies the weaker condition
$\forall \delta > 0.\ \exists n.\ \forall r \geq n.\ d(A_r, A_n) < \delta$.

A metric space is said to be complete if each Cauchy sequence converges to some limit, that is

$$\exists A. \forall \delta > 0.\ \exists n.\ \forall r \geq n. d(A, A_r) < \delta.$$

3.6 Theorem

The metric space defined relative to any normal class of restriction operators is complete.

proof

Suppose that $\langle A_i \mid i \in N \rangle$ is a Cauchy sequence relative to such a class of operators. Define functions r, n, j from Σ^* to N as follows:

$$r(w) = \text{minimal } r \text{ s.t. } \forall B.\ w \in B \Leftrightarrow w \in B\restriction r$$
(such an r exists by 2.49)
$$n(w) = \max\{r(v) \mid v \leq w\}$$
$$j(w) = \text{minimal } j \text{ s.t. } i \geq j \Rightarrow d(A_i, A_j) < \frac{1}{n(w)}$$
(such a j exists by definition of Cauchy seq.)

Now define $\mathbf{A} = \langle w \mid w \in A_{j(w)} \rangle$. Claim that A is the limit of the Cauchy sequence.

A is an element of P for certainly $\langle\rangle \in A_{j(\langle\rangle)}$ and also

$$\begin{aligned}
v < w \in A &\Rightarrow v \in A_{j(w)} \quad (\text{as } A_{j(w)} \in P) \\
&\Rightarrow v \in A_{j(w)}\restriction r(v) \\
&\Rightarrow v \in A_{j(v)}\restriction r(v) \quad \left(\text{as } d(A_{j(v)}, A_{j(w)}) < \frac{1}{r(v)} \Rightarrow A_{j(v)}\restriction r(v) = A_{j(w)}\restriction r(v)\right) \\
&\Rightarrow v \in A_{j(v)}
\end{aligned}$$

Suppose that $\langle A_i \mid i \in N \rangle$ did not converge to the given A. Then there is some $\delta > 0$ s.t. $\forall m. \exists n. n \geq m$ & $d(A_n, A) > \delta$. By the definition of the metric, since $\langle A_i \rangle$ is a Cauchy sequence, for each $n \in N$ there is some m s.t. $k \geq m \Rightarrow A_k \upharpoonright n = A_m \upharpoonright n$. Thus there are some n&m s.t. $k \geq m \Rightarrow A_m \upharpoonright n = A_k \upharpoonright n \neq A \upharpoonright n$ (choose $n > \frac{1}{\delta}$ and m as in the last sentence relative to it).

There would thus either be some $w \in \Sigma^*$ which was an element of $A \upharpoonright n - A_k \upharpoonright n$ for every $k \geq m$ or some $w \in \Sigma^*$ which was an element of $A_k \upharpoonright n - A \upharpoonright n$ for every $k \geq m$.

Suppose first that the first of these two possibilities could arise. Since $\upharpoonright n$ is a continuous function by assumption we have $A \upharpoonright n = \bigcup \{B \upharpoonright n \mid B \subseteq A$ & B is finite$\}$. Hence there is some finite $B \subseteq A$ such that $w \in B \upharpoonright n$.

Suppose that $B = \{v_1, \ldots, v_s\}$; let $r = \max\{m, j(v) \mid v \in B\}$.

Since each $v_i \in A$, we have $v_i \in A_r$ (as $r \geq j(v_i)$). Hence $B \subseteq A_r$, which implies that $w \in A_r \upharpoonright n$, which contradicts the fact that $r \geq m$. Thus the first possibility cannot arise.

The second possibility can similarly be excluded by rev-continuity of $\upharpoonright n$ and the fact that $A \upharpoonright n = \bigcap \{B \upharpoonright n \mid B \supseteq A$ & B ecf$\}$ where an ecf (effectively cofinite) element of P is one which can be written in the form $\{v \mid \neg \exists u \in C. v \geq u\}$ for some finite subset C of Σ^*.

This completes our proof that $\langle A_i \mid i \in N \rangle$ does in fact converge to A as desired.

A metric space is said to be compact if every infinite sequence contains a convergent subsequence.

3.7 Theorem

The metric space defined relative to a normal class of restriction operators is compact if and only if there are only finitely many possible values for $B \upharpoonright n$ for every $n \in N$.

proof

Suppose that there were infinitely many possible values for some n. Then we could pick an infinite sequence $\langle A_i \mid i \in N \rangle$ such that $i \neq j \Rightarrow A_i \upharpoonright n \neq A_j \upharpoonright n$. It is easy to see that this sequence can contain no convergent subsequence as any such subsequence would be a Cauchy sequence in which no two points were within $\frac{1}{n}$ of one another.

Suppose then that there are only finitely many values possible for each restriction operator and that $\langle A_i \mid i \in N \rangle$ is an infinite sequence of processes.

Claim that it is possible to find integers $n_{i,j}$ $(i,j \in N)$ satisfying the following conditions:

(i) $n_{i,j} < n_{i,j+1}$

(ii) $A'_{n_{i,j}} \upharpoonright i = A_{n_{i,o}} \upharpoonright i$

(iii) $\forall j. \forall i. \exists k. k \geqslant j \ \& \ n_{i+1,j} = n_{i,k}$

We will construct these by recursion on i.

Set $n_{o,i} = i$ - these satisfy (ii) as $\upharpoonright o$ is a one-valued function (2.49)

Suppose that we have constructed the $n_{i,j}$. By assumption there are only finitely many values taken by the $A_{n_{i,j}} \upharpoonright i+1$. Therefore there is at least one value (B say) which occurs infinitely often.

Let $n_{i+1,j}$ = jth $n_{i,k}$ s.t. $A_{n_{i,k}} \upharpoonright i+1 = B$.

It is easily seen that these $n_{i+1,j}$ satisfy all that is required of them.

Note that this proof requires the use of the Axiom of Choice in the choosing of value B.

Having constructed the $n_{i,j}$ set $m_i = n_{i,i}$. These satisfy $m_i < m_{i+1}$ (by (i) & (iii) above) and $i,j \geqslant k \Rightarrow A_{m_i} \upharpoonright k = A_{m_j} \upharpoonright k$ (by (ii)).

Thus $i,j \geqslant k \Rightarrow d(A_{m_i}, A_{m_j}) < \frac{1}{k}$ so that the $\langle A_{m_i} \mid i \in N \rangle$ are a Cauchy subsequence of $\langle A_i \mid i \in N \rangle$. By 3.6 this subsequence has a limit and so $\langle A_i \mid i \in N \rangle$ has a convergent subsequence as desired.

Note that this result gives us that (P,d) can only be compact when Σ is countable, for every element of P can in some sense be identified with the sequence of its restrictions $(A \upharpoonright n)$ and the cardinal of these sequences is at most **c** (continuum) if each $\upharpoonright n$ has a finite range. (This also follows more conventionally from the fact that every compact metric space is separable.)

The following result shows that all restriction operators giving rise to compact metric spaces give rise to the same topology.

3.8 Theorem

Over any space of processes all normal classes of restriction operators inducing compact metric spaces induce the same topology. In the case where Σ is countable the restriction operators inducing strong continuity induce a compact metric space (and hence the only possible such topology).

proof

We know that there can be no such metric when Σ is uncountable (by the remarks above) so it is sufficient to consider only the case of countable Σ. That the class of operators described in 2.54 give rise to a compact metric space is an immediate consequence of 3.7, since plainly there is only a finite number of possible values which can be taken by each one.

Suppose that $\{\restriction n \mid n \in N\}$ and $\{\restriction' n \mid n \in N\}$ are two classes of restriction operators giving rise to compact metric spaces, and that their associated metrics are d and e respectively. To show that the topologies are the same it is sufficient, by symmetry, to show that every d-closed set is also e-closed. We will use the fact that a set is closed in a metric space if and only if it contains all its accumulation points (i.e. the points which are the limits of convergent sequences contained entirely within the set).

Suppose that X is a d-closed set. To show that it is e-closed we must show that every e-convergent sequence has its limit in X. Suppose that $\langle A_i \mid i \in N\rangle$ is an e-convergent sequence with limit A. By compactness of the metric space (P,d) there is a d-convergent subsequence, say $\langle A_i' \mid i \in N\rangle$, with a limit A', say. Being a subsequence of $\langle A_i \mid i \in N\rangle$ this must be e-convergent with limit A. It is sufficient to show that A = A', for A' is in X since it is the limit of a d-convergent sequence in X. Suppose $w \in \Sigma^*$; since both classes of operators are normal there exist integers r & s s.t. for all $C \in P$ $w \in C \Leftrightarrow w \in C\restriction r$ and $w \in C \Leftrightarrow w \in C\restriction' s$. Let m = max(r,s); it is easily shown that $w \in C \Leftrightarrow w \in C\restriction m \Leftrightarrow w \in C\restriction' m$. By construction there exists some $n \in N$ s.t. $d(A_n',A') < \frac{1}{m}$ and $e(A_n',A) < \frac{1}{m}$. This tells us that $A_n'\restriction m = A'\restriction m$ and $A_n'\restriction' m = A\restriction' m$.

We thus have $w \in A' \Leftrightarrow w \in A' \lceil m$

$$\Leftrightarrow w \in A'_n \lceil m$$

$$\Leftrightarrow w \in A'_n$$

$$\Leftrightarrow w \in A'_n \lceil' m$$

$$\Leftrightarrow w \in A \lceil' m$$

$$\Leftrightarrow w \in A$$

Since this holds for all w we thus must have A = A' as desired.

Observe that the above proof shows that each d-closed set is e-closed simply by assuming that (P,d) is compact. This shows directly that the topology induced by a compact metric space of the type we are studying is weaker than that induced by any other (i.e. has less closed sets). We will shortly see that this is very much the same result as 2.50.

The opposite role to the compact topology is played in a much less interesting way by the classical discrete topology in which all subsets are open and closed. An example of a normal class of operators giving rise to this topology is the following.

$$A \lceil^* o = \underline{\text{abort}} \qquad (A \in P)$$

$$A \lceil^* n = A \qquad (n > o)$$

Note that the metric which this class induces is the usual discrete metric.

We are now in a position to establish the fundamental connection between the metric spaces described above and the corresponding topologies of continuous predicates which were described in 3.1. We show that the metric topology and the continuous predicate topology are the same. By doing this we are able to use results obtained about the topological properties of spaces of processes to classify the continuous predicates. We are also enabled to prove certain results on the satisfiability of predicates, a topic which is often in practical situations one of the most difficult to deal with.

3.9 Theorem

Suppose that $\{\restriction n \mid n \in N\}$ is a normal class of restriction operators. Let d be the metric defined on P relative to this class (3.2). Then a predicate R is continuous relative to this class if and only if $\{B \mid R(B)\}$ is a closed set in the metric space (P,d). In other words the topology induced by the metric d is the same as that introduced in 3.1.

proof

We use the result that a set is closed if and only if it contains all its accumulation points (i.e. points which are the limits of convergent sequences contained wholly within the set).

Suppose first that R is a continuous predicate and that $\langle A_i \mid i \in N\rangle$ is a convergent sequence contained in the set $\{B \mid R(B)\}$ (with limit A, say). Then for each $n \in N$ there must be some i s.t. $d(A,A_i) < \frac{1}{n}$, so that $A\restriction n = A_i\restriction n$ and $R(A_i)$. R(A) then follows by definition of continuity (2.13).

Thus $\{B \mid R(B)\}$ contains all its accumulation points, and so is a closed set.

Secondly suppose that $\{B \mid R(B)\}$ is a closed set and that $a \in P$ is such that $\forall n.\exists B_n.(A\restriction n = B_n\restriction n)$ & $R(B_n)$.
These B_n clearly form a Cauchy sequence converging to A (as $m \leq n \Rightarrow B_n\restriction m = A\restriction m$). Hence R(A) holds as $\{B \mid R(B)\}$ is closed and so contains its accumulation points.

Thus R is continuous by 2.13.

We can thus now start to apply results of topology to our predicates. The first two results we obtain concern the satisfiability of predicates.

3.10 Theorem

If $\langle R_n \mid n \in N\rangle$ is a sequence of continuous predicates (relative to some normal class of operators) such that $R_{n+1} \Rightarrow R_n$ and if $\langle A_n \mid n \in N\rangle$ is a sequence of processes such that $A_{n+1}\restriction n = A_n\restriction n$ and $R_n(A_n)$ holds for each n, then there is some $A \in P$ s.t.

(i) $A\restriction n = A_n\restriction n$ for all n

(ii) $R_n(A)$ holds for each n.

proof

Denote by $\mathfrak{R}^a$ the subset $\{\bigcup\{B^a_A \mid A \in C\} \mid C \subseteq P\}$ of $\mathfrak{B}$.
Note that $a \geqslant b \Rightarrow \mathfrak{R}^a \subseteq \mathfrak{R}^b$, for if $E \in \mathfrak{R}^b$ then

$E = \bigcup\{B^a_A \mid A \in E\}$.

(The containment of the L.H.S. within the R.H.S. is obvious, the reverse one following from 3.4(i).)

To show that $\mathfrak{B}$ is closed under finite intersections and unions it will thus be sufficient to show that each $\mathfrak{R}^a$ is. (This is because in any finite intersection of elements of $\mathfrak{B}$ there will be some minimal "a".)

The union case is easy: clearly $\bigcup\{B^a_A \mid A \in C\} \cup \bigcup\{B^a_A \mid A \in D\}$ $= \bigcup\{B^a_A \mid A \in (C \cup D)\}$.

In the intersection case we have

$\bigcup\{B^a_A \mid A \in C\} \cap \bigcup\{B^a_A \mid A \in D\} = \bigcup\{B^a_A \cap B^a_B \mid A \in C \ \& \ C \in D\}$

which is an expression of the correct form since each of the $B^a_A \cap B^a_B$ is either empty or B^a_A (by 3.4 (i)).

$\mathfrak{B}$ certainly now contains the basis of 3.4 (ii) for it contains each ball B^a_A individually and is closed under finite unions. Each element of $\mathfrak{B}$ is open by construction (union of open sets). These two facts together tell us that $\mathfrak{B}$ is a basis. Each element is closed since it is the complement of the union of those balls os the same radius which are disjoint from it (an open set). Thus every element is clopen.

If the space is compact then there are only finitely many balls of any given radius (as in 3.7) and so each element of $\mathfrak{B}$ is a finite union, and so is contained in the original basis. Hence in this case the two bases are the same.

If the space is not compact then by 3.7 there is some n such that $\ulcorner n$ takes infinitely many values. Without loss of generality we can assume that n is minimal with respect to this property. There can only be finitely many values taken by all $\ulcorner m$ such that $m < n$, in particular n-1 ($n > 0$ as $\ulcorner o$ is one-valued). Pick elements $A_1, A_2, \ldots$ of P with distinct images under $\ulcorner n$. Since there are only finitely many values of $\ulcorner n-1$ we can pick an infinite number of the A_is which take the same value under $\ulcorner n-1$ ($A'_1, A'_2, \ldots$ say).

Let $\mathbf{C} = \{A'_{2i} \mid i \in N\}$; by construction $\bigcup\{B_A^{\hat{}} \mid A \in \mathbf{C}\}$ $(= \mathbf{X})$ is an element of $\mathfrak{B}$. We need to show that $\mathbf{X}$ cannot be expressed as a finite union of balls. Suppose that $\mathbf{X} = B_1 \cup B_2 \cup .. \cup B_k$ where each B_i is a ball. The radius of each of the B_i must be $\leqslant \frac{1}{n}$, since (i) all the A'_i (and hence every point of X) takes the same value under $\lceil n-1$ (ii) thus the "centre" of such a ball would also have to take this value and so (iii) each A'_{2i+1} is also in the ball (which contradicts the fact that $A_1 \notin \mathbf{X}$). However this implies that each A'_{2i} is contained in a different B_j, since it is easily seen that no ball of radius $\leqslant \frac{1}{n}$ can contain two points whose distance is $\frac{1}{n}$ (in a metric satisfying the conditions of 3.4). This clearly makes impossible our supposition that there are only a finite number of balls in the union.

The next result completely classifies the space of clopen sets in the compact case, showing that in this case the clopen sets correspond exactly to $\mathfrak{B}$ (and hence to the original basis).

3.13 Theorem

In cases where (P,d) is compact a subset $\mathbf{X}$ of P is clopen if and only if it is in $\mathfrak{B}$ (and so can be expressed as a finite union of balls).

proof

Suppose that (P,d) is compact and that $\mathbf{X}$ is a clopen set. Since X is an open set it can be expressed as a countable union of balls (countable since when (P,d) is compact there are only countably many balls). We can assume that this expression is infinite, for otherwise the result is immediate. Since there are only a finite number of balls of any given radius we can assume that the balls are expressed in order of descending radius: $B_1, B_2, \ldots$.
If $\overline{X}$ were expressible as a finite union of balls then so would be $\mathbf{X}$ (this being because then $\mathbf{X} = \bigcup\{B_A^a \mid A \in \mathbf{X}\}$, where a is any number smaller then the smallest radius occurring amongst the expression for $\overline{X}$, but this union is only finite because for any a there are only finitely many a-balls). Since $\overline{X}$ is open it can also be expressed as a countable infinite union of balls of descending radius: $B'_1, B'_2, \ldots$.

We can assume that all of the balls B_i and B_i' are disjoint. Let $Z_i = (B_1 \cup B_1' \cup B_2 \cup \ldots \cup B_i \cup B_i')$. By construction Z_i is open and so $\overline{Z}_i$ is closed. Also $\overline{Z}_i$ is non-empty $(B_{i+1} \subseteq \overline{Z}_i)$ and $\overline{Z}_{i+1} \subseteq \overline{Z}_i$ for every i. The $\overline{Z}_i$ are thus a non-empty descending sequence of closed sets in a compact metric space. Hence there is some point in their intersection, A*, say. This however contradicts the fact that $\bigcup_{i=o}^{\infty} Z_i = P$ $(= X \cup \overline{X})$, so we must conclude that this case, where X is not expressible as a finite union of balls, cannot arise. This completes the proof of our result.

3.14 Corollary

When Σ is countable a predicate R and its negation ¬R are both strongly continuous if and only if R can be expressed finitely in terms of propositions of the form "$w \in A$" $(w \in \Sigma^*)$, "¬" and "∨".

proof

That each predicate R expressed in these terms has both R and ¬R strongly continuous is easily proved by induction, because each of the sets $\{A \mid w \in A\}$ is clopen $(w \in \Sigma^*)$ and the space of clopen sets is closed under complementation and finite unions.

That every R with both R and ¬R strongly continuous can be written in this form follows because the set $\{A \mid R(A)\}$ is clopen, and so can be written as a finite union of balls in the metric space induced by the restriction operators given in 2.54. (Every ball can be written finitely in terms of "¬" and "∧" and "$w \in A$", which is translatable into the desired form.)

Note that it is a corollary to 3.8 that all normal classes of restriction operators which give compact metric spaces have the same class of continuous predicates, and so the above theorem is equally valid for each of them.

It is possible to give a general classification of all clopen sets and hence of all "doubly continuous" predicates for general classes of restriction operators. Before we do this we need a normal form result for closed and for open sets.

3.15 Lemma

If d is the metric defined relative to some normal class of restriction operators then a set **X** is open in (P,d) if and only if it can be written in the form $\bigcup_{i=1}^{\infty} C_i$, where $C_n \in \mathfrak{R}^{\frac{1}{n}}$, (as defined in the proof of 3.12), $C_i \cap C_j = \emptyset$ if $i \neq j$ and for any $a > \frac{1}{n}$ & $A \in P$ we have $B_A^a \notin \bigcup_{i=n}^{\infty} C_i$. This expression for **X** in terms of the C_i is unique for each open set **X**.

Under the same conditions a set **X** is closed in (P,d) if and only if it can be written in the form $\bigcap_{i=1}^{\infty} C_i$, where $C_n \in \mathfrak{R}^{\frac{1}{n}}$, $C_{n+1} \subseteq C_n$ and for all $a \geqslant \frac{1}{n}$ & $A \in P$ we have $B_A^a \cap X = \emptyset \Rightarrow C_n \cap X = \emptyset$. This expression is unique for each closed set **X**.

This result is a fairly easy consequence of 3.12.

It follows that the following four conditions are equivalent for any set **X**.

(a) X is clopen.

(b) X can be written in each of the forms given above.

(c) X and $\overline{X}$ can be written in the first (open) form above.

(d) X and $\overline{X}$ can be written in the second form above.

In particular a set **X** is clopen if and only if there are some C_i and C_i' such that $X = \bigcap_{i=1}^{\infty} C_i$ and $\overline{X} = \bigcap_{i=1}^{\infty} C_i'$, both sequences satisfying the conditions of the second half of 3.15. Consider the sets $D_i = C_i \cap C_i'$. By construction these are closed sets satisfying the conditions $D_{n+1} \subseteq D_n$ and $\bigcap_{i=1}^{\infty} D_i = \emptyset$. This can arise in two essentially different ways: either $D_i = \emptyset$ for some $i \in N$ or not. In the first case we have that $C_j = C_i$ for all $i < j$ (since $C_j \subseteq C_i$ & $C_j \cup \overline{C}_j = P$) which tells us that $X = C_i$. Thus $X \in \mathfrak{B}$. It is also true that if $X \in \mathfrak{B}$ then $D_i = \emptyset$ for some i (any i s.t. $\frac{1}{i} < a$, where a is such that $X = \bigcup\{B_A^a \mid A \in C\}$). Thus the first case arises exactly when **X** lies in the class of clopen sets which we have already identified, namely $\mathfrak{B}$. The other case (which cannot arise when the metric space is compact) is harder to describe exactly.

The following result gives an exact but not very elegant list of all the clopen sets (including the first case). For an example of what one of the second case sets looks like see 3.17.

This characterisation of clopen sets is unfortunately rather too abstract to yield a very useful tool in identifying the "doubly continuous" predicates in a given system. It should be said though that apart from the "finite" ones which are easily identified (i.e. ones which correspond to some element of $\mathfrak{B}$) these predicates are rarely of much practical use. The following is an example of a doubly continuous predicate which is not finite (in the sense defined above), in the system defined by weak continuity (2.6 et seq.) with alphabet containing N, the natural numbers. The derivation of the corresponding set from 3.16 requires one application of rule (ii).

3.17 <u>Example</u>

Let A_n (for $n \in N$) be the process $n \to (n \to ..(n \to$ <u>skip</u>$))..)$ (n "n"s). Then the predicate $R(A) \equiv \exists n.A=A_n$ is doubly continuous but not finite with respect to weak continuity.

The basic principle at work here is that one can tell in a finite time (after one step) exactly how long one has to wait to know whether or not the predicate holds. It is generally true that all doubly continuous predicates result from the compounding of this principle.

Note that <u>neither</u> the above R nor its negation $\neg R$ is <u>strongly</u> continuous. This is because we can find convergent sequences of processes satisfying the predicate ($\langle A_n | n \in N \rangle$ as a sequence has limit <u>abort</u>) and not satisfying it ($\langle A_0 \, \Box \, A_{n+1} | n \in N \rangle$ which has limit A_0) whose limits act oppositely. However this does not extend to a general principle, for if we were to substitute <u>abort</u> into the definition of A_n in place of <u>skip</u> the resulting R would still be doubly <u>weakly</u> continuous but now R would be <u>strongly</u> continuous (though of course not $\neg R$).

There are several results about continuous predicates which follow from the zero-dimensionality of our metric spaces.

3.18 <u>Theorem</u> (Sharpened normality property)

If R and S are two inconsistent predicates, continuous relative to some normal class of restriction operators, then there is a doubly continuous predicate T such that $R \Rightarrow T$ and $S \Rightarrow \neg T$.

3.16 Theorem

In the metric space (P,d) defined relative to some normal class of restriction operators the space $\mathfrak{H}$ of clopen sets satisfies the following:

(i) $\mathfrak{B} \subseteq \mathfrak{H}$.

(ii) If for any $n \in N$ we have a family $\mathfrak{N}$ of clopen sets satisfying $X, Y \in \mathfrak{N}$, $X \neq Y$, $A \in X$ & $B \in Y \Rightarrow A\restriction n \neq B\restriction n$ then $\bigcup\mathfrak{N} \in \mathfrak{H}$.

$\mathfrak{H}$ is the smallest collection of sets satisfying the above.

proof

We will first show that $\mathfrak{H}$ satisfies condition (ii) above (we already know that it satisfies condition (i)). Let $\mathfrak{N}$ be a family of clopen sets satisfying the hypotheses in condition (ii) above for some $n \in N$, and let $X = \bigcup\mathfrak{N}$. That X is open follows from the fact that it is a union of open sets. X is closed since the tail of any convergent sequence contained in X must be contained in one of the elements of $\mathfrak{N}$, and so its limit is contained in that element.

Secondly we will show that every clopen set can be derived from (i) & (ii) above. Define $\mathfrak{G}$ to be the family of clopen sets which can be so derived, and suppose that X is clopen. Observe first that for any n and any such X at least one element of $\{X \cap \{A \mid A\restriction n = B\restriction n\} \mid B \in P\}$ is clopen and not in $\mathfrak{G}$. (Every element of this set is clopen by construction, and if each were in $\mathfrak{G}$ then so would be X as it is the union of a family of elements of $\mathfrak{G}$ satisfying (ii) for n.) Set $X_o = X$, and for any n choose X_{n+1} to be one of the $X_n \cap \{A \mid A\restriction n{+}1 = B\restriction n{+}1\}$ which is of the offending form. Each of the X_i is nonempty (since $\emptyset \in \mathfrak{G}$) so we can pick a sequence $\langle A_i \mid i \in N\rangle$ s.t. $A_i \in X_i$. By construction this sequence is Cauchy, and hence convergent to some $A^* \in P$. Certainly $A^* \in X$, since each $A_i \in X$ and X is closed, but also for each n we must have $X_n \neq \{A' \mid A'\restriction n = A_n\restriction n\}$ (for this set is in $\mathfrak{G}$). We can therefore pick a second sequence $\langle A_n' \mid n \in N\rangle$ such that $A_n'\restriction n = A_n\restriction n$ and such that $A_n' \notin X_n$. This means that $\langle A_n' \mid n \in N\rangle$ also converges to A^*, and also that $A_n' \in \overline{X}$ (it is easy to see that $X_n = X \cap \{A' \mid A'\restriction n = A_n\restriction n\}$). Hence $A^* \in \overline{X}$, as $\overline{X}$ is closed, which is a contradiction. We may thus conclude that our assumption that such an X exists is false.

proof

It is sufficient (by the correspondence theorem 3.9) to show that in the corresponding metric space every two disjoint closed sets can be separated by a clopen set. We will show that this is true in any metric space which satisfies the conditions of 3.4.

Suppose that (P,d) is such a space, and that X and Y are a pair of disjoint closed subsets of P. It is clear that given any point of X there is some $n \in N$ such that the (clopen) ball of radius 2^{-n} about it is disjoint from Y, for otherwise we could find a convergent sequence of points in Y converging to a point in X.

For each $A \in X$ define $B(A)$ to be the $B_A^{2^{-n}}$ of least n such that it is disjoint from Y. It is not hard to see that by 6.4(i) $B(A) \cap B(A') \neq \emptyset \Rightarrow B(A) = B(A')$ for all $A, A' \in X$ (if they are not disjoint then one is contained in the other, but if either had strictly smaller radius than the other it would not have maximal possible radius with respect to being disjoint from Y).

Now define $Z = \bigcup_{A \in X} B(A)$. Claim that Z is clopen (it is trivially disjoint from Y and contains the whole of X as $A \in B(A)$ for all $A \in X$).

Z is open by construction, since it is the union of a set of open balls.

If Z were not closed then there would be a sequence of points $A_1, A_2, \ldots$ which converged to a point A^* not in Z. There are essentially two cases to consider: either infinitely many of the A_i lie in one of the $B(A)$ or not (in which case there must be an infinite collection of the balls $B(A)$ $(A \in X)$ containing them).

In the first case it is easy to see that (if $A \in X$ is such that infinitely many A_i are in $B(A)$) there is an infinite subsequence which lies in $B(A)$, which tells us that $A^* \in B(A)$ as $B(A)$ is closed by 6.4(iii). Thus this case cannot arise.

In the second case either arbitrarily small balls appear amongst the $B(A_i')$, where $A_i' \in X$ is such that $A_i \in B(A_i')$, or not. If this is not the case and b is the smallest radius occurring among the $B(A_i')$ then it is clear that the entire sequence $A_1, A_2, \ldots$ is contained in the clopen (by 3.12) subset $\bigcup_{i=1}^{\infty} B^a_{A_i}$ of Z. But this would imply that $A^* \in Z$, so this possibility cannot arise. If arbitrarily small balls do appear among the $B(A_i')$ then without loss of generality we can assume that the radii of the $B(A_i')$ are strictly decreasing. (This is because some infinite subsequence of the A_i must then have this property.) For all n we then have that $d(A_n, A_n') \leq 2^{-n}$. This is easily seen to imply that $\langle A_i' | i \in N \rangle$ converges to A^*. This however is impossible since then, because X is closed we must have $A^* \in X \subseteq Z$.

It is interesting to see how the topological notions of convergence and continuous functions relate to the ideas we met in the previous two chapters. It is in fact easy to see that a sequence of processes converges in the sense of 2.42 if and only if it converges relative to the corresponding metric (both ideas are equivalent to there being for every $w \in \Sigma^*$ a point in the sequence after which w is either always present or always absent).

It is possible for a function to be continuous in the topological sense (i.e. inverse images of open sets being open) without being continuous in the lattice sense, as a function can be topologically continuous without being monotonic. We can for example divide P into two clopen sets, pick any two points we wish in P and have the function which maps one set to one point and the other to the other topologically continuous. There are though some weaker comparisons possible.

3.19 <u>Theorem</u>

Suppose that (P,d) is a metric space induced by a class of restriction operators giving a compact space. Then a function $f: P \rightarrow P$ which is doubly continuous in the lattice sense is topologically continuous. Furthermore every function which is topologically continuous and monotone is also (lattice) doubly continuous.

<u>proof</u>

The first part of this follows from the facts that a function is continuous in (P,d) if and only if it preserves the limits of convergent sequences, that the two types of convergence are the same (see above), and that a doubly continuous function preserves the limits of convergent sequences (2.48).

The second part follows since as (P,d) is compact Σ must be countable. This means that for any $A \in P$ we can find sequences $\langle A_i \mid i \in N\rangle$ and $\langle A_i' \mid i \in N\rangle$ which consist respectively of increasing finite processes and decreasing ecf processes, each of which converges to A. By topological convergence we get that each of $\langle f(A_i) \mid i \in N\rangle$ and $\langle f(A_i') \mid i \in N\rangle$ is convergent with limit f(A). This combined with monotonicity easily yields the desired result that
$f(A) = \bigcup\{f(B) \mid B \subseteq A \text{ \& } B \text{ is finite}\} = \bigcap\{f(B) \mid B \supseteq A \text{ \& } B \text{ is ecf}\}$.

Note that 2.46(vii) extends the first part of this result to the case of strong continuity over uncountable alphabets, since it tells us that the inverse image of a closed set under a doubly continuous function is closed (in the topology defined as in 3.1 by reference to strong continuity).

The above result can be regarded as an explanation of the connection which we observed between strong continuity of predicates and doubly continuous functions.

Other classes of restriction operators relate less well to lattice continuity. For example in the case of weak continuity (with its usual metric) there are lattice continuous functions which are not topologically continuous, and monotonic topologically continuous functions which are not lattice continuous.

3.20 Examples

(i) The function $f:P \to P$ defined by $f(A) =$ abort if A^o is finite, $f(A) =$ skip if A^o is infinite, is both monotone and topologically continuous but is not lattice continuous whenever Σ is infinite.

(ii) Suppose that $N \subseteq \Sigma$ and that $b \in \Sigma$. The function $f:P \to P$ defined $f(A) = \{\langle\rangle\} \cup \{\langle n\rangle \mid \langle \overbrace{bb\ldots b}^{n\ \text{"b"s}}\rangle \in A\}$ is lattice doubly continuous but not topologically continuous.

The second example works because it maps the convergent sequence $\langle B_i \mid i \in N\rangle$ (where $B_i = \{\langle\rangle, \langle b\rangle, \langle bb\rangle, \ldots, \langle \overbrace{bb\ldots b}^{i\ \text{"b"s}}\rangle\}$) to the non-convergent sequence $\langle \{\langle n\rangle \mid n \leq m\} \cup \{\langle\rangle\} \mid m \in N\rangle$.

It is however possible to recast the usual $\epsilon-\delta$ metric space continuity criterion in a more familiar form:
$\forall B \in P.\exists g:N \to N \text{ s.t. } \forall A.(A\restriction g(n) = B\restriction g(n)) \Rightarrow f(A)\restriction n = f(B)\restriction n$.

The above formula, which holds for a function f if and only if f is continuous in the topological sense, is rather like 2.18(x) (though slightly stronger). That this should be so is of course quite natural, since 2.18(x) plays the same role for weak continuity as 2.46(vii) does for strong continuity.

The next question to ask is how constructive functions behave in our metric spaces. It will be seen immediately that a constructive function has the effect of reducing the distance between two points, and that a non-destructive function is one which guarantees not to reduce the distance between two points. One gets a slightly preferable picture at this point by altering the metric slightly: let d' be the metric defined by setting $d'(A,A) = 0$ and $d'(A,B) = \frac{1}{2^n}$, where n is minimal with respect to $A \upharpoonright n \neq B \upharpoonright n$ (if $A \neq B$). This alteration does not affect our earlier work except in minor computational details, all results still stand in the metric space (P,d'). The advantage gained is that a constructive map now becomes a contraction mapping: for all $A,B \in P$, if f is constructive then $d'(f(A),f(B)) \leqslant \frac{1}{2}d'(A,B)$. It is possible to derive all our basic results about existence of fixed points and recursion induction from this fact alone. This topic and the wider uses of contraction mappings deserve more attention, but we will leave it here.

Following Scott ( ) it is possible to define a topology $\mathfrak{T}$ whose continuous functions correspond exactly to the lattice continuous functions.

3.21 <u>Theorem</u>

Let $\mathfrak{T}$ be the topology generated by the subbasis $\mathfrak{S} = \{\{A \mid A \supseteq B\} \mid B \in P \ \&\ B \text{ finite}\}$. Then a function $f: P \to P$ is continuous in this topology if and only if it is continuous in the usual lattice sense.

The proof of this result is omitted, being very similar to the usual proof over P(N). This topology is very much weaker than any of the ones associated with restriction operators. It is T_0 (i.e. given any pair of distinct points there is an open set containing one and not the other) but not T_1.

Before we summarise the implications of this chapter we will see one more result: an alternative characterization of the topology induced by our compact metric space.

3.22 Theorem

If $A,B \in P$ define the interval $[A,B]$ to be the set $\{C \mid A \subseteq C \subseteq B\}$. The set $\mathfrak{J}$ of intervals is closed under intersection. Let $\mathfrak{U}$ be the weakest topology on P in which all intervals are closed. $\mathfrak{U}$ is the same as the topology induced by strong continuity when Σ is countable (and hence is also the same as the metric topology induced by the operators described in 2.54).

proof

The set $\mathfrak{U}^c$ of closed sets in the topology is the following:

(i) $\mathfrak{J} \subseteq \mathfrak{U}^c$

(ii) $\mathfrak{U}^c$ is closed under taking finite unions.

(iii) $\mathfrak{U}^c$ is closed under taking arbitrary intersections.

(iv) $\mathfrak{U}^c$ is the smallest set satisfying (i)-(iii).

That these are the closed sets of some topology is not hard to prove, and having done this the resulting topology must by construction be the weakest one in which all intervals are closed.

Let $\mathfrak{C}$ be the set of closed sets of the topology induced by strong continuity (3.1). To prove our result it is sufficient to show that $\mathfrak{U}^c = \mathfrak{C}$. That $\mathfrak{U}^c \subseteq \mathfrak{C}$ follows inductively from the fact that the predicate induced by each interval is strongly continuous (2.46(i,ii,viii)) and the fact that $\mathfrak{C}$ is closed under finite unions and arbitrary intersections.

To prove that $\mathfrak{C} \subseteq \mathfrak{U}^c$ it is sufficient to prove that each element of the known basis for $\mathfrak{C}$ is in $\mathfrak{U}^c$ (as the complement of each basis element is also in , so we are showing that each is also open in $\mathfrak{U}$). Since there are only finitely many balls of a given radius and $\mathfrak{U}^c$ is closed under finite intersections it is sufficient to show that all balls B_A^a are in $\mathfrak{U}^c$. It is easily seen that for every ball there is an expression which has the form $Z_0 \cap Z_1 \cap \ldots \cap Z_n$, where Z_i is either $\{A \mid w_i \in A\}$ or $\{A \mid w_i \notin A\}$ ($\{w_0, w_1, \ldots\}$ the enumeration of Σ^* in 2.54) and n is minimal with respect to $\frac{1}{n} < a$ (radius of the ball). It is sufficient therefore to show that each possible Z_i is an interval. To do this we simply observe that

$$\{A \mid w \in A\} = [\{v \mid v \leq w\}, \underline{run}] \quad (\underline{run} \text{ is maximal in } P)$$
$$\{A \mid w \notin A\} = [\underline{abort}, \{v \mid w \not\leq v\}] \quad (\text{if } w \neq \langle\rangle)$$
$$= [\underline{run}, \underline{abort}] \quad (\text{if } w = \langle\rangle).$$

This completes the proof of 3.22.

This result is pleasing, since it shows that the topology induced by all compact metrics and strong continuity is quite a natural one. It can be used to prove results of closed sets (and hence of strongly continuous predicates) inductively. If a property holds of all the basic intervals ([abort,A], [A,run]) and is preserved by finite union and arbitrary intersection then it holds of all closed sets in $\mathfrak{C}$. In the language of predicates this translates to the following inductive principle: if a property holds of each predicate of the form $R(A) = A \supseteq B$ or $R(A) = A \subseteq B$ $(B \in P)$ and is preserved by finite disjunctions and arbitrary conjunctions then it holds of all strongly continuous predicates. (Both of these principles hold only when Σ is countable.) One application of this is an alternative proof of 2.46(vii), which becomes an immediate consequence of 2.46(v,vi,viii,ix). Several of our other results have alternative proofs using the same principle. If desired this principle can be strengthened: it is in fact only necessary to show that a property holds of all of the basic predicates above with B finite and ecf in the respective cases. This is because every other of the basic predicates can be expressed as a conjunction of ones of this form.

To conclude this chapter we will take stock of our results, paying particular attention to how they affect our understanding of the two main classes of continuous predicates we met in chapter 2.

We have seen that given any normal class of restriction operators on P we can define a corresponding metric space in a natural way, and that the closed sets of this metric space correspond in a natural way to the predicates which are continuous relative to the class of restriction operators. This corresponds to the fact that (as we already knew) continuous predicates are closed under finite disjunctions and arbitrary conjunctions. We found that our metric spaces were of a very discrete kind, and were complete. Completeness was used to prove a satisfiability result. We found that the metric space was compact only when the predicates continuous with respect to a class of restriction operators were exactly the strongly continuous ones, and Σ was countable. We re-established the fact that a strongly continuous predicate is continuous with respect to all other normal operator classes, and also discovered that strong continuity is in several ways better behaved than other sorts (e.g. 3.11, 3.13, 3.14, 3.19 & 3.21). In investigating how continuous functions in the metric space behave we discovered another way of treating constructive functions.

Below is a summary of the results affecting our knowledge of weakly and strongly continuous predicates.

a) <u>Weakly continuous</u>

(i) 3.10 (which is obvious in this case anyway)

(ii) 3.15 (which can be adapted to give a normal form for weakly continuous predicates)

(iii) 3.16 (which classifies the predicates which are "doubly continuous")

(iv) 3.18 (the application of which is helped by our classification of doubly continuous predicates)

b) <u>Strongly continuous</u>

Each of the above also holds in this case, except that 3.14 plays the role of 3.15. In addition we have:

(i) 3.11

(ii) The explanation of the connection with doubly continuous functions.

(iii) The inductive principles which come from 3.22.

It is possible to prove most of the above without consideration of topology or metric spaces, and indeed it is possible to do much of the topology (e.g. 3.13 & 3.22) without using metric considerations. The use of metric spaces does however often seem to be the most convenient and elegant way to deal with continuous predicates.

Appendix: A summary of some later results.

The results we have so far met in this chapter do not tell us a great deal about the case of strong continuity when Σ is uncountable. Most of the following results are extensions of existing results to this case (or demonstrations that existing results do not then apply). The first theorem is however an additional classification of strong continuity over countable alphabets.

3.23 Theorem

The topology induced by strong continuity is homeomorphic to the usual metric topology of the Cantor set if and only if Σ is countably infinite.

This follows quite easily from theorem 30.3 of Willard( ), since if Σ is infinite every point is the limit of a sequence of points distinct from itself, whereas if Σ is finite the process abort is not.

3.24 Theorem

The topology induced by strong continuity is, when Σ is uncountable, neither first countable, metrizable nor compact.

That it is not first countable follows from the fact that there exists an uncountable set $\{\{\langle\rangle, \langle a\rangle\} | a \in \Sigma\}$ of nearly disjoint processes. If the topology were first countable there would exist a sequence of closed sets C_j such that abort $\notin C_i$, $C_i \subseteq C_{i+1}$, and whenever X is a closed set not containing abort there exists some j s.t. $X \subseteq C_j$. One can then show that one of the C_j must contain infinitely many of the one point closed sets $\{\{\{\langle\rangle, \langle a\rangle\}\} | a \in \Sigma\}$ as subsets; but this implies that this C_j contains a convergent sequence of points converging to abort, contradicting the fact that abort $\notin C_j$.

That it is not metrizable (and hence not the topology induced by any normal class of restriction operators) follows from the fact that every metric space is first countable.

The fact that it is not compact follows from the fact that we can find (under the continuum hypothesis) an infinite set of points with no limit point. This is because under the continuum hypothesis it is clearly sufficient to show that this can be done for any particular alphabet of cardinal $2^{\aleph_0}$. Now let $\Sigma = \{f \mid f:N \to N\}$, and define $A_i = \{\langle\rangle, \langle f\rangle \mid \exists k.\ f(2k)=i\}$. If the set $\{A_i \mid i \in N\}$ were to contain a limit point it is easy to show that there would have to exist some 1-1 function f such that the sequence $\langle A_{f(i)} \mid i \in N\rangle$ was convergent; but for any such sequence it is easy to see that $\langle f\rangle \in \limsup A_{f(i)}$ but $\langle f\rangle \notin \liminf A_{f(i)}$.

The next few results show that by generalizing the idea of convergent sequences we can find a new class of predicates, the extra continuous ones, which (i) is contained in the class of strongly continuous predicates, coinciding with it when Σ is countable; (ii) gives rise to a more pleasant topology; and (iii) generalizes theorem 2.53.

Let us extend the notion of sequence to include functions from arbitrary non-empty directed sets to P. We will call such a sequence a generalized sequence and a sequence from non-empty directed set D to P a D-sequence. If f is a D-sequence and D* is a subset of D with the property that for each $d \in D$ there exists some $d^* \in D^*$ s.t. $d^* \geqslant d$ we will say that $f \upharpoonright D^*$ is a subsequence of f (it is easy to see that each such D* must be directed). If f is a D-sequence define $\limsup(f) = \bigcap_{d \in D}(\bigcup_{e \geqslant d} f(e))$ and $\liminf(f) = \bigcup_{d \in D}(\bigcap_{e \geqslant d} f(e))$. Say that f converges to A (or $f \to A$) if $\limsup(f) = A = \liminf(f)$. Say that a predicate R is extra-continuous if it satisfies the condition that whenever f is a convergent generalized sequence of points satisfying it then lim(f) satisfies R also. The following is a compilation of some easy results about generalized sequences.

3.24 Theorem

(i) For each generalized sequence f we have $\liminf(f) \subseteq \limsup(f)$.

(ii) All finite generalized sequences converge,

(iii) If f* is a subsequence of f then $\liminf(f) \subseteq \liminf(f^*) \subseteq \limsup(f^*) \subseteq \limsup(f)$.

(iv) There is a topology, $\mathfrak{E}$, in which a set X is closed if and only if there is some extra-continuous predicate R such that $X = \{A \mid R(A)\}$.

The following are four important results which have fairly complicated proofs using the axiom of choice.

3.25 Theorem
The topology $\mathfrak{E}$ has $\{\{A \mid A \subseteq B\}, \{A \mid A \supseteq B\} \mid B \in P\}$ as a subbasis for its <u>closed</u> sets. Thus the class of extra continuous predicates is contained in the class of strongly continuous ones. This containment is strict if and only if Σ is uncountable.

(An example of a strongly continuous but not extra continuous predicate: $R(A) = \ulcorner A$ is countable$\urcorner$.)

3.26 <u>Theorem</u>
The topology $\mathfrak{E}$ is compact (i.e. if $\mathfrak{F}$ is a family of closed sets such that each finite subset of $\mathfrak{F}$ has non-empty intersection then $\bigcap \mathfrak{F}$ is non-empty).

3.27 <u>Theorem</u>
The topology $\mathfrak{E}$ is zero-dimensional (i.e. if X is any closed set and $A \notin X$ then there exists a clopen set Z with the property $A \in Z$ and $X \cap Z = \emptyset$).

3.28 <u>Theorem</u>
a) The clopen sets of $\mathfrak{E}$ are precisely those which can be constructed from sets of the form $\{A \mid w \in A\}$ and $\{A \mid w \notin A\}$ by finite intersections and unions.

b) The predicates R such that both R and $\neg R$ are extra continuous are precisely those which can be defined using the constructs "$w \in A$" ($w \in \Sigma^*$), "$\wedge$" and "$\neg$".

Note that 3.28 extends 3.13 and 3.14 to general alphabets.

The following result is a justification of the use of these extra continuous predicates, since it extends theorem 2.53 to the case of uncountable alphabets, and hence provides an alternative set of conditions justifying the use of 2.1 in an inductive proof. Note that the statement of 3.29 is a translation of the statement of 2.53 into topological terms (as well as being a generalization to arbitrary alphabets).

3.29 <u>Theorem</u>
Suppose that X is a non-empty closed set of $\mathfrak{E}$ and that $f: P \to P$ is a monotonic function with a unique fixed point; then if $f(X) \subseteq X$ we must have $fix(f) \in X$.

proof

The proof is not difficult once we have 3.25 and 3.26. Define $f^{\alpha}(\perp)$ and $f^{\alpha}(\top)$ for arbitrary ordinals α in the same way as we did in 2.53. By 3.25 each of the sets $C_{\alpha} = \{A \mid f^{\alpha}(\perp) \subseteq A \subseteq f^{\alpha}(\top)\}$ is closed. Easy consequences of our definition are that $\beta \in \alpha \Rightarrow C_{\beta} \supseteq C_{\alpha}$ and that $C_{\lambda} = \bigcap_{\beta \in \lambda} C_{\beta}$ if λ is a limit ordinal. Claim that each of the closed sets $X \cap C_{\alpha}$ is non-empty. Proof is by transfinite induction.

When $\alpha = 0$ we have $C_0 \cap X = X$, which is non-empty by assumption.

If $X \cap C_{\alpha}$ is non-empty, then it contains some element A, say. Then $f(A) \in X$, since $f(X) \subseteq X$, and $f(f^{\alpha}(\perp)) \subseteq f(A) \subseteq f(f^{\alpha}(\top))$ since f is monotonic and $f^{\alpha}(\perp) \subseteq A \subseteq f^{\alpha}(\top)$. Thus $f(A) \in X \cap C_{\alpha+1}$, so $X \cap C_{\alpha+1}$ is non-empty.

If λ is a limit ordinal and each $X \cap C_{\alpha}$ $(\alpha \in \lambda)$ is non-empty then $\langle X \cap C_{\alpha} \mid \alpha \in \lambda \rangle$ is a chain of non-empty closed sets. By compactness this chain has a non-empty intersection. But $\bigcap_{\alpha \in \lambda}(X \cap C_{\alpha}) = X \cap \bigcap_{\alpha \in \lambda} C_{\alpha} = X \cap C_{\lambda}$, so $X \cap C_{\lambda}$ is non-empty as claimed. This completes our inductive proof, so we can deduce that $X \cap C_{\alpha}$ is non-empty for all ordinals α.

Since f has a unique fixed point there must exist some ordinal ξ such that $f^{\xi}(\perp) = f^{\xi}(\top) = \text{fix}(f)$. This tells us that $\text{fix}(f) \in X$, as claimed ($C_{\xi} = \{\text{fix}(f)\}$).

It is usual, in spaces defined by convergent sequences, to define compactness by the property that all sequences have convergent subsequences. This result does not translate verbatim to the space $\mathfrak{E}$ (consider the sequence A_i which we defined in 3.24) but there is a corresponding lemma (which can be used to give an alternative proof of 3.29 very much like the proof of 2.53).

3.30 Lemma

If f is any generalized sequence of points in P then there is a point A which is in cl(f), the smallest closed set containing all the points of f, and such that $\text{liminf}(f) \subseteq A$ and $A \subseteq \text{limsup}(f)$.

Since singleton sets are closed in $\mathfrak{E}$ and any ordinary convergent sequence is also a convergent generalized sequence the proof (in 3.24) that the topology is not first countable is still valid for $\mathfrak{E}$ when Σ is uncountable.

Chapter 4 :- A Model for Non-deterministic Processes

We can model the behaviour of non-deterministic machines by observing not only the traces which it is possible for them to execute, but also the sets of symbols which it is possible for them to reject at each stage. The model we use therefore is a subset of $\mathcal{P}(\Sigma^* \times \mathcal{P}(\Sigma))$, i.e. the relations between traces and subsets of Σ (interpreted as refusal sets). As a relation it is possible to regard any process as a function from traces to sets of refusal sets, the image (as a function) of any trace being the set of its (relational) images. It is necessary to impose certain conditions to ensure that a process is realistic. Formally a non-deterministic machine N is a relation which satisfies the following conditions:

4.1 a) The domain of N ($dom(N) = \{w \mid \exists X. (w,X) \in N\}$) is non-empty and prefix closed.

b) $X \in N(w)$ & $Y \subseteq X \Rightarrow Y \in N(w)$
(where $N(w)$ is the set of images of w under N)

c) $X \in N(w)$ & $Y \cap (N \text{ \underline{after} } w)^0 = \emptyset \Rightarrow X \cup Y \in N(w)$

d) If $D \subseteq N(w)$ is a directed set then $\bigcup D \in N(w)$.

$N \text{ \underline{after} } w = \{(v,X) \mid (wv,X) \in N\}$

$N^0 = \{a \in \Sigma \mid \langle a \rangle \in dom(N)\}$

A set of sets is said to be directed if $X,Y \in D \Rightarrow \exists Z \in D.\ Z \supseteq X \cup Y$.

The justifications of these conditions are as follows:

a) If a process has executed any trace it must previously have executed every prefix. It must be possible for a process to do at least nothing ($\langle\rangle$).

b) If a process can refuse every element of a set of symbols then it can refuse every element of a subset.

c) If a process can refuse a set X and it is impossible for it to accept any element of Y then it must be able to refuse the whole of $X \cup Y$.

d) If a process can refuse all approximations to a given set then it can refuse the set itself.

Condition d) is almost always (except in the case of an uncountably infinite alphabet) equivalent to the ascending chain condition:

4.2 d)* If $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_i \subseteq \ldots$ is an ascending chain in $N(w)$ then $\bigcup_{i=1}^{\infty} X_i \in N(w)$,

which is easier to justify intuitively, but breaks down in the uncountable Σ case.

It is possible to consider different versions of these conditions. We could for example re-phrase the definition in terms of acceptance sets (complements of refusal sets). The resulting model is then clearly isomorphic in a simple way to the original.

More fundamentally we could insist that refusal sets be finite. This involves dropping condition (d) and altering (c) slightly. It is fairly easy to show that the two models are isomorphic (by closing up under condition (d)) and that all our definitions of operators are isomorphic except in one case, which we will meet shortly.

I have included infinite refusal sets in this treatment for several reasons. Firstly if we allow infinite alphabets it seems reasonable that we should be able to test a process by offering it an infinite choice (for example the ability to output any integer). This type of behaviour seems to be modelled more naturally by the inclusion of infinite refusal sets. Secondly they give a more natural model to some recent proof rules of C.A.R.Hoare. Thirdly they pose the soundness problem (where the two models may not be isomorphic) explicitly rather than implicitly. This soundness problem (which we will meet in 4.10 et seq) also places a bound on the validity of the above-mentioned proof-rules.

The definitions of the operators are summarized in an appendix to this chapter. The definitions used are motivated in Hoare, Brookes & Roscoe ( ), which also contains much basic material on the model which omitted here.

4.3 Theorem

a) The space M of non-deterministic machines can be partially ordered by reverse inclusion $A \sqsubseteq B$ if $A \supseteq B$. This order can be interpreted $A \sqsubseteq B$ if B is more deterministic than A. Under this order M is a complete partial order in which the maximal elements are the deterministic machines and the minimal element, CHAOS $(= \Sigma^* \times \mathcal{P}(\Sigma))$ represents the process which is absolutely unpredictable.

b) $N \in M \Rightarrow (\langle\rangle, \varnothing) \in N$

c) A process is <u>deterministic</u> if

$$w \in dom(N) \Rightarrow N(w) = \{X \mid X \cap (N \ \underline{after}\ w)^{o} = \varnothing\}$$

d) The non-deterministic <u>or</u> operator is modelled by union.

All the operators defined in the paper with the exceptions of hiding and $\otimes$(intersection) are easy to prove well-defined (map machines to machines) and continuous (preserve directed limits).

The rest of this chapter will consist of an examination of the problems introduced by these operators. The following chapters will show ways of proving correctness properties of individual processes, by adapting and extending the ideas of chapter 2.

Recall the definition of the hiding operator:

4.4 $$N/X = \{(w \restriction (\Sigma - X),\ Y) \mid (w, Y \cup X) \in N\}$$
$$\{(wv, X) \mid \{w' \in dom(N) \mid w' \restriction (\Sigma - X) = w\} \text{ is infinite}\}$$

We cannot hope that this definition will give rise to a continuous operator for infinite X because of the following :

4.5 <u>Example</u>

Let $\Sigma = N \cup \{a\}$ $(a \notin N)$

$A_n = \underline{stop}\ \underline{or}\ (?m:(\{k \mid k > n\}) \to a \to \underline{stop}\ )$

Then it is easy to verify that $\forall n. A_n \sqsubseteq A_{n+1}$ and that $\bigsqcup_{n=1}^{\infty} A_n = \underline{stop}$.

But then $(A_n)/N$ = CHAOS for each n (infinitely many derivations of $\langle\rangle$) and $(\bigsqcup_{n=1}^{\infty} A_n)/N = \underline{stop}$.

Thus $\bigsqcup_{n=1}^{\infty}(A_n/N) \neq (\bigsqcup_{n=1}^{\infty} A_n)/N$

(<u>stop</u> is the process $\{(\langle\rangle, X) \mid X \subseteq \Sigma\}$ which corresponds to <u>abort</u> in this model.)

If we were to alter the second clause of the hiding definition to require arbitrarily long derivations of w, which might seem more natural for infinite X, it would still not make this example continuous. Also it is necessary in this case to make a slight amendment to the operator to make it well-defined with respect to 4.1 (d).

By placing stronger conditions upon the types of process we allow, it is possible to make certain types of infinite hiding continuous. The basic ideas are to divide the alphabet into finitely many portions, and to insist that if an infinite part of one of the portions is available then the whole of it must be. The analysis of this topic is long and complicated, and the results technical. I therefore omit this topic for lack of space, and return to the simpler analysis of finite hiding.

4.6 Theorem

If the set X of hidden symbols is finite, then the hiding operator N/X is well-defined and continuous.

proof

Say that a trace w is a derivation (with respect to X) of v if $w \upharpoonright (\Sigma - X) = v$.

We will prove first that N/X is well-defined.

Throughout this proof A will denote the first (normal) clause of the definition 4.4 of N/X and B will denote the second (infinite chatter) clause.

a) That the domain of N/X is non-empty follows as either $\langle\rangle$ has infinitely many derivations, in which case $(\langle\rangle, \varnothing) \in B$, or it has a maximal one, say w. As w is maximal we must have $(N \text{ \underline{after} } w)^0 \cap X = \varnothing$, which implies $(w,X) \in N$ (by clause (c) of N) and thus $(w, X \cup \varnothing) \in N \Rightarrow (\langle\rangle, \varnothing) \in N/X$.

To prove that the domain of N/X is prefix-closed we use a similar argument. Suppose that $v < w \in \mathrm{dom}(N/X)$. Then either some prefix of v has an infinite number of derivations (in which case $(v, \varnothing) \in B$) or v has at least one derivation (for then either w has a derivation, or the minimal prefix of w with infinitely many derivations is greater than v). If v has a derivation the argument is the same as for $\langle\rangle$ above.

b) Suppose $(w,Y) \in N/X$ and $Y' \subseteq Y$

If $(w, \varnothing) \in B$ the result is elementary.

$$(w,Y) \in A \Rightarrow \exists v.\ v\upharpoonright(\Sigma - X) = w \ \& \ (v, X \cup Y) \in N$$
$$\Rightarrow v\upharpoonright(\Sigma - X) = w \ \& \ (v, X \cup Y') \in N$$
$$\Rightarrow (w,Y') \in A$$

c) Suppose $(w,Y) \in N/X$ and $Z \cap (N \text{ \underline{after} } w)^0 = \varnothing$

If $(w, \varnothing) \in B$ then $(N \text{ \underline{after} } w)^0 = \Sigma$, so the result is trivial.

We may thus suppose that $(w, \emptyset) \notin B$

Hence there is some $v \in dom(N)$ such that

$v \restriction (\Sigma - X) = w \;\;\&\;\; (v, Y \cup X) \in N$

Now $(N\ \underline{after}\ v)^o \cap (\Sigma - X) \subseteq (N/X\ after\ w)^o$

Thus $(\Sigma - X) \cap Z \cap (N\ \underline{after}\ v)^o = \emptyset$

$\Rightarrow (v, Y \cup X \cup ((\Sigma - X) \cap Z)) = (v, Y \cup X \cup Z) \in N$

$\Rightarrow (w, Y \cup Z) \in N/X$ as desired.

d) In this section we use the fact that (as will be proved in 4.14) in these circumstances (given that we have already proved (b)) directed set closure is equivalent to closure under the limits of arbitrary (possibly longer than ω) chains.

Suppose therefore that C is a chain contained within $(N/X)(w)$. Again if $(w, \emptyset) \in B$ the result is trivial, so we may suppose not, so $(w, Y_\alpha) \in A$ for each $Y_\alpha \in C$ (assume the chain is indexed by $\alpha < \xi$ (some initial ordinal) and that $\alpha < \beta \Rightarrow Y_\alpha \leqslant Y_\beta$).
Now as $(w, \emptyset) \in A - B$ there must be finitely many $v_i \in dom(N)$ which are derivations of w $(v_1, \ldots, v_k$, say).
For each $\alpha < \xi$ there must be one of the v_i s.t. $(v_i, Y_\alpha \cup X) \in N$.
We can therefore partition ξ into k sets $\Xi_1, \ldots, \Xi_k$ with the property that $\alpha \in \Xi_i \Rightarrow (v_i, Y_\alpha \cup X) \in N$. It is easy to show that there must be at least one Ξ_i with the property $\alpha < \xi \Rightarrow \exists \beta \in \Xi_i . \alpha \leqslant \beta$.

If we now let $Y'_\alpha = Y_\beta$, where β is minimal in Ξ_i w.r.t. $\beta \geqslant \alpha$ then we have that $C' = \langle Y'_\alpha \cup X \mid \alpha < \xi \rangle$ is a chain contained in $N(v_i)$. Therefore $\bigcup C' = (\bigcup C) \cup X \in N(v_i)$, so $\bigcup C \in (N/X)(w)$ as desired.

To show that N/X is a continuous operator it is necessary to show that if D is a directed set of machines then $(\bigsqcup D)/X = \bigsqcup \{N/X \mid N \in D\}$. It is easy to show that N/X is a monotonic operator, so we have:

$$N \in D \Rightarrow N \sqsubseteq \bigsqcup D$$
$$\Rightarrow (N/X) \sqsubseteq (\bigsqcup D)/X$$
$$\Rightarrow \bigsqcup \{(N/X) \mid N \in D\} \sqsubseteq (\bigsqcup D)/X$$

It therefore only remains to show the reverse inclusion. Because of the reverse nature of the order used, this means showing $(w, Y) \in \bigsqcup \{N/X \mid N \in D\} \Rightarrow (w, Y) \in (\bigsqcup D)/X$.

There are two cases to consider:

a) $\forall N \in D.\ (w, \emptyset) \in B_N$, where B_N is the second clause in the definition of N/X.

b) $\exists N \in D.\ (w, \emptyset) \notin B_N$

In the first case it is clearly sufficient to prove that some prefix of w has infinitely many derivations, for then $(w,X) \in B$ for all $X \subseteq \Sigma$ (where B is the second clause in the definition of $(\bigsqcup D)/X$). There is some prefix v of w which is minimal with respect to there being an infinite derivation set for it in each $N \in D$.

There is therefore some $N_v \in D$ s.t. no proper prefix of v has an infinite derivation set in N_v. Let $D^* = \{N \in D \mid N \sqsupseteq N_v\}$. It is easy to show that $\bigsqcup D^* = \bigsqcup D$ and hence that $(\bigsqcup D^*)/X = (\bigsqcup D)/X$. It is therefore sufficient to show that some prefix of w (namely v) has an infinite derivation in $\bigsqcup D^*$.

As $N \sqsupseteq N_v \Rightarrow N \subseteq N_v$ there must be finitely many derivations of every proper prefix of v for every $N \in D^*$.

Claim that the number of <u>k-minimal</u> derivations of v is finite for each $N \in D^*$ and $k \in \mathbb{N}$ (natural numbers), where a k-minimal derivation of v is one which has precisely k proper prefixes which are derivations of v.

(Thus if $X = \{a\}$ we have -

$\langle aab \rangle$, $\langle ab \rangle$, $\langle b \rangle$ are all o-minimal derivations of $\langle b \rangle$,

$\langle aba$, $\langle ba \rangle$, $\langle aaba \rangle$ are all 1-minimal,

$\langle abaa \rangle$, $\langle baa \rangle$ are 2-minimal, etc.)

Firstly the number of o-minimal derivations is finite. This is certainly true if $v = \langle\rangle$, for then the only one is $\langle\rangle$. If $v = v'\langle a \rangle$ the number must be finite since each o-minimal s must have the form $s'\langle a \rangle$ for some s' which is a derivation of v'. But v' is known only to have finitely many derivations in $N \sqsupseteq N_v$, which gives us the desired result.

If we suppose the number of k-minimal derivations is finite, then since each k+1-minimal derivation has the form $s\langle b \rangle$, where s is k-minimal and $b \in X$ (and X is finite) the number of k+1-minimal derivations must also be finite.

Hence by induction the number of k-minimal derivations is finite for each k. Also, since v has infinitely many derivations each of which is k-minimal for some k, there must be k-minimal derivations present for each k in every $N \in D^*$.

Claim that there is a k-minimal derivation of v present in $\bigsqcup D^*$ for each k. Suppose not (for some k) and that $s_1 \ldots s_r$ are the k-minimal derivations present in N_v. If (for any i) $(s_i, \emptyset) \in N'$ for each $N' \in D^*$ then we would have $(s_i, \emptyset) \in \bigsqcup D^*$, contradicting our assumption. Thus, for each i, there must be some $N_i \in D^*$ s.t. $(s_i, \emptyset) \in N_i$. But then (as D^* is directed) there is some $N^* \in D^*$ s.t. $N^* \sqsupseteq N_i$ for each i. Therefore $\forall i. s_i \notin \mathrm{dom}(N^*)$, but as also $\mathrm{dom}(N^*) \subseteq \mathrm{dom}(N_v)$ there can be no k-minimal derivations of v in N^*. This contradicts the remark on the previous page. Thus the claim that there is some k-minimal derivation of v in $\bigsqcup D^*$ is proven.

But now since every n-minimal derivation of v is clearly distinct from every m-minimal one (if $n \neq m$) there must be an infinity of derivations of v in $\bigsqcup D^*$, which was what we wanted to prove.

This completes the proof of case (a).

Case (b) is rather easier.
If $\exists N_w$ D. $(w, \emptyset) \notin B_{N_w}$ set $D^* = \{N \mid N \sqsupseteq N_w\}$, then as before D^* is itself directed and $\bigsqcup D^* = \bigsqcup D$. Since $N \in D^* \Rightarrow N \sqsubseteq N_w$ we must have $N \in D^* \Rightarrow (w, \emptyset) \notin B_N$.
Thus the number of derivations of w in each $N \in D^*$ is finite, and clearly the derivations in each $N \in D^*$ are included in those in N_w (as $N \sqsubseteq N_w$).
We must have $(w, Y) \in A_N$ for each $N \in D^*$.
For each $N \in D^*$ there is thus a non-empty set S_N of the traces in dom(N) s.t. $s \in S_N \Leftrightarrow s\lceil(\Sigma - X) = w$ and $(s, Y \cup X) \in N$.
By construction each of these S_N is finite and included in S_{N_w}.
Claim that $\exists s \in S_{N_w}. (s, X \cup Y) \in (\bigsqcup D^*)$.
If not then for each $s \in S_{N_w}$ we can find a $N_s \in D^*$ s.t. $s \notin S_{N_s}$.
But then as D^* is directed we can find some $N^* \sqsupseteq N_s$ for every s.
But then we would have $S_{N_w} \cap S_{N^*} = \emptyset$, which contradicts the structure of the S_N. Thus $\exists s \in S_{N_w}. (s, X \cup Y) \in (\bigsqcup D^*)$ as claimed.

But this is the desired result, since then $(w, Y) \in (\bigsqcup D^*)/X$.

By similar methods one can prove (for finite X & Y)

4.7 $(N/X)/Y = N/(X \cup Y)$.

The following commutativity law is an immediate corollary to 4.7 .

4.8 $(N/X)/Y = (N/Y)/X$

There will be further analysis of the hiding operator in chapters 5 & 6, where we will examine the pipe operator "$\gg$" and the Master/Slave operator $(A \| a::B)$ in some depth (both of which use hiding in their definitions).

We now turn our attention to the intersection operator "$\otimes$". This operator is used critically in the definition of the parallel combinator $(A_X\|_Y B)$. Recall its definition:

4.9 $A \otimes B = \{(w, X \cup Y) \mid (w,X) \in A \;\&\; (w,Y) \in B\}$

The interpretation of this is that $A \otimes B$ will only execute traces possible for both A & B and at any stage it can refuse any set which A and B can co-operate in refusing.

It is this last feature (the refusal sets) which cause us the problems. It is easy to show that the domain of $A \otimes B$ is non-empty and prefix closed, that $A \otimes B(w)$ satisfies left-closure (4.1 (b)) and condition 4.1(c). It seems, however, to be far from easy to prove anything about 4.1(d).

The root of this difficulty lies in the fact that if $\{X \cup Y \mid X \in K_1 \;\&\; Y \in K_2\}$ is a directed set there is no reason why K_1 or K_2 should be directed, so it is difficult to prove the existence of elements on the two sides which combine to give the limit of the directed set. The problem can be stated thus:

4.10 If F_1 and F_2 are two families of subsets of Σ which satisfy:

a) <u>left-closure</u> $X \in F \;\&\; Y \subset X \Rightarrow Y \in F$

b) <u>directed closure</u> $D \subseteq F$ directed $\Rightarrow \bigcup D \in F$

does the family $\{X \cup Y \mid X \in F_1 \wedge Y \in F_2\}$ satisfy these conditions?

The conditions (a) & (b) here are the same as 4.1 (b) & (d) respectively.

This problem does not exist for finite alphabets, for then every directed set contains its limit, so we need to examine only the various possible cardinalities of infinite alphabets.

We will adopt the following approach in analysing the problem. Firstly we will see that the answer to 4.10 is affirmative if Σ is countable, which means that in this case $\otimes$ is both well-defined and continuous. Secondly we will see an example to show that if we substitute the countable chain condition 4.2 for directed closure the answer to 4.10 (for uncountable alphabets) is negative. Finally we will see how to prove the well-definedness of $\otimes$ for arbitrary alphabets and that it is a non-trivial set-theoretic tool.

4.11 Theorem

If Σ is countably infinite then the answer to 4.10 is yes.

proof

This result can be proved using directed sets explicitly, using a version of Konig's lemma independent of the Axiom of Choice. We here however prove a version involving chains, since this ties in better with what is to follow.

4.11.1 lemma

If Σ is countable then we can substitute the chain condition

$\langle X_i \mid i \in N \rangle$ a sequence in F s.t. $X_i \subsetneq X_{i+1} \Rightarrow \bigcup_{i=1}^{\infty} X_i \in F$

for directed closure in 4.10 and the effect of the new pair of conditions is equivalent to that of the old pair.

proof

That the above condition is weaker than directed closure is obvious since every ascending chain is a directed set.

It is therefore sufficient to show that if D is a directed set in a family F which is left-closed and ascending chain closed then $\bigcup D \in F$.

Suppose X is any finite subset of $\bigcup D$. Then it is easy to show by techniques akin to some used in the proof of 4.6 that there is some $Y \in D$ s.t. $X \subsetneq Y$. Thus $X \in F$ (by left-closure), so every finite subset of $\bigcup D$ is in F.

As Σ is countable we can enumerate the elements of $\bigcup D$ as $\{a_1, a_2, \ldots, a_i, \ldots\}$ (if $\bigcup D$ is finite then $\bigcup D \in F$ by the above, so the result is trivial).

But then if $X_i = \{a_j \mid j < i\}$ we have that $\langle X_i \rangle$ is an ascending sequence in F (finite subsets of $\bigcup D$) and so $\bigcup_{i=1}^{\infty} X_i = \bigcup D \in F$.

It is thus sufficient to prove that if F_1 and F_2 are two left-closed and ascending chain closed families then so is

$\{X \cup Y \mid X \in F_1 \ \& \ Y \in F_2\}$.

Suppose that these conditions hold of F_1 and F_2 and that

$X_1 \cup Y_1,\ X_2 \cup Y_2,\ \ldots.,\ X_i \cup Y_i,\ \ldots.$

is an ascending chain with limit Z, say and where $X_i \in F_1$, $Y_i \in F_2$.

If Z is finite then the sequence $X_i \cup Y_i$ is ultimately constant (and equal to Z). In that case the desired result, namely the existence of some $X \in F_1$ and $Y \in F_2$ s.t. $X \cup Y = Z$, is trivial. We may therefore assume that Z is infinite and is enumerated as $\{a_1, \ldots, a_i, \ldots.\}$.

Claim that for each $j \in N$ we can find an infinite sequence of natural numbers $\langle n_{j,i} \mid i \in N\rangle$ with the following properties:

a) $i < i' \Rightarrow n_{j,i} < n_{j,i'}$

b) $\langle n_{j+1,i}\rangle_i$ is a subsequence of $\langle n_{j,i}\rangle_i$.

c) If $A_j = \{a_k \mid k < j\}$ then $\forall i . A_j \cap X_{n_{j,i}} = A_j \cap X_{n_{j,1}}$, $\forall i . A_j \cap Y_{n_{j,i}} = A_j \cap Y_{n_{j,1}}$ and $\forall i . A_j \subseteq X_{n_{j,i}} \cup Y_{n_{j,i}}$.

(Note that this is very similar to the construction used in the proof of 2.52.)

We can find such a sequence for j=1 since $A_1 = \emptyset$ so we can put $n_{j,i} = i$.

Suppose that we have constructed such a sequence for j.

By construction (since $n_{j,i+1} > n_{j,i}$) the sequence $\langle X_{n_{j,i}} \cup Y_{n_{j,i}} \mid i \in N\rangle$ is ascending with limit Z. Thus $a_j \in \bigcup_{i=1} (X_{n_{j,i}} \cup Y_{n_{j,i}})$, so there is some k such that $i \geq k \Rightarrow a_j \in (X_{n_{j,i}} \cup Y_{n_{j,i}})$.

It is easy to see that <u>either</u> there is an infinite subset of $\{i \mid i \geq k\}$ such that $a_j \in X_{n_{j,i}} \cap Y_{n_{j,i}}$ for each i, <u>or</u> there is an infinite subset such that $a_j \in X_{n_{j,i}} - Y_{n_{j,i}}$ for each i, <u>or</u> there is an infinite subset such that $a_j \in Y_{n_{j,i}} - X_{n_{j,i}}$ for each i. We now define $n_{j+1,i}$ to be the ith $n_{j,s}$ such that s lies in the first of these infinite sets to exist (that is, if $a_j \in X_{n_{j,i}} \cap Y_{n_{j,i}}$ set exists, choose it, etc.).

It is now easy to show that the new sequence $n_{j+1,i}$ satisfies all that is required of it.

Now choose $m_i = n_{i,i}$. By conditions (a) & (b) above we have that $m_i < m_{i+1}$ for each i.

Now let $W_i = X_{m_i} \cap A_i$ and $V_i = Y_{m_i} \cap A_i$. The following must hold of these W_i an V_i.

a) $W_i \in F_1$ & $V_i \in F_2$ (by left-closure)

b) $W_i \cup V_i = A_i$

c) $W_i \subseteq W_{i+1}$ & $V_i \subseteq V_{i+1}$ (as every $n_{i+1,j}$ is a $n_{i,j}$)

But since F_1 and F_2 are ascending chain closed they must contain $\bigcup_{i=1}^{\infty} W_i$ and $\bigcup_{i=1}^{\infty} V_i$ respectively, and

$$(\bigcup_{i=1}^{\infty} W_i) \cup (\bigcup_{i=1}^{\infty} V_i) = \bigcup_{i=1}^{\infty}(W_i \cup V_i) = \bigcup_{i=1}^{\infty} A_i = Z \quad .$$

Hence Z lies in $\{X \cup Y \mid X \in F_1 \ \& \ Y \in F_2\}$ as desired.

4.12 Theorem

The ascending chain condition 4.2 is insufficient to make $\otimes$ well defined if Σ is uncountable.

proof

First observe that for an uncountable alphabet 4.2 is a strictly weaker condition than directed closure.

An example to show this is the countable subsets of the real numbers. This family is closed under 4.2 (and is left closed) but is not directed closed since the family itself is directed but does not contain its union.

Our aim will be to show there exists an uncountable alphabet which is somehow isomorphic to the set of proofs of membership of families which are left closed and satisfy 4.2.

4.12.1 If $\langle X_i \rangle$ is any sequence of sets define

$$\text{liminf}(X_i) = \bigcup_{j=1}^{\infty}(\bigcap_{i=j}^{\infty}(X_i))$$

(Note the similarity between this and 2.42.)

Claim that if $\langle X_i \rangle$ is any sequence of sets in a chain-closed family (for the rest of this section we will use the term chain closed to mean left closed and satisfying 4.2) then $\text{liminf}(X_i)$ is also contained in the family.

Let $Y_i = \bigcap_{j=i}^{\infty}(X_j)$.

Clearly each Y_i is contained in the family (F say) by left closure (it is greater than X_i).

Also the Y_i are an increasing sequence (being intersections of decreasing sets).

Therefore $\text{liminf}(X_i) = \bigcup_{i=1}^{\infty} Y_i \in F$ as desired.

4.12.2 <u>lemma</u>

If G is any family of sets which is closed under the taking of liminfs then the family $\{X \mid \exists Y \in G. Y \supseteq X\}$ is chain closed.

<u>proof</u>

Let $F = \{X \mid \exists Y \in G. Y \supseteq X\}$.

That F is left closed is trivial since $X \in F \Rightarrow \exists Y \in G. X \subseteq Y$ so $X' \subset X \Rightarrow X' \subset Y$. Thus $X' \in F$ as desired.

Suppose that $\langle X_i \rangle$ is any ascending chain contained in F. Then for each X_i we can choose a $Y_i \in G$ such that $X_i \subseteq Y_i$. But then

$$\begin{aligned} j > i \; &\Rightarrow X_i \subseteq X_j \subseteq Y_j \\ &\Rightarrow X_i \subseteq \bigcap_{j=i}^{\infty} Y_j \\ &\Rightarrow \bigcup_{i=1}^{\infty} X_i \subseteq \bigcup_{i=1}^{\infty}\left(\bigcap_{j=i}^{\infty} Y_j\right) = \text{liminf}(Y_j) \in G \end{aligned}$$

Hence $\bigcup_{i=1}^{\infty} X_i \in F$ as desired.

This result (in a way which will become clear shortly) helps us to bound the number of elements we must include in any chain closed family, given that we wish it to contain an arbitrary collection of sets.

4.12.3 Define a <u>finite path ω-branching tree</u> $t \in T_N$ as follows:

a) t is a directed tree with a single base node.

b) Every non-leaf node of t is unlabelled and has edges labelled 1,2,3,... leading out of it to subtrees $t_1, t_2, \ldots \in T_N$ which are all distinct.

c) Every leaf node of t is labelled by some $n \in N$.

d) t contains no infinite path.

4.12.4 <u>lemma</u>

a) The relation $t_1 < t_2$ if t_1 is a strict subtree* of t_2 is a partial order on T_N.

b) It is a well-founded partial order, and thus every subset of T_N contains its minimal elements and induction and recursion are both possible.

<u>proof</u>

a) If $t_1 < t_2$ and $t_2 < t_3$ then trivially $t_1 < t_2$. If $t_1 > t_2 > t_3 > \ldots > t_i > \ldots$ were an infinite descending

(* - all descendants of some non-base node of t_2)

chain in T_N then t_1 would contain an infinite path (through the successive base nodes of the t_i) contradicting its membership of T_N. Hence there are no such chains in T_N, so in particular for no element of T_N can we have $t > t$ for this would give rise to the descending chain $t > t > t > \ldots$. This completes the proof that T_N is partially ordered by $<$.

b) Suppose S is any non-empty subset of T_N which does not contain any elements which are minimal with respect to it. Then for every element t of S there is some $s \in S$ such that $s < t$. It is then easy to show (given AC) that there is an infinite descending chain (starting from any element) contradicting the above.

Hence every subset S of T_N contains some element t such that $t \in S \Rightarrow \neg(s < t)$.

It is then easy to prove the inductive principle:

$$(\forall t \in T_N . (\forall s < t . R(s)) \Rightarrow R(t)) \Rightarrow (\forall t \in T_N . R(t))$$

for any property R.

Also it is easy to show that if H is any function $H: \{(t,g) \mid t \in T_N \;\&\; g:\{s \mid s < t\} \to A\} \to A$ (for any set A) then there is a unique total function $f:T_N \to A$ which satisfies $f(t) = H(t, f\restriction\{s \mid s < t\})$ for all $t \in T_N$.

Define $t \in T_N$ to be a <u>singleton</u> if it has only a single node (we will denote t by $\langle n \rangle$, where the single (leaf) node is labelled "n").

We will sometimes denote <u>infinite</u> elements of T_N by the elements of T_N at the ends of their lowest level edges, thus $\langle t_i \mid i \in N \rangle$ is the tree with t_i at the end of edge i leading out of the base node.

Define functions $f:T_N \times T_N \to \mathcal{P}(N)$ and $g:T_N \times T_N \to \mathcal{P}(N)$ as follows:

If t & s are both singletons then $f(s,t) = g(s,t) = \emptyset$.

If t is a singleton $\langle n \rangle$ and s is infinite then $f(t,s) = \{n\}$ and $g(t,s) = \{m \mid m$ labels some leaf of s and $n \neq m\}$.

If s is a singleton $\langle n \rangle$ and t is infinite then $g(t,s) = \{n\}$ and $f(t,s) = \{m \mid m$ labels some leaf of t and $n \neq m\}$.

If both t & s are infinite then each contains a finite or countable number of non-leaf nodes with infinitely many leaves attached. (This is easy to prove using the induction principle outlined on the last page.)

Suppose that these nodes of t are $n_1, n_2, \ldots, n_k, \ldots$ and that these nodes of s are $m_1, m_2, \ldots, m_k, \ldots$

and that the leaf nodes of n_i form the sequence $\langle n_{i,j} \rangle (j \geq 0)$
and that the leaf nodes of m_i form the sequence $\langle m_{i,j} \rangle$.

Define $H_0 = L_0 = \emptyset$, and define two cyclical functions:

Let $p(r) = r \pmod n$ if there is a finite number n of n_js
$= r -$ (the greatest triangular number $\leqslant r$) if there are infinitely many n_js

Let $q(r) = r \pmod m$ if there are m m_js
$= r -$ (the greatest triangular number $\leqslant r$) if there are infinitely many m_js

Now let l_r = least element of $\{n_{p(r),j} \mid j \in N\} - (H_{r-1} \cup L_{r-1})$
and h_r = least element of $\{m_{q(r),j} \mid j \in N\} - (H_{r-1} \cup L_{r-1} \cup \{l_r\})$
and $H_r = H_{r-1} \cup \{h_r\}$ & $L_r = L_{r-1} \cup \{l_r\}$ ($r \geqslant 1$ in each case).

This is a well-defined recursion since each H_r and L_r is finite, and the sets $\{n_{i,j} \mid j \in N\}$ and $\{m_{i,j} \mid j \in N\}$ are infinite for every i in the correct ranges as all the leaves on the n_i and m_i are by assumption different.

Note that since by construction all the l_i are distinct from all l_j $(i \neq j)$ and from all h_j the two sets $\{l_i \mid i \in N\}$ and $\{h_i \mid i \in N\}$ are disjoint.

Now set $f(t,s) = \{l_j \mid j \in N\}$
and $g(t,s) = \{h_j \mid j \in N\}$.

This completes the definition of f & g.
Note that in the last case, since the functions p & q each take every node index as their value infinitely many times, there are infinitely many $n_{i,j}$ in $f(t,s)$ for each i and infinitely many $m_{i,j}$ in $g(t,s)$ for each i.

By construction also $f(t,s) \cap g(t,s) = \emptyset$ for all $t,s \in T_N$.

Now define $X_n^* = \{(t,s) \mid n \notin f(t,s)\}$
$Y_n^* = \{(t,s) \mid n \notin g(t,s)\}$
For each n we have $X_n^* \cup Y_n^* = T_N \times T_N$, since $(t,s) \in T_N \times T_N$ implies $f(t,s) \cap g(t,s) = \emptyset$ so either $n \notin f(t,s)$ or $n \notin g(t,s)$.

Now define $X_n = X_n^* \cup \{0,1,\ldots,n\}$
$Y_n = Y_n^* \cup \{0,1,\ldots,n\}$
Thus $X_n \cup Y_n = T_N \times T_N \cup \{0,1,2,\ldots,n\}$ and so the $X_n \cup Y_n$ form an ascending chain with limit $T_N \times T_N \cup N$ (= Σ, say).

Define functions $h,k: T_N \to \mathcal{P}(\Sigma)$ by recursion.

$h(\langle n \rangle) = Y_n \qquad k(\langle n \rangle) = X_n$
$h(\langle t_i \mid i \in N \rangle) = \text{liminf}(h(t_i))$
$k(\langle t_i \mid i \in N \rangle) = \text{liminf}(k(t_i))$

By 4.12.4 this recursion is well-defined.

Consider now the families F_1 and F_2 defined:

$F_1 = \{X \mid \exists t \in T_N . X \subseteq k(t)\}$
$F_2 = \{X \mid \exists t \in T_N . X \subseteq h(t)\}$

These families are both chain closed by a similar argument to 4.12.2 (a little care is required to show that we can get away with our demand that all the first level subtrees of any tree are distinct).

Claim that the limit (Σ) of the above ascending chain cannot be expressed as $X \cup Y$, where $X \in F_1$ and $Y \in F_2$. It is clearly sufficient to show that Σ cannot be expressed as $k(t) \cup h(s)$ for any $(t,s) \in T_N \times T_N$.

If t & s are both singletons, labelled n & m respectively then $\max(n,m)+1 \notin k(t) \cup h(s)$.

Otherwise claim that $(t,s) \notin k(t) \cup h(s)$.
We will show here that $(t,s) \notin k(t)$, the proof that $(t,s) \notin h(s)$ being practically identical.

If t is a singleton labelled n then $n \in f(t,s)$, and thus $(t,s) \notin X_n^*$. Hence $(t,s) \notin X_n = k(\langle n \rangle)$, as required.

If t is infinite and s is a singleton $\langle n \rangle$ then work by induction on the structure of t.
Claim that every infinite $t' < t$ satisfies $(t,s) \notin k(t')$.

Assume that all infinite t_i in $\langle t_i \mid i \in N\rangle \leq t$ satisfy this. Either $\langle t_i \mid i \in N\rangle$ has infinitely many of the t_i infinite or it does not.
In the first case, by induction, there are infinitely many i such that $(t,s) \notin k(t_i)$. But then $(t,s) \notin \liminf(k(t_i))$. Thus $(t,s) \notin k(\langle t_i \mid i \in N\rangle)$ as desired.
In the second case there must be infinitely many leaf nodes amongst the t_i, only one of which (at most) can be labelled n. For each $\langle m\rangle$ s.t. $m \neq n$ we have

$m \in f(t,s)$ (as $m (\neq n)$ labels a leaf node of t)

$\Rightarrow (t,s) \notin k(\langle m\rangle)$.

Thus again $(t,s) \notin k(t_i)$ for infinitely many i, so that $(t,s) \notin \liminf(k(t_i)) = k(\langle t_i \mid i \in N\rangle)$ as required.

Finally we have the case that both t and s are infinite. Again we prove by induction on the infinite $t' \leq t$ that $(t,s) \notin k(t')$.
Assume that all infinite t_i in $\langle t_i \mid i \in N\rangle \leq t$ satisfy this. If there are infinitely many infinite t_i then $(t,s) \notin k(\langle t_i\rangle_i)$ by the same argument as above. If there are not then infinitely many of the t_i must be leaf nodes. Thus in the definition of f(t,s) and g(t,s) the base node of $\langle t_i \mid i \in N\rangle$ must be one of the n_i. Hence infinitely many of the labels of the t_i are included in f(t,s). Thus, as above, there are infinitely many t_i s.t. $(t,s) \notin k(t_i)$ and so $(t,s) \notin \liminf(k(t_i)) = k(\langle t_i \mid i \in N\rangle)$ as required. This completes the induction, and so the property holds of t itself.

This completes the proof that if t & s are not both singletons then $(t,s) \notin h(s) \cup k(t)$.

Hence in any case $(s,t) \in T_N \times T_N \Rightarrow k(t) \cup h(s) \neq \Sigma$, which is known to be the limit of an ascending chain in $\{X \cup Y \mid X \in F_1 \ \& \ Y \in F_2\}$, and so this family is not chain closed.

To complete the proof of 4.12 all we now have to do is set

$$N_1 = \{(\langle\rangle,X) \mid X \in F_1\} \cup \{(\langle a\rangle,X) \mid a \in \Sigma ,\ X \subseteq \Sigma\}$$

$$N_2 = \{(\langle\rangle,X) \mid X \in F_2\} \cup \{(\langle a\rangle,X) \mid a \in \Sigma ,\ X \subseteq \Sigma\}$$

and observe that both N_1 and N_2 satisfy 4.1 (a)-(c) and 4.2 but $N_1 \otimes N_2$ does not.

It is possible to extend the notion of ascending chain (and so also the ascending chain condition 4.2) to include chains indexed by larger ordinals than the usual ω.

If η is any ordinal define a η-chain to be a function $\theta: \eta \to \mathcal{P}(\Sigma)$ which satisfies $\xi \in \pi \Rightarrow \theta(\xi) \subseteq \theta(\pi)$.
We will often use the notation $\langle c_\rho | \rho \in \eta \rangle$ for the η-chain with ρ-component c_ρ.

Clearly the only η-chains to be of interest from the point of view of taking unions are those indexed by limit ordinals (as other ones contain their unions as last members).

We can now extend 4.2 either to insist that a family be closed under the unions of arbitrary length chains or under the unions of all chains of length smaller than some λ.

The following is a technical result in classifying these conditions.

4.13 Lemma

Suppose that Σ is an infinite alphabet with cardinal $|\lambda|$, where λ is an initial ordinal. Then if $C = \langle c_\kappa | \kappa \in \eta \rangle$ is any chain over this Σ there is some subchain $D = \langle d_\kappa | \kappa \in \tau \rangle$ of C such that $\tau \leq \lambda$ is a regular initial ordinal and $\bigcup C = \bigcup D$.

The proof is not difficult but is omitted.

It is fairly easy to see how the methods of 4.12 could be extended to show that no ascending chain condition which expressed a bound on the length of chain could work for general alphabets (in the sense of making the definition of $\otimes$ valid). 4.13 also shows that (using AC) for any fixed alphabet there is maximum length of chain which need be considered.

The next result shows that the arbitrary ascending chain condition is in fact equivalent to directed set closure.

4.14 Lemma

Suppose that Σ is an alphabet of cardinality $|\lambda|$, where λ is an infinite initial ordinal. Suppose further that F is a family of subsets of Σ which is left-closed. Then the following two conditions are equivalent.

(i) F is directed closed.

(ii) F is closed under the limits of η-chains for every regular initial ordinal $\eta \leq \lambda$.

proof

That (i) $\Rightarrow$ (ii) is obvious since every η-chain is a directed set in its own right. In proving the converse note that by 4.13 the second condition is equivalent to closure under the unions of arbitrary length chains.

Suppose then that (ii) holds of F and that $D \subseteq F$ is a directed set. Claim that for each $D' \subseteq D$ (D' not necessarily directed) we have $\bigcup D' \in F$.

Prove this by transfinite induction on $|D'|$ (actually T.I. on the initial ordinal equinumerous with D', so we are using AC here).
If D' is finite then there must be some element of D which contains $\bigcup D'$ since D is directed. Thus $\bigcup D' \in F$ by left-closure.
Suppose then that D' is infinite and that the result holds of all sets with smaller cardinality. Enumerate D' by its initial ordinal (so that $D' = \{X_\kappa | \kappa \in \theta\}$). For each $\alpha \in \theta$ set $D'_\alpha = \{X_\kappa | \kappa \in \alpha\}$. By construction each D'_α has strictly smaller cardinal than D' and so $\bigcup D'_\alpha \in F$ by assumption.

The $\bigcup D'_\alpha$ are an ascending θ-chain (as the D'_α are an ascending sequence of sets) so by 4.13 $\bigcup_{\alpha\in\theta}(\bigcup D'_\alpha) = \bigcup D' \in F$.

Hence by induction $\bigcup D' \in F$ for every $D' \subseteq D$, and in particular $\bigcup D \in F$ as desired, completing the proof of 4.14.

The methods used in proving can be extended to show, using the above, that 4.10 holds for certain uncountable alphabets. These methods become quite involved, and seem to break down at the cardinal $\aleph_\omega$ (which may or may not be less than the cardinal of the real numbers, dependant on the continuum hypothesis).
In order to complete the proof of the truth of 4.10 for general alphabets we appeal to the compactness theorem of propositional calculus. We will in fact see that not only is this implied by propositional compactness but that also compactness is directly provable from 4.10 without recourse to any other powerful set-theoretic tools such as Zorn's lemma (the normal result used to prove compactness).

4.15 Theorem

The truth of 4.10 is both implied by and implies the compactness theorem for propositional calculi with arbitrarily large collections of propositional variables.

proof

Recall the compactness theorem:

If L is a propositional language which consists of the finite formulae formed from a set of propositional variables and the standard connectives then any subset of L which finitely consistent is consistent. A set $K \subseteq L$ is consistent if there is some truth assignment which satisfies every element of K and is finitely consistent if each finite $K' \subseteq K$ is consistent.

We show first that the truth of 4.10 is implied by the above. Suppose that Σ is any alphabet and that F_1 and F_2 are two families of subsets of Σ which satisfy the conditions of 4.10. Let L be the language which contains distinct propositional variables p_α and q_α for each $\alpha \in \Sigma$ and the finite combinations of these by "$\neg$","$\vee$"&"$\wedge$".

Suppose that D is a directed subset of $\{X \cup Y \mid X \in F_1 \ \& \ Y \in F_2\}$. Define sets of formulae K_1, K_2 & K_3 as follows:

$$K_1 = \{ p_\alpha \vee q_\alpha \mid \alpha \in \cup D \}$$
$$K_2 = \{ \neg(p_\alpha \wedge p_\beta \wedge \ldots \wedge p_\eta) \mid \{\alpha, \overset{(\text{finite})}{\beta}, \ldots, \eta\} \notin F_1 \}$$
$$K_3 = \{ \neg(q_\alpha \wedge q_\beta \wedge \ldots \wedge q_\eta) \mid \{\alpha, \overset{(\text{finite})}{\beta}, \ldots, \eta\} \notin F_2 \}$$

Claim that $K = K_1 \cup K_2 \cup K_3$ is finitely consistent.

Suppose that $K' \subseteq K$ is finite. Then $K' \cap K_1$ is finite and so is $U = \{ \alpha \mid p_\alpha \vee q_\alpha \in K' \}$. For each $\alpha \in U$ there is some $X \in D$ such that $\alpha \in X$ and so, as D is directed, there is some $X \in D$ such that $U \subseteq X$. This X can be written $Y \cup Z$ for some $Y \in F_1$ and $Z \in F_2$ (by assumption). Define a truth assignment s as follows:

$s(p_\alpha)$ = <u>true</u> if $\alpha \in Y$
$s(p_\alpha)$ = <u>false</u> otherwise
$s(q_\alpha)$ = <u>true</u> if $\alpha \in Z$
$s(q_\alpha)$ = <u>false</u> otherwise .

By construction $s(\varphi)$ = <u>true</u> for each $\varphi \in K' \cap K_1$.

If $\varphi \in K_2$ then φ can be written $\neg(\bigwedge_{\alpha \in W} p_\alpha)$ for some finite $W \notin F_1$.

Certainly $W \nsubseteq Y$ (as $Y \in F_1$) so that $W - Y \neq \emptyset$. Hence $s(\varphi)$ = <u>true</u>.

Similarly $\psi \in K_3 \Rightarrow s(\psi)$ = <u>true</u>.

Thus $s(\psi)$ = <u>true</u> for each $\psi \in K' \cap K_1$ or $K' \cap K_2$ or $K' \cap K_3$ and so the whole of K' is satisfied by s.

This completes the proof that K is finitely consistent.

By the assumption of the compactness theorem, therefore, there is some truth assignment s* which simultaneously satisfies the whole of K. Define $Y = \{\alpha \in \bigcup D \mid s^*(p_\alpha)\}$ and $Z = \{\alpha \in \bigcup D \mid s^*(q_\alpha)\}$.

Clearly $\bigcup D = Y \cup Z$ as
$$\alpha \in \bigcup D \Rightarrow s^*(p_\alpha \vee q_\alpha) = \underline{true}$$
$$\Rightarrow s^*(p_\alpha) = \underline{true} \vee s^*(q_\alpha) = \underline{true}$$
$$\Rightarrow \alpha \in Y \vee \alpha \in Z$$

Also $Y \in F_1$ as the set $\{Y' \mid Y' \subseteq Y \ \& \ Y' \text{ finite}\}$ is directed with limit Y and is contained in F_1 as
$$(Y' \subseteq Y) \ \& \ Y' \text{ finite} \Rightarrow s^*(\varphi) = \underline{true}, \text{ where } \varphi = \bigwedge_{\alpha \in Y'} p_\alpha$$
$$\Rightarrow s^*(\psi) = \underline{false}, \text{ where } \psi = \neg(\bigwedge_{\alpha \in Y'} p_\alpha)$$
$$\Rightarrow \psi \notin K_2$$
$$\Rightarrow Y' \in F_1 .$$

Similarly $Z \in F_2$ which completes the proof of our result.

Secondly we show that the truth of 4.10 can be used to prove the compactness theorem. Suppose that L is a propositional language of finite formulae with variables V. Suppose further that $K \subseteq L$ is finitely consistent. Define two families F_1 and F_2 as follows:

$$F_1 = \{X \subseteq L \mid X \cup K \text{ is finitely consistent}\}$$
$$F_2 = \{X \subseteq L \mid X \text{ is finitely frustratable}\}$$

where $X \subseteq L$ is finitely frustratable (f.f.) if for every finite $X' \subseteq X$ there is a truth assignment which maps each $\psi \in X'$ to <u>false</u>.

Each of these families satisfies the conditions of 4.10: left-closure is trivial and directed set closure follows from the fact that every finite subset of the union of a directed set is contained in some element of the set.

Claim that the whole of L can be expressed as $Y \cup Z$ for some $Y \in F_1$ and $Z \in F_2$. By the assumption of the truth of 4.10 it will be sufficient to show that each finite subset of L can be so expressed (as these finite sets are a directed set with union L).

Claim that each finite subset of L is a subset of some set of the form $Y \cup Z$, such that $Y \in F_1$, $Z \in F_2$ and $Z \subseteq \{\psi \mid \neg\psi \in Y\}$.

Proof is by induction on the size of the subset.
The result is trivial for the empty set $\emptyset$.
Suppose that it holds of all smaller sets than $X = X^* \cup \{\psi\}$ ($\psi \notin X^*$). By assumption there are $Y^* \in F_1$ and $Z^* \in F_2$ which satisfy our requirements for X^*.
Either $K \cup Y^* \cup \{\psi\}$ is finitely consistent or $K \cup Y^* \cup \{\psi\}$ is. This is because if not there would be finite subsets U & V of $K \cup Y^*$ such that $U \cup \{\psi\}$ and $V \cup \{\neg\psi\}$ are both inconsistent. But then it is easy to see that $U \cup V$ would be an inconsistent finite subset of $K \cup Y^*$, something which by assumption does not exist.

In the first case, let $Y = Y^* \cup \{\psi\}$ and $Z = Z^*$, in the second case let $Y = Y^* \cup \{\neg\psi\}$ and $Z = Z^* \cup \{\psi\}$. It is easy to see that in either case Y and Z satisfy all that is required of them for X.

Hence the result holds for all finite $X \subseteq L$. Thus as stated there must be some $Y \in F_1$ and $Z \in F_2$ such that $Y \cup Z = L$.

For each $\psi \in L$ either $\psi \in Y$ or $\neg\psi \in Y$. This is because the set $\{\psi, \neg\psi\}$ is not f.f. and so is not contained in Z.
Hence in particular this is true of the atomic propositional variables. Define a truth assignment s* by $s^*(q) =$ <u>true</u> if $q \in Y$ and $s^*(q) =$ <u>false</u> if $\neg q \in Y$ (there can be no ambiguity as $\{q, \neg q\}$ is not consistent).

Claim that every statement in K is satisfied by s*. This is true as for each $\psi \in K$ the set $\{\psi, \delta q \mid q \text{ occurs in } \psi\}$ is consistent, where $\delta q = q$ if $q \in Y$, $\delta q = \neg q$ otherwise.

Thus K is consistent as desired, completing the proof of 4.15.

This result means that our result is equivalent to several other results such as the so-called "ultrafilter lemma" (which can itself be easily proved from the truth of 4.10).

The consistency of this model under the $\otimes$ operator (and hence under the ($_X\|_Y$) operator) has the implication that the parallel operator does not introduce any new non-determinism into a system in the following sense:

4.16 <u>Corollary</u>
If $(P_X\|_Y Q) \xrightarrow{s} R$, where $R^O = Z$, then there exist P* & Q* such that $P \xrightarrow{s'} P^*$ and $Q \xrightarrow{s''} Q^*$ and $Z \cap (X \cup Y) = (X \cap P^O) \cup (Y \cap Q^O)$.
($s' = s \upharpoonright X$, $s'' = s \upharpoonright Y$).

The continuity of $\otimes$ follows immediately from its well-definedness in the following manner.

4.17 Theorem

The $\otimes$ operator is continuous.

proof

The continuity of $\otimes$ in both arguments follows from its continuity in its two arguments seperately, and that in one argument follows from that in the other by commutativity. It will thus suffice to show that if D is a directed set of processes then for each N M we have $(\bigsqcup D) \otimes N = \bigsqcup_{Q \in D} (Q \otimes N)$.

That $(\bigsqcup D) \otimes N \sqsupseteq \bigsqcup_{Q \in D} (Q \otimes N)$ follows easily from monotonicity, so it is sufficient to show that $(s,X) \in \bigsqcup_{Q \in D} (Q \otimes N) \Rightarrow (s,X) \in (\bigsqcup D) \otimes N$.

Suppose that $(s,X) \in \bigsqcup_{Q \in D} (Q \otimes N)$. By left-closure $(s,X^*) \in \bigsqcup_{Q \in D} (Q \otimes N)$ for each finite $X^* \subseteq X$. For each $Q \in D$ therefore there is a non-empty set $\Theta(Q) = \{Y \in Q(s) \mid \exists Z \in N(s). Z \cup Y = X^*\}$. Of necessity $\Theta(Q)$ is finite as it is a subset of the finite set $\mathcal{P}(X^*)$, and we also have $Q' \sqsupseteq Q \Rightarrow \Theta(Q') \subseteq \Theta(Q)$. As a downwards-directed set of finite non-empty sets the $\Theta(Q)$ have a non-empty intersection.

There is therefore some $Y \in \bigcap_{Q \in D} Q(s)$ such that $\exists Z \in N(s). Y \cup Z = X^*$. But $\bigcap_{Q \in D} Q(s) = (\bigsqcup D)(s)$, which tells us that $X^* \in (\bigsqcup D) \otimes N(s)$.

Since $(\bigsqcup D) \otimes N(s)$ contains each finite subset of X, and as $(\bigsqcup D) \otimes N$ is a well-defined process by 4.15 we have the desired result that $X \in (\bigsqcup D) \otimes N(s)$.

Appendix to Chapter 4 :- Definitions of Operators

Suppose that A, B, A_x ($x \in T$) are all elements of M, the space of non-deterministic processes, $a \in \Sigma^-$, $X, Y \subseteq \Sigma$, and T is a set of unnamed elements of Σ^-.

(i) $\mathrm{CHAOS} = \{(w,X) \mid w \in \Sigma^* \ \& \ X \subseteq \Sigma\}$

(ii) <u>abort</u> $= \{(\langle\rangle, X) \mid X \subseteq \Sigma\}$

(iii) <u>skip</u> $= \{(\langle\rangle, X), (\langle\surd\rangle, Y) \mid X \subseteq \Sigma^- \ \& \ Y \subseteq \Sigma\}$

(iv) $a \to A = \{(\langle\rangle, X) \mid a \notin X\} \cup \{(\langle a\rangle w, X) \mid (w,X) \in A\}$

(v) $x{:}T \to A_x = \{(\langle\rangle, X) \mid X \cap T = \emptyset\} \cup \{(\langle x\rangle w, X) \mid (w,X) \in A_x \ \& \ x \in T\}$

(vi) $a.x{:}T \to A_x = \{(\langle\rangle, X) \mid X \cap a.T = \emptyset\} \cup \{(\langle a.x\rangle w, X) \mid x \in T \ \& \ (w,X) \ A_x\}$

(vii) $a.T = \{(a.w, X \cup a.Y) \mid (w,Y) \in A \ \& \ X \cap a.\Sigma^- = \emptyset\}$

(viii) A <u>or</u> B $= A \cup B$

(ix) $A \,\square\, B = \{(\langle\rangle, X \cap Y) \mid (\langle\rangle, X) \in A \ \& \ (\langle\rangle, Y) \in B\}$
$\cup \ \{(w,X) \mid w \neq \langle\rangle \ \& \ ((w,X) \in A \vee (w,X) \in B)\}$

(x) $A;B = \{(w,X) \mid w \in (\Sigma^-)^* \ \& \ (w, X \cup \{\surd\}) \in A\}$
$\cup \ \{(wv,X) \mid w \in (\Sigma^-)^* \ \& \ w\langle\surd\rangle \in \mathrm{dom}(A) \ \& \ (v,X) \in B\}$

(xi) $A/X = \{(w/X, Y) \mid (w, X \cup Y) \in A\}$
$\cup \ \{(wv, Y) \mid Y \subseteq \Sigma \ \& \ \{s \in \mathrm{dom}(A) \mid s/X = w\}$ is infinite$\}$

(xii) $(A_X \|_Y B) = f_X(A) \otimes f_Y(B) \otimes \mathrm{RUN}_{X \cup Y}$, where
$C \otimes D = \{(w, Z \cup V) \mid (w,Z) \in C \ \& \ (w,V) \in D\}$;
$f_Z(C) = \{(w,V) \mid (w \upharpoonright Z, V) \in C \ \& \ V \subseteq Z\}$
and $\mathrm{RUN}_Z = \{(w,V) \mid w \in Z^* \ \& \ V \cap Z = \emptyset\}$

(xiii) If $\{\Gamma_1, .., \Gamma_k\}$ is a partition of some non-empty indexing set Λ, and if $A_1, .., A_k$ are functions $A_i : M^\Lambda \times \Gamma_i \to M$ then the recursively defined process
B_λ, where $\xi \in \Gamma_1 \Rightarrow B_\xi \Leftarrow A_1$
$\xi \in \Gamma_2 \Rightarrow B_\xi \Leftarrow A_2$
. . . .
$\xi \in \Gamma_k \Rightarrow B_\xi \Leftarrow A_k$
has the value $(\bigsqcup_{n=0}^{\infty} G^n(\mathrm{CHAOS}^\Lambda))_\lambda$, where $G : M^\Lambda \to M^\Lambda$ is the function defined $G(\underset{\sim}{C})_\xi = A_i(\underset{\sim}{C}, \xi)$ (i chosen so that $\xi \in \Gamma_i$).

Chapter 5 :- Recursion Induction and Buffers

In this chapter we will see how many of the ideas introduced in chapter 2 extend naturally to the non-deterministic model M. We will then make a fairly extensive analysis of one particular predicate, namely "is a buffer", and its relationship with the pipe operator "$\gg$".

The first requirement for this extension is a class of restriction operators $\{\upharpoonright n \mid n \in N\}$. It would be possible to make $A \upharpoonright n$ deterministically "die" after n steps (as was done in 2.6). It is however more in the spirit of our partial order on M to have $A \upharpoonright n$ dissolve into CHAOS after n steps. With this behaviour $A \upharpoonright n$ will in some sense be the minimal process which models A up to stage n whereas in the definition suggested first $A \upharpoonright n$ would be one of many maximal such processes. We will thus normally interpret $\upharpoonright n$ as it is defined below.

5.1 $A \upharpoonright n = \{(w,X) \mid (w,X) \in A \;\&\; |w| < n\} \cup \{(wv,X) \mid w \in dom(A) \;\&\; |w| = n\}$

Below is a summary of a few simple results about these operators.

5.2 Lemma

a) $A \upharpoonright o = CHAOS$ for all $A \in M$.

b) $(A \upharpoonright n) \upharpoonright m = A \upharpoonright \min(m,n)$

c) $A \upharpoonright n \sqsubseteq A \upharpoonright n{+}1 \sqsubseteq A$

d) $\bigsqcup_{n=0}^{\infty} (A \upharpoonright n) = A$

e) $\forall (w,X).\exists n.\forall A.\ (w,X) \in A \Leftrightarrow (w,X) \in A \upharpoonright n$

We extend the definition of $\upharpoonright n$ to vectors of processes in M^{Λ} as follows:

5.3 $(\underset{\sim}{A} \upharpoonright n)_{\lambda} = (A_{\lambda}) \upharpoonright n$.

The definitions of constructive and non-destructive functions and of continuous predicates are exactly the same as before.

5.4 a) A function $F: M^{\Lambda} \to M^{\Gamma}$ is said to be <u>constructive</u> if it satisfies $\forall n.\forall \underset{\sim}{A} \in M^{\Lambda}.\ F(\underset{\sim}{A}) \upharpoonright n{+}1 = F(\underset{\sim}{A} \upharpoonright n) \upharpoonright n{+}1$.

b) A function $F: M^{\Lambda} \to M^{\Gamma}$ is said to be <u>non-destructive</u> if it satisfies $\forall n.\forall \underset{\sim}{A} \in M^{\Lambda}.\ F(\underset{\sim}{A}) \upharpoonright n = F(\underset{\sim}{A} \upharpoonright n) \upharpoonright n$.

5.5 A predicate on M^{Λ} is said to be <u>continuous</u> if it satisfies $\forall \underset{\sim}{A} \in M^{\Lambda}.(\forall n.\exists \underset{\sim}{B}.R(\underset{\sim}{B}) \;\&\; (\underset{\sim}{A} \upharpoonright n = \underset{\sim}{B} \upharpoonright n)) \Rightarrow R(\underset{\sim}{A})$.

5.6 Lemma (analogue of 2.11)

a) If $F:M^{\wedge} \to M^{\Gamma}$ and $G:M^{\Gamma} \to M^{\Delta}$ are non-destructive then so is GoF.

b) If a function is constructive then it is non-destructive.

c) If one of $F:M^{\wedge} \to M^{\Gamma}$ and $G:M^{\Gamma} \to M^{\Delta}$ is constructive and the other is non-destructive then GoF is constructive.

d) If $F:M^{\wedge} \to M^{\wedge}$ is constructive then it has a unique fixpoint.

proof

The proof is identical to that of 2.11 since it only depends on the properties of $\upharpoonright n$, namely 2.8 (≡ 5.2), which hold in both models. The analysis required to show that an arbitrary monotonic function has a fixed point is more complicated on the complete partial order $M^{\wedge}$ than on the complete lattice $P^{\wedge}$.

5.7 Theorem

If $F:M^{\wedge} \to M^{\wedge}$ is a constructive function and if R is a continuous satisfiable predicate then the translation of rule 2.1 into the non-deterministic model is valid.

$$\text{(i.e } \quad (\forall \underset{\sim}{B}.\ R(\underset{\sim}{B}) \Rightarrow R(F(\underset{\sim}{B}))) \Rightarrow R(fix(F)) \quad)$$

Again the proof of this is identical to that of 2.14.

It is possible to develop in this model a similar calculus to that used in the deterministic model for proving functions constructive and predicates continuous and satisfiable.

5.8 Theorem

The following predicates are all continuous:

$R(\underset{\sim}{A}) \equiv$

(i) $A_\lambda = B$

(ii) $A_\lambda = A_\kappa$

(iii) $A_\lambda \sqsubseteq B$

(iv) $A_\lambda \sqsupseteq B$

(v) $A_\lambda \sqsupseteq A_\xi$

(vi) $A_\lambda \neq B$ if there is an upper bound on $\{|w| \mid (w,\emptyset) \in B\}$

(vii) "A_λ is deadlock-free" (in this model this is equivalent to $w \in (\Sigma - \{\surd\})^* \Rightarrow \Sigma \notin A_\lambda(w)$)

(viii) $(w,X) \in A_\lambda \Rightarrow p(w,X)$ if p is any predicate on $\Sigma^* \times \wp(\Sigma)$

(ix) $w \in dom(A_\lambda) \Rightarrow p_w((A_\lambda \underline{after}\ w)^o)$ if p_w are predicates on $\mathcal{P}(\Sigma)$

(x) $w \in dom(A_\lambda) \Rightarrow p_w(A_\lambda(w))$ if p_w are predicates on $\mathcal{P}(\mathcal{P}(\Sigma))$

(xi) $R_1(F(\underset{\sim}{A}))$ if $\exists g:N \to N$ s.t. $\forall \underset{\sim}{B}.\forall n.F(\underset{\sim}{B})\lceil n = F(\underset{\sim}{B}\lceil g(n))\lceil n$
($F:M^\Lambda \to M^\Gamma$ monotonic and R_1 a predicate on M^Γ)

(xii) $F(\underset{\sim}{A}) \sqsubseteq \underset{\sim}{B}$ for any continuous $F:M^\Lambda \to M^\Gamma$

(xiii) $\bigwedge_{\gamma\in\Gamma} R_\gamma(\underset{\sim}{A})$ for any set Γ

(xiv) $R_1(\underset{\sim}{A}) \vee R_2(\underset{\sim}{A})$

where B is any constant process R_1, R_2 and R_γ are all continuous and $\lambda \in \Lambda$.

The proofs of these results are similar to the proofs of 2.18, for example:

(v) $\neg(A_\lambda \sqsupseteq A_\xi) \Rightarrow \exists(w,X) \in A_\lambda - A_\xi$
If $n = |w|+1$ then clearly $\underset{\sim}{A}\lceil n = \underset{\sim}{B}\lceil n$ implies $(w,X) \in B_\lambda - B_\xi$, so $\neg(B_\lambda \sqsupseteq B_\xi)$.

5.9 Theorem

a) Each of the combinators "a → A", ‖ , ▯ , ";" , or, "a.A" defines a non-destructive function of its variable(s).

b) If a function $F:M^\Gamma \times M^\Lambda \to M^\Lambda$ is constructive (non-destructive) in its first variable* then $G:M^\Gamma \to M^\Lambda$ defined $G(\underset{\sim}{A}) = fix(\lambda\underset{\sim}{B}.F(\underset{\sim}{A},\underset{\sim}{B}))$ is constructive (non-destructive).

c) Suppose that the function $F:M^\Lambda \to M^\Lambda$ is such that each component of $F(\underset{\sim}{A})$ is a syntactic expression involving only process variables, expressions independent of all process variables, the combinators a → B, a?x:T → B(x), ?x:T → B(x), B;C, B▯C, a.B and $(B_X\|_Y C)$, and iterated recursions which bind all instances of process variables which are not A_λs. Then provided that every free recursive call of an A_λ is guarded directly or indirectly the function F is constructive.

proof

These are all similar to the correponding results in the deterministic model (namely 2.15, 2.36 & 2.37), the only difference being in the analysis of the individual combinators. The only one to present a slight difficulty is ";" because of the way it "hides" the "✓".

(* and non-destructive in its second variable)

Recall the definition of sequential composition:

$$A;B = \{(w,X) \mid w/\{\surd\} = w \ \& \ (w, X\ \{\surd\}) \in A\}$$
$$\{(wv,X) \mid w/\{\surd\} = w \ \& \ w\langle\surd\rangle \in dom(A) \ \& \ (v,X) \in B\}$$

We would like to show that $(A\restriction n;B\restriction n)\restriction n = (A;B)\restriction n$.

It is clearly sufficient to show that

a) $(A\restriction n;B\restriction n) \cap \{(w,X) \mid |w|<n\} \Rightarrow A;B \cap \{(w,X) \mid |w|<n\}$

b) $dom(A\restriction n;B\restriction n) \cap \{w \mid |w|=n\} = dom(A;B) \cap \{w \mid |w|=n\}$

In each case the containment of the R.H.S. within the L.H.S. follows from monotonicity.

$(w,X) \in (A\restriction n;B\restriction n) \cap \{(w,X) \mid |w|<n\}$

$\Rightarrow$ <u>either</u> $(w,X) \in \{(w,X) \mid w/\{\surd\} = w \ \& \ (w,X\ \{\surd\}) \in A\restriction n\}$

$\Rightarrow$ $(w,X) \in \{(w,X) \mid w/\{\surd\} = w \ \& \ (w,X\ \{\surd\}) \in A\}$ as $|w|<n$

$\Rightarrow$ $(w,X) \in A;B$

<u>or</u> $(w,X) = (uv,X)$ for some u,v s.t. $u/\{\surd\} = u$

$u\langle\surd\rangle \in dom(A\restriction n)$ & $(v,X) \in B\restriction n$

now $|v|<n$ so $(v,X) \in B$

and $|u\langle\surd\rangle| \leq n$ so $u\langle\surd\rangle \in dom(A)$

thus $(uv,X) \in A;B$.

$w \in dom(A\restriction n;B\restriction n) \cap \{w \mid |w|=n\}$

$\Rightarrow$ <u>either</u> $w \in \{w \in dom(A\restriction n) \mid w/\{\surd\} = w\}$

$\Rightarrow$ $w \in \{w \in dom(A) \mid w/\{\surd\} = w\}$ as $|w| = n$

<u>or</u> $w \in \{uv \mid u/\{\surd\} = u \ \& \ u\langle\surd\rangle \in dom(A\restriction n) \ \& \ v \in dom(B)\}$

now either $|u\langle\surd\rangle| \leq n$ & $|v| \leq n$, in which case $uv \in dom(A;B)$ as required,

or $|u\langle\surd\rangle| = n+1$ and $v = \langle\rangle$ in which case $w = u$ and $w \in$ (first clause of A;B).

The proofs of the non-destructive nature of the other operators require similar tedious analysis.

Other parts of the deterministic theory to be valid in this model are the extensions for partial predicates and constructiveness relative to partial orders (and also the simple 2.21). In each case it will be seen that the proofs depend only on properties shared by the two models and the classes of functions and operators defined on them. These results are not stated as formal theorems in this model but will be used wherever necessary. It is noteworthy that all the examples of program-proving in

chapter 2 are equally valid over the non-deterministic model M. With the exception of the buffers example, where the predicate needs translation to make sense in M, all the proofs can equally well be read as proofs in the non-deterministic model. This is because the various combinators have much the same properties (commutativity, distributivity, etc.) in both models (with a few exceptions which are not used in any of these proofs).
The analysis needed to justify the constructiveness of the master/slave operator in 2.39 over M will be found in the next chapter.

The only part of the theory in chapter 2 which does not seem to transfer effectively to M is the work on unique fixed points and strongly continuous predicates. The main reason for this is that M has no "top" element. It is still possible to define a strongly continuous predicate and show that such predicates are continuous w.r.t. every normal restriction operator class. This is useful, since it saves work when we wish to use different operators.

5.10 Define a sequence of processes to be <u>convergent</u> if it satisfies $\limsup(A_i) = \liminf(A_i)$ where these operators act setwise on M, and if this limit is in M.

$$\liminf(A_i) = \bigcup_{j=1}^{\infty}(\bigcap_{i=j}^{\infty} A_i), \quad \limsup(A_i) = \bigcap_{j=1}^{\infty}(\bigcup_{i=j}^{\infty} A_i)$$

(Observe that in general $\liminf(A_i)$ and $\limsup(A_i)$ are not necessarily elements of M.)

Define a predicate on M to be <u>strongly continuous</u> if it satisfies "$\langle A_i \mid i \in N \rangle$ convergent with limit A and $\forall i.R(A_i)$ implies $R(A)$".

5.11 Define a class $\{\upharpoonright n \mid n \in N\}$ of restriction operators on M to be <u>normal</u> if they satisfy:

a) $\forall A.\ \forall B.\ A\upharpoonright o = B\upharpoonright o$

b) $\forall A.\ \forall n.\forall m.\ (A\upharpoonright n)\upharpoonright m = A\upharpoonright \min(m,n)$

c) $\forall (w,X).\exists n.\forall A.\ (w,X) \in A \Leftrightarrow w \in A\upharpoonright n$

(This is just a translation of 2.49. Observe that we have already shown that the canonical class of restriction operators is normal (5.2).)

5.12 Theorem

If a predicate R is strongly continuous then it is continuous with respect to every normal class of restriction operators.

The proof of this is the same as that of 2.50.

5.13 Theorem

The following predicates are all strongly continuous.

$R(A) \equiv$ (i) $A = B$

(ii) $A \sqsupseteq B$

(iii) $A \sqsubseteq B$

(iv) $(w,X) \in A \Rightarrow p(w,X)$ where p is any predicate on $\Sigma^* \times \mathcal{P}(\Sigma)$

(v) "A is free of deadlock"

(vi) $F(A) \sqsubseteq B$ if F is continuous

(vii) $\bigwedge_{\gamma \in \Gamma} R_\gamma(A)$

(viii) $R_1(A) \vee R_2(A)$

where B is any constant process and R_1, R_2 and R_γ are all strongly continuous.

Note that freedom from deadlock is strongly continuous in this model, whereas it is not in the deterministic model. The reason for this is that absence of deadlock is represented in M by "$\Sigma \notin A(w)$ for any $w \in dom(A)$ s.t. w has not already terminated successfully". Thus, for any such w, if none of a sequence $\langle A_i \mid i \in N \rangle$ deadlocks after w then $\forall i.(w, \Sigma) \notin A_i$. Therefore $(w, \Sigma) \notin \limsup(A_i)$ so $\lim(A_i)$ cannot deadlock after w either.

One consequence of this is that the function $F(A) = a.A$ cannot be constructive relative to any normal class of restriction operators (see 2.41). (a.A can be made constructive in P, by defining $A\upharpoonright n = A \cap \Sigma_n^*$, where Σ_n contains all those elements of Σ with less than n "components".)

Henceforth "$\upharpoonright n$" will always be the canonical operator (5.1, 5.3) unless specifically stated otherwise.

We will now prove the two extensions (stated in 2.33 and 2.34 for P) whose proofs were delayed until this chapter.

5.14 Theorem

If $A \in \mathcal{P}(M^\wedge)$ define $A \upharpoonright n = \{\underset{\sim}{B} \upharpoonright n \mid \underset{\sim}{B} \in A\}$.
Define a function $F: \mathcal{P}(M^\wedge) \to \mathcal{P}(M^\wedge)$ to be constructive if it satisfies $\forall A. \forall n.\ F(A) \upharpoonright n+1 = F(A \upharpoonright n) \upharpoonright n+1$.
Suppose the predicate R on $M^\wedge$ is satisfiable and continuous, then we have

$(\forall A'.(\forall \underset{\sim}{B}.(\underset{\sim}{B} \in A' \Rightarrow R(\underset{\sim}{B})))) \Rightarrow (\forall \underset{\sim}{B}.(\underset{\sim}{B} \in F(A') \Rightarrow R(\underset{\sim}{B})))) \ \&\ (A \subseteq F(A))$
implies $(\forall \underset{\sim}{B}. \underset{\sim}{B} \in A \Rightarrow R(\underset{\sim}{B}))$.

proof

If A is empty then the result is trivial, so we may assume that it is not. Clearly for all non-empty sets A' we have $A' \upharpoonright o = \{CHAOS^\wedge\}$ so in particular $A_o \upharpoonright o = A \upharpoonright o$, where A_o is the set of processes which satisfy R (non-empty by satisfiability).
Suppose that $(\forall \underset{\sim}{B}. \underset{\sim}{B} \in A \Rightarrow R(\underset{\sim}{B}))$ does not hold. Then there must be some $n \in N$ which is maximal with respect to $\exists A_n . A_n \upharpoonright n = A \upharpoonright n \ \&\ (\forall \underset{\sim}{B}. \underset{\sim}{B} \in A_n \Rightarrow R(\underset{\sim}{B}))$ (by continuity of R).
Then $F(A) \upharpoonright n+1 = F(A \upharpoonright n) \upharpoonright n+1 = F(A_n \upharpoonright n) \upharpoonright n+1 = F(A_n) \upharpoonright n+1$ by constructiveness of F.
Let $C = F(A_n)$. By our assumptions we must have $(\forall \underset{\sim}{B}. \underset{\sim}{B} \in C \Rightarrow R(\underset{\sim}{B}))$.
Also $A \subseteq F(A)$, so $A \upharpoonright n+1 \subseteq F(A) \upharpoonright n+1 = C \upharpoonright n+1$.
Thus $\forall \underset{\sim}{B} \in A.\ \exists \underset{\sim}{B}'_{\underset{\sim}{B}} \in C. \underset{\sim}{B} \upharpoonright n+1 = \underset{\sim}{B}'_{\underset{\sim}{B}} \upharpoonright n+1$.
Let $A_{n+1} = \{\underset{\sim}{B}'_{\underset{\sim}{B}} \mid \underset{\sim}{B} \in A\}$. By definition $A_{n+1} \upharpoonright n+1 = A \upharpoonright n+1$ and $(\forall \underset{\sim}{B}. \underset{\sim}{B} \in A_{n+1} \Rightarrow R(\underset{\sim}{B}))$ since $A_{n+1} \subseteq C$. This contradicts our choice of n, so $(\forall \underset{\sim}{B}. \underset{\sim}{B} \in A \Rightarrow R(\underset{\sim}{B}))$ does hold as claimed.

5.15 Theorem

Suppose that $R_1, \ldots, R_n$ are predicates which are all continuous and satisfiable (but possibly not simultaneously satisfiable). Suppose further that $F: M^\wedge \to M^\wedge$ is a function which, for each $i \in \{1,2,\ldots n\}$ can be written in the form $F^*_i \circ D$, for some $F:(M^\wedge)^n \to M^\wedge$ which is constructive and where $D(\underset{\sim}{A}) = (\underset{\sim}{A}, \underset{\sim}{A}, \ldots, \underset{\sim}{A})$ for $\underset{\sim}{A} \in M^\wedge$.
Then if for each i we can prove for all $\underset{\sim}{A}_1, \ldots, \underset{\sim}{A}_n \in M^\wedge$

$R_1(\underset{\sim}{A}_1)\ \&\ \ldots\ \&\ R_n(\underset{\sim}{A}_n) \Rightarrow R_i(F^*_i(\underset{\sim}{A}_1, \ldots, \underset{\sim}{A}_n))$

we can infer $\forall i. R_i(\text{fix}(F))$.

proof

This is just an application of 5.7. Define $G:(M^{\wedge})^n \to (M^{\wedge})^n$ by $G(\underset{\sim}{A}_1,\ldots \underset{\sim}{A}_n) = (F_1^*(\underset{\sim}{A}_1,\ldots,\underset{\sim}{A}_n),\ldots,F_n^*(\underset{\sim}{A}_1,\ldots,\underset{\sim}{A}_n))$.
Define the compound predicate R* on $(M^{\wedge})^n$ by
$R^*(\underset{\sim}{A}_1,\ldots \underset{\sim}{A}_n) \equiv R_1(\underset{\sim}{A}_1) \;\&\; R_2(\underset{\sim}{A}_2) \;\&\ldots\&\; R_n(\underset{\sim}{A}_n)$.
G is constructive since each of the F_i^* is.
R* is continuous since each of the R* is.
R* is satisfiable, by $(\underset{\sim}{A}_1^*,\ldots,\underset{\sim}{A}_n^*)$ where $\underset{\sim}{A}_i^*$ satisfies R_i.
$\underset{\sim}{A} \in (M^{\wedge})^n \Rightarrow (R^*(\underset{\sim}{A}) \Rightarrow R^*(G(\underset{\sim}{A}))$ by the assumptions in the statement of the theorem.
We can therefore infer R*(fix(G)) by 5.7.

Claim that fix(G) = (fix(F),fix(F),...,fix(F)) (= $\underset{\sim}{C}$, say).

G has a unique fixed point by 5.6 (d), but $G(\underset{\sim}{C}) = \underset{\sim}{C}$ since $F_i^*(\underset{\sim}{C}) = F_i^*(D(\text{fix}(F))) = F(\text{fix}(F)) = \text{fix}(F)$.
Thus $\underset{\sim}{C}$ = fix(G) as claimed, and so we can infer $R^*(\underset{\sim}{C})$, which is the result we desired.

As was said in chapter 2 this result will allow us to prove several results of fix(F) by mutual induction, even though we may not know them to be consistent. The proof used must only assume one of the said properties of each recursive call in the proof of each hypothesis. It is however permissible to assume different properties of different calls of the same process, and to assume different properties of the same call provided that these assumptions are in the proofs of different hypotheses.

An example of the use of this rule will be found in 6. , and an example of rule 5.14 in 5.27.

We will now turn our attention to the detailed examination of a specific predicate, namely "is a buffer". This, in addition to being a useful exercise in demonstrating the use of our techniques, is also a useful predicate to have knowledge of. This is because we wish to prove either this property or a very similar one of such processes as operating systems, communication channels, etc.

By the predicate Buff = "is a buffer" we would like not only to be able to prove partial correctness, but also total correctness as was done in 2.20.

We therefore take as our definition of a buffer the following reworking of the predicate used in 2.20.

5.16 $\mathrm{Buff}(B) \equiv$

$$\begin{aligned} &(i) \quad w \in \mathrm{dom}(B) \Rightarrow w \in (?T \cup !T)^* \quad \& \quad \mathrm{ins}(w) \geq \mathrm{outs}(w) \\ \& \; &(ii) \quad (w \in \mathrm{dom}(B) \;\&\; \mathrm{ins}(w) = \mathrm{outs}(w)) \Rightarrow B(w) = \{X \mid X \cap ?T = \emptyset\} \\ \& \; &(iii) \quad (w \in \mathrm{dom}(B) \;\&\; \mathrm{ins}(w) > \mathrm{outs}(w)) \Rightarrow !T \notin B(w) \end{aligned}$$

The motivation for these conditions (i),(ii) & (iii) is the same as that in 2.20.

In almost exactly the same fashion as 2.20 we could prove that the above three conditions are simultaneously satisfied by the canonical one-place buffer $B \Leftarrow ?x{:}T \rightarrow (!x \rightarrow B)$. That Buff is continuous follows immediately from 5.8. It is in fact strongly continuous, since (ii) and (iii) above are easily rewritten in the form 5.13 (iv).

As we will see shortly, the theory of buffers links closely with the theory of the pipe operator "≫". This can be modelled more reasonably in this model since the non-determinism of the hiding can now be expressed. Its formal definition is as follows:

5.17 $(A \gg B) = (\mathrm{strip}!(A)\,{}_{T\cup ?T}\|_{T\cup !T}\,\mathrm{strip}?(B))/T$

$$\text{where} \quad \mathrm{strip}a(A) = \{(\mathrm{strip}a(w), \mathrm{strip}a(X - T) \cup Y) \mid (w,X) \in A \;\&\; Y \subseteq a.T\}.$$

The definition of strip on strings is the same as in chapter 2 and that on sets the natural extension of that on Σ.

Because of the hiding used in its definition, we will assume that the set T of basic values for communication is finite whenever "≫" is used. It is also necessary to assume that $?T = \{?.x \mid x \in T\}$ and $!T = \{!.x \mid x \in T\}$ are both disjoint from T.

This definition is in some sense only reasonable if the processes A and B only communicate in the alphabet $!T \cup ?T$, for otherwise the "strip" operator identifies events which should not be identified. We will therefore ensure that all processes which we expect to to act sensibly when combined by "≫" satisfy this.

We will now spend a little time developing a calculus for "$\gg$" before we start to apply it to buffers.

5.18 Lemma

"$\gg$" is a well-defined continuous operator in $M \times M \to M$ providing that T is finite. Furthermore we have the following criterion for membership of $(A \gg B)$:

$(w,W) \in (A \gg B)$ if and only if

either there is some $w' \leq w$ such that

$\{t \mid (t\upharpoonright(T \cup ?T) \in \text{strip!}(\text{dom}(A))) \;\&\; (t\upharpoonright(!T \cup ?T) = w') \;\&$
$(t\upharpoonright(T \cup !T) \in \text{strip?}(\text{dom}(B)))\}$ is infinite

or there are some $(u,U) \in A$ and $(v,V) \in B$ such that

$(W \cap (!T \cup ?T)) \cup T = \text{strip!}(U - T) \cup \text{strip?}(V - T)$ and
$t\upharpoonright(T \cup ?T) = \text{strip!}(u) \;\&\; t\upharpoonright(T \cup !T) = \text{strip?}(v) \;\&$
$t\upharpoonright(?T \cup !T) = w$.

(In this last case say that $((u,U),(v,V))$ is a derivation for (w,W) in $(A \gg B)$ or that $((u,U),(v,V)) \Leftrightarrow (w,W)$ in $(A \gg B)$.)

proof

The first part of this follows from the same result of the various operators from which "$\gg$" is defined. These results are already known except for the "strip" operator, which is easy to verify.

The second part comes straight from the definition of the operator, using the following result on the parallel operator.

$$(w,W) \in (A_X\|_Y B) \Leftrightarrow \exists (u,U) \in A.\ \exists (v,V) \in B \text{ s.t.}$$
$$w\upharpoonright(X \cup Y) = w \;\&\; w\upharpoonright X = u \;\&\; w\upharpoonright Y = v \;\&$$
$$W \cap (X \cup Y) = (U \cap X) \cup (V \cap Y)$$

If the either clause in the above definition is satisfied by some $(w,W) \in (A \gg B)$ then we say that $(A \gg B)$ contains infinite internal chatter.

5.19 Theorem

If $\text{dom}(A \text{ or } B \text{ or } C) \subseteq (!T \cup ?T)^*$ and both $(A \gg B)$ and $(B \gg C)$ are free of infinite internal chatter then the associative law holds, viz

$$((A \gg B) \gg C) = (A \gg (B \gg C)) \quad .$$

proof

We use the following lemma, which will be proved in an appendix to this chapter (5.35).

If A is free of infinite X-chatter and $X \cap Z = \emptyset$ then

$$(A/X_{Y}\|_{Z}B) = (A_{X\cup Y}\|_{Z}B)/X \quad .$$

"$\gg$" works by identifying the outputs of its first variable with the inputs of its second and hiding the resulting internal communication. It is possible to change the method of identification without changing the result. Instead of transforming the joint communications to T we can transform them to a.T for "a" any suitable label. Define a replacement operator repab (for replacing label "a" by label "b") as follows for any a,b such that $a \neq b$ and $a.T \cap b.T = \emptyset$.

For $c \in \Sigma$ $\quad$ repab(c) = b.x $\quad$ if c=a.x for any $x \in T$

= c $\quad$ otherwise

For $w \in \Sigma^*$ and $X \in \mathcal{P}(\Sigma)$ repab(w) and repab(X) are the natural elementwise extensions of the above.

For $A \in M$ $\quad$ repab(A) =

$$\{(\text{repab}(w), \text{repab}(X - b.T) \cup Y) \mid (w,X) \in A \ \& \ Y \subseteq a.\Sigma\}$$

With this definition it is quite easy to show that provided "a" is chosen so that a.T is disjoint from both ?T and !T and A,B satisfy dom(A <u>or</u> B) $\subseteq$ (!T $\cup$?T)*
then $(A \gg B) = (\text{rep!a}(A)_{a.T\cup ?T}\|_{a.T\cup !T}\text{rep?a}(B))/a.T$.

Thus under the conditions of the theorem, if "a" and "b" are chosen so that !T, ?T, a.T and b.T are all disjoint (if they do not exist then enlarge Σ) then $((A \gg B) \gg C)$ =

$(\text{rep!b}((\text{rep!a}(A)_{X}\|_{Y}\text{rep?a}(B))/a.T)_{Z}\|_{W}\text{rep?b}(C))/b.T$

where X = ?T$\cup$a.T, Y = !T$\cup$a.T, Z = ?T$\cup$b.T, W = !T$\cup$b.T

= $((\text{rep!a}(A)_{X}\|_{V}\text{rep!b}(\text{rep?a}(B)))/a.T_{Z}\|_{W}\text{rep?b}(C))/b.T$

where V = a.T $\cup$ b.T $\quad$ (by various properties of "rep")

= $((\text{rep!a}(A)_{X}\|_{V}\text{rep!b}(\text{rep}(\text{?a}(B)))_{Z\cup a.T}\|_{W}\text{rep?b}(C))/(a.T \cup b.T)$

(by 4.7 and the lemma at the head of the page)

A symmetric expression can be derived for $(A \gg (B \gg C))$ using "a" again for the first channel and "b" for the second. These two are then equal by the associative law of $\|$ and the commutativity of rep!b and rep?a .

The next result gives us a useful technique for proving that processes of the form $(A \gg B)$ are free of infinite chatter (a desirable result in its own right as well as in its use in proving associativity).

5.20 Theorem

Each of the two predicates:

$P_1(A) \equiv \neg\exists\, w_1 < w_2 < w_3 < \ldots \in dom(A)$ s.t. $\forall i. w_i \upharpoonright ?T = w_1 \upharpoonright ?T$

(A cannot output for ever without inputting)

$P_2(A) \equiv \neg\exists\, w_1 < w_2 < w_3 < \ldots \in dom(A)$ s.t. $\forall i. w_i \upharpoonright !T = w_1 \upharpoonright !T$

(A cannot input for ever without outputting)

satisfies $(i \in \{1,2\})$ (for $dom(A$ or $B) \subseteq (?T \cup !T)^*$)

$P_i(A)$ & $P_i(B) \Rightarrow (P_i(A \gg B)$ & $(A \gg B)$ is free of infinite internal chatter) .

proof

We will prove the result for i=1, the proof for i=2 being very similar.

Suppose That P_1 holds of both A & B. We will prove first that $(A \gg B)$ is free of infinite chatter.

If not there is some minimal $w \in dom(A \gg B)$ such that

$\{t \mid (t \upharpoonright (T \cup ?T) \in strip!(dom(A)))$ & $(t \upharpoonright (!T \cup T) \in strip?(dom(B)))$ & $(t \upharpoonright (!T \cup T) = w)\}$ is infinite.

It is easy to show (for the same reasons as 4.6) that the number of minimal elements of this set is finite. König's lemma then gives us that it contains an infinite ascending chain $t_1 < t_2 < t_3 < \ldots$ (because T is finite).

There must therefore be an infinite sequence $\langle u_i \mid i \in N \rangle$ in dom(A) such that $strip!(u_i) = t_i \upharpoonright (T\ ?T)$.

Since $dom(A) \subseteq (!T \cup ?T)^*$ the u_i must be an ascending sequence and $u_i \upharpoonright ?T = (strip!(u_i)) \upharpoonright ?T = t_i \upharpoonright ?T = w \upharpoonright ?T$. Hence the u_i contradict $P_1(A)$. $(A \gg B)$ is thus free of infinite internal chatter as claimed.

In proving that P_1 holds of $(A \gg B)$ we may thus assume that all elements of it arise from the "or" clause in 5.18.

Suppose that $w_1 < w_2 < w_3 < \ldots$ is an infinite sequence in $dom(A \gg B)$ with $w_i \upharpoonright ?T$ constant.

For each w_i there must be some $(u_i,U) \in A$ and $(v_i,V) \in B$ such that $((u_i,U),(v_i,V)) \Leftrightarrow (w_i,\emptyset)$ in $(A \gg B)$.

These must satisfy the relations:

$$u_i\upharpoonright ?T = w_i\upharpoonright ?T = w_1\upharpoonright ?T \quad \text{(all the same)}$$
$$\text{strip!}(u_i\upharpoonright !T) = \text{strip?}(v_i\upharpoonright ?T) \; (= s_i \text{ , say})$$
$$v_i\upharpoonright !T = w_i\upharpoonright !T > v_{i-1}\upharpoonright !T \quad \text{(all different)} .$$

It is easy to see from this that either there are infinitely many s_i or there is some i such that $\{j \mid s_i = s_j\}$ is infinite. The first case contradicts $P_1(A)$ for then the tree of $u \in \text{dom}(A)$ s.t. $u\upharpoonright ?T = u_1\upharpoonright ?T$ is infinite.
Claim this would contradict $P_1(A)$ for any $u' \in ?T^*$ (corresponding to $u_1\upharpoonright ?T$ in the above). If $|u'| = 0$ then $u' = \langle\rangle$ so the tree has one minimal element, and is finite branching as T is finite, and so has an infinite path which contradicts $P_1(A)$.
Assume true of all shorter u". If the tree has infinitely many minimal elements for u' ($=u''\langle a\rangle$, say) then each of these is of the form $u\langle a\rangle$. Thus the tree for u" is infinite, contradicting $P_1(A)$ by induction. If the tree has finitely many minimal elements then by the same argument as above it contains an infinite path which contradicts $P_1(A)$. This completes the induction, and so in particular the infinitude of $\{u \in \text{dom}(A) \mid u\upharpoonright ?T = u_1\upharpoonright ?T\}$ contradicts $P_1(A)$ as claimed.
The same argument shows that the second case above contradicts $P_1(B)$, for then $\exists v \in \text{dom}(B).\{i \mid v\upharpoonright ?T = v_i\upharpoonright ?T\}$ is infinite.
Hence there can be no such sequence $w_1 < w_2 < w_3 < \ldots.$, so $P_1(A \gg B)$ holds as claimed.

Note that under the conditions of the theorem we in addition have $\text{dom}(A \gg B) \subseteq (!T \cup ?T)^*$, this result being a consequence of the lack of infinite chatter and 5.18.

The two predicates P_1 and P_2 are both discontinuous, since at no finite time can their negations be decided. There are however a large class of continuous and strongly continuous predicates which imply one or other of them. For example Buff $\Rightarrow P_1$ (by line (i) of 5.16).

5.21 Corollary .

If for $i \in \{1,2\}$ each of A_1, A_2, ..., A_k satisfies P_i then we may bracket $A_1 \gg A_2 \gg \ldots \gg A_k$ however we please and get the same answer. Furthermore the result is free of infinite internal chatter and satisfies P_i.

The proof of this is an easy induction on k using 5.19 and 5.20.

Note that if A & B both satisfy P_i then most of the normal combinators applied to A (& B) produce a process which satisfies P_i, for example "$a \rightarrow A$" ($a \in (?T \cup !T)$), "$?x:T \rightarrow A$", "$A \Box B$". We will therefore be quite informal in the use of 5.20 & 5.21 in proofs, not always justifying their application to a particular set of processes if they are known to be justified for similar ones. For example if A,B,C are buffers then

$$((A \gg (b \rightarrow B)) \gg C) = (A \gg ((b \rightarrow B) \gg C))$$

(The inclusion of such details would clutter up the proofs, so they are omitted on the basis that we could insert them if challenged.)

The following lemma (which will be used freely and informally in proofs) allows us to do basic "handle turning" in proofs involving "$\gg$".

5.22 Lemma

a) $((!y \rightarrow A) \gg (?x:T \rightarrow B(x))) = (A \gg B(y)) \quad (y \in T)$

b) $((?x:T \rightarrow A(x)) \gg B) = ?x:T \rightarrow (A(x) \gg B) \quad$ if $B^O \subseteq ?T$

c) $(A \gg (!y \rightarrow B)) = !y \rightarrow (A \gg B) \quad$ if $A^O \subseteq !T$ & $y \in T$

d) $((?x:T \rightarrow A(x)) \gg (!y \rightarrow B)) = ?x:T \rightarrow (A(x) \gg (!y \rightarrow B))$
$\qquad \Box\, !y \rightarrow ((?x:T \rightarrow A(x)) \gg B)$

e) $((?x:T \rightarrow A(x)) \gg C \gg (!y \rightarrow B)) =$
$(?x:T \rightarrow (A(x) \gg C \gg (!y \rightarrow B))) \Box (!y \rightarrow ((?x:T \rightarrow A(x)) \gg C \gg B))$
provided that the associative law holds ($y \in T$)

f) $(A \text{ or } B \gg C \text{ or } D) = (A \gg C) \text{ or } (A \gg D) \text{ or } (B \gg C) \text{ or } (B \gg D)$

g) If $A^O \cup C^O \subseteq ?T$ and $B^O \cup D^O \subseteq !T$ then let
$E = (\text{strip}!(B)_{T \cup ?T} \|_{T \cup !T} \text{strip}?(C))$.
If $\Sigma \notin E(\langle\rangle)$ then $((A \Box B) \gg (C \Box D)) \sqsubseteq (B \gg C)$.
(A lower bound can be obtained from (f) by monotonicity.)

provided that the domains of all processes $\subseteq (?T \cup !T)^*$.

The proofs of 5.22 are all tedious manipulations using 5.18 and the definitions of the various operators.

We are now in a position to apply our knowledge to the study of the relationship between pipes and buffers.

5.23 Theorem

If any two of A,B & $(A \gg B)$ are buffers then so is the third (for any $A,B \in M$ s.t. dom(A or B) $\subseteq (?T \cup !T)^*$).

proof

Observe that in each case there can be no infinite chatter in $(A \gg B)$, either because buffers satisfy P_1 or because no process which contains infinite chatter can be a buffer.

Thus in each case we can restrict attention to the "or" case of 5.18.

We will examine only the "A & B buffers imply $(A \gg B)$ is a buffer" case in detail here. The proofs of the other two cases are similar in spirit and equally tedious.

Suppose Buff(A) & Buff(B).

To prove Buff$(A \gg B)$ we will prove the three conditions 5.16 (i),(ii) & (iii) in turn.

(i) Suppose $w \in \text{dom}(A \gg B)$. $w \in (!T \cup ?T)^*$ follows by the absence of infinite chatter.

There must be some $(u,U) \in A$ & $(v,V) \in B$ such that

$$((u,U),(v,V)) \Leftrightarrow (w,\emptyset) \text{ in } (A \gg B) .$$

But then
$$\begin{aligned} \text{ins}(w) &= \text{ins}(u) && \text{by definition of ins} \\ &\geq \text{outs}(u) && \text{as A is a buffer} \\ &\geq \text{ins}(v) && \text{as ins}(v) = \text{outs}(u) \\ &\geq \text{outs}(v) && \text{as B is a buffer} \\ &\geq \text{outs}(w) && \text{as outs}(v) = \text{outs}(w) \end{aligned}$$

(For proving this condition in the other two cases we use the fact that all strings of the process in question must also be strings of $(A \gg B)$ as the one which is known to be a buffer must have all strings in $\{\langle ?x!x\rangle \mid x \in T\}^*$ in its domain.)

(ii) Suppose $w \in \text{dom}(A \gg B)$ and that ins(w) = outs(w).

Since $(A \gg B)$ satisfies condition (i) we must have $((A \gg B) \text{ after } w)^0 \subseteq ?T$.

Hence $X \cap ?T = \emptyset \Rightarrow X \in (A \gg B)(w)$ (by 4.1 (c))
so $\{X \mid X \cap ?T = \emptyset\} \subseteq (A \gg B)(w)$. (*)
Suppose then that $X \in (A \gg B)(w)$. By 5.18 there exist $(u,U) \in A$, $(v,V) \in B$ such that $ins(w) = ins(u) \geqslant outs(u) = ins(v) \geqslant outs(v) = outs(w)$ (using Buff(A) & Buff(B)) and $X \cap (?T \cup !T) = (U \cap ?T) \cup (V \cap !T)$.
But then $ins(u) = outs(u)$ (since $ins(w) = outs(w)$) so $X \cap ?T = U \cap ?T = \emptyset$ (by line (ii) of Buff(A)).

Thus $(A \gg B)(w) \subseteq \{X \mid X \cap ?T = \emptyset\}$ which together with (*) above proves line (ii).

(iii) Suppose $w \in dom(A \gg B)$ and $ins(w) > outs(w)$.
We require to show that $!T \notin (A \gg B)(w)$.
Suppose to the contrary that $(w, !T) \in (A \gg B)$.
Then there are some $(u,U) \in A$ and $(v,V) \in B$ such that $ins(w) = ins(u) \geqslant outs(u) = ins(v) \geqslant outs(v) = outs(w)$ and $!T \cup T = ((T \cup ?T) \cap (strip!(U - T))) \cup ((T \cup !T) \cap (strip?(V - T)))$.
<u>Either</u> $ins(v) > outs(v)$, in which case $V \cap !T \neq !T$ by line (iii) of Buff(B), contradicting above relation.
<u>Or</u> $ins(v) = outs(v)$, in which case $ins(u) > outs(u)$.
Then $V \cap ?T = \emptyset$, so $strip?(V - T) \cap T = \emptyset$.
Also $U \cap !T \neq !T$, so $strip!(U - T) \cap T \neq \emptyset$.

But then we have $T = (T \cap strip!(U - T)) \cup (T \cap strip?(V - T))$ by the above, giving a contradiction.

Hence $!T \notin (A \gg B)(w)$ as required, completing the proof that $(A \gg B)$ is a buffer.

One corollary to this is that if $B^n = B \gg B \gg B \gg \ldots \gg B$ (n"B"s) where B is the canonical one place buffer of 5.16 then B^n is a buffer.

The following two results on nested buffers can be proved in much the same way as 5.24.

5.24 <u>Theorem</u>

a) If $A \gg B \gg C$ and B are both buffers then so is $A \gg C$.

b) If $A \gg C$ and B are buffers and A satisfies the "single inevitable output" condition SIO(A) (see over) <u>or</u> C satisfies $\forall w. \forall x. (w\langle ?x \rangle \in dom(C) \Rightarrow (\forall y. w\langle ?y \rangle \in dom(C)))$, then $A \gg B \gg C$ is a buffer.

$$SIO(A) = \forall w.\forall x.(w\langle !x\rangle \in dom(A) \Rightarrow (\forall v \in dom(A).v \geq w \Rightarrow (outs(v)\langle x\rangle \geq outs(w)\langle x\rangle)))$$

This condition is interpreted as "if at any stage it is possible for A to output x, then the next output of A <u>must</u> be x".

The need for one of these additional conditions to hold in case (b) is created by the possibility of C (in $(A \gg C)$) being able to selectively input from A. When this selective input is necessary for the correct behaviour of $(A \gg C)$ the insertion of B introduces the possibility of deadlock.

e.g. Let $A \Leftarrow ?x:T \rightarrow ((!a \rightarrow !x \rightarrow A) \square (!b \rightarrow \underline{stop}))$

$B \Leftarrow ?x:T \rightarrow !x \rightarrow B$

$C \Leftarrow ?a \rightarrow (?x:T \rightarrow !x \rightarrow C)$

(for a,b two distinct elements of T)

Then $(A \gg C)$ and B are buffers but $A \gg B \gg C$ is not as B, unlike C, cannot prevent A deadlocking.

5.25 <u>Example</u>

For each $n \geq 1$ define a canonical n-place buffer $B^n_{\langle\rangle}$ by mutual recursion on M^{Λ_n}, where $\Lambda_n = \{w \in T^* \mid |w| < n\}$.

$$B^n_{\langle\rangle} \Leftarrow ?x:T \rightarrow B^n_{\langle x\rangle}$$

$$B^n_{w\langle y\rangle} \Leftarrow (?x:T \rightarrow B^n_{\langle x\rangle w\langle y\rangle}) \square (!y \rightarrow B^n_w) \quad \text{if } |w| < n-1,\ y \in T$$

$$B^n_{w\langle y\rangle} \Leftarrow (!y \rightarrow B^n_w) \quad \text{if } |w| = n-1,\ y \in T$$

Claim that $B^n_{\langle\rangle} = B^n$ (as previously defined).

For $x \in T$ define $B_x = !x \rightarrow B$ (for B the one place buffer).

Observe that $(B_x \gg B) = (!x \rightarrow B) \gg (?x:T \rightarrow (!x \rightarrow B))$

$= (B \gg B_x)$ (by 5.22 (i)).

Define processes C^n_w for each $n \geq 1$ and $|w| \leq n$ as follows.

$$C^n_{\langle\rangle} = B^n, \quad C^1_{\langle a\rangle} = B_a, \quad C^{n+1}_{w\langle a\rangle} = (C^n_w \gg B_a)$$

Thus C^n_w is a string of one-place buffers, with the last ones containing the elements of w, for example $C^3_{\langle ab\rangle} = B \gg B_a \gg B_b$ and $C^3_{\langle abc\rangle} = B_a \gg B_b \gg B_c$.

Claim that $\forall w.\forall n.\ C^n_w = B^n_w$ (this clearly implies the above claim that $B^n_{\langle\rangle} = B^n$). We will prove this by induction on the definitions of the B_w^n.

For each n the predicate $R_n(\underset{\sim}{B}) \equiv \forall w \in \Lambda_n . B_w = C_w^n$ is clearly continuous and satisfiable. Also the recursion defining the B_w^n is clearly constructive. It thus suffices to show that $\forall \underset{\sim}{B}' \in M^{\Lambda_n} . R_n(\underset{\sim}{B}') \Rightarrow R_n(F_n(\underset{\sim}{B}'))$, where F_n is the function associated with the B_w^n recursion.

If n=1 then $R_1(\underset{\sim}{B}') \Rightarrow F_1(\underset{\sim}{B}')_{\langle\rangle} = ?x{:}T \to C^1_{\langle x\rangle}$

$= ?x{:}T \to B_x$ by definition of C_x^1

$= B$ by definition of B

$= C^1_{\langle\rangle}$

$F_1(\underset{\sim}{B}')_{\langle x\rangle} = !x \to C^1_{\langle\rangle}$

$= !x \to B$

$= B_x = C^1_{\langle x\rangle}$

$\Rightarrow R_1(F_1(\underset{\sim}{B}'))$

If n>2 then $R_n(\underset{\sim}{B}') \Rightarrow F_n(\underset{\sim}{B}')_{\langle\rangle} = ?x{:}T \to C^n_{\langle x\rangle}$

$= ?x{:}T \to (B^{n-1} \gg B_x)$

$= ?x{:}T \to (B_x \gg B^{n-1})$

(by repeated use of $(B \gg B_x) = (B_x \gg B)$)

$= (?x{:}T \to B_x) \gg (B^{n-1})$ (by 5.22)

$= B \gg B^{n-1} = C^n_{\langle\rangle}$

if $|w| < n-1$ then $F_n(\underset{\sim}{B}')_{w\langle y\rangle} = (?x{:}T \to C^n_{\langle x\rangle w\langle y\rangle}) \,\square\, (!y \to C^n_w)$

$= (?x{:}T \to (B_x \gg C_w^{n-2} \gg B_y)) \,\square\, (!y \to (B \gg C_w^{n-2} \gg B))$

(by repeated use of $(B \gg B_x) = (B_x \gg B)$)

$= ((?x{:}T \to B_x) \gg C_w^{n-2} \gg (!y \to B))$

(by 5.22 (e))

$= B \gg C_w^{n-2} \gg B_y$

$= C^n_{w\langle y\rangle}$

if $|w| = n-1$ then $F_n(\underset{\sim}{B}')_{w\langle y\rangle} = (!y \to C_w^n)$

$= !y \to (C_w^{n-1} \gg B)$

(by repeated use of $(B \gg B_x) = (B_x \gg B)$)

$= (C_w^{n-1} \gg (!y \to B))$ (it is easily shown by induction that $|v| = m \Rightarrow (C_v^m)^\circ \subseteq !T$)

$= C^n_{w\langle y\rangle}$

The case n=2 is almost identical with the $n > 2$ case, the only difference being in the middle case, where C_w^{n-2} is not now defined. There is of course no need for it to be present in this case (n=2) and we use 5.22 (d) instead of (e).

This completes the proof that $B_w^n = C_w^n$ for all n & w.

We have the following immediate corollaries to this result.

5.25.1 $B_{\langle\rangle}^n$ is a buffer for each $n \geqslant 1$.

5.25.2 $B_w^n \gg B_v^m = B_{wv}^{n+m}$ (by repeated application of $(B_x \gg B) = (B \gg B_x)$ to $C_w^n \gg C_v^m$)

We will henceforth identify the symbols $B_{\langle\rangle}^n$ and B^n as we are justified in doing by the preceding example.

Observe that in the above example we proved that the "large" process $B_{\langle\rangle}^n$ is a buffer by breaking it down into a lot of small parallel components. We will see this idea employed often from now on.

The next two results, the second of which is a type of inductive generalisation of the first, are both very useful in dealing with practical examples of proofs of correctness of buffers.

5.26 <u>Theorem</u>

Suppose that (for any set S) we are given two sets of processes $\{A_s \mid s \in S\}$ and $\{C_s \mid s \in S\}$ such that for all s $A_s \gg C_s$ is a buffer. Then for any function $g:T \to S$ the process $?x:T \to (A_{g(x)} \gg (!x \to C_{g(x)}))$ is a buffer.

(Note also that the above process is equal to

$$((?x:T \to !x \to A_{g(x)}) \gg (?x:T \to !x \to C_{g(x)})) \quad .)$$

We will find that this result is a corollary to the proof of the next result.

5.27 Theorem

Suppose that for any set S we are given two sets of processes $\{A_s \mid s \in S\}$ and $\{C_s \mid s \in S\}$ and a function $g: S \times T \to S$ such that for all $s \in S$

$$(A_s \gg B_s) = ?x{:}T \to (A_{g(s,x)} \gg (!x \to C_{g(s,x)})) \ .$$

Then for all $s \in S$ $A_s \gg C_s$ is a buffer.

proof

(This is an application of 5.14.)

Define $F: \mathcal{P}(M) \to \mathcal{P}(M)$ as follows:

$$A \in F(E) \iff$$
$$\text{(i)} \quad A^{0} = ?T \quad \text{and} \quad A(\langle\rangle) = \{X \mid X \cap ?T = \emptyset\}$$
$$\& \ \text{(ii)} \quad \forall t \in T.\ \exists B_t \in E \ \text{s.t.}\ \langle ?t\rangle w \in dom(A) \ \&\ w \in (?T)^* \Rightarrow w \in dom(B_t)$$
$$\&\ A(\langle ?t\rangle w) \subseteq \{X \mid !t \notin X\}$$
$$\langle ?t\rangle w \langle !s\rangle v \in dom(A) \ \&\ w \in (?T)^* \Rightarrow s = t \ \&$$
$$\&\ A(\langle ?t\rangle w \langle !s\rangle v) \subseteq B_t(wv)$$

5.27.1 lemma

F is constructive in the sense of 5.14.

5.27.2 lemma

$(\forall B \in A.Buff(B)) \Rightarrow (\forall B \in F(A).Buff(B))$

5.27.3 lemma

For any $g: S \times T \to S$ and $s \in S$

$$(?x{:}T \to (A_{g(s,x)} \gg (!x \to C_{g(s,x)}))) \in F(\{A_s \gg C_s \mid s \in S\})$$

The proofs of these lemmas are just tedious analysis of cases and are omitted.

Lemma 3 above gives us that under the hypotheses of the theorem $\{A_s \gg C_s \mid s \in S\} \subseteq F(\{A_s \gg C_s \mid s \in S\})$.

Now since Buff is continuous and satisfiable 5.14 gives us the desired result that $s \in S \Rightarrow Buff(A_s \gg C_s)$.

Observe that lemmas 2 & 3 above combine to prove 5.26.

The above result admits much "degeneralisation" by simplifying the form of g. For example by setting S to be a one element set we get for any $A, B \in M$:

$$A \gg B = ?x{:}T \to (A \gg (!x \to B)) \Rightarrow Buff(A \gg B) \ .$$

5.28 Example

Define $A \Leftarrow ?x{:}T \to !x \to !x \to A$

$C \Leftarrow ?x{:}T \to !x \to ?x \to C$

Now
$$\begin{aligned} A \gg C &= (?x{:}T \to !x \to !x \to A) \gg (?y{:}T \to !y \to ?y \to C) \\ &= ?x{:}T \to ((!x \to !x \to A) \gg (?y{:}T \to !y \to ?y \to C)) \\ &= ?x{:}T \to ((!x \to A) \gg (!x \to ?x \to C)) \quad (*) \\ &= ?x{:}T \to !x \to ((!x \to A) \gg (?x \to C)) \\ &= ?x{:}T \to !x \to (A \gg C) \end{aligned}$$

(by many applications of 5.22)

It is thus an easy induction to show that $A \gg C = B$, the one place buffer.

For $x \in T$ define $A_x = !x \to A$ and $C_x = ?x \to C$.
Thus $A \gg C = ?x{:}T \to (A_x \gg !x \to C_x)$ (by (*) above)
and for each $y \in T$ $A_y \gg C_y = A \gg C = ?x{:}T \to (A_x \gg !x \to C_x)$.

Thus if we put $S = T \cup \{a\}$ for any $a \notin T$ and set $A_a = A$ & $C_a = C$ the $\{A_s \mid s \in S\}$ and $\{C_s \mid s \in S\}$ satisfy the conditions of 5.27 (putting $g(s,x) = x$ for all $s \in S$).

This serves as an alternative proof that $B = A \gg C$ is a buffer (though it is of course circular as we assumed that Buff was satisfiable, and the most basic instance of Buff we have seen was B, all others having been proved from it). It also shows that B, as well as all the other B^n, can be expressed in the form $A \gg C$ for some $A, C \in M$.

5.29 Example (C.A.R.H./A.W.R.)

This example models a simple method for overcoming a "gremlin" on a transmission line which occasionally randomizes information.

Define error generating processes E_i as follows ($i \in \{0,1,2,3\}$).

$$E_0 \Leftarrow ?x{:}T \to \bigvee_{y \in T}(!y \to E_3), \quad E_{i+1} \Leftarrow ?x{:}T \to !x \to E_i$$

The E_i randomize every fourth bit, i determining the phasing. To counteract a line behaving like E_i we send every message in triplicate, and take a majority vote at the receiving end.

$$X \Leftarrow ?x{:}T \to !x \to !x \to !x \to X$$

$$\begin{aligned} Y \Leftarrow ?x{:}T \to (&\ (?x \to (?y{:}T \to Y)) \\ &\ \square\ (?y{:}T-\{x\} \to ((?x \to !x \to Y)\ \square\ (?y \to !y \to Y)))) \end{aligned}$$

To show that this device is correct we would like to show that each $X \gg E_i \gg Y$ is a buffer, so that these processes both transmit information reliably and are free of deadlock (observe that Y can deadlock if it gets no clear majority in any group of three symbols).

Putting $S = \{0,1,2,3\}$, $A_i = X \gg E_i$ and $C_i = Y$, by 5.27 it will clearly be sufficient to show that $i \in S$ implies

$$(X \gg E_i) \gg Y = ?x{:}T \rightarrow ((X \gg E_{i \oplus 1}) \gg (!x \rightarrow Y)),$$

where $\oplus$ represents addition modulo 4.

We can show each of these by repeated application of 5.22.

e.g. $X \gg E_0 \gg Y$ (each process satisfies P_1 so associative law is justified)

$$= ?x{:}T \rightarrow ((!x \rightarrow !x \rightarrow !x \rightarrow X) \gg (?x{:}T \rightarrow (\bigvee_{y \in T} !y \rightarrow E_3)) \gg Y)$$

$$= ?x{:}T \rightarrow ((!x \rightarrow !x \rightarrow X) \gg (\bigvee_{y \in T} !y \rightarrow E_3) \gg Y)$$

$$= ?x{:}T \rightarrow (\bigvee_{y \in T} ((!x \rightarrow !x \rightarrow X) \gg E_3 \gg Y_y)) \quad \text{(where } Y_y = Y \ \underline{\text{after}}\ ?y)$$

$$= ?x{:}T \rightarrow (((!x \rightarrow X) \gg (!x \rightarrow E_2) \gg Y_x)) \ \underline{\text{or}}\ (\bigvee_{y \neq x} (!x \rightarrow X \gg !x \rightarrow E_2 \gg Y_y)))$$

$$= ?x{:}T \rightarrow (((!x \rightarrow X) \gg E_2 \gg (?y{:}T \rightarrow !x \rightarrow Y))$$

$$\underline{\text{or}}\ (\bigvee_{y \neq x} ((!x \rightarrow X) \gg E_2 \gg ((?y \rightarrow !y \rightarrow Y) \,\square\, (?x \rightarrow !x \rightarrow Y)))))$$

$$= ?x{:}T \rightarrow ((X \gg E_1 \gg (!x \rightarrow Y)) \ \underline{\text{or}}\ (\bigvee_{y \neq x} (X \gg (!x \rightarrow E_1) \gg (***))))$$

$$= ?x{:}T \rightarrow ((X \gg E_1 \gg (!x \rightarrow Y)) \ \underline{\text{or}}\ (\bigvee_{y \neq x} (X \gg E_1 \gg (!x \rightarrow Y)))$$

$$= ?x{:}T \rightarrow (X \gg E_1 \gg (!x \rightarrow Y)) \qquad \text{as desired}$$

(*** represents the same term as at that place in the previous line.)

The other cases are easier than this one.

Many other examples in this vein are possible, for example we might design processes to insert parity checks in streams, and others to check them and remove them. We would then wish to show that when we combine these processes the result is a buffer.

We now turn our attention to recursive definitions which involve "$\gg$". Since its definition involves hiding, this operator is not in general non-destructive. The next result identifies two cases where it is.

5.30 <u>Theorem</u>

a) If $w \in dom(C) \Rightarrow |ins(w)| \leqslant |outs(w)|$ then $(A \gg C)$ is a non-destructive function of A (if $dom(C) \subseteq (?T \cup !T)^*$).

b) If $w \in dom(C) \Rightarrow |ins(w)| \geqslant |outs(w)|$ then $(C \gg A)$ is a non-destructive function of A (if $dom(C) \subseteq (?T \cup !T)^*$).

<u>proof</u>

We will give a proof of (a), result (b) following by symmetry. Suppose that $w \in dom(C) = |ins(w)| \leqslant |outs(w)|$.

This condition clearly implies P_2 (of 5.20). For arbitrary A we must have $(A \gg C)$ free of infinite internal chatter, for the proof of this part of 5.20 only depends on P_2 of the second variable (just as the proof that if A & C satisfy P_1 then $A \gg C$ is chatter-free depends only on $P_1(A)$).

Hence for all A the entirety of $A \gg C$ comes from the "<u>or</u>" clause of 5.18.

Suppose $A \in M$. We wish to show that $(A \gg C)\upharpoonright n = (A\upharpoonright n \gg C)\upharpoonright n$. We have $(A \gg C)\upharpoonright n \sqsupseteq (A\upharpoonright n \gg C)\upharpoonright n \equiv (A \gg C)\upharpoonright n \subseteq (A\upharpoonright n \gg C)\upharpoonright n$ by monotonicity. It is thus sufficient to show that if $|w| < n$ and $(w,X) \in (A\upharpoonright n \gg C)$ then $(w,X) \in (A \gg C)$ and that if $|w| = n$ and $w \in dom(A\upharpoonright n \gg C)$ then $w \in dom(A \gg C)$.

In the first case there must be some $(u,U) \in A\upharpoonright n$ and $(v,V) \in C$ such that $((u,U),(v,V)) \Leftrightarrow (w,X)$ in $(A\upharpoonright n \gg B)$.

But then $|n| > |w| = |ins(w)| + |outs(w)| = |ins(u)| + |outs(v)|$
$\geqslant |ins(u)| + |ins(v)| = |ins(u)| + |outs(u)| = |u|$.

Hence $(u,U) \in A$ so $((u,U),(v,V)) \Leftrightarrow (w,X)$ in $(A \gg C)$.

The second case follows by a very similar argument.

This completes the proof of 5.30.

5.31 <u>Example</u>

Consider the process defined $C \Leftarrow ?x:T \rightarrow (C \gg (!x \rightarrow B))$. By the above this recursion is constructive. It is easy to show that C is a buffer by induction. We know that Buff is satisfiable and continuous, so suppose that Buff(D) holds. Then $D \gg B$ is a buffer by 5.24 as both B and D are. Therefore $?x{:}T \to (D \gg (!x \to B))$ is a buffer.

Thus Buff(D) $\Rightarrow$ Buff(F(D)), where F is the function of the C-recursion. Thus Buff(C) holds as claimed.

$$\begin{aligned}
\text{Now } C \gg B &= (?x{:}T \to (C \gg (!x \to B))) \gg B \\
&= ?x{:}T \to ((C \gg (!x \to B)) \gg B) \quad \text{as } B^{O} \subseteq ?T \\
&= ?x{:}T \to (C \gg ((!x \to B) \gg B)) \quad \text{(buffers are associative)} \\
&= ?x{:}T \to (C \gg (B \gg (!x \to B))) \quad \text{(as in 5.25)} \\
&= ?x{:}T \to ((C \gg B) \gg (!x \to B))
\end{aligned}$$

Hence $C \gg B$ is a fixed point of the recursive equation of C, but this equation has a unique fixed point, namely C, so $C = C \gg B$.

$$\begin{aligned}
\text{Also } C \gg C &= (?x{:}T \to (C \gg (!x \to B))) \gg C \\
&= ?x{:}T \to ((C \gg (!x \to B)) \gg C) \quad \text{as } C^{O} \subseteq ?T \\
&= ?x{:}T \to (C \gg ((!x \to B) \gg (?y{:}T \to (C \gg (!y \to B))))) \\
&= ?x{:}T \to (C \gg B \gg C \gg (!x \to B)) \\
&= ?x{:}T \to ((C \gg C) \gg (!x \to B)) \quad \text{by the above.}
\end{aligned}$$

Thus by the same argument as above $C \gg C = C$.

Define processes C_w $(w \in T^*)$ as follows:

$$C_{\langle\rangle} = C \qquad C_{w\langle y\rangle} = (C_w \gg (!y \to B)) \quad .$$

It is easy to show (by $(C \gg B) = C$, $(B_x \gg B) = (B \gg B_x)$ and definition of C & B) that

$$C_{\langle\rangle} = ?x{:}T \to C_x$$

$$C_{w\langle y\rangle} = (?x{:}T \to C_{\langle x\rangle w\langle y\rangle}) \,\Box\, (!y \to C_w)$$

It is then an easy induction to show that $C = B^{\infty}$, where B^{∞} is the canonical infinite buffer $B^{\infty}_{\langle\rangle}$, where

$$B^{\infty}_{\langle\rangle} \Leftarrow ?x{:}T \to B^{\infty}_{\langle x\rangle}$$

$$B^{\infty}_{w\langle y\rangle} \Leftarrow (?x{:}T \to B^{\infty}_{\langle x\rangle w\langle y\rangle}) \,\Box\, (!y \to B^{\infty}_{w}) \quad .$$

Note that as corollaries to the above proof that B^{∞} is a buffer, we have the following identities:

$$B^{\infty} \gg B^{n} = B^{\infty}, \quad B^{\infty}_{w} \gg B^{n}_{v} = B^{\infty}_{wv}, \quad B^{\infty} \gg B^{\infty} = B^{\infty}, \quad B^{\infty}_{w} \gg B^{\infty}_{v} = B^{\infty}_{wv} \quad .$$

The only obvious omissions from this list are
$B^n \gg B^\infty = B^\infty$ and $B^n_w \gg B^\infty_v = B^\infty_{wv}$.

These two results are both easily proved from $B \gg B^\infty = B^\infty$, which is easily proved by defining $B^*_w = B \gg B^\infty_w$ (for $w \in T^*$) and showing that these B^*_w satisfy the recursive equations of the B^∞_w, so the two systems are the same as these equations are certainly constructive.

In cases where a recursion does not satisfy the conditions of 5.30 the analysis is a little harder. One trick is to show that any fixed point of an equation must be deterministic, for then the equation has a unique fixed point as otherwise it could not have a minimal one. We can then prove predicates which are equalities by the satisfaction of equations with unique fixed points (as was done in 5.31) but we cannot directly prove predicates like Buff.

5.32 Example

Define a process $C^* \Leftarrow ?x{:}T \rightarrow (C^* \gg (!x \rightarrow C^*))$.
This recursion is not constructive in our usual sense so it has to be treated with some care.

Suppose that D is any fixed point of this equation. It is quite easy to show, using induction on $|w|$ and 5.18, that all elements of dom(D) satisfy $|ins(w)| \geq |outs(w)|$ and that D is free of infinite chatter. Assuming this result we are justified in using the assocative law on D and its derivatives. Claim that for each $n > 0$ we have

$$D^n = ?x{:}T \rightarrow (D^{2n-1} \gg (!x \rightarrow D)) \quad (\text{where } D^m = \underbrace{D \gg D \gg \ldots \gg D}_{(m \text{ terms})})$$

This is true for n=1 as D is a fixed point of the C* equation.

Assume true for n.

$$\begin{aligned} \text{Then } D^{n+1} &= (?x{:}T \rightarrow (D \gg (!x \rightarrow D)) \gg D^n \\ &= ?x{:}T \rightarrow (D \gg (!x \rightarrow D) \gg D^n) \quad \text{as } (D^n)^0 \subseteq ?T \\ &= ?x{:}T \rightarrow (D \gg (!x \rightarrow D) \gg (?y{:}T \rightarrow (D^{2n-1} \gg (!y \rightarrow D))) \\ &= ?x{:}T \rightarrow (D \gg D \gg D^{2n-1} \gg (!x \rightarrow D)) \\ &= ?x{:}T \rightarrow (D^{2n+1} \gg (!x \rightarrow D)) \quad \text{as desired} \end{aligned}$$

Hence it is true for all n by induction.

We also have $(!x \to D) \gg D \;=\; (!x \to D) \gg (?x{:}T \to (D \gg (!x \to D)))$
$= D \gg D \gg (!x - D)$

Using these two results and 5.22 we have that each process of the form D^n or $D^n \gg (!x \to D) \gg \ldots \gg (!z \to D)$ can either be written in the form $?x{:}T \to A(x)$, where each $A(x)$ is of the same form, or in the form $(?x{:}T \to A(x)) \Box (!y - A')$ where $A(x)$ and A' all have the same form.

Thus define (for $E \in \wp(M)$) $F(E) =$
$$\{?x{:}T \to A(x),\ (?x{:}T \to A(x)) \Box (!y \to A') \mid y \in T,\ A(x) \in E,\ A' \in E\}$$

This function is easily seen to be constructive in the sense of 5.14. It also preserves the (satisfiable and continuous) predicate "is deterministic".

Setting $E = \{D^n,\ D^n(!x \to D)\ \ldots\ (!z \to D) \mid n \in N^+,\ x,\ldots,z \in T\}$ the above shows that $E \subseteq F(E)$. Thus by rule 5.14 we are entitled to deduce that each element of E (and in particular D) is deterministic.

As T is finite hiding is continuous and thus so is H, the function of the C*-recursion.
Thus fix(H) (which we now know to be its unique fixpoint) is equal to $\bigsqcup_{n=1}^{\infty} (H^n(\mathrm{CHAOS}))$
Claim that $\forall m.\forall n.H^n(\mathrm{CHAOS}) \sqsubseteq C^{*m}$. Prove this by induction on n. It is certainly true for n=0, as CHAOS is the minimal element of M.
Suppose true for n.
Then $C^{*m} = ?x{:}T \to ((C^*)^{2m-1} \gg (!x \to C^*))$ (as on the last page)
$\sqsupseteq ?x{:}T \to (H^n(\mathrm{CHAOS}) \gg (!x \to H^n(\mathrm{CHAOS})))$ by induction
$\sqsupseteq H^{n+1}(\mathrm{CHAOS})$ by definition of H

Hence by induction the result holds for all n & m.
But then for each m we have $(C^*)^m \sqsupseteq \bigsqcup_{n=1}^{\infty} (H^n(\mathrm{CHAOS})) = C^*$ and we know that C* is maximal in M as it is deterministic, which gives us the relation $(C^*)^m = C^*$ for all m.

One consequence of this is that C* is a buffer, for we then have $C^* \gg C^* = ?x{:}T \to (C^* \gg (!x \to C^*))$ and $(C^* \gg C^*) \gg C^* = ?x{:}T \to ((C^* \gg C^*) \gg (!x \to C^*))$, which respectively imply $C^* \gg C^*$ and $(C^* \gg C^*) \gg C^*$ are buffers by 2.27. Thus C* is a buffer by 5.23.

If we now define processes C'_w ($w \in T$) corresponding to the C_w of 5.31 we can show that $C^* = B^\infty$ in very much the same way. Let $C'_{\langle\rangle} = C^*$ and $C'_{w\langle y\rangle} = C'_w \gg (!y \to C^*)$. Then as $C^* \gg C^* = C^*$ and $(!x \to C^*) \gg C^* = C^* \gg C^* \gg (!x \to C^*) = C^* \gg (!x \to C^*)$ we can easily show that the C'_w satisfy the recursive equations of B^∞_w, that is

$$C'_{\langle\rangle} = ?x{:}T \to C'_x$$

$$C'_{w\langle y\rangle} = (?x{:}T \to C'_{\langle x\rangle w\langle y\rangle}) \,\square\, (!y \to C'_w) \quad .$$

and so the two systems must be equal, as in the last example. Hence everything that is true of B^∞ is also true of C^*.

All the buffers we have happened to meet so far have been deterministic (even the "gremlins" example 5.29, where the explicit non-determinism of the definition disappears from the point of view of the external environment). This is certainly not true in general, however. In fact it is easy to show from the definition of Buff (5.16) that if B_1 and B_2 are buffers then so is B_1 <u>or</u> B_2 (and there are certainly more than one buffer). Indeed this result extends to infinite disjunctions and so, for example, the process $\bigvee_{n=1}^{\infty}(B^n)$ is a buffer. (It is a process because T is finite.) It is easy to show that "$\gg$" is a distributive operator in the limited sense that if $(\bigvee_{i=1}^{\infty}(A_i) \gg B)$ is free of infinite chatter then it is equal to the process $\bigvee_{i=1}^{\infty}(A_i \gg B)$. (This comes from the nature of the "<u>or</u>" clause of 5.18.) There is of course a corresponding result for the second variable.

Let $B^* = \bigvee_{n=1}^{\infty}(B^n)$. Clearly $B^* \gg (!x \to B)$ is free of infinite chatter as B^* is a buffer, so we are entitled to use the above law in this case.

Hence

$$\begin{aligned} ?x{:}T \to (B^* \gg (!x \to B)) &= \bigvee_{n=1}^{\infty}(?x{:}T \to (B^n \gg (!x \to B))) \\ &= \bigvee_{n=1}^{\infty}(?x{:}T \to B^{n+1}_{\langle x\rangle}) \qquad \text{(by 5.25)} \\ &= \bigvee_{n=1}^{\infty}(B^{n+1}) \end{aligned}$$

This proves that $B^* \sqsubseteq F(B^*)$, where F is the recursive equation of C $(= B^\infty)$. It is easy then to show that F must have some fixed point above B^*. Thus by the unique fixed point property of F we have $B^\infty \sqsupseteq \bigvee_{n=1}^{\infty}(B^n)$ (the canonical infinite buffer is stronger than the disjunction of all the finite ones).

Postscript: the expressive power of "$\gg$"

To round of our study of buffers and their relationship with the pipe operator we ask the question "are all buffers expressible as $A \gg C$ for some $A, C \in M$?". All the buffers we have met so far with the exception of B* have either been directly expressed in this form or later shown to be so expressible. In fact B* can be expressed in this form by a combination of 5.28 and the distributivity principle of the last page ($B^* = ((A \text{ or } (\overset{\infty}{\underset{n=1}{\nabla}} (B^n \gg A))) \gg C)$, where A & C are as in 2.28).

In fact there are (unfortunately?) certain pathological buffers which cannot be so expressed.

5.33 Example

$$\text{Let} \quad A \Leftarrow ?x{:}T \rightarrow (!x \rightarrow B$$
$$\qquad \Box ?y{:}T \rightarrow (?z{:}T \rightarrow (!x \rightarrow !y \rightarrow !z \rightarrow B)$$
$$\qquad\qquad \Box !x \rightarrow !y \rightarrow B) \quad)$$

Then A is a buffer not expressible as $A_1 \gg A_2$ for $A_1, A_2 \in M$. The fact that A is a buffer can be proved easily from the fact that B is a buffer.
The main reason for A not being expressible as $(A_1 \gg A_2)$ is that when it contains two items it will deterministically accept another, but on outputting its first symbol it immediately loses the ability to output. The point is that if $A = A_1 \gg A_2$ then it must be A_1 doing the inputting and A_2 doing the outputting. The only way that A_1 can lose the ability to input is by a signal passing between A_1 and A_2 after A_2 has output. This however leaves the possibility that A_1 might accept an input before this signal has been executed but after A_2's output.
A formal proof that A is not so expressible will follow easily from the next result.

5.34 Theorem

If A is a buffer expressible as $A_1 \gg A_2$ then A satisfies
$$w\langle !a\rangle \in dom(A) \Rightarrow \{?x \mid w\langle !a?x\rangle \notin dom(A)\} \in A(w)$$

(A can refuse before it outputs "a" the whole of what it must refuse after outputting "a".)

proof

Suppose that $w\langle !a\rangle \notin dom(A)$ and that A is so expressible.

$A = (A_1 \gg A_2)$ must be free of infinite internal chatter as it is a buffer. Thus only the "or" clause of 5.18 applies. Hence $(w,\emptyset)$ must have some derivation $((u,U),(v,V))$ in $(A_1 \gg A_2)$.

Let $X = \{?x \mid w\langle !a?x\rangle \notin dom(A)\}$.
Suppose $?x \in X$ and $u\langle ?x\rangle \in dom(A_1)$.

Now $v\langle !a\rangle \in dom(A_2)$ or else $((u,U),(v,V\cup\{!a\}))$ would be a derivation in $(A_1 \gg A_2)$ of $(w,\{!a\})$, which would contradict the fact that A is a buffer (lines (i) & (iii) of 5.16).
Thus $t\langle !a?x\rangle \in dom(strip!(A_1)_{T \cup ?T} \|_{T \cup !T} strip?(A_2))$ where t corresponds with u & v in the sense of 5.18.
Hence $(t\langle !a?x\rangle)\lceil(?T \cup !T) = w\langle !a?x\rangle \in dom(A_1 \gg A_2)$, which contradicts the fact that $u\langle ?x\rangle \in dom(A_1)$.

Hence $X \cap (A_1 \text{ \underline{after} } u)^{o} = \emptyset$, so $((u,U\cup X),(v,V))$ is a derivation for (w,X) in $(A_1 \gg A_2)$, which gives us the desired result.

The author conjectures that the condition on the last page is also sufficient to ensure expressiblity of buffers, but at the time of writing has not had time to prove or disprove this.

This concludes our detailed study of buffers. Note that in the next chapter there is a different method given of defining B^{∞} (in terms of the master/slave) operator.

Appendix: Proof of the lemma needed in 5.19

5.35 Lemma

If A is free of infinite X-chatter and $X \cap Z = \emptyset$ then $(A/X_{Y}\|_{Z}B) = (A_{X\cup Y}\|_{Z}B)/X$.

proof

Use the notation $s_{X}\|_{Y}t \to w$ to mean (for $X,Y \subseteq \Sigma$ and $s,t,w \in \Sigma^*$) that $w\upharpoonright X = s$, $w\upharpoonright Y = t$ and $w \in (X\cup Y)^*$.

It is easy to check that (for any A,B,X,Y)

$$(w,W) \in (A_{X}\|_{Y}B) \Leftrightarrow \exists (s,S) \in A,\ (t,T) \in B \text{ s.t. } s_{X}\|_{Y}t \to w \ \&\ W\cap(X\cup Y) \subseteq (X\cap S)\cup(Y\cap T) \quad . \qquad (*)$$

It is also easy to verify that $X \cap Z = \emptyset$ implies

$$s_{X\cup Y}\|_{Z}t \to w \Rightarrow (s/X)_{Y}\|_{Z}t \to (w/X)$$

We will show first that under the stated conditions

$$(A/X_{Y}\|_{Z}B) \subseteq (A_{X\cup Y}\|_{Z}B)/X \ .$$

Suppose $(w,W) \in (A/X_{Y}\|_{Z}B)$, then by (*) above, and since A is free of infinite X-chatter, there exist s,t,S,T such that $(s,S\cup X) \in A$, $(t,T) \in B$, $(s/X)_{Y}\|_{Z}t \to w$

$$\& \ W\cap(Y\cup Z) \subseteq (S\cap Y)\cup(T\cap Z) \ . \qquad (**)$$

It is easy to see that since $X \cap Z = \emptyset$ and $(s/X)_{Y}\|_{Z}t \to w$ there exists some w^* such that $s_{X\cup Y}\|_{Z}t \to w^*$ and $w^*/X = w$. (For example such a w^* is obtained by placing each maximal substring of X-symbols occurring in s in w at the leftmost place at which the number of Y-X symbols is the same as at the place the substring occurs in s. If $X = \{x\}$, $Y = \{y\}$ and $Z = \{z\}$, $s = \langle xyxxyy\rangle$, $t = \langle zzz\rangle$ and $w = \langle yzyzyz\rangle$ then in this case w^* would be $\langle xyxxzyzyz\rangle$.)

Now we have $(s,S\cup X) \in S$, $(t,T) \in B$ and $(W\cup X)\cap(X\cup Y\cup Z) \subseteq ((S\cup X)\cap(Y\cup X))\cup(T\cap Z)$. (The set relation is obtained by taking the union of each side of (**) with X.)

Hence $(w^*,W\cup X) \in (A_{X\cup Y}\|_{Z}B)$. (By (*))

This implies $(w^*/X,W) \in (A_{X\cup Y}\|_{Z}B)/X$ by definition of "/X".

Since $w = w^*/X$ this just says $(w,W) \in (A_{X\cup Y}\|_{Z}B)/X$, which is what we wanted to prove.

This completes the proof that $(A/X_{Y}\|_{Z}B) \subseteq (A_{X\cup Y}\|_{Z}B)/X$.

To prove that $(A_{X\cup Y}\|_Z B)/X \subseteq (A/X_Y\|_Z B)$ the first step is to show that $(A_{X\cup Y}\|_Z B)$ is free of infinite X-chatter because A is. If this were not so there would be an infinite sequence $w_o < w_1 < \ldots < w_i < \ldots$ of elements of $dom(A_{X\cup Y}\|_Z B)$ such that $w_i \restriction X = w_o \restriction X$ for all i. But then $w_i \restriction X\cup Y \in domA$ for all i, and

$$
\begin{aligned}
(w_i \restriction X\cup Y)/X &= (w_i/X)\restriction X\cup Y \\
&= (w_o/X)\restriction X\cup Y \\
&= (w_o \restriction X\cup Y)/X
\end{aligned}
$$

Also it is easily seen that $w_{i+1}\restriction X\cup Y > w_i \restriction X\cup Y$, since the difference between w_i and w_{i+1} occurs in X and is so preserved by restriction to $X\cup Y$.

This contradicts the fact that A is free of infinite X-chatter, so we can conclude that $(A_{X\cup Y}\|_Z B)$ is indeed free of infinite X-chatter.

Now suppose that $(w,W) \in (A_{X\cup Y}\|_Z B)/X$. By the above there is some w^* such that $w^*/X = w$ and $(w^*, W\cup X) \in (A_{X\cup Y}\|_Z B)$.

Thus there are some $(s,S) \in A$ and $(t,T) \in B$ such that $s_{X\cup Y}\|_Z t \to w^*$ and $(W\cup X)\cap(X\cup Y\cup Z) \subseteq ((X\cup Y)\cap S)\cup(Z\cap T)$.

Since $X\cap Z = \emptyset$ we see that $X \subseteq S$ (i.e. $S \cup X = S$) and also $W\cap(Y\cup Z) \subseteq (Y\cap S)\cup(Z\cap T)$. (+)

Now $s_{X\cup Y}\|_Z t \to w^*$ & $X\cap Z = \emptyset$

$\Rightarrow s/X_Y\|_Z t \to (w^*/X)$ (by our earlier remark) (++)

Putting these facts together we obtain

$(s/X,S) \in A/X$ (as $S\cup X = S$)

$(t,T) \in B$

$\Rightarrow (w^*/X,W) \in (A/X_Y\|_Z B)$ (by (+) & (++))

This tells us $(w,W) \in (A/X_Y\|_Z B)$, which completes the proof that $(A_{X\cup Y}\|_Z B)/X \subseteq (A/X_Y\|_Z B)$.

This completes the proof of lemma 5.35.

We will see that this lemma is important in several proofs of the well-definedness of parallel/hiding combinators. The reasons why it does not hold in general (i.e. without the assumption of freedom from infinite chatter) will be discussed in chapter 8.

Chapter 6 : The Master/Slave Operator

This operator, which was introduced in chapter 2, can be modelled in a similar (though preferable) manner over M. Because of the use made of hiding in its definition it will be necessary to restrict its definition to cases where the set T of communicated values is finite.

6.1 $(A \| a::B) = (A_{\Sigma} \|_{a.\Gamma} a.(swap!?(B)))/(a.\Gamma)$

where $\Gamma = T \cup ?T \cup !T$ and swap is defined

$swapab(A) = \{(swapab(w), swapab(X)) \mid (w,X) \in A\}$

(swapab(X) for $X \subseteq \Sigma$ is defined to be the natural extension of the definition for Σ given in chapter 2).

6.2 Examples

a) If $A = A_1 \| A_2 \| \ldots\ldots \| A_k$ and some of the A_i might have to execute a critical section which needs to be executed in parallel with no other critical section then if we guard each c.s. with a request to a suitable (joint) slave this condition can be ensured.

e.g. $A_i' = B_i;(a!i \rightarrow \underline{skip});CS_i;(a?i \rightarrow \underline{skip});C_i$

(where $A_i = B_i;CS_i;C_i$)

$D = ?n:\{1,2,..,k\} \rightarrow !n \rightarrow D$

$A^* = ((A_1' \| A_2' \| \ldots\ldots \| A_k') \| a::D)$

b) Five dining philosophers: ($\boxplus$, $\boxminus$ ≡ mod 5 arithmetic)

If $PHIL_i \Leftarrow a.sit.i \rightarrow i.pickup_i \rightarrow i.pickup_{i \boxplus 1} \rightarrow i.eat \rightarrow i.putdown_{i \boxplus 1} \rightarrow i.putdown_i \rightarrow a.getup.i \rightarrow PHIL_i$

$FORK_i \Leftarrow i.pickup_i \rightarrow i.putdown_i \rightarrow FORK$
$\quad \square\, i \boxminus 1.pickup_i \rightarrow i \boxminus 1.putdown_i \rightarrow FORK$

$B = B_0$ where $B_0 \Leftarrow sit.j:\{0,..4\} \rightarrow B_1$

$i \in \{1,2,3\} \Rightarrow B_i \Leftarrow sit.j:\{0,..4\} \rightarrow B_{i+1}$
$\quad \square\, getup.j:\{0,..4\} \rightarrow B_{i-1}$

$B_4 \Leftarrow getup.j:\{0,..4\} \rightarrow B_3$

Then $(\|_{i=0}^{4} PHIL_i) \| a::B) \| (\|_{i=0}^{4} FORK_i)$ is free of deadlock.

This last example is reasonably easy to prove using the techniques that were developed in chapter 5 and those which we will develop in this chapter and the next.

6.3 <u>Lemma</u>

If T is finite then $(A \| a::B)$ is a well-defined continuous function of its parameters. Furthermore we have the following criterion for membership of $(A \| a::B)$:

$(w,X) \in (A \| a::B)$ if and only if

<u>either</u> there is some $w' \leqslant w$ such that

$$\{t \in dom(A) \mid (t \upharpoonright a.\Gamma \in a.swap!?(dom(B))) \ \& \ (t/a.\Gamma = w')\}$$

is infinite

<u>or</u> there is some $(u,U) \in A$ and $(v,V) \in B$ such that

$$X \cup a.\Gamma = U \cup (a.swap!?(V \cap \Gamma))$$

The proof of this is very similar to that of 5.18, the corresponding result on the $(A \gg B)$ operator.

If the first case holds of any (w,X) then we say that $(A \| a::B)$ contains infinite internal chatter.

In the <u>or</u> clause we say that $((u,U),(v,V))$ is a derivation for (w,X) in $(A \| a::B)$ or equivalently that $((u,U),(v,V)) \Leftrightarrow (w,X)$ in $(A \| a::B)$.

6.4 <u>Theorem</u>

Suppose that each of $(A \| a::B)$ and $(A \| b::C)$ is free of infinite internal chatter and that $a.\Gamma \cap b.\Gamma = \emptyset$.

Then $((A \| a::B) \| b::C) = (A \| b::C) \| a::B)$.

The proof of this is an immediate corollary to lemma 5.35, the same result used in proving 5.19.

Theorem 2.38 also carries straight over to this model:

6.5 <u>Theorem</u>

Define C^a_j (for $a \in \Sigma$ and $j \in Z$) as follows:

$$C^a_j(A) \triangleq \forall w \in dom(A).\ (|w \upharpoonright \Gamma| - |w \upharpoonright a.\Gamma|) \geqslant \min(j, |w \upharpoonright (\Gamma \cup a.\Gamma)|)\ .$$

a) $C^a_j(A)$ is a (strongly) continuous predicate.

b) $C^a_j(A) \Rightarrow (A \| a::B)$ is free of infinite internal chatter.

c) $C^a_k(A) \Rightarrow \forall n. \forall B. (A \| a::B) \upharpoonright n+k+1 = (A \| a::B \upharpoonright n) \upharpoonright n+k+1$.

d) $C^a_j(A) \Rightarrow C^a_j(A \| b::B)$.

The proof of (a) is the same as 2.38(a).
The proof of (b) is similar to (though easier than) that of 5.20.
The proofs of (c) and (d) are similar to 2.38(b)&(c).

We will shortly see that the master/slave operator is a very useful tool in defining processes by recursion. As in chapter 2 the above results help us to identify the cases where it is a constructive or non-destructive function of its second (slave) argument. The following is a list of technical lemmas of a computational nature which will subsequently be used freely and informally in proofs.

6.6 Lemma Rules for $(A \| a::B)$

(i) If $A^o \cap a.\Sigma = \emptyset$ then $(A \| a::B)(\langle\rangle) = A(\langle\rangle)$, $(A \| a::B)^o = A^o$ and $(A \| a::B)\,\underline{\text{after}}\,x = (A\,\underline{\text{after}}\,x \| a::B)$

(ii) $(A\ \underline{\text{or}}\ B \| a::C) = (A \| a::C)\ \underline{\text{or}}\ (B \| a::C)$
$(A \| a::(B\ \underline{\text{or}}\ C)) = (A \| a::B)\ \underline{\text{or}}\ (A \| a::C)$

(iii) $(a!x \rightarrow A \| a::(?x:T \rightarrow B_x)) = (A \| a::B_x)$
$= ((a!x \rightarrow A) \Box (a?x:T \rightarrow C_x) \| a::(?x:T \rightarrow B_x))$
$= (a!x \rightarrow A \| a::((?x:T \rightarrow B_x) \Box (!y \rightarrow C)))$

(iv) $(a?x:T \rightarrow A_x \| a::(!x \rightarrow B)) = (A_x \| a::B)$
$= ((a?x:T \rightarrow A_x) \Box (!y \rightarrow C) \| a::(!x \rightarrow B))$
$= (a?x:T \rightarrow A_x \| a::((!x \rightarrow B) \Box (?y:T \rightarrow C_y)))$

(v) Define a function $f:\Sigma \rightarrow \Sigma$ as follows:

$f(y) = a!x$ if $y = ?x$ for any x
$= a?x$ if $y = !x$ for any x
$= y$ otherwise.

If $B^o \subseteq a.\Sigma$ and $B^o \cap f(C^o) = \emptyset$ then $(A \Box B \| a::C) = (A \| a::C)$

(vi) If $A^o \cap a.\Sigma = \emptyset$ & $B^o \subseteq a.\Sigma$ & $\neg\exists X \in B(\langle\rangle), Y \in C(\langle\rangle).X \cup f(Y) = a.\Sigma$ then $(A \Box B \| a::C) = ((A \| a::C) \Box (B \| a::C))\ \underline{\text{or}}\ (B \| a::C)$.
(The condition here ensures that B and C cannot deadlock, thus ensuring that some hidden transition will eventually take place if nothing else is offered.)
e.g. $(?x:T \rightarrow A_x \Box a?x:T \rightarrow B_x \| a::(!y \rightarrow C)) =$
$((?x:T \rightarrow (A_x \| a::(!y \rightarrow C))) \Box (B_y \| a::C))\ \underline{\text{or}}\ (B_y \| a::C)$

(vii) If $A^o \cap a.\Sigma = \emptyset$ and $B^o \subseteq a.\Sigma$ and $B^o \cap f(C^o) \neq \emptyset$ and $\exists X \in B(\langle\rangle), Y \in C(\langle\rangle)$ s.t. $X \cup f(Y) = a.\Sigma$ then $(A \Box B \| a::C) = (A \| a::C)\ \underline{\text{or}}\ (B^* \| a::C^*)$
where $D^*(w) = \{X \mid X \cap D^o = \emptyset\}$ if $w = \langle\rangle$
$= D(w)$ otherwise.
(If B & C deadlock on the first step then A must be allowed to take precedence.)

Rules (i) - (vii) deal essentially with the first step behaviour of $(A \| a::B)$. It is possible however to derive a few more general ones. ((viii) is just a restatement of 6.4.)

(viii) If $a \neq b$ and each of $(A \| a::B)$ and $(A \| b::C)$ is free of infinite internal chatter then
$((A \| a::B) \| b::C) = ((A \| b::C) \| a::B)$.

(ix) If $(A \| a::B)$ is free of infinite internal chatter and $a.\Sigma \cap Y = \emptyset$ then $((A_{X \cup a.\Sigma} \|_Y B) \| a::C) = ((A \| a::C)_X \|_Y B)$.

Each of (i) - (vii) is quite easy to prove directly from the definitions of the operators involved. (ix), like (viii) is a consequence of 5.35.

We will concentrate in the next few pages on seeing how $(A \| a::B)$ can be used as a recursive tool, and on how processes defined recursively with this operator can be proved correct by our methods.

In the following examples the recursive scheme used is $R \Leftarrow (X \| a::R)$ (or some slight variant), where X is a known process. This says that R is the process which X is when it calls R as its slave. One can think of R as a process which is allowed to call itself as a subroutine running in parallel. In 6.7 and 6.8 exactly this scheme is used, modelling a stack and a buffer respectively; in 6.9 we allow the process to call two independent versions of itself and model C.A.R. Hoare's Quicksort.

6.7 <u>Example</u>: <u>Stack</u>
Define the process R recursively by

$$R \Leftarrow (X \| a::R)$$

where
$$\begin{aligned} X &\Leftarrow ?x:T \rightarrow Y_x \\ Y_x &\Leftarrow ?y:T \rightarrow a!x \rightarrow Y_y \\ &\qquad \square\ !x \rightarrow Z \\ Z &\Leftarrow ?x:T \rightarrow Y_x \\ &\qquad \square\ a?x:T \rightarrow Y_x \quad . \end{aligned}$$

R is intended to model an empty stack. Intuitively we can interpret its actions as follows (bearing in mind that it has as a slave a copy of itself labelled "a"):

Initially it is empty and so cannot output. It can however input any element of T and store it (without reference to its slave).

If subsequently it is storing an element of T it can output it to its environment. It can also input any element of T from its environment; if it does this it outputs the old element it was storing to its slave and stores the new one itself (the new one becoming the new top of the stack).

If (at a time other than the start) the control process finds itself not storing any element of T it does not know whether or not its slave is empty. It is therefore prepared either to input the new top of the stack from its slave or from its environment.

We can define a more obviously correct stack as follows by infinite mutual recursion over T*:

$$S_w \Leftarrow ?x{:}T \rightarrow S_{\langle x\rangle} \qquad \text{(if } w = \langle\rangle)$$

$$S_w \Leftarrow ?x{:}T \rightarrow S_{\langle x\rangle w} \;\square\; !y \rightarrow S_v \qquad \text{(if } w = \langle y\rangle v,\ y \in T \;\&\; v \in T^*)$$

Here S_w represents the stack with contents w; the top of the stack being the leftmost component of w. An empty stack can only input; a stack with top element y can either output y (and lose y from the top of the stack) or input some element which it adds to the stack at the top.

We have not formally defined any correctness criterion for stacks. If this were done it is likely that $S_{\langle\rangle}$ would be much easier to prove correct than R. Indeed one very plausible correcness condition for infinite stacks is congruence with $S_{\langle\rangle}$, a process which it is very easy to prove (for example) free of deadlock. We will therefore content ourselves with a proof of the congruence $R = S_{\langle\rangle}$, leaving out any further work that needs to be done proving $S_{\langle\rangle}$ to be correct.

Define $R_{\langle\rangle} = R$ and $R_{\langle y\rangle w} = (Y_y \,\|\, a{::}R_w)$.

Claim that (i) $\forall w.(Z \,\|\, a{::}R_w) = R_w$

and (ii) $R_{\langle\rangle} = ?x{:}T \rightarrow R_{\langle x\rangle}$ and $\forall y,w.R_{\langle y\rangle w} = ?x{:}T \rightarrow R_{\langle xy\rangle w} \;\square\; !y \rightarrow R_w$.

We will prove these two results not by recursion induction but by ordinary mutual induction on the length of w.

a) <u>$w = \langle\rangle$</u>

i) $R_w = (X \| a::R)$

$\quad = ?x:T \to (Y_x \| a::R)$ (by rule (i))

$\quad = ?x:T \to R_{\langle x\rangle}$ (by defn. of $R_{\langle x\rangle}$)

ii) $(Z \| a::R_w) = ((?x:T \to Y_x \Box a?x:T \to Y_x) \| a::R)$

$\quad = (?x:T \to Y_x \| a::R)$ (by rule (v) as $f(R^{o}) = a!T$)

$\quad = (X \| a::R)$

$\quad = R_w$

b) <u>$w = \langle y\rangle v$</u> (assuming the results to hold of all shorter w')

(ii) $R_w = (Y_y \| a::R_v)$

$\quad = !y \to (Z \| a::R_v)$

$\quad\quad \Box\ ?x:T \to (a!y \to Y_x \| a::R_v)$ (by rule (i))

$\quad = !y \to R_v$ (inductive hypothesis (i))

$\quad\quad \Box\ ?x:T \to (a!y \to Y_x \| a::(?y:T \to R_{\langle y\rangle v}(\Box\ !z \to R_u)))$

(by inductive hypothesis (ii) where if $v=\langle\rangle$ the bracket is not present otherwise $v = \langle z\rangle u$, say)

$\quad = !y \to R_v$

$\quad\quad \Box\ ?x:T \to (Y_x \| a::R_{\langle y\rangle v})$ (by rule (ii) in either case)

$\quad = !y \to R_v$

$\quad\quad \Box\ ?x:T \to R_{\langle xy\rangle v}$ as desired.

(i) $(Z \| a::R_w) = ((?x:T \to Y_x \Box a?x:T \to Y_x) \| a::(!y \to R_v \Box ?x:T \to R_{\langle xy\rangle v}$

(by (ii) above)

$\quad = ((?x:T \to (Y_x \| a::R_{\langle y\rangle v}) \Box P)$ <u>or</u> $P)$

where $P = (Y_y \| a::R_v)$ (by rules (vi), (i) & (iv

now $P = R_{\langle y\rangle v}$

$\quad = !y \to R_v \Box ?x:T \to R_{\langle xy\rangle v}$ (by (ii) above)

$\quad = ((?x:T \to R_{\langle xy\rangle v}) \Box ((!y \to R_v) \Box ?x:T \to R_{\langle xy\rangle v}))$

<u>or</u> $((!y \to R_v) \Box ?x:T \to R_{\langle xy\rangle v})$

$\quad = (?x:T \to R_{\langle xy\rangle v}) \Box (!y \to R_v)$ (by laws of $\Box$ and <u>or</u>)

$\quad = R_{\langle y\rangle v}$ (by (ii) above)

$\quad = R_w$ as desired.

This completes the proof of our two inductive hypotheses, so the two results are proved for all w.

It is now a simple (recursion) induction on the definition of $\underset{\sim}{S}$ to show that $\forall w.\ S_w = R_w$ (the predicate $R(\underset{\sim}{X}) = \forall w. X_w = R_w$ is trivially continuous and satisfiable and the $\underset{\sim}{S}$-recursion is clearly constructive).

$$R(\underset{\sim}{X}) \Rightarrow \left\{ \begin{array}{l} ?x{:}T \to X_{\langle x\rangle} = ?x{:}T \to R_{\langle x\rangle} = R_{\langle\rangle} \\ \left(\begin{array}{l} ?x{:}T \to X_{\langle xy\rangle w} \\ \square\ !y \to X_w \end{array}\right) = \left(\begin{array}{l} ?x{:}T \to R_{\langle xy\rangle w} \\ \square\ !y \to R_w \end{array}\right) = R_{\langle y\rangle w} \end{array} \right\} \Rightarrow R(F(\underset{\sim}{X}))$$

This completes the proof that $R = S_{\langle\rangle}$, since by definition $R = R_{\langle\rangle}$.

The above illustrates one possible scheme of proof which is possible for processes defined in the way we are studying. We never needed the fact that the R-recursion is constructive, though this fact is quite easy to prove. To do this one shows that X, Y_x and Z respectively satisfy the conditions C_1^a, C_o^a and C_{-1}^a by mutual (recursion) induction. A tabular method for doing this is decribed in 6. .

Intuitively the master/slave operator is well adapted to the definition of stacks, just as "»" is well adapted to modelling buffers. It is easy to imagine how a process defined recursively by $R \Leftarrow (X \| a{::}R)$ might act as a stack but harder to imagine how one might act as a buffer. This is because there must then be some leap-frogging of information. In the case of a stack information input from the environment is put first on the queue for re-output; this naturally ties in with the fact that the output of an "$R \Leftarrow (X \| a{::}R)$" process is in the same place as its input. This is not the case with a buffer, where an element input from the environment must be put last on the queue for output. The next example shows how this might be done. In 6.8 the method adopted is to throw any input down to the bottom of the recursion, so that except when some input is being processed the process settles down to be like a queue of information waiting to get out: the further from the output/input port the longer its wait.

This method requires a slightly less constructive recursion: we can no longer induct on the amount contained in the buffer.

6.8 Buffer

This example is modelled closely on the previous one. The proof is rather more involved because of the rather less constructive recursion.

We use the same scheme of recursion:

$$
\begin{aligned}
R &\Leftarrow (X \,\|\, a::R) \\
\text{where}\quad X &\Leftarrow ?x{:}T \rightarrow Y_x \\
Y_x &\Leftarrow ?y{:}T \rightarrow a!y \rightarrow Y_x \\
&\qquad \square\ !x \rightarrow Z \\
Z &\Leftarrow ?x{:}T \rightarrow a!x \rightarrow Z \\
&\qquad \square\ a?x{:}T \rightarrow Y_x
\end{aligned}
$$

It is easily shown by induction on their joint definition that X satisfies C^a_1, each Y_x satisfies C^a_o and that Z satisfies C^a_{-1}. Thus the R-recursion above is constructive by 6.5(c).

<u>Theorem</u> $R = B^\infty$

Recall that $B^\infty = B^\infty_{\langle\rangle}$, where $B^\infty_{\langle\rangle} \Leftarrow ?x{:}T \rightarrow B^\infty_{\langle x\rangle}$

$$B^\infty_{w\langle y\rangle} \Leftarrow (?x{:}T \rightarrow B^\infty_{\langle x\rangle w\langle y\rangle}) \;\square\; (!y \rightarrow B_w) \;.$$

The proof of this result depends on two inductions, one on the definition of B and one on the definition of R.

Firstly claim that (i) $\forall w.\ (Z \,\|\, a::B^\infty_w) = B^\infty_w$

$$
\begin{aligned}
\text{(ii)}\quad & B^\infty_{\langle\rangle} = (X \,\|\, a::B^\infty_{\langle\rangle}) \\
& \forall w. \forall y.\ B^\infty_{w\langle y\rangle} = (Y_y \,\|\, a::B^\infty_w) \;.
\end{aligned}
$$

To prove these results define $\underset{\sim}{C} \in M^{T^*}$ as follows:

$$
\begin{aligned}
C_{\langle\rangle} &= (X \,\|\, a::B^\infty_{\langle\rangle}) \ \underline{\text{or}}\ (Z \,\|\, a::B^\infty_{\langle\rangle}) \\
C_{w\langle y\rangle} &= (Y_y \,\|\, a::B^\infty_w) \ \underline{\text{or}}\ (Z \,\|\, a::B^\infty_{w\langle y\rangle}) \;.
\end{aligned}
$$

We then have (using many applications of rules 6.6)

$$
\begin{aligned}
C_{\langle\rangle} &= ?x{:}T \rightarrow (Y_x \,\|\, a::B^\infty_{\langle\rangle}) \ \underline{\text{or}}\ ?x{:}T \rightarrow (a!x \rightarrow Z \,\|\, a::(?x{:}T \rightarrow B^\infty_x)) \\
&= ?x{:}T \rightarrow ((Y_x \,\|\, a::B^\infty_{\langle\rangle}) \ \underline{\text{or}}\ (Z \,\|\, a::B^\infty_x)) \\
&= ?x{:}T \rightarrow C_x
\end{aligned}
$$

$$
\begin{aligned}
C_{w\langle y\rangle} = {} & (!y \rightarrow (Z \,\|\, a::B^\infty_w)) \,\square\, (?x{:}T \rightarrow (a!x \rightarrow Y_y \,\|\, a::(?x{:}T \rightarrow B^\infty_{\langle x\rangle w}(\square\, *)))) \\
& \underline{\text{or}}\ ((?x{:}T \rightarrow ((a!x \rightarrow Z) \,\|\, a::(?x{:}T \rightarrow B^\infty_{\langle x\rangle w\langle y\rangle} \square\, !y \rightarrow B^\infty_w)) \,\square\, P)\ \underline{\text{or}}\ P)
\end{aligned}
$$

where * (present if $w \neq \langle\rangle$) has the form $(!z \rightarrow B^\infty_v)$ and

$$
\begin{aligned}
P = (Y_y \,\|\, a::B^\infty_w) &= ?x{:}T \rightarrow (a!x \rightarrow Y_y \,\|\, a::(?x{:}T \rightarrow B^\infty_{\langle x\rangle w}(\ *)) \\
&\qquad \square\ !y \rightarrow (Z \,\|\, a::B^\infty_w) \\
&= (?x{:}T \rightarrow (Y_y \,\|\, a::B^\infty_{\langle x\rangle w})) \,\square\, (!y \rightarrow (Z \,\|\, a::B^\infty_w))
\end{aligned}
$$

Hence

$$C_{w\langle y\rangle} = \,!y \rightarrow (Z \| a::B^{\infty}_{w}) \,\square\, ?x:T \rightarrow ((Y_y \| a::B^{\infty}_{\langle x\rangle w}) \text{ or } (Z \| a::B^{\infty}_{\langle x\rangle w\langle y\rangle}))$$
$$\sqsupseteq (!y \rightarrow C_w) \square (?x:T \rightarrow C_{\langle x\rangle w\langle y\rangle})$$

Thus $\underset{\sim}{C} \sqsupseteq F(\underset{\sim}{C})$, where F is the function of the $\underset{\sim}{B}^{\infty}$-recursion. This implies that F has some fixed point below $\underset{\sim}{C}$ in the complete partial order M^{T^*}.
But fix(F) is maximal in M^{T^*}, since F is a constructive function with a unique fixed point which is a vector of deterministic processes (the fact that this is so is easily proved by induction on the definition of $\underset{\sim}{B}^{\infty}$).

Hence $\underset{\sim}{C} = \underset{\sim}{B}^{\infty}$, and this is easily seen to imply (i) & (ii), again because all the B^{∞}_{w} are maximal in M.

We thus have that $(X \| a::B^{\infty}) = B^{\infty}$, and since the R-recursion is already known to be constructive it is an easy induction to show that $R = B^{\infty}$.

Notice that we have also shown that $B^{\infty}_{w} = (Z \| a::B_w)$, which shows that the recursion $Q \Leftarrow (Z \| a::Q)$ cannot possibly be constructive, for we could then prove that $Q = B_w$ for each $w \in T^*$. This shows that 6.5(c) is as strong as possible, for Z satisfies C^{a}_{-1}.

6.9 Quicksort (After C.A.R.H.)

Suppose that T'is given some total order $<$, and $T = T' \cup \{e,f\}$. It is possible to model a version of the "quicksort" algorithm using the same scheme of recursion used in the last two examples, except that here the process will call two independent versions of itself as slaves.

$$Q \Leftarrow ((X \| u::Q) \| d::Q)$$
$$\text{where } X \Leftarrow (?x:T' \rightarrow Y_x) \square (?e \rightarrow !f \rightarrow X)$$
$$Y_x \Leftarrow (?y:\{y \mid y > x\} \rightarrow u!y \rightarrow Y_x) \square (?y:\{y \mid y \leq x\} \rightarrow d!y \rightarrow Y_x)$$
$$\square (?e \rightarrow u!e \rightarrow d!e \rightarrow Z_x)$$
$$Z_x \Leftarrow (u?y:T' \rightarrow !y \rightarrow Z_x) \square (u?f \rightarrow !x \rightarrow Z^*)$$
$$Z^* \Leftarrow (d?y:T' \rightarrow !y \rightarrow Z^*) \square (d?f \rightarrow !f \rightarrow X)$$

The intention is that Q should receive a stream of input symbols terminated by the symbol "e" which is not in T' and then output them in descending order, ending the output stream by "f", another special symbol. Q goes about this by using the first symbol in T' that it receives (assuming that the input stream is not empty) as a pivot. All the symbols subsequently are either sent to an "up" copy of Q if they are greater than the first, and to a"down"copy if not. These two copies of Q then sort the symbols by recursion. When the input stream of Q terminates, Q tells its slaves and they output their contents, sorted, to Q which relays this information to the environment. When the "up" slave (which is activated first) has finished its outputting, Q interrupts the two slaves by inserting the pivotal first element.

The way we will prove this implementation correct is to prove it congruent to a process which is defined in a simple way (like the S_w of 6.7) which it is easy to prove things about.

Define processes A_w, B_w ($w \in K$) as below, where K is the set of linearly ordered strings of T' (in descending order).

$$A_w = (?e \rightarrow B_w) \,\square\, (?x{:}T' \rightarrow A_{u(w,x)})$$
$$B_{\langle\rangle} = !f \rightarrow A_{\langle\rangle}$$
$$B_{\langle y\rangle w} = !y \rightarrow B_w$$

where $u{:}K{\times}T' \rightarrow K$ is defined

$$u(\langle\rangle,y) = \langle y\rangle$$
$$u(w\langle x\rangle,y) = w\langle xy\rangle \text{ if } x \geqslant y$$
$$= u(w,y)\langle x\rangle \text{ otherwise}$$

(That u is a well-defined function in $K{\times}T' \rightarrow K$ is easily proved by induction.)

The main result of this section will be to prove the theorem $Q = A_{\langle\rangle}$.

It is an easy induction on the definitions of X,Y_x,Z_x,Z^* to show that they satisfy (in that order)

$$(C^u_o,C^u_{-1},C^u_{-1},C^u_1) \text{ and also } (C^d_1,C^d_o,C^d_o,C^d_{-1}).$$

We thus know (by 6.5(b) and 6.4) that for all processes R&S $((X^* \| u{::}R) \| d{::}S) = ((X^* \| u{::}S) \| u{::}R)$ and that they are free of internal chatter, where X^* is any one of the four terms above.

It is also easily shown by 6.5(c) and (d) that the Q-recursion is constructive:

$((X \| u::R) \| d::R)\upharpoonright n+1 = ((X \| u::R) \| d::(R\upharpoonright n))\upharpoonright n+1$

(by $C_1^d(X \| u::R)$ which comes by 6.5(d))

$= ((X \| d::(R\upharpoonright n)) \| u::R)\upharpoonright n+1$ (by 6.4)

$= ((X \| d::(R\upharpoonright n)) \| u::(R\upharpoonright n))\upharpoonright n+1$

(by $C_o^u(X \| d::(R\upharpoonright n))$ which comes by 6.5(d))

$= ((X \| u::(R\upharpoonright n)) \| d::(R\upharpoonright n))\upharpoonright n+1$

as required

Claim that $A_{\langle\rangle} = ((X \| u::A_{\langle\rangle}) \| d::A_{\langle\rangle})$

and $w\langle x\rangle v \in K \Rightarrow A_{w\langle x\rangle v} = ((Y_x \| u::A_w) \| d::A_v)$

and $w\langle x\rangle v \in K \Rightarrow B_{w\langle x\rangle v} = ((Z_x \| u::B_w) \| d::B_v)$

and $v \in K \Rightarrow B_v = ((Z^* \| u::A_{\langle\rangle}) \| d::B_v)$

This can be proved by several methods, including 5.15, but we will content ourselves by using the same device as that used in the last example:

let $C_{\langle\rangle} = ((X \| u::A_{\langle\rangle}) \| d::A_{\langle\rangle})$

$\langle\rangle \neq w \in K \Rightarrow C_w = \bigvee_{s\langle x\rangle t=w}((Y_x \| u::A_s) \| d::A_t)$

$\langle\rangle \neq w \in K \Rightarrow D_w = \bigvee_{s\langle x\rangle t=w}((Z_x \| u::B_s) \| d::B_t)$ or $((Z^* \| u::A_{\langle\rangle}) \| d::B_w)$

$D_{\langle\rangle} = ((Z^* \| u::A_{\langle\rangle}) \| d::B\ )$

Now $C_{\langle\rangle} = ?x:T' \rightarrow ((Y_x \| u::A_{\langle\rangle}) \| d::A_{\langle\rangle})$

$\square ?e \rightarrow !f \rightarrow ((X \| u::A_{\langle\rangle})\ \ d::A_{\langle\rangle})$

$= ?x:T' \rightarrow C_x$

$\square ?e \rightarrow ((Z^* \| u::A\ ) \| d::B_{\langle\rangle})$

$= (?x:T \rightarrow C_x) \square (?e \rightarrow D_{\langle\rangle})$

and $C_w = ?y:T' \rightarrow (\bigvee_{\substack{s\langle x\rangle t=w \\ y>x}}((u!y \rightarrow Y_x \| u::A_s) \| d::A_t)$, or

$\bigvee_{\substack{s\langle x\rangle t=w \\ y\leq x}}((d!y \rightarrow Y_x \| u::A_s) \| d::A_t)\ )$

$\square ?e \rightarrow (\bigvee_{s\langle x\rangle t=w}((u!e \rightarrow d!e \rightarrow Z_x \| u::A_s) \| d::A_t))$

$= ?y:T' \rightarrow (\bigvee_{\substack{s\langle x\rangle t=w \\ y>x}}((Y_x \| u::A_{u(s,y)}) \| d::A_t)$ or

$\bigvee_{\substack{s\langle x\rangle t=w \\ y\leq x}}((Y_x \| u::A_s) \| d::A_{u(t,y)})\ )$

$\square ?e \rightarrow (\bigvee_{s\langle x\rangle t=w}((Z_x \| u::B_s) \| d::B_t)\ )$

Now $s\langle x\rangle t \in K$ and $y>x$ implies $u(s\langle x\rangle t,y) = u(s,y)\langle x\rangle t$ and also $y\leq x$ implies $u(s\langle x\rangle t,y) = s\langle x\rangle u(t,y)$, these facts being easy to show by definition of u.

Thus $C_w \sqsupseteq ?y:T' \rightarrow (\bigvee_{s\langle x\rangle t=u(w,y)}((Y_x \| u::A_s) \| d::A_t))$

$\square ?e \rightarrow (\bigvee_{s\langle x\rangle t=w}((Z_x \| u::B_s) \| d::B_t))$

$\sqsupseteq (?y:T' \rightarrow C_{u(w,y)}) \square (?e \rightarrow\ \ _w)$

Similarly (and more easily) it can be shown that

$$D_{\langle y\rangle w} \sqsupseteq !y \to D_w$$

and $D_{\langle\rangle} = !f \to C_{\langle\rangle}$.

The manipulations on the last page can be justified by repeated use of 6.4 and 6.6.

We thus have that $F(\underset{\frown}{C},\underset{\frown}{D}) \sqsubseteq (\underset{\frown}{C},\underset{\frown}{D})$, where F is the function of the combined $\underset{\frown}{A},\underset{\frown}{B}$-recursion. Hence, since it is an easy induction to show that all the A_w and B_w are deterministic, we must have that $\underset{\frown}{A} = \underset{\frown}{C}$ and $\underset{\frown}{B} = \underset{\frown}{D}$ (by the same argument as in the last example).

This gives us that $A_{\langle\rangle} = ((X \| u::A_{\langle\rangle}) \| d::A_{\langle\rangle})$, and since we have already seen that the Q-recursion is constructive this implies that $A_{\langle\rangle} = Q$, by induction.

By this result it is easy to prove that Q sorts its input correctly and is free of deadlock, amongst other things.

It is also possible, using exactly the same recursive scheme, to model the dual algorithm of quicksort, namely "shell sorting". Instead of pivoting on one of the elements of the input stream and recursively sorting the elements above and below the pivot, this works by splitting the input stream into two halves, sorting them and merging the two output streams. The process S below has this behaviour. It can be proved correct in very much the same way as Q (in fact $Q = S = A_{\langle\rangle}$).

$$S \Leftarrow ((X \| a::S) \| b::S)$$

where

$$X \Leftarrow (?x:T' \to (?e \to !x \to !f \to X) \,\square\, (?y:T' \to a!x \to b!y \to X_1)) \,\square\, (?e \to !f \to X)$$

$$X_1 \Leftarrow (?x:T' \to a!x \to X_2) \,\square\, (?e \to a!e \to b!e \to Y)$$

$$X_2 \Leftarrow (?x:T' \to b!x \to X_1) \,\square\, (?e \to a!e \to b!e \to Y)$$

$$Y \Leftarrow a?x:T' \to Y_x$$

$$Y_x \Leftarrow (b?y:\{y \mid y \geq x\} \to !y \to Y_x) \,\square\, (b?y:\{y \mid y < x\} \to !x \to Y'_y) \,\square\, (b?f \to !x \to Z_2)$$

$$Y'_x \Leftarrow (a?y:\{y \mid y \geq x\} \to !y \to Y'_x) \,\square\, (a?y:\{y \mid y < x\} \to !x \to Y_y) \,\square\, (a?f \to !x \to Z_1)$$

$$Z_1 \Leftarrow (b?f \to !f \to X) \,\square\, (b?x:T' \to !x \to Z_1)$$

$$Z_2 \Leftarrow (a?f \to !f \to X) \,\square\, (a?x:T' \to !x \to Z_2)$$

Note that this process still makes a special case of the first symbol it inputs, even though it does not require it as a pivot. It is this special treatment of the first symbol, not sending it to a slave until a second symbol is input, which makes the recursion constructive. Intuitively this behaviour corresponds to not bothering to sort a list of one symbol, since any list of one symbol is already sorted. It is quite easy to show that X satisfies the conditions C_o^a and C_o^b.

Both these recursive algorithms require O(n) time and O(n) processors to sort a list of n symbols. It is obviously possible to refine the definitions of the processes to make them slightly more efficient, and if this were done the following observations would probably remain true. The Q algorithm has the advantage of requiring rather less processors and being more economical on data transmission (both because of the retention of the pivot). The S algorithm has the advantages that both the recursive structure generated in any run and the work load of any given processor are much more predictable (both these being because of the certain division of the input stream into two nearly equal halves, which does not always occur in Q because of the random nature of the pivot).

There is of course no reason why more complicated recursive structures should not be invoked when defining processes with the $(A \| a::B)$ operator. For a simple example of a mutual recursion let X be as it was in defining Q above and let Y be the "X" used in defining S above. Then each of the processes T & U defined below is equal to $S_{\langle\rangle}$.

$$T \Leftarrow ((X \| u::T) \| d::U)$$
$$U \Leftarrow ((Y \| a::T) \| b::U)$$

(This follows quite easily from what we already know, namely that the pair $(S_{\langle\rangle}, S_{\langle\rangle})$ is indeed a fixed point of the recursive function and that this function is constructive and so has a unique fixed point.)

There is a clear sense in which each of the recursions we have seen so far in this chapter using $(A \| a::B)$ can be thought of as defining a network of intercommunicating processes. This network has the form of a directed tree in which the base node communicates with the environment and with its successors (slaves), and all other nodes communicate with their unique predecessors (masters) and their successors (slaves). This tree will normally be infinite and the real world is finite, so it would be unfortunate if in carrying out some finite computation an infinite portion of the tree were used. Consider the following example.

Let $X = ?x \rightarrow a!x \rightarrow \underline{abort}$ (x any element of T)
Trivially X satisfies C_0^a, and so the recursion

$A \Leftarrow (X \| a::A)$ is constructive (with value $?x \rightarrow \underline{abort}$).
However in the network which one expects to correspond with A (an infinite linear tree of "X"s) the simple act of communicating "?x" generates an infinite sequence of communications like an infinite row of toppling dominoes.

It is worrying that the condition C_0^a which is satisfied by the "X" above is exactly the same one which we used to prove some of our example processes correct. (We will in fact find that C_0^a is the only constructive C_i^a which allows this pathological behaviour.) All of our examples do in fact avoid this sort of behaviour, but there is no way of telling this from the final values of the processes. For example, consider the X of example 6.7; let $X^* = ?on \rightarrow a!on \rightarrow X$ and $X' = ?x:T \rightarrow a!on \rightarrow Y_x$ (where "on" is for the purposes of this example some symbol not in T but in the alphabet of the $(A \| a::B)$ operator). Now let $R^* \Leftarrow (X^* \| a::R^*)$ and $R' \Leftarrow (X' \| a:;R^*)$ be defined by recursion. Then each of X' and X* satisfies C_0^a (because X satisfies C_1^a) and $R = R'$ but the definition of R' gives rise to very much the same sort of pathological behaviour as the earlier example.

This sort of pathological behaviour clearly has a lot in common with "infinite internal chatter": both are caused by an infinite number of internal actions occurring in a process after it has only communicated finitely with its environment. The difference is that in the case we are now studying, let us call it network chatter, these actions are spread between an infinite number of distinct hiding operations and it is possible that no individual hiding operation (i.e. link between two processes) gives trouble.

Since we cannot detect the presence or otherwise of network chatter in a complete network of processes from its final value in our model we must try to find some criterion for deciding whether or not it is present from the values of the components of a network. For simplicity, and since it corresponds best with our experience, we will restrict our selves to the consideration of networks defined by recursions of the following type:

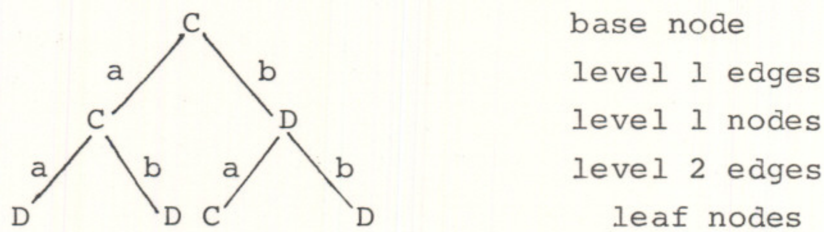
$$B_1 \Leftarrow ((\ldots((A_1 \parallel a_1::B_1) \parallel a_2::B_2) \ldots) \parallel a_n::B_n)$$
$$B_2 \Leftarrow ((\ldots((A_2 \parallel a_1::B_2) \parallel a_2::B_2) \ldots) \parallel a_n::B_n)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$B_n \Leftarrow ((\ldots((A_n \parallel a_1::B_2) \parallel a_2::B_2) \ldots) \parallel a_n::B_n)$$

where each of the A_i contains no recursive calls of any of the B_j in its own definition and all of the names a_i are distinct. We will make two further simplifying assumptions, namely that it is impossible for any of the A_i to engage in infinite internal chatter with any slaves named $a_1,\ldots,a_n$ presented to it in any order, and that the above recursion has a unique fixed point. The justifications for these two assumptions are that firstly it is not worth considering the possibility of network chatter in a network if there is a possibility of the (worse?) condition infinite chatter between two components, and secondly that we need both of them to be able unambiguously to regard such a recursive definition as a network.

The following are a few remarks of a more general nature on the above recursive scheme. Fistly note that this scheme does not prevent one process calling two independent versions of itself or another process (as was done in the two sorting examples) since this can be achieved by inserting two identical A_is. Secondly note that the convention introduced above that the same slave is always addressed by the same name throughout the recursion is of little significance, since the names by which a process addresses its slaves in no way affects the final value of the combination (so long as a few simple rules are observed). Finally note that the fact that we might not want some of the B_i to call all of the other B_j as slaves does not matter either since $(A \parallel a::C) = A$ if dom(A) contains no string with any symbols named by "a".

Remember that the n-tuple $\underset{\sim}{B}$ is defined to be $\bigsqcup_{i=o}^{\infty} F^i(\text{CHAOS}^n)$, where $F(\underset{\sim}{C})_i = ((\ldots(A_i \| a_1::C_1)\ldots) \| a_n::C_n)$ for $\underset{\sim}{C} \in M^n$. Use the notation B_{ij} for the jth approximation $(F^j(\text{CHAOS}^n))_i$ to B_i. Because of our assumptions about freedom from infinite internal chatter we can unambiguously regard each B_{ij} as a finite tree of processes. This tree has a special form, which we must specify and annotate before we can continue.

Define (for A and X non-empty sets) an A-tree of depth n over X as follows. If n=o then it is a tree consisting solely of a base node, which is labelled by some element of X. If n=k+l then it is a tree with base node labelled by some element of X, and from which there is a collection of edges, one labelled with each element of A, each edge leading to the base node of an A-tree of depth k over X. For example, the following is a $\{a,b\}$-tree of depth 2 over $\{C,D\}$.



In an A-tree of depth n one can specify a node by its co-ordinates, which are a string of elements of A with length n or less: if $\langle c_1 \ldots c_k \rangle$ is such a string then it specifies the node which is reached from the base by the path which consists of edges labelled $c_1, \ldots, c_k$ (in that order). If T is such a tree and $\underline{a}$ such a string use the notation $T(\underline{a})$ for the node with co-ordinates $\underline{a}$ in T. If $n \geq 1$ and $a \in A$ denote by $T[a]$ the tree of depth n-1 which is at the end of the level 1 edge labelled a.

With this notation we can think of B_{ij} as a $\{a_1, \ldots, a_n\}$-tree of depth j over M. Define T_{ij}, the tree corresponding to B_{ij}, as follows: if j=o then $T_{ij}(\langle\rangle) = \text{CHAOS}$, otherwise $T_{ij}(\langle\rangle) = A_i$. If $\underline{a}$ has length j then $T_{ij}(\underline{a}) = \text{CHAOS}$; if $\underline{a}$ has non-zero length $< j$ with last component a_k then $T_{ij}(\underline{a}) = A_k$.

Because of the lack of infinite internal chatter it is easy to compound 6.2 to obtain the following result.

6.10 <u>Lemma</u>

For each $1 \leq i \leq n$ and $w \in \Sigma^*$ we have $w \in \text{dom}(B_{ij})$ if and only if there exists a $\{a_1, \ldots, a_n\}$-tree of depth j over Σ^* with the following properties:

(i) For each co-ordinate $\underline{a}$ we have $t(\underline{a}) \in dom(T_{ij}(\underline{a}))$

(ii) $t(\langle\rangle) \upharpoonright (\Sigma - (a_1.\Gamma \cup \ldots \cup a_n.\Gamma)) = w$

(iii) If $\underline{a}$ and $\underline{b}$ are two co-ordinates such that $\underline{a} = \underline{b}\langle a_k\rangle$ then $t(\underline{a}) \upharpoonright (\Sigma - (a_1.\Gamma \cup \ldots \cup a_n.\Gamma)) =$

$$swap?!(strip(a_k)(t(\underline{b}) \upharpoonright a_k.\Gamma))$$

(iv) If $\underline{a}$ is a co-ordinate of length j then <u>either</u> j=0 <u>or</u> $t(\underline{a}) \in \Gamma^*$

(In the above $\Gamma = T \cup ?T \cup !T$, as before.)

Say that such a tree is a tree for w in B_{ij}. Intuitively the tree t represents one possible way in which the processes which make up B_{ij} can co-operate to execute trace w. Say that t is a <u>perfect</u> tree if all its terminal nodes are labelled "<>", and <u>imperfect</u> otherwise. A perfect t corresponds to all of the work being done by the "proper", fully defined components of the network; an imperfect t corresponds to the situation where some of the work is left to the improper "CHAOS" components of the tree.

Suppose t is a $\{a_1,\ldots,a_n\}$-tree of depth r over Σ^* and $s \leq r$. Define $t\upharpoonright s$ to be the tree of depth s such that:

(i) if $\underline{a}$ has length $<s$ then $t\upharpoonright s(\underline{a}) = t(\underline{a})$, and

(ii) if $\underline{a}$ has length s then $t\upharpoonright s(\underline{a}) = t(\underline{a}) \upharpoonright (\Sigma - (a_1.\Gamma \cup \ldots \cup a_n.\Gamma))$.

The following result is a simple consequence of 6.10.

6.11 <u>Lemma</u>

a) If $k<j$ and t is a tree for w in B_{ij} then $t\upharpoonright k$ is a tree for w in B_{ik}.

b) If $k>j$ and t is a perfect tree for w in B_{ij} then $t\uparrow k$ is a tree for w in B_{ik}, where $t\uparrow k(\underline{a}) = t(\underline{a})$ if $|\underline{a}| \leq j$, $t\uparrow k(\underline{a}) = \langle\rangle$ otherwise.

c) If $j>0$ and t is a tree for w in B_{ij} then $t[a_k]$ is a tree for some w' in B_{kj-1}. If t is perfect then so is $t[a_k]$ and at least one of the $t[a_k]$ is imperfect if t is.

Since $dom(B_i) = \bigcap_{j=0}^{\infty} dom(B_{ij})$ we clearly have by (b) above that the existence of a perfect tree for w at any level ensures that $w \in dom(B_i)$. If w has only imperfect trees in B_{ij} we cannot tell whether w $dom(B_i)$; but if $w \in dom(B_i)$ we can interpret the existence of imperfect trees for w in B_{ij} as signifying that B_i, while executing the string w, might make use of its j+1st level. We can use this information to give a precise definition of network chatter in the B_is. Say that B_i admits network chatter on w if for each $m \in N$ there exists $r \geq m$ and tree t such that t is an imperfect tree for w in B_{ir}. Say that B_i contains network chatter if it admits it for any string w. There are several points worth noting about these definitions:

a) They apply only to the case of a recursion of the stated type with a unique fixed point and infinite chatter free components.

b) Because of the finite branching nature of the network of processes which makes up each B_i and the fact that T (the set of communicated symbols) is finite one can apply Konig's lemma to obtain the result that if B_i admits network chatter on w there exists a sequence $t_1, t_2, \ldots$ of trees with the properties that firstly each t_j is a tree for w in B_{ij} and $t_{i,j+1} \upharpoonright j = t_{ij}$ and secondly that infinitely many of the t_j are imperfect. There is a clear sense in which one can think of such a nested sequence of trees as representing a single behaviour of B_i. In this sense a sequence in which infinitely many components are imperfect can be seen to represent a behaviour which includes network chatter (since it involves the use of infinitely many components of the network).

c) There is the possibility that there might exist an imperfect tree for some string w in B_{ij+1} when none exists in B_{ij}. This seems a little paradoxical as it implies that it is possible for B_i to call its j+2th level of recursively defined processes without seeming to call its j+1th level (the level which calls the j+2th level). The situation this represents is the spontaneous occurrence of communication between the j+1th and j+2th levels of processes without prompting by lower levels. This can certainly occur in the type of network we are considering, and while this type of behaviour cannot influence the external behaviour of the system it seems correct to include it in our consideration of network chatter.

The next stage of our work will be to seek sets of conditions which, if satisfied by the processes A_j, ensure that the B_i are free of network chatter. Our first aim will be to ensure that (under certain conditions) the "spontaneous communication" described above cannot occur. This will mean that all activity in the network is the result of chains of command originating at the base node. This will mean that we can to a large extent restrict ourselves to the study of these chains of command. By far the most convenient and practical condition which ensures our first end is the following EF (environment first):

$$EF(A) \Leftrightarrow \forall a \in \Sigma . \langle a \rangle \in dom(A) \Rightarrow a \in \Gamma$$

6.12 <u>Lemma</u>

If each of the A_i in our usual recursion satisfies the condition EF, then whenever t is an imperfect tree for some string w in B_{ij+1} $t \upharpoonright j$ is an imperfect tree for w in B_{ij}. This means that "spontaneous communication", in the sense described above, is impossible.

The proof of this is an easy application of 6.11(a).

Intuitively the condition EF demands that the first communication of a process is with its master/environment and not with any of its slaves.

Note that if A is a process such that $dom(A) \subseteq (\Gamma \cup a_1.\Gamma \cup \ldots \cup a_n.\Gamma)^*$ and $C_o^a(A)$ holds for each j then EF(A) holds also.

Lemma 6.12 formalizes the idea that in studying any condition which is stronger than EF we need only worry about the activity which is in some sense the direct or indirect result of some communication with the environment. What we would like to find is some condition which ensures that all chains of command through the network are finite. The obvious interpretation of the word "command" here is the strings of symbols which pass between masters and their slaves. The obvious method for ensuring that all chains of command are finite is to verify that at all times and for each process in the network the commands given out to a process' slaves are strict reductions in some well-founded partial order of the command received from its own master. This idea is formalized in the next result.

6.13 Theorem

Suppose that $<'$ is some well-founded partial order on Σ^* with unique minimal element $\langle\rangle$. Suppose that the A_i in our usual recursive definition of the B_j all satisfy the following condition:

C) If $w \in dom(A)$ and $w \neq \langle\rangle$ then, for all $1 \leq j \leq n$
   $w \upharpoonright \Gamma >' swap?!(strip(a_j)(w \upharpoonright a_j.\Gamma))$

then none of the B_j contain network chatter.

proof

Suppose for contradiction that the conditions of the theorem hold but that network chatter does exist in some of the B_i. We may suppose without loss of generality that w is minimal in the p.o. $<'$ with respect to giving rise to network chatter in any of the B_i and that B_r is one of the B_i which admits network chatter on w.

Observe first that condition C above implies EF since if $\langle a\rangle \in dom(A)$ we must have $a \in \Gamma$, for otherwise (as $<'$ has unique minimal element $\langle\rangle$) the inequality in C would not be satisfied for any j if this were not so.

By assumption there exists some imperfect tree t for w in B_{rj} for some $j \geq 1$. Applying 6.12 j-1 times we see that there must be an imperfect tree t' for w in B_{r1}. This t' consists of a base node (labelled $v \in dom(A_r)$, say) and n leaf nodes (labelled $v_1,..,v_n$, say). Since t' is imperfect there is some j s.t. $v_j \neq \langle\rangle$. This is easily seen to imply that $v \neq \langle\rangle$, so we can deduce that $w\ (=v \upharpoonright \Gamma) \neq \langle\rangle$ (by $EF(A_r)$).

By definition of network chatter in B_r there must exist some infinite sequence of trees $t_1,t_2,..$ with the properties that each t_i is an imperfect tree for w in some $B_{rj(i)}$, and that the resulting sequence j(i) is strictly increasing. By 6.12 and the fact that $EF(A_j)$ holds for each j we can assume that each t_i is an imperfect tree for w in B_{ri}. By 6.11(c) there must be, for each i, some j such that $t_i[a_j]$ is imperfect. One j at least must be repeated infinitely; we can therefore assume (applying 6.12 once again) that there is some j such that all the t_i have $t_i[a_j]$ imperfect. Each one of these is a tree for some w_i in B_{ji-1}. If there were infinitely many possible values w_i then infinite internal chatter would be possible in the process $((..((A_r \parallel a_1::C)..a_{j-1}::C) \parallel a_{j+1}::C)..a_n::C) \parallel a_j::C)$ at the highest level, where C is an abbreviation for CHAOS. This would contradict our assumptions on the nature of the A_i. We may therefore conclude that there is some w* with the property that infinitely many of the $t_i[a_j]$ are imperfect trees for w* in B_{ji-1}. This tells us that B_j admits network chatter on w*.

However by construction there exists some $v \in dom(A_r)$ (for example any of the base nodes of the t_i such that $t_i[a_j]$ is a tree for w*) such that $w = v\upharpoonright(\Sigma - (a_1.\Gamma \cup .. a_n.\Gamma))$ $w^* = swap?!(strip(a_j)(v\upharpoonright a_j.\Gamma))$. Since condition C holds of A_r we can infer that $w^* <' w$, but this contradicts our assumption that w is minimal with respect to giving rise to network chatter in any of the A_is.

We may therefore conclude that network chatter is impossible in the processes B_j as claimed.

The above result can be strengthened slightly: in place of a partial order on Σ^* one can instead use a partial order on $\Sigma^* \times \{a_1,..,a_n\}$ with joint minimal elements $(\langle\rangle,a_1),..,(\langle\rangle,a_n)$ and require that if process A_i sends command w' to its a_j-slave when it has itself received command w then we must have $w=\langle\rangle$ <u>or</u> $(w,a_i) >' (w',a_j)$. The proof of this strengthened result is a simple adaptation of the above.

Having established a set of conditions which ensure freedom from network chatter we should check to see that each of the examples seen earlier (6.7, 6.8, 6.9) is free from it. The one obvious difficulty here is that all our definitions and results apply only to the standard form of recursion which was set out earlier, and that the two sorting processes do not conform to that pattern since they make multiple calls of identical slaves. As was stated earlier it is possible to recast such recursions into our canonical form by defining two processes by mutual recursion with identical "master" processes. It is easy to see that the various approximating trees (T_{ij}) would be identical in the two versions of such a process. It is of course possible to extend all our definitions and proofs to the case of "multiple calls" with the penalty of requiring more complex notation. When one does this is the natural way it is not hard to show that network chatter exists in a process with "multiple calls" if and only if it exists in the version of the process recast in our canonical form in the natural way.

As suggested earlier our task is simplest in the cases of 6.7 and 6.8 because the "X"s used satisfy C_1^a. That this implies the freedom from network chatter of the stack R and buffer R is a consequence of the following result, itself a corollary to 6.13.

6.14 Theorem

If L is some set of labels and Π some subset of Γ define the condition D_Π^L as follows:

$\forall a \in L . \forall w \in dom(A) .\ w \neq \langle\rangle \Rightarrow |w \restriction \Pi| > |swap?!(stripa(w \restriction a.\Gamma)) \restriction \Pi|$.

Suppose that in our usual recursion each of the processes $A_1,..,A_n$ satisfies D_Π^L, where $L = \{a_1,..,a_n\}$, then each of the processes B_j defined by the usual recursive scheme is free of network chatter.

The partial order used in applying 6.13 to prove this is the one defined $w > v$ if either $|w \restriction \Pi| > |v \restriction \Pi|$ or $w \neq \langle\rangle$ & $v = \langle\rangle$ (or both).

It it easy to see that if a process satisfies each of C_1^a, with "a" ranging over a set of labels L, and if further the process satisfies $dom(A) \subseteq (\bigcup\{\Gamma, a.\Gamma \mid a \in L\})^*$ then it satisfies D_Γ^L. It is this fact which tells us that the definitions of the stack and buffer are free of network chatter.

It remains to show that the two sorting examples are free from network chatter. One way in which one might seek to show a recursion to be "well-defined" is to check that no process outputs as much to its slaves as it has input from its environment. Indeed a close examination of each of the "X"s used in 6.9 will reveal that at no time have they performed as many "a!x"s (outputs to slaves) as they have performed "?x"s (inputs from environment). This suggests that ?T might be a good "Π" to use in applying 6.14. This is in fact so; if we introduce the condition E^L as below it is easy to show that $E^L = D_{?T}^L$ and that a suitable E^L is satisfied by each of the "X" processes of 6.9.

$E^L(A) \Leftrightarrow \forall a \in L . \forall w \in dom(A) .\ w \neq \langle\rangle \Rightarrow |w \restriction ?T| > |w \restriction a!T|$ ($\Leftrightarrow \forall a \in L . E^a(A)$, say)

It is easy to see that all conditions of the form D_{Π}^{L} are strongly continuous. However unlike the C_i^a they do not in themselves imply freedom from infinite internal chatter with arbitrary processes (or even other D_{Π}^{L}-processes) as slaves. If this can be done by some other means there is though a class of restriction operators closely related to D_{Π}^{L} which has an interesting and useful theory in relation to master/slave recursion via D_{Π}^{L}-processes. This class of operators is the one where $A\upharpoonright n$ behaves like A until it has communicated n elements of π and then dissolves into CHAOS. We will not follow up this subject here however.

Let us sum up this section. Having identified a certain unfortunate type of possible behaviour in recursively defined networks we discovered that its presence did not appear to influence the external behaviour of a network. We established a plausible formal definition of network chatter over the class of recursive definitions which we could most easily regard as networks, and in that case derived laws which could be shown to imply that it was absent (in the formally defined sense). One might choose to regard the insensitiveness of our model to this condition as a weakness in the model; this and other related problems will be discussed in chapter eight. The crucial fact about the type of recursive tree discussed in the latter part of this chapter was that it could be thought of as the limit of a sequence of finite networks. It should be reasonably easy to extend the ideas used to other infinite networks with this property, provided that one could prove that the notions of network chatter introduced were well-defined in that they were independent of the convergent sequence of networks used. (This is quite easy to do for the networks we have already met.) It is not quite so clear what one should do in cases (common in chapter five) where recursive calls of processes are made not only through parallel operators but also guarded (5.31 for example). Intuitively these recursions define networks which are of a less clear-cut type, but which somehow appear to be less prone to this type of behaviour. Again one could possibly define corresponding trees of processes (of a less simple nature) relative to which one could define and avoid infinite chains of recursive calls.

Postscript: Proving C_i^a and E^L of simple processes.

The conditions C_i^a and E^L admit a fairly simple and mechanical method of proof which can be applied to processes defined by tail recursion. Say that a process is defined by tail recursion if it is written in the form:

$$B_\lambda, \text{ where } \begin{array}{l} \xi \in \Gamma_1 \Rightarrow B_\xi \Leftarrow A_1 \\ \cdots\cdots \\ \cdots\cdots \\ \xi \in \Gamma_s \Rightarrow B_\xi \Leftarrow A_s \end{array}$$

and each of the A_i is formed from the syntax consisting of "skip", "abort", "a →", "x:U →", "a.x:U →" and "□" together with recursive calls of the B_λ, each of which has the property that it can be syntactically deduced exactly which of the Γ_i it must inevitably fall in. Note that each of the "X"s used in examples 6.7 - 6.9 is of this form.

If "a" is a label define the condition E_i^a as follows:

$$E_i^a(A) \Leftrightarrow \forall w \in \mathrm{dom}(A).\ |w\restriction ?T| \geqslant \min(|w\restriction a!T|+i, |w\restriction(?T\cup a!T)|)$$

This condition is plainly both strongly continuous and of a very similar form to C_i^a. It is easy to see that $E^L(A) \Leftrightarrow (\forall a \in L.E_1^a(A))$ & $A^O \subseteq ?T$. This means that any method we develop to prove the E_i^a can be used to prove the E^L.

In order to develop our method we will need a list of technical lemmas, which we will later combine to produce inductive proofs.

6.15 Lemma

a) Suppose that A & B are processes satisfying C_i^a and C_j^a respectively, then

(i) $C_k(c \to A)$ holds, where $k = i+1$ if $c \in \Gamma$
$= \min(-1, i-1)$ if $c \in a.\Gamma$
$= i$ otherwise;

(ii) $C_k(A \,\square\, B)$ holds, where $k = \min(i,j)$.

b) Suppose that A & B are processes satisfying E_i^a and E_j^a respectively, then

(i) $C_k(c \to A)$ holds, where $k = i+1$ if $c \in ?T$
$= \min(-1, i-1)$ if $c \in a!T$
$= i$ otherwise;

(ii) $C_k(A \,\square\, B)$ holds, where $k = \min(i,j)$.

c) C_i^a and E_i^a always hold of skip and abort (because "√" can never be in T).

6.16 Lemma

a) Suppose that $U \subseteq T$ and that for each $x \in U$ we have $C_i^a(A_x)$, then

(i) $C_{i+1}^a(x{:}U \to A_x)$, $C_{i+1}^a(!x{:}U \to A_x)$, $C_{i+1}^a(?x{:}U \to A_x)$;

(ii) $C_i^a(b.x{:}U \to A_x)$ if $b \notin \{a, a?, a!, ?, !\}$;

(iii) $C_j^a(a.x{:}U \to A_x)$, $C_j^a(a!x{:}U \to A_x)$, $C_j^a(a?x{:}U \to A_x)$, where $j = \min(-1, i-1)$.

b) Suppose that $U \subseteq T$ and that for each $x \in U$ we have $E_i^a(A_x)$, then

(i) $E_{i+1}^a(?x{:}U \to A_x)$;

(ii) $E_j^a(a!x{:}U \to A_x)$, where $j = \min(-1, i-1)$;

(iii) $E_i^a(b.x{:}U \to A_x)$, $E_i^a(x{:}U \to A_x)$ if $b \notin \{?, a!\}$

Now suppose that we have a process which is defined by tail recursion, and that it is written

$$
\begin{array}{lll}
B \text{ , where} & \zeta \in \Gamma_1 \Rightarrow & B_\zeta \Leftarrow A_1 \\
& \cdots\cdots & \\
& \cdots\cdots & \\
& \zeta \in \Gamma_s \Rightarrow & B_\zeta \Leftarrow A_s \quad .
\end{array}
$$

One will often be able to find upper bounds for the strengths of conditions satisfied by some of the above clauses from their lowest recursive level (by application of the "min" clauses). For example, recalling the definition of Y_x in the buffer example $Y_x \Leftarrow (?y{:}T \to a!x \to Y_x) \,\square\, (!x \to Z)$, it is clear that Y_x cannot satisfy either of C_1^a or E_1^a. Using knowledge of this type and one or two iterations of the recursion one can extend these bounds throughout the recursion. Thus in the buffer example we cannot expect "X" to satisfy C_2^a because it is defined $X \Leftarrow ?x{:}T \to Y_x$ and Y_x never satisfied C_1^a.

Suppose now that one has developed a hypothesis of the form <u>either</u> $(\zeta \in \Gamma_i \Rightarrow C_{k(i)}^a(B_\zeta))$ <u>or</u> $(\zeta \in \Gamma_i \Rightarrow E_{k(i)}^a(B_\zeta))$. A <u>test</u> of such a hypothesis will be the action of assuming that it is true of the vector $\underset{\sim}{B}$ and seeing if it remains true of $F(\underset{\sim}{B})$, where F is the function associated with the $\underset{\sim}{B}$-recursion. Such a test can be carried out very easily because of our assumption that each recursive call must fall within exactly one of the Γ_i, and by Lemmas 6.15 and 6.16.

It is also extremely easy to check that recursions of our given form are constructive (for example such recursions are constructive if all recursive calls are guarded). Since all predicates of the form C_i^a and E_i^a are trivially satisfiable the following is a trivial application of our inductive principle.

6.17 Theorem

If in a constructive tail recursion of the form given above we have a hypothesis either of the form $(\zeta \in \Gamma_i \Rightarrow C^a_{k(i)}(A_\zeta))$ or $(\zeta \in \Gamma_i \Rightarrow E^a_{k(i)}(A_\zeta))$ which is testable then we can infer that it holds of the vector $\underset{\sim}{B}$.

6.18 Example

To prove E_1^u and E_1^d of the "X" used in 6.9 (Quicksort).

step 1

Recall our recursive definition:

$$X \Leftarrow (?x{:}T' \to Y_x) \,\square\, (?e \to\, !f \to X)$$
$$Y_x \Leftarrow (?y{:}\{y \mid y \geqslant x\} \to u!y \to Y_x) \,\square\, (?y{:}\{y \mid y < x\} \to d!y \to Y_x)$$
$$\square (?e \to u!e \to d!e \to Z_x)$$
$$Z_x \Leftarrow (u?y{:}T' \to\, !y \to Z_x) \,\square\, (u?f \to\, !x \to Z^*)$$
$$Z^* \Leftarrow (d?y{:}T' \to\, !y \to Z^*) \,\square\, (d?f \to\, !f \to X)$$

It is clear that we cannot expect Y_x to satisfy any stronger E^u condition than E_o^u or any stronger E^d condition than E_o^d. By substituting this into X we see that E_1^u and E_1^d are the maximum conditions satisfied by X, and also by Z* and Z_x. Let us therefore take as our hypothesis that X satisfies E_1^u and E_1^d, each Y_x satisfies E_o^u and E_o^d, each Z_x satisfies E_1^u and E_1^d, and that Z* satisfies E_1^u and E_1^d. We will just check the E^u-hypothesis since the other is very similar.

step 2

Assume that the vector $(X', \underset{\sim}{Y}', \underset{\sim}{Z}', Z^{*\prime})$ satisfies our E^u hypothesis. Then we get

$E_o^u(Y'_x)$ $(x \in T')$	$E_1^u(X')$
$\Rightarrow E_1^u(?x{:}T' \to Y'_x)$	$\Rightarrow E_1^u(!f \to X')$
	$\Rightarrow E_2^u(?e \to\, !f \to X')$

$\Rightarrow E_1^u((?x{:}T' \to Y'_x) \,\square\, (?e \to\, !f \to X'))$ as required.

. . . .

$E_o^u(Y'_x)$	$E_o^u(Y'_x)$	$(x \in T')$
$\Rightarrow E_{-1}^u(u!y \to Y'_x)$	$= E_o^u(d!y \to Y'_x)$	$(x,y \in T')$

$\Rightarrow E^u_0(?y:\{y|y\geq x\}\rightarrow u!y \rightarrow Y'_x)$ | $\Rightarrow E^u_1(?y:\{y|y<x\}\rightarrow d!y \rightarrow Y'_x)$

$\Rightarrow E^u_0((?y:\{y|y\geq x \rightarrow u!y \rightarrow Y'_x) \,\square\, (?y:\{y|y<x\}\rightarrow d!y \rightarrow Y'_x))$

and $E^u_1(Z'_x)$

$\Rightarrow E^u_1(d!e \rightarrow Z'_x)$

$\Rightarrow E^u_0(u!e \rightarrow d!e \rightarrow Z'_x)$

$\Rightarrow E^u_1(?e \rightarrow u!e \rightarrow d!e \rightarrow Z'_x)$

$= E^u_0(((?y:\{y|y\geq x\}\rightarrow u!y \rightarrow Y'_x) \,\square\, (?y:\{y|y<x\}\rightarrow d!y \rightarrow Y'_x))$
$\square(?e \rightarrow u!e \rightarrow d!e \rightarrow Z'_x))$

as required.

. . . .

$E^u_1(Z'_x)$ | $E^u_1(Z^{*\prime})$ $(x\in T')$

$\Rightarrow E^u_1(!y \rightarrow Z'_x)$ | $\Rightarrow E^u_1(!x \rightarrow Z^{*\prime})$

$\Rightarrow E^u_1(u?y:T'\rightarrow !y \rightarrow Z'_x)$ | $\Rightarrow E^u_1(u?f \rightarrow !x \rightarrow Z^{*\prime})$

$= E^u_1((u?y:T'\rightarrow !y \rightarrow Z'_x) \,\square\, (u?f \rightarrow !x \rightarrow Z^{*\prime}))$ as required

. . . .

$E^u_1(Z^{*\prime})$ | $E^u_1(X')$

$\Rightarrow E^u_1(!y \rightarrow Z^{*\prime})$ | $\Rightarrow E^u_1(!f \rightarrow X')$

$\Rightarrow E^u_1(d?y:T'\rightarrow !y \rightarrow Z^{*\prime})$ | $\Rightarrow E^u_1(d?f \rightarrow !f \rightarrow X')$

$\Rightarrow E^u_1((d?y:T'\rightarrow !y \rightarrow Z^*) \,\square\, (d?f \rightarrow !f \rightarrow X'))$ as required

The above, together with the fact that the recursion is constructive, tells that $E^u_1(X)$ holds as desired.

These proofs can be recast in a tabular form, assigning a number to each point in the syntax of a process. For example the second clause of the above proof could be re-written

```
(((?y:{y|y≥x}→ u!y → Y'x) □ (?y:{y|y<x}→ d!y → Y'x))
      o       -1   o o    1          o    o
□ (?e → u!e → d!e → Z'x))
o   1     o     1     1
```

(The underlined "o" represents the highest syntactic level.)

Chapter 7 :- Alternative Parallel Combinators

In the first half of this chapter we will briefly study the theories of a few more parallel/hiding combinators. In the second half we will examine the important problem of how one can prove networks of processes free from deadlock.

The following is a list of a few possible parallel/hiding combinators which we might wish to use.

a) A bidirectional "pipe" operator "$\bowtie$" in which processes are connected very much in the same way as in the old pipe "$\gg$". A process will now though be expected to be able to input and/or output down two named channels "l" and "r"; the "left" process will have its right hand ("r") outputs and inputs connected to the left hand ("l") inputs and outputs of the other "right" process.

$$(A \bowtie B) = (swap?!(stripr(A))\ {}_X\|_Y\ stripl(B)))/(?T \cup !T),$$

where $X = l!T \cup l?T \cup ?T \cup !T$

$Y = r!T \cup r?T \cup ?T \cup !T$

b) $(A_X \leftrightarrow_Y B) = (A_X\|_Y B)/(X \cap Y)$, the operator in which two processes running ordinarily in parallel have all their intercommunication hidden.

c) $[A_{ij}]_{m,n}$, in which an mxn matrix of processes operate in parallel, each communicating with its four immediate neighbours.

```
A11 - A12 - ...... - A1n
 |     |              |
A21 - A22 - ...... - A2n
 |     |              |
 . . . . . . . . . . . .
 |     |              |
Am1 - Am2 - ...... - Amn
```

For a precise definition of this operator see later.

d) $\langle A_{ij} \rangle_{m,n}$, in which an mxn matrix of processes is arranged in a hexagonally connected array instead of the square above.

```
A11 - A12 - A13 - ..... - A1n
  \   / \   / \          / \
   A21 - A22 - ........... - A2n
     \   / \             / \
      . . . . . . . . . . . . .
       \ /   \ /             \
       Am1 - Am2 - ........... - Amn
```

To some extent the operator "$\leftrightarrow$" defined in (b) above is the most basic operator of its type, all the others being derived from it and alphabet transformers. Observe how both of our old operators may be defined:

$$A \gg B = (\text{strip}!A_{T\cup ?T}\!\leftrightarrow_{T\cup !T}\text{strip}?B)$$

$$(A \parallel a::B) = (A_{\Sigma}\!\leftrightarrow_{a.\Gamma} a.\text{swap}!?(B)) \quad .$$

The conditional associativity of "$\gg$" and the conditional commutativity of $(A \parallel a::B)$ can both be deduced from the above definitions and the following lemma, which is as one might expect a consequence of 5.35.

7.1 <u>Lemma</u>

If $(A_X\!\leftrightarrow_Y B)$ and $(B_Y\!\leftrightarrow_Z C)$ are both free of infinite internal chatter (in the obvious sense) and $X\cap Y\cap Z = \emptyset$ then

$$((A_X\!\leftrightarrow_Y B)_{X\cup Y}\!\leftrightarrow_Z C) = (A_X\!\leftrightarrow_{Y\cup Z}(B_Y\!\leftrightarrow_Z C)) \quad .$$

<u>proof</u>

Suppose that the hypotheses of the lemma hold, then

$$\begin{aligned}((A_X\!\leftrightarrow_Y B)_{X\cup Y}\!\leftrightarrow_Z C) &= ((A_X\|_Y B)/(X\cap Y)_{X\cup Y}\|_Z C)/((X\cup Y)\cap Z) \\ &= ((A_X\|_Y B)_{X\cup Y}\|_Z B)/((X\cap Y)\cup(X\cap Z)\cup(Y\cap Z)) \quad (5.35) \\ &= (A_X\|_{Y\cup Z}(B_Y\|_Z C))/((X\cap Y)\cup(X\cap Z)\cup(Y\cap Z)) \\ &= (A_X\|_{Y\cup Z}(B_Y\|_Z C)/(Y\cap Z))/(X\cap(Y\cup Z)) \quad (5.35) \\ &= (A_X\!\leftrightarrow_{Y\cup Z}(B_Y\!\leftrightarrow_Z C)) \quad \text{as desired.}\end{aligned}$$

The "meaning" of this lemma is that in a network so long as there can be no confusion about the destination of any message ($X\cap Y\cap Z = \emptyset$) and there is no infinite chatter, it does not matter how the network was constructed.

Let us study the structure of the combinator $[A_{ij}]_{mn}$ introduced in (c) above. It is fundamentally different from our other ones in that the structures it creates are not normally trees, and therefore it is likely to introduce loops. So far we have not specified the exact nature of the operator; we will expect it to achieve the following:

a) Each A_{ij} will have four channels u,d,l,r. The "u" channel of $A_{i+1,j}$ will be connected to the "d" channel of A_{ij}, and the "r" channel of A_{ij} will be connected to the "l" channel of $A_{i,j+1}$ (all internal communication being hidden).

b) The row of n accessible "u" channels will be addressed by the names u.i ($i\in\{1,..,n\}$), the m accessible "l" channels by the names l.i ($i\in\{1,..,m\}$), etc.

c) The communications down each channel will be in T, the usual finite set of unnamed symbols. (If desired the operators we define can easily be adapted to assuming communication in ?T∪!T with inputs being connected to outputs and vice-versa.)

From our experience with other operators it appears that there is some ambiguity left in this definition, this resulting from the many possible orders of putting such a network together and hiding the internal communication. We might also expect this ambiguity to disappear when the network is free of infinite internal chatter, as we should then be in a position to apply 7.1.

It is clear that it is possible to construct arbitrarily large matrices with the following combinators.

7.2 <u>Definitions</u>

(i) Define $X(a,b,c,d,r,s,t,u) = \bigcup_{i=r}^{s}(a.i.T \cup b.i.T) \cup \bigcup_{i=t}^{u}(c.i.T \cup d.i.T)$ for labels a,b,c,d and integers r,s,t,u.

(ii) $[A]$ = swapu(u.1)(swapd(d.1)(swapl(l.1)(swapr(r.1)(A)))) This is the combinator which produces a 1x1 matrix from a single process with u,d,l,r channels.

(iii) ${}_n[A|_m B]_s$ = (swaprb(A) $_Y\!\leftrightarrow_Z$ swaplb(inc{u,d}(r,n)(B))) where $Y = X(l,b,u,d,1,m,1,n)$, $Z = X(b,r,u,d,1,m,n+1,n+s)$, "b" is a label distinct from u,d,l and r, and inc{e,f}(t,s)(A) = swap(e.1)(e.s+1)(swap(f.1)(f.s+1)(..swap(f.t)(f.t+s)(C))..).

(iv) $\left[\frac{A}{B}\right]^{m}_{\substack{n\\s}}$ = (swapda(A) $_Y\!\leftrightarrow_Z$ swapua(inc{l,r}(s,m)(B))) , where $Y = X(l,r,u,a,1,m,1,n)$, $Z = X(l,r,a,d,m+1,m+s,1,n)$, and "a" is a label distinct from b,u,d,l and r.

These combinators join blocks together, joining mxn and mxs blocks to make a mx(n+s) matrix, and mxn and sxn blocks to make a (m+s)xn matrix respectively.

In future we will habitually suppress the dimension parameters (n,m,s above) when they are obvious from their context. One might think that it is possible to drop the "central" parameter (m in (iii) and n in (iv)), but if we were to do this it would be necessary to hide an infinite alphabet (there being no bound on the possible integer part of the labels of communications).

There are clearly many ways in which one could produce a definition of $[A_{ij}]_{nm}$ from these three combinators. For example, the number of different ways of producing a 1xn or a nx1 matrix is $\frac{1}{n}\binom{2n-2}{n-1}$, and the first few terms in the table of the number of ways of producing a nxm matrix are shown below.

m \ n	1	2	3	4	5	6
1	1	1	2	5	14	42
2	1	2	8	45	318	2644
3	2	8	64	770	13008	290544
4	5	45	770	19450	729148	41031312
5	14	318	13008	729148	57378464	7222570064
6	42	2644	290544	41031312	7222570064	****

where **** = 1816170558336

The generating relations of this table are

$$t_{11} = 1$$

$$i \neq 1 \text{ or } j \neq 1 \Rightarrow t_{ij} = \sum_{k=1}^{j-1} t_{ik} \cdot t_{ij-k} + \sum_{k=1}^{i-1} t_{kj} \cdot t_{i-kj} \quad .$$

In proving the (conditional) independence of the final value from the method of construction the following three lemmas are vital.

7.3 <u>Lemma</u>

If A,B,C represent mxn, rxn and sxn matrices respectively (i.e. if their alphabets are consistent with this) and each of $\left[\frac{A}{B}\right]$ and $\left[\frac{B}{C}\right]$ is free of infinite internal chatter then

$$\left[\frac{\left[\frac{A}{B}\right]}{C}\right] = \left[\frac{A}{\left[\frac{B}{C}\right]}\right] .$$

7.4 <u>Lemma</u>

If A, B, C represent nxm,nxr and nxs matrices respectively and each of [A|B] and [B|C] is free of infinite internal chatter then [[A|B]|C] = [A|[B|C]].

7.5 <u>Lemma</u>

If A,B,C,D represent nxm, nxs, txm and txs matrices respectively and each of [A|B], [C|D], $\left[\frac{A}{C}\right]$, $\left[\frac{B}{D}\right]$, $\left[\frac{[A|B]}{[C|D]}\right]$ and $\left[\left[\frac{A}{C}\right]\middle|\left[\frac{B}{D}\right]\right]$ is free of infinite internal chatter, then

$$\left[\frac{[A|B]}{[C|D]}\right] = \left[\left[\frac{A}{C}\right]\middle|\left[\frac{B}{D}\right]\right].$$

proof

The proofs of 7.3 and 7.4 are very similar to the proof of 5.19 (the associativity of "$\gg$") for obvious reasons. We will therefore content ourselves with a sketch proof of 7.5 (a result which is another elaborate corollary to lemma 7.1).

$$\left[\frac{[A|B]}{[C|D]}\right] = (\mathrm{swapda}[A|B]\ {}_{Y}\!\leftrightarrow_{Z} \mathrm{swapua}(\mathrm{inc}\{1,r\}(t,n)[C|D]))$$

where $Y = X(1,r,u,a,1,n,1,m+s)$, $Z = X(1,r,a,d,n+1,n+t,1,m+s)$

$$= ((A^*\ {}_{U}\!\leftrightarrow_{V} B^*)\ {}_{Y}\!\leftrightarrow_{Z} (C^*\ {}_{W}\!\leftrightarrow_{R} D^*)), \qquad (*)$$

where
$A^* = \mathrm{swapda}(\mathrm{swaprb}(A))$
$B^* = \mathrm{swapda}(\mathrm{swaplb}(\mathrm{inc}\{u,d\}(s,m)(B)))$
$C^* = \mathrm{swapua}(\mathrm{inc}\{1,b\}(t,n)(\mathrm{swaprb}(C)))$
$D^* = \mathrm{swapua}(\mathrm{inc}\{b,r\}(t,n)(\mathrm{swapbl}(\mathrm{inc}\{u,d\}(s,m)(D))))$

$U = X(1,b,u,a,1,n,1,m)$, $V = X(b,r,u,a,1,n,m+1,m+s)$
$W = X(1,b,a,d,n+1,n+t,1,m)$, $R = X(b,r,a,d,n+1,n+t,m+1,m+s)$.

Now $U \cup V \supseteq Y$ and $W \cup R \supseteq Z$, and so it is easy to see that $(U \cup V)$ and $(W \cup R)$ can be substituted for Y and Z in (*) above without changing the value.

Also $(U \cup V) \cap W \cap R = \emptyset$, and since infinite internal chatter is absent by assumption we get that (*) is equal to

$$(((A^*\ {}_{U}\!\leftrightarrow_{V} B^*)\ {}_{U\cup V}\!\leftrightarrow_{W} C^*)\ {}_{U\cup V\cup W}\!\leftrightarrow_{R} D^*) \qquad \text{by 7.1}$$

$$= (((A^*\ {}_{U}\!\leftrightarrow_{W} C^*)\ {}_{U\cup W}\!\leftrightarrow_{V} B^*)\ {}_{U\cup V\cup W}\!\leftrightarrow_{R} D^*) \qquad \text{" "}$$

$$= ((A^*\ {}_{U}\!\leftrightarrow_{W} C^*)\ {}_{U\cup W}\!\leftrightarrow_{V\cup R} (C^*\ {}_{V}\!\leftrightarrow_{R} D^*)) \qquad \text{" "}$$

This is readily shown to be equal to $\left[\left[\frac{A}{C}\right]\middle|\left[\frac{B}{D}\right]\right]$, as desired.

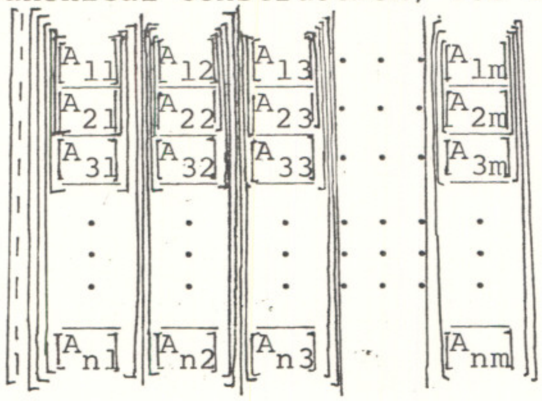
(The various manipulations required to fill out this outline proof are easy but tedious.)

Having established these results we can now prove that under certain conditions we can disregard the order of construction of matrices.

7.6 Lemma

Suppose that $\{A_{ij} \mid 1 \leq i \leq n,\ 1 \leq j \leq m\}$ is a set of processes such that every construction of a (not necessarily proper) sub-matrix of A_{ij} using the constructs of 7.2 is free of infinite internal chatter. Then all possible constructions of the matrix $[A_{ij}]_{nm}$ give rise to the same value.

This result is not difficult to prove by induction on the dimensions of the matrix. One proves that all possible ways of constructing the matrix are equivalent to some canonical construction, for example



(This is the construction where the columns are put together first, associating to the top; then the complete columns are put together, associating from the left.)

Let us conventionally adopt the definition that $[A_{ij}]_{nm}$ is the above form (in every case, whether or not it is free of infinite internal chatter). The force of lemma 7.6 is that the particular canonical form chosen is irrelevant in all cases where infinite internal chatter is impossible.

The table which we saw earlier is a demonstration of the fact that we cannot expect to prove that the conditions of 7.6 are satisfied by examination of cases. We therefore need to find some general method for proving the absence of infinite internal chatter from networks. An example of such a method is provided by the next result, which is similar in statement, effect and proof to 5.20.

7.7 <u>Lemma</u>

If "a" is any label define the predicate P^a on M as follows:

$$P^a(A) \equiv \neg\exists\, w_o < w_1 < \ldots < w_i < \ldots \in \mathrm{dom}(A).\ \forall i.(w_i \restriction a.\Sigma) = (w_o \restriction a.\Sigma)\ .$$

(i) If A and B are two processes satisfying P^a $(a \in \{u,d,l,r\})$ then so do $[A]$, $[A|B]$ and $\left[\frac{A}{B}\right]$.

(ii) If each of A and B satisfies P^a $(a \in \{u,d,l,r\})$ then $[A|B]$ and $\left[\frac{A}{B}\right]$ are both free of infinite internal chatter.

The above result tells us that if each A_{ij} of some matrix satisfies the <u>same</u> P^a $(a \in \{u,d,l,r\})$ then the conditions of 7.6 are satisfied. The critical feature about any predicate with this property is that it satisfies (i) above, as well as implying freedom from infinite internal chatter, for then it implies that all the partial constructions of $[A_{ij}]_{nm}$ also satisfy it. The four conditions P^u, P^d, P^l and P^r correspond intuitively to ensuring that there is always some flow of information towards one of the four faces of a matrix. This motivates the following four conditions, which correspond to the corners of the matrix in much the same way as the P^a correspond to the faces.

7.8 Lemma

If a and b are two distinct labels define the predicate Q^a_b as follows:

$$Q^a_b(A) = \neg\exists\, w_0 < w_1 < \ldots < w_i \ldots \in \mathrm{dom}(A).\forall i.\ w_i \upharpoonright (a.\Sigma \cup b.\Sigma) = w_0 \upharpoonright (a.\Sigma \cup b.\Sigma).$$

(i) If $a \in \{u,d\}$ and $b \in \{l,r\}$ and A,B are two processes satisfying Q^a_b then [A], [A|B] and $\left[\frac{A}{B}\right]$ all satisfy Q^a_b.

(ii) If $a \in \{u,d\}$, $b \in \{l,r\}$ and each of A and B satisfies Q^a_b then [A|B] and $\left[\frac{A}{B}\right]$ are both free of infinite internal chatter.

Note that $P^a \Rightarrow Q^a_b$ and $P^b \Rightarrow Q^a_b$, so this second set of predicates is more general than the first.

Note also that the conditions Q^u_d and Q^l_r satisfy neither (i) nor (ii) above. For an example of this consider the two processes $A \Leftarrow d.a \rightarrow A$ and $B \Leftarrow u.a \rightarrow B$, which both satisfy Q^u_d trivially but for which $\left[\frac{[A]}{[B]}\right]$ does nothing but infinite internal chatter.

In common with the conditions used in a similar way in 5.20 the predicates P^a and Q^a_b are unfortunately not continuous, but again there are large classes of continuous predicates which imply them. (For example any predicate which expresses a bound on the number of "wrong" symbols which can appear before every "correct" one.)

It is clear that the "hexagonally connected array" can be defined in a very similar manner to the rectangular matrix of the above discussion. There are several excellent algorithms making use of arrays of these forms for such things as matrix multiplication and inversion as well as the more obvious uses such as the numerical solution of partial differential equations. For a description of a number of these see Mead and Conway ( ).

There are clearly many other possible configurations for parallel processes which we have not defined or studied so far. These include three (and higher) dimensional arrays; arrays with more complex interconnection; arrays in which all processes can be individually addressed by the environment; rings and even spheres of processes. It is not hard to adapt the techniques we have used so far to produce a reasonable definition for any of these. It is clear from the work we have done to date that we can expect each to have its own characteristic set of theorems, but that many of these theorems will follow set patterns.

Deadlock in networks

Deadlock is an important subject in the study of networks of parallel processes. So far when we have studied particular processes which have been defined as networks (in chapters 5 and 6) the desirable feature "freedom from deadlock" has almost always been proved as a corollary to a more powerful result. We have either proved that our networks were equivalent to other processes which were known to be free of deadlock (as in 6.8 and 6.9) or proved that the value of a network satisfied some predicate which implied freedom from deadlock (as in many of the buffer examples of chapter 5). These two techniques both have a worthwile place in our repertoire, but there are certainly going to be times when neither is applicable. Since freedom from deadlock is of such fundamental importance it is worthwile to try to find other methods for establishing it. The following is not an extensive treatment of such methods, merely a summary of a few ways in which the techniques we have developed might be applied to the problem.

The author believes that theorem 5.14 could be applied in many cases, in much the same way as it was applied in the proof of 5.27 (the "fundamental buffer theorem"). If one could identify a finite or infinite set of "states" of a network, and could prove some simple constructive relation between these states (very much as in 5.27) it would be possible to deduce freedom from deadlock (so long as the constructive relation preserves it).

So long as we can use 7.1 (and other similar results) to bring all the hiding of a definition to the outside (so that it has the form A/X, where $\surd\notin\Sigma$ and the definition of A is free from hiding), and if we can show that the definition is free from infinite chatter, then we restrict ourselves to proving freedom from deadlock in the process before any hiding is carried out ("A" above). This is because, in the absence of infinite internal chatter, the only way deadlock (the refusal of "Σ" after some string) can occur in A/X is when A itself can refuse "Σ" after some possibly different string.

For example, in the process $A = ((B\,{}_X\|_Y C)/Z\,{}_{X\cup Y}\|_{U\cup V}(D\,{}_U\|_V E)/W)/S$, where $Z\cap(U\cap V) = W\cap(X\cap Y) = \emptyset$, if each of the hiding operators individually is free of infinite chatter, then to prove A free from deadlock it is sufficient to prove it in $((B\,{}_X\|_Y C)\,{}_{X\cup Y}\|_{U\cup V}(D\,{}_U\|_V E))$.

This fact has several uses, not the least of which is the fact that by effectively eliminating hiding from our consideration we can generally expect it to be much easier to find the constructive relations between states required for the previous method suggested. It can also be used to reduce some apparently complex problems to forms to which the next class of methods is applicable.

It is well known that networks which have the form of trees are generally easier to prove free from deadlock than those which possess loops. This fact is brought out by the next few results.

7.9 Theorem

Suppose that $A_1,\ldots,A_n$ is a finite collection of processes $(n\geq 2)$ with associated alphabets $X_1,\ldots,X_n$, and that A^* is the result $((\ldots((A_1\,{}_{Y_1}\|_{X_2}A_2)\,{}_{Y_2}\|_{X_3}A_3)\,{}_{Y_3}\|\ldots)\,{}_{Y_{n-1}}\|_{X_n}A_n)$ of combining them in parallel (where $Y_i = X_1\cup\ldots\cup X_i$). Suppose further that the A_i and X_i satisfy the following conditions:

(i) each pair $(A_i\,{}_{X_i}\|_{X_j}A_j)$ $(i\neq j)$ is free of deadlock;

(ii) if i,j,k are all different then $X_i\cap X_j\cap X_k = \emptyset$;

(iii) for each i and string w $(A_i\ \underline{\text{after}}\ w)^o$ has non-empty intersection with at most one of the X_i s.t. $i\neq j$,

then A^* can deadlock on string w only if there is a sequence $n_1,\ldots,n_k$ of distinct elements of $\{1,\ldots,n\}$ with the properties set out below.

(i) k (the length of the sequence) is at least three.

(ii) Letting $w_i = w \upharpoonright X_{n_i}$, $Z_i = X_{n_i}$ and $B_i = A_{n_i}$, we have $w_i \in dom(B_i)$ for each $i \in \{1,..,k\}$.

(iii) There is a sequence $W_1,...,W_k$ of subsets of Σ such that (for each i) W_i is a maximal element of $B_i(w_i)$ and $Z_i - W_i$ is a non-empty subset of Z_{i+1} (or Z_1 when $i=k$).

<u>proof</u>

If A* can deadlock after w then $\Sigma \in A^*(w)$ (by definition of deadlock). It is an easy consequence of the definition of the parallel combinator that for any string v and set V we have $(v,V) \in A^*$ if and only if $v \in (X_1 \cup ... \cup X_n)^*$ and there exist sets $V_1,...,V_n$ such that

a) $V_i \in A_i(v \upharpoonright X_i)$ for each i, and
b) $V \cap (X_1 \cup ... \cup X_n) = (V_1 \cap X_1) \cup ... \cup (V_n \cap X_n)$.

In the case when $V = \Sigma$ we may clearly assume that each of the sets V_i is maximal in $A_i(v \upharpoonright X_i)$. Thus if A* can deadlock after string w we can deduce that there exist sets $V_1,..,V_n$ such that

a) V_i is a maximal element of $A(w \upharpoonright X_i)$ for each i, and
b) $(X_1 \cup ... \cup X_n) = (V_1 \cap X_1) \cup ... \cup (V_n \cap X_n)$.

Let us suppose that the conditions of the theorem hold, that A* can deadlock after w, and that $V_1,...,V_n$ are as above. At most one of the $V_i \cap X_i$ can equal X_i, for otherwise there would be some $i \neq j$ such that the pair $(A_i \, {}_{X_i}\|_{X_j} A_j)$ would be able to deadlock after $w \upharpoonright (X_i \cup X_j)$. From relation (b) above it can be seen that for each i $(X_i - \bigcup_{j \neq i} X_j) \subseteq V_i$; hence $X_i - V_i \subseteq \bigcup_{j \neq i} X_j$ for each i. Since by assumption each V_i is maximal in $A_i(w \upharpoonright X_i)$ we can infer that $X_i - V_i \subseteq (A_i \, \underline{after} \, (w \upharpoonright X_i))^{\circ}$. Putting these facts together, and using the fact that $(A_i \, \underline{after} \; v)^{\circ}$ has non-empty intersection with at most one X_j s.t. $i \neq j$ for all v, we see that for all i, with possibly one exception, there is some $j(i) \neq i$ such that $X_i - V_i$ is a non-empty subset of $X_{j(i)}$.

Any k such that j(k) is not defined must have $X_k \subseteq V_k$. Suppose there were some i with $j(i) = k$ where $X_k \subseteq V_k$; then it is easy to see that $(V_i \cap X_i) \cup (X_k \cap V_k) = (X_i \cup X_k)$. This tells us that it is possible that $(A_i \, {}_X\|_X A_k)$ deadlock after string $w\upharpoonright(X_i \cup X_k)$, which contradicts our assumptions. We can thus deduce that any k such that j(k) is not defined is not the image under j of any i. "j" is thus a function from I into I, where I is the finite set of indices on which it is defined. It is easy to see that, for any element r of I, in the sequence $r, j(r), j^2(r), j^3(r), \ldots$ there must be some repetition. In other words there is some finite sequence $n_1, \ldots, n_k$ of distinct elements of I such that $j(n_i) = n_{i+1}$ $(i<k)$ and $j(n_k) = n_1$. Since $j(i) \neq i$ for all i we can deduce that $k \neq 1$. Suppose that k=2, then there are some i,j such that $i \neq j$, $(X_i - V_i) \subseteq X_j$ and $(X_j - V_j) \subseteq X_i$. By relation (b) in the construction of the V_k, and since $X_i \cap X_j \cap X_k = \emptyset$ for all k distinct from i and j, we get the relation $X_i \cap X_j = (V_i \cup V_j) \cap (X_i \cap X_j)$, which in turn implies that $(X_i - V_i) \subseteq V_j$ and $(X_j - V_i) \subseteq V_i$. Hence $(X_i \cap V_i) \cup (X_j \cap V_j) = X_i \cup X_j$, contradicting the fact that $(A_i \,{}_{X_i}\|_{X_j}\, A_j)$ cannot deadlock after $w\upharpoonright(X_i \| X_j)$. We can thus infer that $k \geqslant 3$, and it is easy to see that by construction the sequence $n_1, \ldots, n_k$ satisfies all that is required of it.

The above theorem, interpreted informally, means that in a network of processes satisfying conditions (ii) and (iii) (which we will interpret shortly), if all pairs of processes are free of deadlock then whenever deadlock occurs it must contain a ring of at least three distinct processes each demanding to communicate with one of its neighbours and refusing to communicate with its other neighbour (which wants to communicate with it). Condition (ii) of the theorem says that every communication is participated in by at most two processes. Condition (iii) says that each process can never be willing to communicate with two of its neighbours (i.e. it can never have the option to communicate with either one neighbour or the other).

7.10 <u>Corollary</u>

In any network which both satisfies the conditions of 7.9 and for which the graph formed with nodes A_i and edges between A_i & A_j when $X_i \cap X_j \neq \emptyset$ is a tree, there can be no deadlock.

proof

If $n_1,..,n_k$ is the sequence which is produced by 7.9 when there is any deadlock then $A_{n_1},...,A_{n_k}$ is a circuit in the graph.

Since any tree of processes automatically satisfies condition (ii) of 7.9 this result tells us that in any tree whose elements satisfy condition (iii) it is possible to eliminate global deadlock by showing that it is impossible between any pair of its components.

In any graph with only a few circuits (or a lot of circuits of only a few types) it is often possible to reduce the proof of freedom from deadlock to the checking of a few cases. (If there are n circuits in a graph there are 2n possible sequences $n_1,..,n_k$ arising from 7.9). Thus in a ring of processes there are only two possible circuits.

The following example (after E.W. Dijkstra) represents a ring of processes, any of which might be requested to obtain some "token" (which is passed round the ring) so that it can carry out some action, and then release the token for use by other processes. Suppose $n \geqslant 3$, we can define processes X_i, Y_i for $i \in \{0,1,..,n-1\}$ thus

$$X_i \Leftarrow i.get \rightarrow i{+}1.find \rightarrow i.pri \rightarrow i.cri \rightarrow i.rel \rightarrow Y_i$$
$$\square\ i.find \rightarrow i{+}1.find \rightarrow i.pri \rightarrow i{-}1.pri \rightarrow X_i$$

$$Y_i \Leftarrow i.get \rightarrow i.cri \rightarrow i.rel \rightarrow Y_i$$
$$\square\ i.find \rightarrow i{-}1.pri \rightarrow X_i$$

(all arithmetic is modulo n)

(X_i represents a process without the token "pri", which before it allows the environment to perform its critical action "cri" must put in a request to its neighbour to find it and pass it back. Y_i represents a process with the token, which will allow the environment to perform "cri" or will pass it to its neighbour if requested.)

If we set up a ring with one token, which initially is in the 0-process (each process being given as its alphabet the set of symbols which it can potentially use), then it would be useful to be able to prove it free of deadlock. It is easy to see that if R is the process which results from combining $Y_0,X_1,...,X_{n-1}$ in parallel, the graph produced in the manner of

7.10 is a ring, there being edges between the i-process and i+1-process for each i (addition modulo n).

It is not hard to prove that the processes and alphabets which make up the networks satisfy conditions (ii) and (iii) of 7.9. Also the proof that they satisfy condition (i) can be reduced to a fairly easy analysis of cases, proving by mutual induction that $(X_{i+1}\ {}_{Z_{i+1}}\|_{Z_i} X_i)$, $(Y_{i+1}\ {}_{Z_{i+1}}\|_{Z_i} Y_i)$ and $(Y_{i+1}\ {}_{Z_{i+1}}\|_{Z_i} X_i)$ are free of deadlock, all other cases (non-adjacent processes) being trivial. (Z_i is used to denote the alphabet of the i-process.)

We can thus infer that the network R can only deadlock if each process is waiting for its i+1 neighbour or if each process is waiting for its i-1 neighbour. It is quite easy to prove that at each point in the ring's history there is exactly one process with the token, in the sense that <u>either</u> it has never left Y_0 and no other process has been passed it (communicated i.pri) <u>or</u> there is exactly one process i which has not passed the token on (communicated i-1.pri) since it last received it (communicated i.pri). However a process can only be waiting for its i+1 neighbour if it does not have the token, and can only be waiting for its i-1 neighbour if it does have the token. We can thus infer that the network is free of deadlock.

The above argument, while it is not an absolutely rigorous proof, can easily be extended to one.

It is not hard to extend the above to the cases when any non-zero number of tokens are initially in the network. If all the processes are without tokens initially (i.e. are all equal to X_i) then deadlock occurs.

In the above example, and others where it is applicable, 7.9 seems to formalize the intuitive reasons why one expects a network to be free of deadlock. This makes it a useful tool in eliminating deadlock. Condition (iii) of 7.9 is fairly restricive in its nature (it means that the result is not applicable to networks such as the "five dining philosophers" of 6.2). It is possible if we drop condition (iii) to prove a weaker version of 7.9, which is stated below and has a similar proof.

7.11 Theorem

Suppose that $A_1,..,A_n$ is a finite collection of processes ($n \geqslant 2$) with associated alphabets $X_1,..,X_n$, and that A^* is the result $((..(A_1 {}_{Y_1}\|_{X_2} A_2)_{Y_2}..)_{Y_{n-1}}\|_{X_n} A_n)$ of combining them in parallel ($Y_r = X_1 \cup .. \cup X_r$). Suppose also that whenever i,j,k are distinct $X_i \cap X_j \cap X_k = \emptyset$. Define a ternary relation on $\{1,..,n\}$ as follows:

$$R(i,j,k) \Leftrightarrow i \neq j \ \& \ j \neq k \ \& \ k \neq i \ \& \ \exists w.(((A_i \underline{\text{after}}\ w)^{\circ} \cap X_j) \neq \emptyset) \ \& \ (((A_i \underline{\text{after}}\ w)^{\circ} \cap X_k) \neq \emptyset)$$

If i and j are distinct elements of $\{1,..,n\}$ define clique(i,j) to be the smallest subset of $\{1,..,n\}$ which both contains i and j and is closed under R in the sense $s,t \in X \ \& \ R(s,t,u) \Rightarrow u \in X$. (clique$(i,j)$ places an upper bound on the sets of processes which might interfere directly with any communication between A_i and A_j.)

If X is any non-empty subset of $\{1,..,n\}$ define A_X to be the result of combining in parallel all the processes indexed by elements of X:

$$A_X = A_i \quad \text{if } X = \{i\}$$

$$A_X = ((..(A_i {}_{X_i}\|_{X_j} A_j)_{X_i}\|_{X_l}..)_{X_i \cup .. \cup X_s}\|_{X_k} A_k) \quad \text{if } X = \{i,j,..,s,k\} \text{ has at least two elements.}$$

If the network A^* is free from local deadlock in the sense that all subsets of cliques are deadlock-free, then deadlock can only occur in the complete network A^* on string w when there exist $Z_1,..,Z_n \subseteq \Sigma$ which satisfy the following:

(i) Z_i is maximal in $A_i(w \upharpoonright X_i)$;

(ii) $X_1 \cup .. \cup X_n = (Z_1 \cap X_1) \cup .. \cup (Z_n \cap X_n)$;

(iii) for each pair $i \neq j$ there is some element k of clique(i,j) with the property that $(X_k - Z_k) \subseteq \bigcup\{X_r \mid r \notin \text{clique}(i,j)\}$;

(iv) there exists a sequence $n_1,..,n_k$, not contained within any one clique, such that $(X_{n_i} - Z_{n_i}) \cap X_{n_{i+1}} \neq \emptyset$ for each i (where $k+1$ is interpreted as 1).

* * * * * * * * * *

The power of this result depends critically on the structure of the space of cliques in a given network. In general one can easily show that clique(i,j) = clique(j,i) for each pair (i,j), and that $k \in \text{clique}(i,j) \Rightarrow (\text{clique}(i,k) \subseteq \text{clique}(i,j))$. When all processes in a network are different (as will usually be the case in a network without hiding) one can unambiguously think of cliques as being sets of processes and being defined by pairs of processes.

If $X_i \cap X_j = \emptyset$ we must have clique(i,j) $=\{i,j\}$, and it is easy to see that so long as there is at least one pair r&s such that $X_r \cap X_s \neq \emptyset$ we can disregard all such cliques. Bearing this in mind the only cliques we need consider in the "five dining philosophers" example are $\{PHIL_i, FORK_i, PHIL_{i\boxminus 1}\}$ (which is both clique($PHIL_i$,$FORK_i$) and clique($FORK_i$,$PHIL_{i\boxminus 1}$)) and $\{PHIL_1,\ldots,PHIL_5,B\}$ (which is clique(B,$PHIL_i$) for each i).

Note that under the assumptions of 7.9 we have clique(i,j) $=\{i,j\}$ for all i and j, so that the conclusions of 7.11 more or less imply those of 7.9. We can prove an analogous result to 7.10, though the proof is harder:

7.12 Theorem

Suppose that A* is the network described in 7.11, and that in addition to satisfying the hypotheses of 7.11 it is a tree in the sense of 7.10; then it is free of deadlock.

proof

If there were deadlock possible in A* on string w then there would exist sets $Z_1,\ldots,Z_n$ and a sequence $n_1,\ldots,n_k$ satisfying the conditions of 7.11(i-iv).

Claim that every sequence $m_1,\ldots,m_r$ of elements of $\{1,\ldots,n\}$ with the properties that (i) $(X_{m_i} - Z_{m_i}) \cap X_{m_{i+1}} \neq \emptyset$ $(i < r)$ & $(X_{m_r} - Z_{m_r}) \cap X_{m_1} \neq \emptyset$, (ii) $r \geqslant 2$ and (iii) $m_i \neq m_{i+1}$ $(i < r)$ & $m_r \neq m_1$ has the property that $\{m_1,\ldots,m_r\} \subseteq$ clique(m_1,m_2).

We will prove this by induction on r. The result is trivially true when r=2, so let us suppose that it holds for all such sequences with length less than r and that $m_1,\ldots,m_r$ is such a sequence. There are two cases to consider: either m_1 is repeated later in the sequence or it is not.

In either case it is easy to see that the processes indexed by the m_i form a connected subtree of the network, and that A_{m_i} is joined to $A_{m_{i+1}}$ by an edge, as is A_{m_r} to A_{m_1}. In the second (or) case we can thus deduce that $m_2 = m_r$, which means that by our inductive hypothesis $\{m_2..m_{r-1}\} \subseteq$ clique(m_2,m_3) (we must have $r > 3$ for otherwise $m_2 = m_{2+1}$). However by construction $(((A_{m_2}\ \underline{after}\ w\restriction X_{m_2})^{\circ} \cap X_{m_3}) \neq \emptyset)$ & $(((A_{m_2}\ \underline{after}\ w\restriction X_{m_2})^{\circ} \cap X_{m_1}) \neq \emptyset)$ (since $m_2 = m_r$ and $(X_i - Z_i) \subseteq (A_i\ \underline{after}\ w\restriction X_i)^{\circ}$). This tells us that $m_3 \in clique(m_1,m_2)$, which as we observed earlier implies that $clique(m_2,m_3) \subseteq clique(m_1,m_2)$. This completes the proof in this case.

In the "either" case we must have $m_1 = m_s$ for some $s \neq r$, $s \neq 1$. Clearly each of $m_1,..,m_{s-1}$ and $m_s,..,m_r$ is a sequence which satisfies the hypotheses of this result, so we can deduce that $\{m_1,..,m_{s-1}\} \subseteq clique(m_1,m_2)$ and that $\{m_s..m_r\} \subseteq clique(m_1,m_{s+1})$ from our inductive hypothesis. However by construction $(((A_{m_1} \underline{after}\ w \upharpoonright X_{m_1})^{\circ} \cap X_{m_2}) \neq \emptyset)$ & $(((A_{m_1} \underline{after}\ w \upharpoonright X_{m_1})^{\circ} \cap X_{m_{s+1}}) \neq \emptyset)$ (as before), which implies that $m_{s+1} \in clique(m_1,m_2)$. Thus $clique(m_1,m_{s+1}) \subseteq clique(m_1,m_2)$, which completes the proof of the "either" case.

We may therefore conclude that this holds of all such sequences. This contradicts our assumption of deadlock, for the sequence $n_1,...,n_k$ satisfies the hypotheses but not the conclusion of the above result (by 7.11(iv) it is not contained in any clique).

The cliques of a tree have a very regular form, as shown by the next result (the proof of which is similar to parts of the above).

7.13 Lemma

If the network A* of 7.11 has the form of a tree, then if A_i and A_j are not adjacent we have $clique(i,j) = \{i,j\}$. If A_i and A_j are adjacent then the nodes indexed by clique(i,j) form a connected subtree with the property that whenever A_r and A_s are two other adjacent nodes either clique(i,j) = clique(r,s) or $clique(i,j) \cap clique(r,s)$ has at most one element.

7.12 provides us with a very general technique for proving freedom from deadlock in finite trees. As an example of this consider the system formed by combining n philosophers and n+1 forks in a line:

$FORK_0 \| PHIL_0 \| FORK_1 \| PHIL_1 \| \ldots \| FORK_{n-1} \| PHIL_{n-1} \| FORK_n$

(processes defined as in 6.2 but with ordinary rather than mod 5 arithmetic).

The cliques of this system are just $\{FORK_0, PHIL_0\}$, $\{FORK_n, PHIL_{n-1}\}$ and $\{FORK_i, PHIL_i, PHIL_{i-1}\}$ $(1 \leq i \leq n-1)$ whose subsets are easy to prove free from deadlock. Having done this we can infer that the whole system is deadlock-free.

Many of the arguments and results in this section have been (implicitly or explicitly) graph-theoretic. The basic type of graph we have been interested in is that of "requests" on deadlock (the directed graph which results from drawing in an edge leading from each element of a deadlocked system to the elements of the system from which it would accept communication). Theorems 7.9 and 7.11 tell us things about the gross structure of these graphs (essentially that they contain loops of certain types). It is often possible to limit the possibilities for these graphs if we have some local knowledge of structure. One might for example be able to show that if in such a deadlock graph some component of the network has an edge leading to it from one of its neighbours then the edges leading from it have some particular structure. In particular one might be able either to completely eliminate the possibility of the loops implied by 7.9 or 7.11 or cut down the number of possible loops to manag le proportions.

It is not very hard to apply this type of technique to prove freedom from deadlock in the "five dining philosophers" example of 6.2. The outline of such a proof is set out below.

(i) Show that in any deadlock graph an edge from $PHIL_i$ to $FORK_i$ implies that $FORK_i$ has a unique edge leading out of it which leads to $PHIL_{i \boxminus 1}$, and that when $PHIL_{i \boxminus 1}$ has an edge leading to $FORK_i$ there is a unique edge leading from $FORK_i$ (which leads to $PHIL_i$).

(ii) Show by consideration of $PHIL_i \| FORK_i$ that whenever $FORK_i$ has a unique edge from it leading to $PHIL_i$ then $PHIL_i$ has a unique edge leading from it, to $FORK_{i \boxplus 1}$. Correspondingly whenever $FORK_{i \boxplus 1}$ has a unique edge from it, leading to $PHIL_i$, then $PHIL_i$ has a unique edge from it, to $FORK_i$.

(iii) By 7.11 there must be some edge leading out of the clique $\{B, PHIL_1, PHIL_2, .., PHIL_5\}$, which means by (i) and (ii) that there is a ring of edges leading round the "outside" of the graph, and that these are the only edges leading out of the $FORK_i$s and $PHIL_i$s.

(iv) Show that this is impossible by the structure of B (by counting "getup"s and "sitdown"s).

It ought to be possible to develop a better notation for this type of argument, and to develop a simple theory to make its application easier.

This concludes our brief discussion of deadlock. We have seen that it is often possible to reduce the problem of proving freedom from deadlock in a general network to the corresponding problem in a network free from hiding. We conjectured that it might be reasonably easy to attack some such networks by means of constructive relations between states, and developed various graph-theoretic methods for application to such networks. There are other, similar, predicates which one might wish to prove in lieu of simple freedom from deadlock. These might include proving that every component of a network remains alive (in some sense), or proving that a process is "live" in the sense that at all times there is some way of getting it back to its initial state. It should be possible to adapt several of our methods to such problems (of the two examples quoted here the first is likely to be easier than the second).

<u>footnote</u>

Note that the identification of deadlock with the appearance of "Σ" in the image of strings is critical in this section. This would not have been the case if we had chosen the finite refusal sets model in chapter 4 instead of the directed-closed model. If we had done this it would have been necessary to prove a result equivalent to the difficult consistency theorem 4.15 in order to prove 4.9, 4.11 and their corollaries.

Chapter 8 :- Assigning Meanings to Models

So far in this thesis we have studied in some depth the mathematical structure of two models for computation. We have derived many interesting results which are in many cases correctness results for the model processes which exist within our two models. It is natural to ask how these models relate to any more "real" space of processes and how our correctness proofs translate into theorems of the more concrete processes.

Before examining our particular models it is interesting to develop as general a calculus as possible for this type of relative study. This should give us a better general understanding of the problem of modelling processes, and suggest alternative models with chosen properties. Let us therefore suppose that C is some space of concrete processes and that M is intended to be a mathematical model for C. We will for the moment make no assumptions about the natures of C and M. Since M is a model for C we may suppose that there is some function $\Phi : C \rightarrow M$ such that for any $c \in C$ $\Phi(c)$ is the representation in M of c.

Our particular interest is in the translation of proofs in M to results in C (i.e deducing results about real processes from the model). It is therefore very important to us to know how predicates translate. Suppose that Ψ is some predicate we want to prove of an element of C (Ψ is thus a one-place relation on C). It is clearly important that we should know just how much we need to prove of $\Phi(c)$ $(c \in C)$ in order to know for certain that $\Psi(c)$ holds. Define the predicate Ψ' on M by $\Psi'(A) \equiv \forall c.(\Phi(c) = A) \Rightarrow \Psi(c)$. By construction Ψ' is the weakest predicate on M which ensures this (i.e. $\Psi'(\Phi(c)) \Rightarrow \Psi(c)$). Correspondingly if Π is a predicate on M we can define the predicate Π^* on C by $\Pi^*(c) \equiv \Pi(\Phi(c))$. Π^* is the strongest predicate on C which is proved by $\Pi(\Phi(c))$. These two constructs relate in the following way.

8.1 Theorem

If $\Phi : C \rightarrow M$, Ψ is a predicate of C, Π is a predicate of M and the constructs "'" and "*" are as described above then we have the following.

a) $\Pi(A) \Rightarrow (\Pi^*)'(A)$ ($\Leftrightarrow$ if Φ is onto) for each $A \in M$

b) $\Psi(c) \Leftarrow (\Psi')^*(c)$ ($\Leftrightarrow$ if Φ is one-one) for each $c \in C$

c) $\Pi^* \equiv ((\Pi^*)')^*$

d) $\Psi' \equiv ((\Psi')^*)'$

<u>proof</u>

The following lemma is useful in proving the above (amongst other things).

8.1.1 <u>lemma</u>

a) If H and Θ are two predicates over M such that H is weaker than Θ (i.e. $\Theta(A) \Rightarrow H(A)$ for all $A \in M$) then H^* is weaker than Θ^* (i.e. $\Theta^*(c) \Rightarrow H^*(c)$ for all $c \in C$).

b) If X and Ω are two predicates over C such that Ω is weaker than X then Ω' is weaker than X'.

<u>proof</u>

Both these results follow immediately from the definitions of Ψ' and Π^*.

We can now prove a) - d) above.

a) $\Pi(A) \Rightarrow \Pi^*(c)$ for each c s.t. $\Phi(c) = A$ (by defn of Π^*)

$\Rightarrow (\Pi^*)'(A)$ (by defn of Ψ')

If Φ is onto then $A = \Phi(c)$ for some $c \in C$, so

$(\Pi^*)'(A) \Rightarrow (\Pi^*)'(\Phi(c))$

$\Rightarrow \Pi^*(d)$ for each d s.t. $\Phi(d) = \Phi(c)$ (by defn of Ψ')

$\Rightarrow \Pi^*(c)$ (as $\Phi(c) = \Phi(c)$)

b) $(\Psi')^*(c) \Rightarrow \Psi'(\Phi(c))$ (by defn of Π^*)

$\Rightarrow \Psi(d)$ for each d s.t. $\Phi(d) = \Phi(c)$ (defn of Ψ')

$\Rightarrow \Psi(c)$ (as $\Phi(c) = \Phi(c)$)

If Φ is one-one then

$\Psi(c) \Rightarrow \Psi(d)$ for each d s.t. $\Phi(c) = \Phi(d)$ (as there is only one such d (i.e. c))

$\Rightarrow \Psi'(\Phi(c))$ (by defn of Ψ')

$\Rightarrow (\Psi')^*(c)$ (by defn of Π^*)

c) $(\Pi^*)'$ is weaker than Π by (a) above. Thus $((\Pi^*)')^*$ is weaker than Π^* by 8.1.1(a). Π^* is weaker than $((\Pi^*)')^*$ by (b) above. These two clauses together give us the desired result.

d) Ψ is weaker than $(\Psi')^*$ by (b) above. This tells us that Ψ' is weaker than $((\Psi')^*)'$ by 8.1.1(b). Also $((\Psi')^*)'$ is weaker than Ψ' by (a) above. Again these two clauses combine to give the desired result.

Note that 8.1 (a) and (b) tell us (as we might have expected) that if Φ is a bijection then also the predicate spaces have "'" = $"*"^{-1}$ as a bijection between them. We will however find that it is unusual for Φ to be either one-one or onto.

From the nature of the mapping $\Psi \to \Psi'$ we see that the more discriminating the map Φ the more likely we are to get a reasonable translation into M of a predicate Ψ. This is because there are less additional $d \in C$ which need to satisfy Ψ in order to make $\Psi'(\Phi(c))$ true for any c. It is clearly more important that a useful predicate should translate well than one which we are unlikely to want to prove of a process. We would like (for as many and as useful Ψ as possible) that there should be predicates Π of M s.t.

(i) $\Pi \Rightarrow \Psi'$

(ii) Π is reasonable to prove (is continuous, perhaps)

(iii) Π is not ridiculously strong, so that generally $\Pi(\Phi(c))$ does in fact hold of a process c s.t. $\Psi(c)$.

Note that everything we have said so far is equally true of product spaces and their predicates because of the extremely general nature of the spaces we have considered. Though we treat only single predicates in the following, for simplicity, almost the whole of the rest of what follows can be translated to accommodate predicates of several (or many) variables.

We must now be more specific about the nature of the objects we are studying. Let us suppose that elements of C have things called behaviours, which we will think of as representing possible sequences of actions carried out by processes (relative to any control exerted by outside agents). We will suppose that the behaviours of a process are a powerful language for specifying correctness; we will restrict ourselves to the study of predicates of the form $\underset{\sim}{X}(c) \Leftrightarrow \forall b.$ b is a behaviour of c $\Rightarrow X(b)$ (X any predicate of behaviours). For $c \in C$ we will suppose that B(c) is the set of possible behaviours of c. Note that the type of predicate specified above corresponds quite closely to normal ideas of what a correctness predicate of a process should be: a process is "correct" if and only if it must inevitably behave correctly.

We will further suppose that the map $\Phi: C \rightarrow M$ is closely based on the possible behaviours of a process. This assumption is prompted by the nature of the models we already possess, which are both intended to reflect some aspect of a process' behaviour. Specifically we will suppose that $\Phi(c) = \{\theta(b) \mid b \in B(c)\}$ for some function θ. (For example in the "traces" model P of chapters 1-3 we would expect that $\theta(b)$ = the sequence of external communications occurring in b.) This assumption clearly carries with it the requirement that M should be (at least up to isomorphism) a subset of some powerset (a requirement which is met by both our existing models).

Clearly the more discriminating the function θ, the more expressive becomes the model.

For the predicate $\underset{\sim}{X}$ and function Φ described above we have

$$\begin{aligned}
\underset{\sim}{X}'(A) &\equiv \forall c.\,(\Phi(c) = A) \Rightarrow \underset{\sim}{X}(c) \\
&\equiv \forall c.(\{\theta(b) \mid b \in B(c)\} = A) \Rightarrow (\forall b \in B(c).\ X(b)) \\
&\Leftarrow \forall b \in \theta^{-1}(A).\ X(b) \\
&\equiv \forall d \in A.\,(\forall b \in \theta^{-1}(d).\ X(b)) \\
&\equiv X^{+}, \text{ say.}
\end{aligned}$$

X^{+} is the predicate (of M) saying that all behaviours which might occur in a process mapping to A are correct, the phrase "might occur" being interpreted as element-wise possibility. (We are thus considering the behaviours which appear to be possibilities, even though it might be that we could eliminate them by consideration of the gross structure of A. For an example of the influence this has see the later examination of the traces model.)

Since $X^{+} \Rightarrow \underset{\sim}{X}'$ we have $X^{+}(\Phi(c)) \Rightarrow \underset{\sim}{X}(c)$ for any $c \in C$.

We will also reserve the right to make θ a <u>relation</u>, in which case $\Phi(c) = \bigcup\{\theta(b) \mid b \in B(c)\}$. Behaviours then have any number of representatives in the image. To simplify matters, from here on we will always assume that θ is a function unless we clearly state the contrary. Except where we indicate otherwise all results stated or proved for "Φ"s defined using functions are also true for those defined using relations. We will have to wait until the last two sections of this chapter for examples of the use of relations in defining maps to models.

The question of the expressiveness of "Φ"s defined using relations is less clear-cut than with functions, though it is still usually true that a "θ" which makes more distinctions will give rise to a more expressive model. It is now possible, however, for a behaviour which has a large image under "θ" to obscure much detail in the image (it is precisely for this reason that we will later use relations in some cases).

It is possible to define the "$^+$" operator on predicates of behaviour in much the same way as before. Suppose that "Φ" is a modelling function defined using a relation "θ". Then we have

$$
\begin{aligned}
\underset{\smile}{X}'(A) &\equiv \forall c.(\Phi(c) = A) \Rightarrow X(c) \\
&\equiv \forall c.(\bigcup\{\theta(b) \mid b \in B(c)\} = A) \Rightarrow (\forall b \in B(c).\ X(b)) \\
&\Leftarrow \forall b.\ \theta(b) \subseteq A \Rightarrow X(b) \\
&\equiv X^+, \text{ say.}
\end{aligned}
$$

Once again X^+ is the predicate of M saying that all behaviours which might occur in a process mapping to A are correct. This time we do not consider A element by element, since the result of doing this, $\forall d \in A. \forall b.((\theta(b) = \emptyset \vee d \in \theta(b)) \Rightarrow X(b))$, is in practice too strong to be of any use (or at least this so in the two examples of relations which we meet later). The penalty we pay for this is that the predicate X^+ defined relative to a relation may not have such a workable form. It is, for example, easy to see that any X^+ defined relative to a function must be strongly continuous when the map Φ is to either of our existing models, but that this need not always be so for relations. Also the results 8.2 and 8.3 proved below only hold in general when θ is a function. The most important single result about the predicates X^+, namely 8.4 does hold in general, however.

8.2 Theorem (holds only when θ is a function)

If X is any predicate of behaviour then there is another one ψ, such that $(X^+)^* = \underset{\sim}{\psi}$ and $X^+ = \psi^+$.

proof

By our earlier definitions

$$
\begin{aligned}
(\chi^{+})^{*}(c) &\Leftrightarrow \chi^{+}(\Phi(c)) \\
&\Leftrightarrow \forall d \in \Phi(c).(\forall b \in \theta^{-1}(d).\ \chi(b)) \\
&\Leftrightarrow \forall b \in B(c).(\forall b' \in \theta^{-1}(\theta(b)).\ \chi(b')) \\
&\Leftrightarrow \underset{\sim}{\psi}(c), \text{ where } \psi(b) \Leftrightarrow \forall b' \in \theta^{-1}(\theta(b)).\ \chi(b')
\end{aligned}
$$

For this same ψ we have

$$
\begin{aligned}
\psi^{+}(A) &\Leftrightarrow \forall d \in A.(\forall b \in \theta^{-1}(d).\ \psi(b)) \\
&\Leftrightarrow \forall d \in A.(\forall b \in \theta^{-1}(d).(\forall b' \in \theta^{-1}(\theta(b)).\ \chi(b'))) \\
&\Leftrightarrow \forall d \in A.(\forall b \in \theta^{-1}(d).(\forall b' \in \theta^{-1}(d).\ \chi(b'))) \\
&\Leftrightarrow \forall d \in A.(\forall b' \in \theta^{-1}(d).\ \chi(b')) \\
&\Leftrightarrow \chi^{+}(A)
\end{aligned}
$$

If more restrictive conditions are placed upon the nature of the space of behaviours and upon M we can prove stronger results than the above.

8.3 Theorem

Suppose that for every behaviour b and $A \in M$ we have $\theta(b) \in A \Rightarrow \exists c \in C. b \in B(c)\ \&\ \Phi(c) \subseteq A$. Then for all predicates χ & ψ of behaviours we have $(\underset{\sim}{\chi} \equiv \underset{\sim}{\psi}) \Rightarrow (\chi^{+} \equiv \psi^{+})$. Furthermore if $\underset{\sim}{\chi} \equiv (\psi^{+})^{*}$ then $\chi^{+} \equiv \psi^{+}$.

proof

Suppose χ and ψ are such that $\underset{\sim}{\chi} \equiv \underset{\sim}{\psi}$. By symmetry, to prove $\chi^{+} \equiv \psi^{+}$ it is enough to show $(\chi^{+})(A) \Rightarrow (\psi^{+})(A)$ for all $A \in M$. Suppose $(\chi^{+})(A)$ holds and that $d \in A\ \&\ b \in \theta^{-1}(d)$.

$$(\chi^{+})(A) \Rightarrow \forall d \in A.(\forall b \in \theta^{-1}(d).\ \chi(b)) \qquad (*)$$

By assumption there exists some $c \in C$ such that $b \in B(c)$ and $\Phi(c) \subseteq A$. If $b' \in B(c)$ then by construction $\theta(b') \in A$, so that $b' \in \theta^{-1}(d)$ for some $d \in A$. Thus $\chi(b')$ holds by (*).

Hence $\forall b' \in B(c). \chi(b')$ $(\Leftrightarrow \underset{\sim}{\chi}(c))$ holds.

Since by assumption $\underset{\sim}{\chi} \equiv \underset{\sim}{\psi}$ we have $\underset{\sim}{\psi}(c)$.

Therefore $\psi(b)$ holds by definition of $\underset{\sim}{\psi}$ since $b \in B(c)$.

Thus on the assumptions $(\chi^{+})(A)$, $d \in A$ and $b \in \theta^{-1}(d)$ we have proved $\psi(b)$. Hence

$$(\chi^{+})(A) \Rightarrow \forall d \in A.(\forall b \in \theta^{-1}(d).\ \psi(b))\ (\Leftrightarrow (\psi^{+})(A)),$$

which was what we wanted to prove.

This result shows that (under the stated conditions) the operator "$^{+}$" can be regarded as an operator on predicates of C (rather than on predicates of behaviour) without ambiguity (as long as the predicate has the form $\underset{\sim}{\chi}$ for some χ).

To prove the second half of 8.3 we need merely observe that if $\underset{\sim}{\chi} \equiv (\psi^+)^*$ then by 8.2 there is some τ such that $\underset{\sim}{\tau} \equiv (\psi^+)^*$ and $\tau^+ \equiv \psi^+$. But then $\underset{\sim}{\chi} \equiv \underset{\sim}{\tau}$, so $\chi^+ \equiv \tau^+$ by the first part of 8.3. Hence $\chi^+ \equiv \psi^+$ as desired.

This shows that when the conditions of 8.3 hold so that "$^+$" is a well-defined operator on predicates on C we have $((Y^+)^*)^+ \equiv Y^+$ for all predicates Y of the form $\underset{\sim}{\chi}$.

The condition stated in 8.3 is one which will tend to be met in all examples. It simply states that if a behaviour appears to be possible for processes mapping to $A \in M$ by elementwise consideration of A then there is some $c \in C$ which has this behaviour and whose behaviour does not exceed the bounds defined by A. See below for further clarification of the use of "subset" rather than "equality" in the condition.

Define a monotonic predicate on M to be one which satisfies the condition $A \subseteq B \ \& \ \Pi(B) \Rightarrow \Pi(A)$. The following result is obvious (from the definition of φ^+).

8.4 Lemma

For any predicate φ of behaviours the predicate φ^+ is monotonic, and furthermore if $N \in M$ and $\forall A \in N.\ \varphi^+(A)$ and $B \in M$ is such that $B \subseteq \bigcup N$ then also $\varphi^+(B)$.

Note that the simple structure of the predicate φ^+ ($\varphi^+(A) \equiv \forall d \in A.\ \xi(d)$) ensures that it will be pleasant in other ways. It will for example be strongly continuous in both our existing models (see 2.44, 5.10).

Suppose we were to restrict ourselves to the use of monotonic predicates in directly proving things about elements of C (from their values in M). This would have several advantages and several disadvantages. The advantages we will meet shortly. The chief disadvantage of this approach is that we lose a certain amount of expressive power by our abandonment of a large class of possible predicates to prove of $\Phi(c)$, in particular the predicates of the form $\underset{\sim}{\chi}'$ (not in general monotonic). The natural predicate to prove of $\Phi(c)$ in order to prove $\underset{\sim}{\chi}(c)$ is then χ^+, which may well not be true of $\Phi(c)$ when $\underset{\sim}{\chi}'$ is. In some cases χ^+ is very much stronger than $\underset{\sim}{\chi}'$, and can even be "false" when $\underset{\sim}{\chi}'$ is quite reasonable. The question of whether a particular triple (C,M,Φ) is suitable for use with monotonic predicates will vary both with the structure of the triple and with the things we wish to prove. As we will see there are some cases where monotonic predicates are usually adequate and others where they are often not. We will also see that there are considerable advantages in the monotonic case in the modelling and implementation of operators, inasmuch as there is a large class of implementations which can be considered correct relative to the proving of monotonic predicates but not of general predicates. Say we are treating (M,Φ) as a class 1 model for C if we only seek to prove monotonic predicates and as a class 2 model if we seek to be able to prove general predicates.

Class 1 Models

We are likely to want to model in M the operators we wish to use on the space C. The purpose of this will be to prove things about the result of applying the operator in C from the value(s) in M of its operand(s). Modelling of operators is a two-way process: an operator in M can model in C or an operator in C can implement an operator in M. We must seek workable definitions of these terms. Define an operator "op" over M to be reasonable if it is both monotonic (in the sense $op(A,*) \subseteq op(B,*)$ if $A \subseteq B$) and there is a possible correct implementation of it over C (in the sense to follow). The reason that a "reasonable" operator should be implementable is obvious. The reason that it should be monotonic is that the more possible behaviours of its operands, the more behaviours might be expected possible of the result of applying it.

Say that $op^*:C^n \to C$ is an operator implementing $op:M^n \to M$ (or alternatively op models op*) if for all $c_1,..,c_n \in C$

$$\Phi(op^*(c_1,...,c_n)) \subseteq op(\Phi(c_1),...,\Phi(c_n)) .$$

This means that the result of applying op* has behaviours contained within the bounds predicted by applying op to the images in M of its operands.

This implies that if Π is any (monotonic) predicate and op correctly models op* then

$$\Pi(op(\Phi(c_1),...,\Phi(c_n))) \Rightarrow \Pi^*(op^*(c_1,...,c_n))$$

so that for any predicate χ of behaviour

$$\chi^{+}(op(\Phi(c_1),..,\Phi(c_n))) \Rightarrow (\chi^{+})^{*}(op^{*}(c_1,..,c_n)) \Rightarrow \underset{\sim}{\chi}(op^{*}(c_1,..c_n))$$

8.5 Lemma

If each of a set of "basic" operators over M is correctly implemented then any composition of them is correctly implemented by the operator over C which results by composing the implementations of the basic operators in the natural way.

The proof of this result is an easy induction using the monotonicity of the basic operators over M.

For example, if f and g are two and one place operators on M respectively, and they are correctly implemented by f* and g* respectively, then the compound operator $h(A,B) = f(f(A,B),g(B))$ is correctly implemented by $h^{*}(c,d) = f^{*}(f^{*}(c,d),g^{*}(d))$.

As we have previously discovered, the simple notion of operators over M alone is unlikely to be sufficient to give a semantics to a language. We need to be able to cope with variables for such things as recursion and input to/assignment to variables. The obvious way to do this is to bring some kind of "state" into our calculations. Let us assume that there is some set Θ of formal parameters taking "process" values and a set Ξ of parameters taking values of other sorts. For the sake of simplicity we will assume in what follows that the sets of parameters and the values taken by elements of Ξ are the same in the two models. One could doubtless extend what follows to the case where there are "real" and "model" values taken by non process parameters and "real" parameters become locations. We will suppose that the constructs of our language are given values in the set M' of monotonic functions in $M^{\Theta} \times S \rightarrow M$, where S is the set containing all possible values of that component of the state which maps elements of Ξ to their possible values in T, a set of "tokens". Constructs in the space C will have the form $C^{\Theta} \times S \rightarrow C$ (= C'). We will shortly see examples of what these constructs might look like when we examine the semantics of our usual language over various models.

We will assume that the semantics of our language are built up from a set B of basic monotonic operators on M'. Say that $A \in M''$, the set of expressible elements of M', if and only if A is the finite composition of a number of elements of B (there will always be several elements of B which are constants (no arguments), so this definition is not vacuous). Assume also that there is a set B* of basic operators over C' from which all constructible elements of C' are composed. We do not assume that there is any sense in which elements of B* attempt to implement elements of B.

If $e \in M'$ and $e^* \in C'$ say that e* is a correct implementation of e if for all $\underset{\sim}{c} \in C^{\Theta}$ and $s \in S$ we have $e(\Phi(\underset{\sim}{c}), s) \supseteq \Phi(e^*(\underset{\sim}{c}, s))$, where Φ is extended in the natural (componentwise) way to C^{Θ}. This definition clearly has a similar effect to the earlier one (op* implementing op) but is different in that here the objects we are discussing are effectively at a lower level: we still need to study the implementations of operators over M'. Suppose that $f: M'^k \rightarrow M'$ is monotonic and $f^*: C'^k \rightarrow C'$. Say that f* <u>imp</u> f if for all $e_1, .., e_k \in M'$ & $e_1^*, .., e_k^* \in C'$ such that e_i^* implements e_i correctly $f^*(e_1^*, .., e_k^*)$ is a correct implementation of $f(e_1, .., e_k)$. It is quite easy to check that this definition corresponds exactly to the old definition of correctness of operators in the degenerate cases covered by that definition.

Say that an element g of B is <u>reasonable</u> if there is some finite composition f_g^* of elements of B* which satisfies f_g^* <u>imp</u> g. The justification for this definition is that g is only a reasonable operator if it is possible to implement it; if we construct elements of M' using non-reasonable operators we can have no guarantee that we will be able to construct real processes to implement them. The following result is obvious, but is nevertheless important.

8.6 <u>Theorem</u>

If for each $g \in B$ there corresponds an f_g^* with the above properties then every expressible element of M' is correctly implemented by a constructible element of C', the construct which results from composing the f_g^*s in the natural way modelling the composition of the expressible element.

For example, if f (0-place), g (1-place) and k (2-place) are all elements of B implemented by f*, g* and k* respectively then k*(g*(f*),f*) is a correct implementation of k(g(f),f).

From the above work we see that if Π is any monotonic predicate of M we get, for any $e^* \in C'$ correctly implementing $e \in M'$, that if $\underset{\sim}{A} \in M^{\Theta}$, $\underset{\sim}{c} \in C^{\Theta}$ and $s \in S$ are such that $\Phi(\underset{\sim}{c}) \subseteq \underset{\sim}{A}$ then $\Pi(e(\underset{\sim}{A},s)) \Rightarrow \Pi^*(e^*(\underset{\sim}{c},s))$. It is worth noting the common special case in which e is independent of $\underset{\sim}{A}$, s or both, in that then $\Pi^*(e^*(\underset{\sim}{c},s))$ is true independently of one or both components of the state. In the case of our usual language we would expect an expression with no free variables of any sort to give rise to an "e" independent of both $\underset{\sim}{A}$ and s.

The technicalities of the above and lack of examples obscure the advantages gained from the assumption that our model is class 1. The fundamental advantage is the possibility of using "$\subseteq$" in our definitions of correct implementation, where otherwise we would have had to use "=". This informally means that an implementor merely has to restrict the behaviours of his resultant processes to be within the bounds specified by an operator over M, and need not make sure that in every case of applying his implementation of the operator there is a possible behaviour mapping under θ to every element of the predicted element of M. This corresponds to allowing the implementor to resolve non-determinism inherent in an operator. It also helps in that the map Φ may well identify large classes of structurally different elements of C. When defining an operator op it may be necessary to allow for this by including in op(A) ($A \in M$, for simplicity) values which arise from applying a natural implementation of op to some c mapping to A but not to others. We will see examples of both these points later.

The most important feature of the above work is that it provides an exact calculus by which we can prove results about the value in C of a process by consideration only of the language used to define the process.

As indicated above, in a <u>class 2</u> model, where monotonic predicates are not considered sufficiently expressive, one can derive a very similar calculus for proving predicates true of implemented processes. It is necessary to demand that all implementations are exact: $e^* \in C'$ will correctly implement $e \in M'$ only if for each $\underset{\sim}{c} \in C$ and $s \in S$ we have $\Phi(e^*(\underset{\sim}{c},s)) = e(\Phi(\underset{\sim}{c}),s)$.

Compositions of Mappings

Suppose that M is a class 1 model for a space C and that C is in turn a model for some space D. It is convenient to think of D as a space of processes and C as a space of "idealized" processes. We will suppose that we have the usual map $\Phi:C \to M$ and in addition a map $Y:D \to C$. Suppose that the set of behaviours of elements of C and D are denoted by B(c) and B'(d) respectively, which are subsets of U and V respectively (types of "behaviour"). It is, as we will later find, useful to know to what extent and under what conditions the composite map $T = \Phi \circ Y$ is useful in modelling D by M. Let us assume that there is some translation function $\eta:V \to U$ (the identity if U = V) such that $B(Y(d)) \supseteq \{\eta(b') \mid b' \in B'(d)\}$ for each $d \in D$. This condition effectively says that the value $Y(d)$ assigned to each $d \in D$ (as its idealization) must not ignore possible behaviours of d.

It is clear from the above definitions that for all $d \in D$ we have $T(d) \supseteq \{\theta(\eta(b')) \mid b' \in B'(d)\}$. This allows us to place a bound on the behaviour of elements of D from a knowledge of $T(d)$. In the case where U = V and η is the identity this bound is the same one as we had before. In this U = V case the predicates of C which have the form $\underset{\sim}{X}$ (for a predicate X of U) clearly extend in a natural way to the space D, with the result that $X^+(T(d)) \Rightarrow \underset{\sim}{X}(d)$. In cases where the function η is many-one there is the possibility that a predicate of V might not translate reasonably to a predicate of U, but we always have that for any predicate X of V $(X'')^+(T(d)) \Rightarrow \underset{\sim}{X}(d)$, where $X''(b) \Leftrightarrow \forall b'.(\eta(b')=b) \Rightarrow X(b')$. Note that the predicate X'' of U is to X just what Ψ' is to Ψ for a predicate Ψ of C, with the same result that the more discriminating the function η the more likely X'' is to be reasonable.

Note that because of the decisions made above the space M is not a model for D in the sense introduced earlier since we do not in general have that $T(d) = \{\theta^*(b') \mid b' \in B'(d)\}$ for any θ^*. It is clear that as long as we restrict ourselves to monotonic predicates this is of little consequence, if we are reasonably careful.

It is desirable to find a criterion by which to judge the implementations over D of operators over C which is "transitive" in the sense that a correct implementation op** (over D) of op* (over C) which in turn is a correct implementation of op (over M) is a correct implementation of op.

Define $D' = D^{\Theta} \times S$ to be the space of constructs over D (analogous to M' and C'). Say that $e^{**} \in D'$ implements $e^* \in C'$ correctly if for all $(\underset{\sim}{d}, s) \in D^{\Theta} \times S$ we have $B(\Upsilon(e^{**}(\underset{\sim}{d}, s))) \subseteq B(e^*(\Upsilon(\underset{\sim}{d}), s))$ (Υ extended in the natural way to the product space). This definition (similar in form to our earlier ones) is justified by the following result.

8.7 Lemma

If $e^{**} \in D'$, $e^* \in C'$ and $e \in M'$ are such that e^{**} correctly implements e^* and e^* correctly implements e then for all $(\underset{\sim}{d}, s) \in D^{\Theta} \times S$ we have $T(e^{**}(\underset{\sim}{d}, s)) \subseteq e(T(\underset{\sim}{d}), s)$, (where T is extended in the natural way to D).

proof

$$
\begin{aligned}
T(e^{**}(\underset{\sim}{d}, s)) &= \Phi(\Upsilon(e^{**}(\underset{\sim}{d}, s))) \\
&= \{\theta(b) \mid b \in B(\Upsilon(e^{**}(\underset{\sim}{d}, s)))\} \\
&\subseteq \{\theta(b) \mid b \in B(e^*(\Upsilon(\underset{\sim}{d}), s))\} && (+) \\
&\subseteq \Phi(e^*(\Upsilon(\underset{\sim}{d}), s)) \\
&\subseteq e(\Phi(\Upsilon(\underset{\sim}{d})), s) = e(T(\underset{\sim}{d}), s) && (++)
\end{aligned}
$$

(+) follows by definition of e** correctly implementing e*, and (++) follows by definition of e* correctly implementing e.

Suppose op** is a k-place operator over D' and op* is a k-place operator over C'. Say that op** imp' op* if whenever $e_1^{**}, .., e_k^{**} \in D'$ and $e_1^*, .., e_k^*$ are such that each e_i^{**} correctly implements e_i^* then $op^{**}(e_1^{**}, .., e_k^{**})$ correctly implements $op^*(e_1^*, .., e_k^*)$.

The corresponding result to 8.6 clearly holds for a set of basic operators B** over D' which can be combined to implement each basic operator (in B*) over C'.

This result, together with 8.6, 8.7 and the fact that for any $d \in D$ we have $T(d) \subseteq \{\theta(\eta(b')) \mid b' \in B'(d)\}$ allows us to form a linked system of implementations with respect to which it is possible to prove predicates of the form $\underset{\sim}{X}$ (X a predicate of V) by consideration of the value taken by a program in M.

There is of course no reason why we must restrict ourselves to three levels of abstraction (D,C,M). We can introduce as many extra levels between D and C as we like, so long as the relationship between consecutive spaces has the same form as that between the old D and C. We would then have the following:

$$D=C_n \overset{\gamma_n}{\rightarrow} C_{n-1} \overset{\gamma_{n-1}}{\rightarrow} \cdots \overset{\gamma_1}{\rightarrow} C_0 \overset{\Phi}{\rightarrow} M \qquad (C_0 = C)$$
$$U_n \overset{\eta_n}{\rightarrow} U_{n-1} \overset{\eta_{n-1}}{\rightarrow} \cdots \overset{\eta_1}{\rightarrow} U_0 \qquad (U_n = V,\ U_0 = U)$$

for spaces of processes C_i with associated spaces of behaviours U_i. We would demand that they satisfy (for all $1 \leq i \leq n$) $B_{i-1}(\gamma_i(d_i)) \supseteq \{\eta_i(b) \mid b \in B_i(d_i)\}$ for each $d_i \in C_i$ (where $B_i(d_i)$ represents the set of behaviours associated with d_i). Having done this we could prove very much the same sort of result about this system and its implementations as before.

The work of this section allows us to break down our analysis of the relationships between systems into easier steps. We can construct one or more idealized versions of a system of "real" processes for use as stepping-stone(s) between our model M and the real world. We are allowed to break up the problems of proving correctness of implementations into two or more distinct parts. This work also has the advantage that once we have established the relationship between M and C we can use our knowledge of this for many different "D"s (and vice-versa).

The application of our theory

The rest of this chapter is devoted to analyzing our existing models and seeing how they might be improved. The first thing we must do is to establish the space C of "real" processes by which we are to judge them. We will first form an informal picture of what we expect a process to look like. We will then formalize these ideas into postulates on the behaviour of processes, and as a result produce a fairly abstract system C which is in a sense a unique model for them.

It should of course be noted that the following is only one possible system by which to judge our models. It does however have the advantages of its very general nature and the fact that it corresponds closely to our intuitive idea of what a process should look like.

Let us therefore suppose the following of the processes which we seek to model:

(i) That at all times a process has an internal state, and that this state and the process' environment are the only factors influencing its behaviour. The state can change only through discrete <u>actions</u> (or <u>transitions</u>). An action is instantaneous, and can either be <u>internal</u> or <u>external</u>. Internal actions are uncontrollable by the environment and indistinguishable to it (if it can observe them at all). External actions, or <u>communications</u>, can take place only with the co-operation of the environment. Only finitely many actions can occur in any given finite interval.

(ii) External actions are named by elements of some <u>alphabet</u> Σ; this name is the only feature of an external action visible to the environment. The only device by which the environment can influence the behaviour of a process is the subset of Σ in which it is willing to co-operate. This influence is restricted to what communications actually take place, not any other feature of the set.

(iii) There is some finite uniform time for which it is not possible for an action to be possible for a process without some (possibly different) action occurring. Apart from this the behaviour of a process is independent of the length of time it has been in a state, subject to the condition that only finitely many actions can occur in a finite time.

Our next step will be to reinforce the above postulates with some more formal ones. To do this we will need a notation for describing behaviour. Denote by $(\rho,X) \dot{\rightarrow} \tau$ the fact that a process in state ρ, while the environment offers set X, might perform some internal action which results in it coming into state τ. Similarly denote by $(\rho,X) \xrightarrow{a} \tau$ the fact that under the same conditions a process in state ρ might perform some external action named "a" and come into state τ. Postulates (i), (ii) and (iii) essentially imply that once we know both the initial state of a process and the possible transitions in the above form of all states, we know exactly which sequences of actions are possible for the process (relative to its environment). This will be made more precise below. The following three postulates formalize and extend another set of ideas introduced in (i), (ii) and (iii).

(P1) $(\sigma,X) \xrightarrow{a} \rho \Rightarrow a \in X$

(P2) $(\sigma,X) \dot{\rightarrow} \rho \Leftrightarrow (\sigma,\emptyset) \dot{\rightarrow} \rho$

(P3) $(\sigma,X \cup \{a\}) \xrightarrow{a} \rho \Leftrightarrow (\sigma,\{a\}) \xrightarrow{a} \rho$

The first of these simply says that a process can only carry out an external action with the co-operation of the environment. The second formalizes the notion that the environment cannot influence the possibility of an internal action occurring. The third formalizes the notion that the only influence which the environment has is through what actually occurs, not what might have occurred.

Our next step will be to bring in a set of postulates which specify how a process will behave in any environment, this behaviour being a function of the possible state transitions. To this end we will introduce the "behaviours" which we will use to describe individual execution sequences of a process. We must be careful in our choice of which type of "behaviour" to use since the sets of behaviours of processes play a central role in the first sections of this chapter. One type of "behaviour" which one might suggest is the observed response of the process to some type of experiment carried out by the environment: this would correspond well to our intuition about the meanings of our existing models, and such behaviours would lend themselves easily to the construction of correctness predicates. It is however rather early to have to decide either the nature of the experiments to be carried out by the environment or exactly what is observable by the environment (e.g. whether or not internal actions are discernable in any way by the environment).

To avoid having to make these decisions, and so as to make our space better able to model less abstract ones, we will choose a type of behaviour which attempts to record as much relevant detail as possible about an execution sequence without attempting to extract the "observable" aspects. The extraction of "observable" aspects of a behaviour is a task better left to the function "θ" which maps behaviours to their representatives in the image in the model of a process. It is necessary however to decide a few general points about our environments. Let us therefore decide that an environment has the ability to apply only a finite number of sets to a process in a finite time, but that there is no necessity for these sets to be themselves finite. Both of these decisions are consistent with the thought that the environment might itself be a process; this idea would not be so easy to entertain if the sets were always finite.

The obvious choice of what a behaviour is (bearing the above in mind) is a finite or infinite sequence of pairs (state and environment) linked by any state transitions which occur between them. In a more general space of processes it might be desirable to include a third component representing the time for which the state/environment pair persists, but because of the idealized nature of our space and the independence of our models from time there seems little need to include this extra component here. It is important however to be able to distinguish finite from infinite behaviours: there is no problem in the case of infinite sequences, since by our assumptions about the nature of processes and environments these can only represent infinite behaviours (i.e. behaviours which occupy an infinite amount of time). There is however the possibility that a finite sequence might represent an infinite experiment, for a stubborn environment might encounter a state which was unwilling either to accept any of the environment's symbols or to perform any internal action. For obvious reasons this can only occur at the end of a behaviour.

The most convenient form of the behaviours described above is finite or infinite sequences of triples of the form (σ,X,δ), for σ a state, X a set offered by the environment and δ either an element of Σ (representing a communication),

"." (representing an internal action), "*" (representing a finite or infinite fruitless wait which is longer than the bound on the inactivity of a state which can do something), or "-" (representing a finite fruitless wait which is shorter than this bound). The only possible "δ"s for the final triple of a finite sequence are "*" and "-".

In the following we will typically use $\underline{a}$, $\underline{b}$,... to denote sequences of triples. If c is a process we assume that it is endowed with a set $B(c)$ of behaviours, the set of behaviours which can actually occur for c. Thus when we define a system of processes it is necessary to define the possible transitions of the processes' states and the sets of behaviours of the processes. It is natural to expect transitions and behaviours to be closely related; the following postulates describe this relation, and also complete the formalization of the "informal" postulates (i), (ii) and (iii).

(Q1) $\langle\rangle \in B(c)$

(Q2) $\langle(\sigma,X,-)\rangle \in B(c)$ iff σ is the initial state of c.

(Q3) $\underline{a}\langle(\sigma,X,-)\rangle \in B(c)$ & $(\sigma,X) \overset{\delta}{\rightarrow} \rho \Leftrightarrow \underline{a}\langle(\sigma,X,\delta)(\rho,Y,-)\rangle \in B(c)$

(Q4) $\underline{a}\langle(\sigma,X,-)(\rho,Y,\delta)\rangle\underline{b} \in B(c) \Leftrightarrow \sigma=\rho$ & $\underline{a}\langle(\sigma,Y,\delta)\rangle\underline{b} \in B(c)$

(Q5) $\underline{a}\langle(\sigma,X,-)\rangle\underline{b} \in B(c)$ & $(\neg\exists\rho,\delta.\delta\in X\cup\{\cdot\}$ & $(\sigma,X) \overset{\delta}{\rightarrow} \rho)$
$\Leftrightarrow \underline{a}\langle(\sigma,X,*)\rangle\underline{b} \in B(c)$

(Q6) $\underline{a}\langle(\sigma,X,-)\rangle\underline{b} \in B(c) \Rightarrow \underline{a}\langle(\sigma,X,-)\rangle \in B(c)$

(Q7)* If $\underline{a} = \langle(\sigma_o,X_o,\delta_o)\ldots(\sigma_i,X_i,\delta_i)\ldots\rangle$ is an infinite behaviour sequence such that infinitely many of the δ_i are elements of $\Sigma\cup\{\cdot\}$ and such that $\langle(\sigma_o,X_o,\delta_o)\ldots(\sigma_i,X_i,-)\rangle$ is an element of $B(c)$ for all i, then $\underline{a} \in B(c)$.

(Q7) If $\underline{a} = \langle(\sigma_o,X_o,\delta_o)\ldots(\sigma_i,X_i,\delta_i)\ldots\rangle$ is an infinite behaviour sequence such that δ_i = "-" for all $i \geqslant k$ and such that $\langle(\sigma_o,X_o,\delta_o)\ldots(\sigma_i,X_i,-)\rangle \in B(c)$ for all i, then $\underline{a} \in B(A)$ iff $\langle(\sigma_o,X_o,\delta_o)\ldots(\sigma_k,Z,*)\rangle \in B(c)$, where $Z = \bigcup_{j=k}^{\infty}(\bigcap_{i=j}^{\infty} X_i)$.

(*) On certain occasions it will be necessary to replace Q7 with a less severe postulate.

The first two of these are easy to interpret. Q3 says that a transition can occur in behaviours exactly when it is implied possible by the transition relations. Q4 says both that a finite application of a set without response by the environment cannot influence the behaviour of a process and (together with Q2 and Q3) that it is always possible that the state may fail to respond to any set only applied for a short time. (This can be regarded as a convenient fiction which makes the structure of behaviours more tractable, and therefore makes these postulates simpler. For more about this assumption see later.) Q5 says that a fruitless wait which is longer than our bound on the bound on inactivity is possible when, and only when, there is no internal or external transition possible for the state in the environment which is offered. Q6 says that any initial part of a possible behaviour is possible. Q7 essentially postulates the absence of fairness, saying that an infinite sequence of transitions is possible if each of its finite approximations is possible. If we are able to assume this it removes much complexity from our arguments; we will find however that it is sometimes impossible to define reasonable operators without an element of fairness. Q8 reflects the fact that the environment can only apply a finite number of sets in a finite time: thus no transition can be possible during the whole of a final portion of an infinite sequence of fruitless waits, as this would contradict our assumption that no transition can be possible infinitely without something occurring.

It is possible to simplify the form of some of the Q-postulates if we assume the P-postulates. It is however desirable to keep them separate so that we can change the form of the P-postulates without having to alter all the Q-postulates as well.

One useful feature of the system of postulates set out above (Q1 - Q8) is that whether or not the system satisfies P1 - P3 the behaviour set B(c) of a process is always exactly determined by its initial state and the transitions $(\sigma,X) \dot{\rightarrow} \rho$ and $(\sigma,X) \overset{a}{\rightarrow} \rho$ which are possible for its states. Thus we can exactly specify the nature of a system of processes by defining which transitions are possible for its states and saying that it satisfies Q1 - Q8. This is a freedom which we will often make use of later in our abstract discussion of spaces of processes. If one were to decide to alter one of Q1 - Q8 (for example Q7) it would be useful if this "exact determination of behaviour" could in some sense be preserved.

An immediate consequence of the postulates P1 - P3 is that we can simplify the structure of our transition relations. If we write $\sigma \stackrel{\cdot}{\to} \rho$ for $(\sigma,\emptyset) \stackrel{\cdot}{\to} \rho$ and $\sigma \stackrel{a}{\to} \rho$ for $(\sigma,\Sigma) \stackrel{a}{\to} \rho$ then it is easy to see that the possible old-style transitions are exactly determined by a knowledge of which new-style ones are possible. Thus for all states σ & ρ, environments X and $a \in \Sigma$ we have:

$$(\sigma,X) \stackrel{\cdot}{\to} \rho \iff \sigma \stackrel{\cdot}{\to} \rho$$
$$(\sigma,X) \stackrel{a}{\to} \rho \iff \sigma \stackrel{a}{\to} \rho \ \& \ a \in X.$$

Before we continue with our study of the systems which satisfy these postulates it is perhaps desirable to consider briefly just how valid and useful the postulates are. The first thing which we should remark on is the fact that our postulates do indeed seem to paint a very idealized picture of what a process is likely to be. It is certainly tempting, and also justifiable, to relax some of them somewhat. It is important to note however that exception of our assumption of the existence of a uniform bound on the "idle time" of a state (a postulate which is by no means critical), each of the postulates which might be regarded as controversial has the effect of assuming maximal unpredictability on the part of our processes. This is because of our emphasis on <u>possible</u>, rather than certain, behaviours, which means that the absence of a behaviour is never checkable by any experimenter. The following paragraphs are brief analyses of some of the points at which this is true.

a) One might regard our postulate that actions occur instantaneously as suspect, and also perhaps our assumption that the behaviour of a state is independent of time. One might prefer to regard actions as events which take a non-zero time to complete, and to believe that as a state develops it can acquire more possible actions (reductions can be modelled by internal actions). In many ways the best way of dealing with the first of these points is to identify the start of each action with the action itself, so that "actions" are again thought of as instantaneous. It is interesting that this identification is also inconsistent with the postulate that a state's possible transitions remain constant throughout its life. We are forced to the conclusion that the possible transitions of a state may increase gradually during the first part of its life (on the completion of the various actions which may be in progress by the possibly distributed state).

To take account of this type of time dependance we would have to modify our transition relations to include a time component, so that for example $(\sigma,X,t) \overset{\cdot}{\rightarrow} \rho$ would mean that after spending time t in state σ a process might, in environment X, perform some internal action which transforms it into state ρ. We would have to modify some of our postulates. Firstly we would assume that the bound on "idle time" was increased to take account of the possible "warming up" time of states. Most of the necessary modifications to the P and Q postulates are fairly obvious. The only major changes would be in the form of Q5 and the addition of a P4.

(P4) $\quad (\sigma,X,t) \overset{\delta}{\rightarrow} \rho \ \ \& \ \ t < t' \ \ \Rightarrow \ \ (\sigma,X,t') \overset{\delta}{\rightarrow} \rho$

(Q5') $\quad \underline{a}\langle(\sigma,X,-)\rangle\underline{b} \in B(c) \ \ \& \ \ \neg(\exists t,\delta,\rho.\ \delta \in X \cup \{\cdot\} \ \ \& \ \ (\sigma,X,t) \overset{\delta}{\rightarrow} \rho)$

$\qquad \Leftrightarrow \ \ \underline{a}\langle(\sigma,X,*)\rangle\underline{b} \in B(c)$

It should not be too hard to prove that if we took a process whose states satisfied the modified postulates, and mapped it to another with the same transitions but independent of time, then the two would have identical sets of behaviours. While P4 would probably not be necessary for such a proof, its assumption is crucial in making Q5' reasonable.

b) The "$\Leftarrow$" halves of P2 and P3 might be regarded as being slightly suspect. This is because we might like to believe that a process might prefer one (external) action to another (internal or external). If this were possible then it might occur that some transition $(\rho,X) \overset{\cdot}{\rightarrow} \tau$ or $(\rho,X) \overset{a}{\rightarrow} \tau$ be possible only when some symbol b is not an element of the set X. It would be reasonable to expect that in this case there is some state υ such that $(\rho,X\cup\{b\}) \overset{b}{\rightarrow} \upsilon$ for all X such that the original transition $(\rho,X) \overset{a}{\rightarrow} \tau$ or $(\rho,X) \overset{\cdot}{\rightarrow} \tau$ is possible. We would need a single extra postulate to replace the "$\Leftarrow$" halves of P2 and P3, for example P4' below.

(P4') There is some partial order ">" on the set $\{(a,\tau) \mid (\rho,\{a\}) \xrightarrow{a} \tau\} \cup \{(\cdot,\tau) \mid (\rho,\emptyset) \dot{\rightarrow} \tau\}$ with the following properties:

a) The partial order has no infinite ascending chains (so that all its non-empty subsets have maximal elements).

b) Every pair of the form $(\cdot,\tau)$ is minimal; there can be no a,τ & υ such that $(a,\tau) > (a,\upsilon)$.

c) For each set X we have $(\rho,X) \xrightarrow{\delta} \tau$ if and only if (δ,τ) is maximal in $\{(a,\upsilon) \mid a \in X \;\&\; (\rho,\{a\}) \xrightarrow{a} \upsilon\} \cup \{(\cdot,\upsilon) \mid (\rho,\emptyset) \dot{\rightarrow} \upsilon\}$ (with respect to the partial order).

Note that (c) above actually implies (b), and also the "$\Rightarrow$" clauses of P2 and P3. The P4' which results from the vacuous partial order is equivalent to P2 and P3. The necessity for condition (a) above arises from the ridiculous situations which would arise if it did not hold: the state having a sequence of possible actions each of which is made impossible by the presence of another, so that in fact no action actually takes place.

One should be able to prove that if we took a system whose states satisfied the above postulate and P1 and Q1-Q8 and mapped it to another which satisfied our original postulates, each state being mapped to one with exactly the same transitions of the form $(\rho,\emptyset) \dot{\rightarrow} \tau$ and $(\rho,\{a\}) \xrightarrow{a} \tau$, then the set of behaviours of each process is contained in the set of behaviours of its image. The critical feature of a proof of this will be (for Q5) the fact that for each state ρ (with image ρ') and set X we have $\neg\exists\tau,\delta.(\rho,X) \xrightarrow{\delta} \tau \;\Rightarrow\; \neg\exists\tau,\delta.(\rho',X) \xrightarrow{\delta} \tau$.

c) When it is assumed the absence of fairness (Q7) might be regarded a being a little restrictive. It is easy to see however that the set of behaviours of a process in a space not satisfying Q7 is contained in that of the process' image under the obvious map to the corresponding space which does satisfy it. If we were to drop the "convenient fiction" of the "$\Leftarrow$" half of Q4 and bring in weaker postulates the same would be true.

Each of the sections (a), (b) & (c) demonstrates that we should be able to produce maps from "weaker" spaces to our idealized spaces which increase the sets of behaviours. There is a strong sense in which a process with more behaviours than another can be regarded as being more non-deterministic than it, so that by idealizing a process we are in fact assuming it to be worse than it really is. It is important to note that this work shows that spaces which satisfy our postulates are good candidates for being the space "C" in the section on "compositions of mappings", since the correctness condition for the function $Y:D \to C$ was that it increased the sets of behaviours of processes.

There is clearly much scope for further work on the above topic. Firstly there is a need for formal proofs of the various results which were derived informally above. Secondly it would be interesting to know how many of the postulates we could weaken simultaneously (e.g. can we weaken P2 and P3 as well as removing independence of states from time?). There should be no great difficulty in solving these problems. The formal analysis of behaviour sets which is necessary will be made easier by the following result.

8.8 Theorem

In a system satisfying Q1 - Q8 (whether or not it satisfies P1, P2 and P3) the behaviour set $B(c)$ of a process is uniquely determined by a knowledge of its initial state and a knowledge of which transitions $(\rho,X) \overset{a}{\to} \tau$ and $(\rho,X) \overset{\cdot}{\to} \tau$ are possible for all its states.

The proof of this is an easy induction on the length of finite elements of $B(c)$. The infinite elements fall into place using Q6, Q7 and Q8.

In systems which satisfy P1, P2 and P3 in addition to Q1-Q8 we can deduce that the behaviour sets of processes are uniquely determined by a knowledge of the possible transitions $\rho \overset{\cdot}{\to} \tau$ and $\rho \overset{a}{\to} \tau$. We can use this fact not only to help to justify the use of such spaces to model weaker ones, but also to prove the existence of canonical spaces satisfying our postulates which can be held to model **many others.** It is convenient at this stage to identify processes with their initial states, thereby embedding spaces of processes into their underlying spaces of states. Without any significant loss of generality we can clearly identify the spaces of processes and states.

Suppose that C and C' are two spaces of processes (states) satisfying our postulates, and that F:C → C' is a function from one to the other. Define F to be a morphism if it satisfies the conditions:

(i) $\forall\sigma.\forall\rho.\forall\delta\in\Sigma\cup\{\cdot\}.\ \sigma \overset{\delta}{\Rightarrow} \rho \Rightarrow F(\sigma) \overset{\delta}{\Rightarrow} F(\rho)$

(ii) $\forall\sigma.\forall\rho.\forall\delta\in\Sigma\cup\{\cdot\}.\ F(\sigma) \overset{\delta}{\Rightarrow} \rho \Rightarrow \exists\tau\in F^{-1}(\rho).\ \sigma \overset{\delta}{\Rightarrow} \tau$.

Condition (i) essentially says that all transitions possible for an element of C must also be possible for its image, and that this is also true of the process after all transitions. Condition (ii) says that all transitions possible for the image of a process are also possible for the process itself.

8.9 Lemma

If C, C' and C" are three spaces of processes with morphisms F:C → C' and G:C' → C", then G∘F is a morphism from C to C".

proof

(i) If σ,ρ C' and $\delta\in\Sigma\cup\{\cdot\}$ are such that $\sigma \overset{\delta}{\Rightarrow} \rho$ then
$F(\sigma) \overset{\delta}{\Rightarrow} F(\rho)$ (as F is a morphism)
$\Rightarrow G(F(\sigma)) \overset{\delta}{\Rightarrow} G(F(\rho))$ (as G is a morphism).

(ii) If $\sigma\in C$, $\rho\in C''$ and $\delta\in\Sigma\cup\{\cdot\}$ are such that $G(F(\sigma)) \overset{\delta}{\Rightarrow} \rho$ then there is some $\tau\in C'$ s.t. $F(\sigma) \overset{\delta}{\rightarrow} \tau$ and $G(\tau) = \rho$. This in turn implies that there is some $\upsilon\in C$ s.t. $\sigma \overset{\delta}{\Rightarrow} \upsilon$ and $F(\upsilon) = \tau$. We thus have $\sigma \overset{\delta}{\Rightarrow} \upsilon$ and $\upsilon\in(G\circ F)^{-1}(\rho)$.

From here on let us use the term P,Q-space for a space of processes (states) which satisfies P1-3 and Q1-8. If C and C' are two P,Q-spaces and F:C → C' is any function we can extend F to the spaces of triples used in forming behaviours (and hence to the spaces of behaviours themselves) by $F(\sigma,X,\delta) = (F(\sigma),X,\delta)$. The following is a result which can be proved by induction which shows that when F is a morphism the spaces of behaviours of a process and its image under F correspond in a useful way.

8.10 Lemma

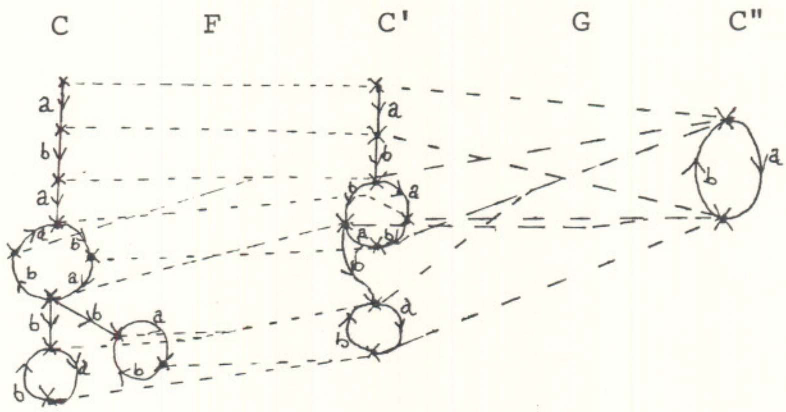
If C and C' are two P,Q-spaces and F:C → C' is a morphism, then for all $\rho\in C$ we have $B(F(\rho)) = F(B(\rho))$.

The proof of this result is omitted, being a fairly lengthy argument by cases. The author's proof of the case of infinite sequences uses the axiom of choice.

The chief significance of this result is that it tells us that so far as the environment is concerned (assuming that it cannot "see" the actual internal state of a process) there is nothing to distinguish a process from its image under a morphism.

8.11 <u>Example</u>

The following diagram illustrates the concept of morphisms between spaces of processes. C, C' and C" are three P,Q-spaces, their possible actions indicated by unbroken lines. The two maps F and G (indicated by broken lines) are morphisms.



The following results will help us in our search for spaces which are canonical in the sense of having unique morphisms to them from large classes of P,Q-spaces.

8.12 <u>Lemma</u>

Suppose that C, C' and C" are three P,Q-spaces with morphisms $F:C \rightarrow C'$ and $G:C \rightarrow C''$. We can define an equivalence relation on C by $\sigma \sim \rho \Leftrightarrow \exists k, \tau_1, \upsilon_1, \ldots, \upsilon_k . \sigma = \tau_1 \;\&\; \rho = \upsilon_k \;\&\; \forall i . F(\tau_i) = F(\upsilon_i) \;\&\; \forall i . G(\upsilon_i) = G(\tau_{i+1})$.

The quotient space $C/\sim$ can be made into a P,Q-space by defining the transitions by $\overline{\sigma} \xrightarrow{\delta} \overline{\rho} \Leftrightarrow \exists \tau \in \overline{\sigma}, \upsilon \in \overline{\rho} . \tau \xrightarrow{\delta} \upsilon$ and the behaviours by Q1-8. This is a well-defined P,Q-space, and the map $H(\rho) = \overline{\rho}$ is a morphism.

The proof of this result is technical and is omitted.

The chief purpose of 8.12 is to show that if F and G are two morphisms of a P,Q-space C then there is a third which identifies all pairs of processes identified by either F or G. An immediate corollary to 8.12 is thus the fact that the relation on C defined by "$\sigma \backsim' \rho \Leftrightarrow \exists$ some morphism F which identifies σ & ρ" is an equivalence relation. This equivalence relation is used to prove 8.14 below.

If C is any P,Q-space let us define a non-empty subset D of C to be a <u>subspace</u> of C if it is closed under "$\rightarrow$" in the sense $\exists \sigma \in D.\ \exists \delta\ .\ \sigma \xrightarrow{\delta} \rho \Rightarrow \rho \in D$. (An element of D has the same transitions and behaviours which it has as an element of C.) It is clear that with the spaces of transitions and behaviours described D is a P,Q-space.

8.13 <u>Lemma</u>

a) If C" is a subspace of C' and C' is a subspace of C then C" is a subspace of C.

b) If F:C → C' is a morphism and D is a subspace of C then F(D) is a subspace of C'. (Corollary: Im(F) is a subspace of C'.)

c) If F:C → C' is a morphism and D is a subspace of C' then $F^{-1}(D)$ is a subspace of C.

d) If D is a subspace of C and F:D → D' is a morphism such that $F(\sigma) = F(\rho)$ $(\rho, \sigma \in D)$, then there is a space C' and a morphism F':C → C' such that $F'(\sigma) = F'(\rho)$.

<u>proof</u>

(a) is obvious; (b) follows from clause (ii) of our definition of morphisms; (c) follows from clause (i) of our definition of morphisms; (d) follows by consideration of the equivalence relation $\backsim''$ defined $\upsilon \backsim'' \tau$ if and only if <u>either</u> $\upsilon = \tau$ and $\upsilon \notin D$ <u>or</u> $\tau, \upsilon \in D$ and $F(\upsilon) = F(\tau)$. We can make $C/\backsim''$ into a P,Q-space by endowing it with the transitions $\bar{\upsilon} \xrightarrow{\delta} \bar{\tau}$ s.t. $\exists \upsilon' \in \bar{\upsilon},\ \tau' \in \bar{\tau}.\ \upsilon' \xrightarrow{\delta} \tau'$. It is not hard to show that the map $F'(\upsilon) = \bar{\upsilon}$ is the morphism we require.

Having established the technical results 8.11 and 8.12 we are in a position to prove the following important result.

8.14 <u>Theorem</u>

If C is any P,Q-space and $\backsim'$ is the equivalence relation defined on it as above then the quotient space $C/\backsim'$, when made into a P,Q-space with transitions $\overline{\sigma} \xrightarrow{\delta} \overline{\rho} \Leftrightarrow \exists \tau \in \overline{\sigma},\ \upsilon \in \overline{\rho}.\ \tau \xrightarrow{\delta} \upsilon$, has the property that the map $F^*: C \to C/\backsim'$ defined by $F^*(\sigma) = \overline{\sigma}$ is a morphism. Call $C/\backsim'$ by the name C^*. F^* is the only morphism from C to C^*, and if G is any morphism from C to a P,Q-space D then there is a unique morphism from Im(G) to C^*, and this morphism G^* satisfies $G^* \circ G = F^*$.

proof

We will first show that F^* is a morphism. Trivially $\sigma \xrightarrow{\delta} \rho \Rightarrow F^*(\sigma) \xrightarrow{\delta} F^*(\rho)$ for all $\sigma, \rho \in C$ (since $\sigma \in \overline{\sigma}$ and $\rho \in \overline{\rho}$). We must therefore show that $\overline{\sigma} \xrightarrow{\delta} \overline{\rho} \Rightarrow \exists \tau \in \overline{\rho}.\ \sigma \xrightarrow{\delta} \tau$. Suppose then that $\overline{\sigma} \xrightarrow{\delta} \overline{\rho}$; this means that there exist some $\sigma' \in \overline{\sigma}$ and $\rho' \in \overline{\rho}$ s.t. $\sigma' \xrightarrow{\delta} \rho'$. By definition of our equivalence relation there exist morphisms F and G of C such that $F(\sigma) = F(\sigma')$ and $G(\rho) = G(\rho')$. By 8.12 this implies the existence of some morphism $H: C \to C'$ (for some C') such that $H(\rho) = H(\rho')$ and $H(\sigma) = H(\sigma')$. Since H is a morphism we must have that $H(\sigma) \xrightarrow{\delta} H(\rho)$, which in turn implies that there exists some ρ'' s.t. $H(\rho) = H(\rho'')$ and $\sigma \xrightarrow{\delta} \rho''$. We have thus shown the existence of some H and ρ'' s.t. $H(\rho) = H(\rho'')$ (so that $\overline{\rho} = \overline{\rho''}$) and $\sigma \xrightarrow{\delta} \rho''$. This completes the proof that F^* is a morphism.

If C' is any other P,Q-space with a morphism $G: C \to C'$ then Im(G) is a subspace of C' by 8.13. We can define a map $G^*: Im(G) \to C^*$ by $G^*(G(\rho)) = F^*(\rho)$. This map is well-defined since whenever $G(\upsilon) = G(\tau)$ we have $\tau \backsim' \upsilon$, which implies that $F^*(\tau) = F^*(\upsilon)$. G^* is a morphism since $G(\tau) \xrightarrow{\delta} G(\upsilon) \Rightarrow \exists \upsilon' \in G^{-1}(\upsilon).\ \tau \xrightarrow{\delta} \upsilon'$; we then have $F^*(\tau) \xrightarrow{\delta} F^*(\upsilon')$, which implies $G^*(G(\tau)) \xrightarrow{\delta} G^*(G(\upsilon))$ as required (as $G(\upsilon') = G(\upsilon)$). Secondly if $G^*(G(\rho)) \xrightarrow{\delta} \tau$ then $\tau' \in F^{*-1}(\tau).\ \rho \xrightarrow{\delta} \tau'$ (as F^* is a morphism); since G is a morphism we then get $G(\rho) \xrightarrow{\delta} G(\tau')$, which is what we require since $G(\tau') \in G^{*-1}(\tau)$ by construction. Thus G^* is indeed a morphism as claimed, and it is obvious from its definition that $G^* \circ G = F^*$.

It remains to examine the question of uniqueness. If F' were a second morphism $F': C \to C^*$ then by the above there exists a morphism $F'': Im(F') \to C^*$ such that $F^* = F'' \circ F'$. We must show that F" is the identity morphism. Claim first that F" is one-one. If it were not then we could (by 8.13(d)) find a space C" and a morphism $F^+: C^* \to C''$ which was not one-one. However $F^+ \circ F^*$ is a morphism of C, so by the above there exists a morphism $F^{**}:C'' \to C^*$ such that $F^{**} \circ F^{+} \circ F^* = F^*$. This means that $F^{**} = (F^{+})^{-1}$, contradicting the fact that F^{+} is not injective. Hence F'' is injective as claimed.

Since F'' is injective and surjective we can define an equivalence relation $\backsim^*$ on C^* by $\upsilon \backsim^* \tau \Leftrightarrow \exists k. F''^{k}(\tau) = \upsilon$ or $F''^{k}(\upsilon) = \tau$. If we (as usual) consider $C^*/\backsim^*$ as a P,Q-space with transitions $\overline{\upsilon} \xrightarrow{\delta} \overline{\tau} \Leftrightarrow \exists \upsilon' \in \overline{\upsilon}, \tau' \in \overline{\tau}. \upsilon' \xrightarrow{\delta} \tau'$ the map $G:C^* \to C^*/\backsim^*$ defined $G(\sigma) = \overline{\sigma}$ can be shown to be a morphism. Now $G \circ F^*$ is a morphism from C to $C^*/\backsim^*$, so by our earlier work there exists a morphism $G^*:C^*/\backsim^* \to C^*$ such that $G^* \circ G \circ F^* = F^*$. Hence G is injective, so all the equivalence classes of $\backsim^*$ have only one element. This is easily seen to imply that F'' is the identity map from C^* to C^*.

We have thus shown that $F^* = I \circ F'$ (I the identity map on C^*), which implies that F^* is indeed the only morphism from C to C^*. The uniqueness of the maps $G^*:\mathrm{Im}(G) \to C^*$ follows immediately from the uniqueness of F^*.

This result has a neat statement in category theory: C^* is a terminal object in the category of onto morphic images of C (with morphisms as arrows).

The next result, which has the same type of proof, is an extension to the above.

8.15 Theorem

Suppose that C and D are P,Q-spaces and that $F:C \to D$ is a morphism, then if D^* is the abstraction of D produced by 8.14 and $F^*:D \to D^*$ is the (unique) morphism between them, we have that there is a unique morphism from D to C^*, and its image is isomorphic to C^* (the abstraction of C produced by 8.14). Thus the compound map $F^* \circ F$ is independent of our choice of F.

Our next step will be to attempt to build up a "universal" P,Q-space into which all others can be mapped by a unique morphism. One can attempt to do this in two essentially different ways. The most obvious approach is to try to construct such a space from scratch. This would have the advantage that we would know exactly how it was constructed, making it easier to calculate with.

The other approach is non-constructive. Suppose we are given a set S of P,Q-spaces (for example a representative of each of the isomorphism classes of P,Q-spaces with less than or equal to any given cardinality), then let us form a "separated union" $\underline{S}$ of these spaces by attaching to each of its elements the name of the set from which it originally came. ($\underline{S} = \{(\sigma,C) \mid C \in S \ \&\ \sigma \in C\}$) If the space is given the transitions inherited from the elements of S ($(\sigma,C) \overset{\delta}{\to} (\rho,D)$ iff $C = D \ \&\ \sigma \overset{\delta}{\to} \rho$ in C), then $\underline{S}$ can be thought of as a P,Q-space containing a copy of each element of S as a subspace. The space $\underline{S}^*$ can be regarded as a canonical space for S, since each $C \in S$ trivially has a morphism into $\underline{S}$, and so by 8.15 there is a unique morphism from each element of S to $\underline{S}^*$.

If as suggested we take S to be a set of representatives of isomorphism classes of spaces with less than a given cardinality, then it is clear that there is a unique morphism from every P,Q-space with less than this cardinality to $\underline{S}^*$.

If it were possible to find a bound on the cardinalities of the spaces $\underline{S}^*$ (though we will shortly be able to deduce that it is not possible) then it would be easy to extend the above work to a production of a completely universal space. The above gives a sufficient taste of the type of methods which might be adopted to produce universal spaces in non-constructive ways. With a little more sophistication in category theory one might extend it further, but without further ado we will switch over to the more constructive approach.

Our idea of what a morphism is is of a map which preserves the exact shape of the possible behaviours of a process. Instead of analysing spaces directly through their morphisms as we have done hitherto it does not seem unreasonable to try to analyse them directly through their shapes of transition spaces. We can define a space (for any alphabet Σ) which attempts to record all the possible shapes of transitions up to depth n (for any $n \in N$).

$T_0 = \{\emptyset\}$ (there is only one possible shape of depth 0)

$T_{n+1} = \mathcal{P}((\Sigma \cup \{\cdot\}) \times T_n)$ (a possible shape of depth n+1 can be regarded as a relation between the possible transitions and the shapes of depth n).

It is easy to see that for all n $T_n \subseteq T_{n+1}$. Because of this we can regard T_n as a P,Q-space in the obvious way, with the transitions $\sigma \xrightarrow{\delta} \rho \Leftrightarrow (\delta,\rho) \in \sigma$. It is possible to show that every morphism of T_n is injective, which tells us that T_n is isomorphic as a P,Q-space to T_n^*.

8.16 Lemma

Every morphism of T_n is injective.

proof

We prove this by induction on n; the result is trivially true when n=0, since T_0 has only one element.

Suppose true for T_n, and that $F:T_{n+1} \to C$ is a morphism for some space C. T_n is a subspace of T_{n+1}, so by induction F is injective on this subspace.

Suppose that σ and ρ are two distinct elements of T_{n+1}. There must be some $\delta \in \Sigma \cup \{\bullet\}$ and $\tau \in T_n$ such that (δ,τ) is contained in $\sigma - \rho$ (without loss of generality). We must show that $F(\sigma) \neq F(\rho)$. Since F is a morphism we have $F(\sigma) \xrightarrow{\delta} F(\tau)$; if $F(\sigma) = F(\rho)$ then there would be some $\gamma \in T_{n+1}$ s.t. $\rho \xrightarrow{\delta} \gamma$ and $F(\gamma) = F(\tau)$. However $\rho \xrightarrow{\delta} \gamma \Rightarrow \gamma \in T_n$, and F is injective on T_n, which implies that $\gamma = \tau$, contradicting the fact that $(\delta,\tau) \notin \rho$. Thus F is injective, completing our inductive proof.

The next step is to define functions which, given a process, produce the representation of its depth n behaviour. Given a P,Q-space we can expect to find a function $H_n^C : C \to T_n$ for each $n \in N$.

$$H_0^C(\sigma) = \emptyset$$
$$H_{n+1}^C(\sigma) = \{(\delta, H_n^C(\rho)) \mid \sigma \xrightarrow{\delta} \rho\}$$

8.17 Lemma

a) If D is a subspace of C and $\sigma \in D$, then the values $H_n^D(\sigma)$ produced relative to D are the same as those $H_n^C(\sigma)$ produced relative to C.

b) If $F:C \to D$ is a morphism then $H_n^C(\sigma) = H_n^D(F(\sigma))$ for all $\sigma \in C$.

c) If $F:C \to D$ is a morphism and $H_n^C(\sigma) \neq H_n^C(\rho)$ then we can be certain that $F(\sigma) \neq F(\rho)$.

The proofs of (a) and (b) are straightforward deductions from our definitions. Part (c) is a corollary to part (b).

One of the effects of 8.17 is that it allows us to ignore the superscript "C" in the notation $H_n^C(\sigma)$ without being likely to introduce errors. Henceforth we will omit it on the understanding that it could be inserted if desired.

8.18 Lemma

If we regard the spaces T_n as P,Q-spaces in the manner described earlier then the following are true:

a) If $\sigma \in T_n$ and $m \geqslant n$ then $H_m(\sigma) = \sigma$.

b) For any space C and $\sigma \in C$ we have $H_n(H_m(\sigma)) = H_k(\sigma)$ for all n,m,k s.t. k=min(n,m).

The proofs of these results are just easy inductions.

One might hope that since we have established (8.17(c)) that two processes mapped to different elements of any T_n must be kept separate by any morphism, there might be some reverse implication: if two processes are mapped to the same element of T_n for each n then they can be identified by some morphism. With this hope in mind we can define a space of behaviour spaces T_ω, which is the set of sequences of elements from the T_i which match up under the functions H_i.

$$T_\omega = \{(\sigma_0, \sigma_1, \ldots, \sigma_i, \ldots) \mid \forall i. \sigma_i \in T_i \;\&\; H_{i+1}(\sigma_{i+1}) = \sigma_i\}$$

(T_ω is just the inverse limit of the spaces T_i with projection functions H_i.)

We can easily define a function $H_\omega : C \rightarrow T_\omega$ for any P,Q-space C by $H_\omega(\sigma) = (H_0(\sigma), H_1(\sigma), \ldots, H_i(\sigma), \ldots)$. The space T_ω is naturally made into a P,Q-space by the transitions $\underline{\sigma} \overset{\delta}{\rightarrow} \underline{\rho} \Leftrightarrow \forall i. \sigma_{i+1} \overset{\delta}{\rightarrow} \rho_i$ (where σ_i represents the ith component of the sequence $\underline{\sigma}$).

8.19 Lemma

If we regard T_ω as a P,Q-space as above, then:

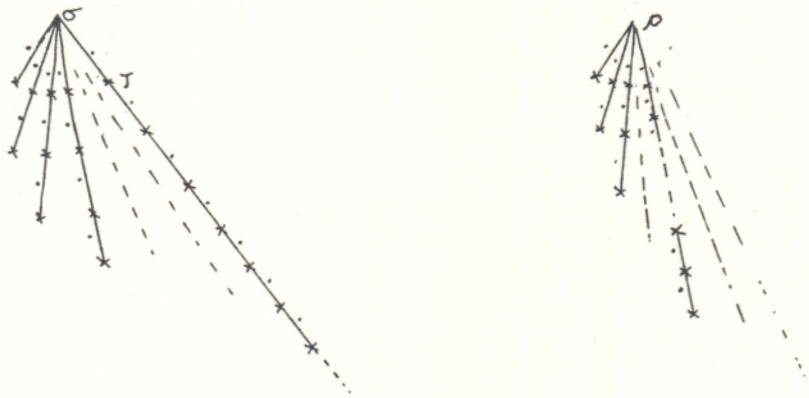
a) $H_n(\underline{\sigma}) = \sigma_n$; $H_\omega(\underline{\sigma}) = \underline{\sigma}$ for all $\underline{\sigma} \in T_\omega$.

b) Every morphism of T_ω is injective.

The proof of (a) is an induction on n; part (b) follows by part (a) and 8.17(c).

Note that (a) implies that for any space C and $\sigma \in C$ we have $H_n(H_\omega(\sigma)) = H_n(\sigma)$ for all n, and $H_\omega(\sigma) = H_\omega(H_\omega(\sigma))$.

One might hope that the map H_ω is a morphism; unfortunately however this is not so. This is because even though two processes have the same shapes of behaviour for all finite depths it does not mean that they are sufficiently similar to be identified by a morphism. As an example consider the two processes represented by the following diagram:



"σ" has the potential to perform an infinite sequence of internal actions whereas "ρ" does not. There can be no morphism F which identifies σ and ρ, for otherwise there would be some "γ" such that $\rho \dot{\rightarrow} \gamma$ and $F(\gamma) = F(\tau)$. It is easy to prove inductively that this cannot be so. It is also not hard to verify that $H_n(\sigma) = H_n(\rho)$ for all $n \in N$, which implies that $H_\omega(\sigma) = H_\omega(\rho)$.

It is an easy consequence of our results that the higher the index of the function H_α, the more distinctions it makes. It is not unnatural to define such functions for higher ordinals than ω, therefore. For <u>countable</u> ordinals this can be done in a very similar way to the above. We define spaces T_α and projection functions H_α by mutual recursion.

$T_0 = \{\emptyset\}$

$H_0(\sigma) = \emptyset$

$\emptyset$ has no transitions;

$T_{\alpha+1} = \wp((\Sigma \cup \{\cdot\}) \times T_\alpha)$

$H_{\alpha+1}(\sigma) = \{(\delta, H(\rho)) \mid \sigma \xrightarrow{\delta} \rho\}$

$T_{\alpha+1}$ has transitions $\sigma \xrightarrow{\delta} H_{\alpha+1}(\rho)$ if $(\delta, \rho) \in \sigma$

$T_\lambda = \{\underline{\sigma} \mid \underline{\sigma}$ is a λ-sequence of elements of $\bigcup_{\beta<\lambda} T_\beta$ s.t.

$\beta < \lambda \Rightarrow (\sigma_\beta \in T_\beta \ \& \ ((\gamma < \beta) \Rightarrow H_\gamma(\sigma_\beta) = \sigma_\gamma))\}$

$H_\lambda(\sigma) = \underline{\rho}$, where $\rho_\alpha = H_\alpha(\sigma)$ for $\alpha < \lambda$.

$\underline{\sigma}$ has transition $\underline{\sigma} \xrightarrow{\delta} \underline{\rho} \Leftrightarrow (\delta, \rho_\gamma) \in \sigma_{\gamma+1}$ for all $\gamma < \lambda$.

For countable α it is possible to prove several results about T_α and H_α which are extensions of our earlier ones. Note that (b) below is necessary for the well-definedness of H_λ (λ limit ordinal) in that $H_\lambda(\sigma)$ is only an element of T_λ if $H_\beta(H_\alpha(\sigma)) = H_\beta(\sigma)$ for all $\beta<\alpha<\lambda$.

8.20 Theorem

If we regard each of the spaces T_α as a P,Q-space with the transitions described above then each of the following holds.

a) If $\sigma \in T_\alpha$ then $H_\alpha(\sigma) = \sigma$.

b) If C is a P,Q-space and $\sigma \in C$ then $\beta \leq \alpha \Rightarrow H_\beta(H_\alpha(\sigma)) = H_\beta(\sigma)$.

c) If λ is a limit ordinal and $\underline{\sigma} \in T_\lambda$ then $\alpha<\lambda \Rightarrow H_\alpha(\underline{\sigma}) = \sigma_\alpha$.

d) Every morphism of T_α is injective.

e) If $\sigma \in C$ and $F: C \to D$ is a morphism, then $H_\gamma(\sigma) = H_\gamma(F(\sigma))$.

These results can all be proved by technical manipulations, some of which (in the author's proof) depend on the fact that every countable limit ordinal λ has an ω-sequence of ordinals $\langle \alpha_i \mid i \in N \rangle$ such that $\forall i . \alpha_i < \lambda$ and $\bigcup_{i=0}^{\infty} \alpha_i = \lambda$.

Note that we need only look to the function $H_{\omega+1}$ to separate the two processes described earlier (which were identified by H_ω but not by any morphism). We can however deduce from 8.20(d) and the fact that none of the T_α has a largest cardinality (among the T_α), that none of the functions H_α can be a morphism on every P,Q-space C (it is not one on $T_{\alpha+1}$).

Note also that even for the countable ordinals which we have used so far, the spaces T_α get very large indeed by normal standards.

The author's proof of 8.20(c) (which is vital to his proof of (a), (b) and (d)) breaks down at limit ordinals with cofinality greater than ω. At the time of writing he does not know whether it holds or not. Because of this it is not possible even to define the pairs (T_α, H_α) for any ordinal greater than or equal to $\omega_1+\omega$ (ω_1 the first uncountable ordinal) using our existing definition. It is however possible to adjust our definition (in such a way that if 8.20 does in fact hold for spaces defined in the old way for arbitrary α the two correspond) and obtain workable spaces for all ordinals.

We adopt the same definition as before except that when λ is a limit ordinal with cofinality greater than ω we take:

$$T'_\lambda = \{\underline{\sigma} \mid \underline{\sigma} \text{ is a } \lambda\text{-sequence of elements of } \bigcup_{\alpha\in\lambda} T_\alpha \text{ such that } \alpha<\lambda \Rightarrow (\sigma_\alpha \in T_\alpha \ \& \ ((\beta<\alpha) \Rightarrow H_\beta(\sigma_\alpha) = \sigma_\beta))\}$$

$H_\lambda(\sigma) = \underline{\rho}$, where $\rho_\alpha = H_\alpha(\sigma)$ for $\alpha<\lambda$.

In T'_λ $\underline{\sigma}$ has transitions $\underline{\sigma} \xrightarrow{\delta} \underline{\rho}$ if $(\delta, \rho_\gamma) \in \sigma_{\gamma+1}$ for all $\gamma<\lambda$.

Now let $T_\lambda = \{H_\lambda(\underline{\sigma}) \mid \underline{\sigma} \in T'_\lambda\}$, and redefine the transitions to be those of T'_λ with both states in the restricted space T_λ. (i.e. $\underline{\sigma} \xrightarrow{\delta} \underline{\rho}$ in T_λ if $\underline{\sigma}, \underline{\rho} \in T$ and $\underline{\sigma} \xrightarrow{\delta} \underline{\rho}$ in T'_λ.)

8.21 Theorem

With the definition given above each of the clauses of 8.20 holds for spaces T_α and functions H_α defined for arbitrary ordinals α. In addition H_α is a well-defined function from every P,Q-space to T_α for every α (this is not quite obvious in the case of non-ω-cofinal limit ordinals).

This result has a long and technical proof which is just an expansion of the one for 8.20.

Because clause (d) of 8.20 carries over to the general case we must give up all hope of finding a completely universal P,Q-space, as we can now find spaces with arbitrarily large cardinality whose only morphisms are injective. One can however prove that given any P,Q-space C then for sufficiently large ordinals α the maps $H_\alpha : C \to T_\alpha$ are morphisms. Given a P,Q-space C we can define a cardinal I(C) which is the index of non-determinism of C by

I(C) = smallest infinite regular cardinal strictly greater than that of $\{\rho \in C \mid \sigma \xrightarrow{\delta} \rho\}$ for every $\sigma \in C$ and $\delta \in \Sigma \cup \{\cdot\}$.

Thus $I(C) = \aleph_0$ if and only if there is no $\sigma \in C$ and $\delta \in \Sigma \cup \{\cdot\}$ s.t. $\{\rho \in C \mid \sigma \xrightarrow{\delta} \rho\}$ is infinite.

8.22 Theorem

If C is a P,Q-space and α is an ordinal such that $|\alpha| \geq I(C)$ then $H_\alpha : C \to T_\alpha$ is a morphism.

The proof of this result is not very difficult.

Corollaries to 8.22 are the facts that if C is finite-branching in the sense $I(C) = \aleph_0$ then the map $H_\omega : C \to T_\omega$ is a morphism, and if $I(C) = \aleph_1$ then $H_{\omega_1} : C \to T_{\omega_1}$ is a morphism (ω_1 the smallest uncountable ordinal).

8.22 tells us that for every alphabet Σ and cardinal $\aleph$ we can construct a space U into which there is a morphism from every space with this alphabet whose index of non-determinism does not exceed $\aleph$. The fact that these morphisms into U are unique is implied by the fact that all morphisms of U are injective, which means that U is isomorphic to U*, so we can apply 8.15. Thus U can be regarded as a universal space (in the sense described earlier) for a restricted class of spaces.

In most cases this universal space is much bigger than is necessary. As an example of this consider the result of constructing the universal space for finite branching processes over a countably infinite alphabet Σ. The successive approximations T_i to this space have cardinal $\beth_i$, where $\beth_0 = 1$; $\beth_1 = 2^{\aleph_0}$, and $n \geqslant 1 \Rightarrow \beth_{n+1} = 2^{\beth_n}$. The cardinal of the space T_ω is at least as large as the least upper bound of these cardinals. It is possible to cut this down very considerably by considering only those elements of T_n which can be the image under H_n of a finitely branching process. Let us define subspaces T^*_n and T^*_ω of T_n and T_ω respectively by the following:

$$T^*_0 = \{\emptyset\}$$

$$T^*_{n+1} = \{X \subseteq (\Sigma \cup \{\cdot\} \times T^*_n) \mid \forall \delta . \{\sigma \mid (\delta, \sigma) \in X\} \text{ is finite}\}$$

$$T^*_\omega = \{\underline{\sigma} \in T \mid \forall n . \sigma_n \in T^*_n\} .$$

It is clear that T^*_ω is indeed a subspace of T_ω, and that the image under H_ω of any C such that $I(C) = \aleph_0$ is contained in T^*_ω. We can therefore regard T^*_ω as a universal space for finitely branching processes. The cardinals of the spaces T^*_n are successively $1, 2^{\aleph_0}, 2^{\aleph_0}, 2^{\aleph_0}, \ldots$, and the cardinal of T^*_ω is $2^{\aleph_0}$.

In fact T^*_ω, though it has minimal possible cardinal with respect to being a universal space for finitely branching processes, still contains elements which cannot be the image under H_ω of any element of a finite branching space. This can be seen by consideration of the processes quoted earlier as examples of processes identified by H_ω but not by any morphism. It is easy to see that their joint image under H_ω is an element of T^*_ω, but that this element of T^*_ω is one which cannot be the image of an element of a finite branching space (there are infinitely many elements of T^*_ω to which it can be transformed by a single internal action). If we let U be the union of all the images in T^*_ω of finite branching spaces it is not hard to see that U is a (proper) subspace of T^*_ω. Furthermore U is itself a finite branching space, and so has a unique morphism into T^*_ω (which must be the identity morphism). Thus U is a universal space for all finite branching spaces, and there can be no smaller one. It is possible to construct U explicitly as follows:

$$\text{Let } U_o = T^*_\omega$$
$$U_{n+1} = \{\underline{\sigma} \in T^*_\omega \mid (\forall \delta . \exists n \in N . \forall m . |\{\rho \mid (\delta,\rho) \in \sigma_m\}| < n)\}$$
$$\cap \{\underline{\sigma} \in T^*_\omega \mid \forall \delta . \forall \underline{\rho} . \ \underline{\sigma} \xrightarrow{\delta} \underline{\rho} \Rightarrow \underline{\rho} \in U_n\}$$
$$U = \bigcap_{n=o}^{\infty} U_n$$

There is no reason why "T_ω" should not be used in place of "T^*_ω" in either of the above constructions of U, since we would have got the same answer. The only advantage gained from using T^*_ω is that it gives us a much better bound on the cardinal of the space U. It is also possible to use either of the above tricks to obtain a universal space U for P,Q-spaces with any bound on their index of non-determinism, and any size of alphabet Σ, such that U falls into the class of spaces which it models.

So far we have only proved the non-existence of completely universal spaces as a corollary to the difficult construction we devized for partially universal spaces. It does in fact have quite an easy proof by contradiction.

8.23 <u>Theorem</u>
For no alphabet Σ can there be a P,Q-space U to which there is a morphism from every P,Q-space with this alphabet.

<u>proof</u>
Suppose to the contrary that such a U does exist for some alphabet Σ. As we have seen U* would have a unique morphism to it from every P,Q-space with the given alphabet, and also every morphism of U* is injective. Consider the P,Q-space $V = (\{a\} \times U^*) \cup (\{b\} \times \wp(U^*))$ $(a \neq b)$, which has transitions

$(a,\sigma) \xrightarrow{\delta} (a,\rho)$ iff $\sigma \xrightarrow{\delta} \rho$ in U*;

$(b,X) \xrightarrow{\cdot} (a,\sigma)$ iff $\sigma \in X$;

and no others.

Since V is a P,Q-space with alphabet Σ there must be some (unique) morphism $F:V \to U^*$. The cardinal of $\mathcal{P}(U^*)$ is strictly greater than that of U^*, so there must be two distinct subsets X and Y of U^* such that $F(b,X) = F(b,Y)$. We may without loss of generality assume that there is some $\sigma \in X - Y$. It is easy to prove from the definition of morphisms that there must be some $\rho \in Y$ such that $F(a,\rho) = F(a,\sigma)$. Thus F is not injective on the subspace $\{a\} \times U^*$ of V. However this contradicts the fact that this subspace is isomorphic to U^*, since all morphisms of U^* are injective.

We may thus conclude that, as claimed, no such U can exist.

We have already demonstrated that we can, amongst all the P,Q-spaces with any given bound on their index of non-determinism, find one which is universal. Sometimes we will wish to make additional assumptions about the type of P,Q-space we are using, and when we do this it will be useful to be able to construct a universal space for spaces with this property (where possible). The following result shows one way in which this can be done.

8.24 Theorem

Suppose that "X" is a property of P,Q-spaces which satisfies the following laws:

(i) If C is a space with property X and $F:C \to D$ is a morphism, then Im(F) has property X.

(ii) If C is a space with subspaces $\{C_\alpha | \alpha \in A\}$ such that $C = \bigcup_{\alpha \in A} C_\alpha$ and each C_α has property X, then C also has property X.

then we may conclude that for each cardinal $\aleph$ there exists a P,Q-space U^X such that U^X has property X, $I(U^X) \leqslant \aleph$, and such that whenever C is a space with property X such that $I(C) \leqslant \aleph$ there exists a unique morphism $F:C \to U^X$.

proof

Let U be a universal space for spaces such that $I(C) \leqslant \aleph$ (for example T_α, where α is the initial ordinal with cardinal $\aleph^+$). Define U^X to be the union of all the images in U of spaces C s.t. $I(C) \leqslant \aleph$ and C has property X. It is easily seen that U^X, when so defined, satisfies all that is required of it.

In the last few pages we have developed a calculus which allows us to relate P,Q-spaces by means of maps called "morphisms", which are in some sense behaviour preserving maps between them. We have succeeded in producing universal spaces which can be held to model large classes of spaces in unique ways. The spaces we have been studying are in essence simple relational structures, which in various guises are used throughout mathematics. There is therefore much similarity between the above work and that of other authors which the author is aware of, and probably more with other work with which he is not familiar. For example the concept of "operational equivalence", as introduced in Hennessy and Milner ( ) and other works, is extremely similar to the equivalence induced by the operator H_ω. The chief difference is in the treatment of internal actions. Different types of "morphisms", similarly defined, can be used to analyze other, rather more complicated, types of process-spaces. For example this can be done (with almost exactly the same effect) for spaces where the relations represent finite sequences of visible actions, when given a suitable axiomatization.

Having built up enough machinery for our own purpose we are in a position to return to the main theme of this chapter, and to find out how the spaces we have constructed relate to the models we used in earlier chapters. We are essentially seeking functions "θ" such that, given processes "c" in a P,Q-space "C", the "c"s are mapped in a useful and realistic way to one of our models by the map $\Phi(c) = \{\theta(\underline{a}) \mid \underline{a} \in B(c)\}$. Our intuition about the models is that they represent some aspect of the observable behaviour of processes. Given a behaviour, which is an abstraction of one possible sequence of interactions between process and state, we must ask just which parts of it are observable by an experimenter who manipulates the environment. The two things which are certain to be observable are the environment component "X" of the triples making up behaviours, and any external actions which occur. We can also assume that the experimenter is aware of the postulates (general properties of processes) which processes satisfy, so that for example if he can be sure that no action has occurred for sufficiently long (while one set, X, was offered) he can deduce that no action will ever occur (if he persists in offering X). We will assume that the internal state "σ" of a process is invisible to the experimenter. The only question to be decided is whether or not the experimenter can observe the presence or absence of internal actions. One might imagine that there is a light on the side of a machine which lights up when there is internal activity. We cannot expect the experimenter to be able to discern anything more about internal activity than its presence, in the loosest sense (if he can detect it at all): for example it does not seem reasonable to expect him to be able to count the number of internal actions which occur. We will find that there are two distinct maps from P,Q-spaces to the non-deterministic model, the choice between them being largely dependent on whether or not we believe internal activity to be observable.

The principle that the internal state of a process is invisible is important in justifying the use of morphisms and universal spaces. This is because of 8.10 which tells us that apart from the state components of behaviour, the behaviours of a process and its image under a morphism are identical. By this principle it seems reasonable to expect that the function "θ" which maps behaviours to their representations will be independent of the "state" components of behaviours. We will therefore expect it to be induced in the natural way by some function of $(\wp(\Sigma)\times(\Sigma\cup\{\cdot,-,*\}))^{\circledast}$ (= H, say). In future when "θ" is a function of H we will regard "$\hat{\theta}$" as being its natural extension to behaviours. This is formed by defining a projection function h to be the natural extension to sequences of the function $h(\sigma,X,\delta) = (X,\delta)$. "$\hat{\theta}$" then becomes $\theta\circ h$. If C and D are two P,Q-spaces then whenever F is a morphism from C to D and θ is a function of H we have $\{\theta(\underline{a}) \mid \underline{a}\in B(c)\} = \{\theta(\underline{a}) \mid \underline{a}\in B(F(c))\}$ for all $c\in C$.

An extension of this principle is to decree that not only will our modelling functions be independent of states, but also the correctness conditions "χ" of behaviours which we wish to prove will be independent of states. This can be interpreted as saying that we shall judge our processes only by what they do (or fail to do) either internally or externally, and not by how they are actually constructed. We will thus generally expect our predicates of behaviour to be induced in the natural way by predicates of H. If "χ" is a predicate of H we will write "$\hat{\chi}$" for the predicate which it induces on whatever space of behaviours we are currently using. ($\hat{\chi}(\underline{a}) \equiv \chi(h(\underline{a}))$). It is easy to see that if C and D are P,Q-spaces and χ is a predicate of H then $\underset{\smile}{\hat{\chi}}(c) \Leftrightarrow \underset{\smile}{\hat{\chi}}(F(c))$ for all $c \in C$, whenever $F:C \rightarrow D$ is a morphism.

We may thus conclude that, so long as our functions and predicates are "forgetful" of internal states, both a process' image in a model and the truth of predicates about them are invariant through morphisms. Thus in these circumstances, to prove a predicate of a process "c" from any space C it is sufficient to prove the corresponding predicate of the process' image in any suitable universal space.

When we are using functions and predicates of this "forgetful" type, the set of the projections into H of the behaviours of a process is very important. If we define $B^*(c)$, the "reduced behaviour set", of $c \in C$ to be $\{h(\underline{a}) \mid \underline{a} \in B(c)\}$, it is easy to see that for any function θ of H we have $\{\hat{\theta}(\underline{a}) \mid \underline{a} \in B(c)\} = \{\theta(\underline{a}) \mid \underline{a} \in B^*(c)\}$. Similarly when χ is a predicate of H we get $\underset{\smile}{\hat{\chi}}(c) \Leftrightarrow \forall \underline{a} \in B^*(c) . \chi(\underline{a})$. Note that $B^*(c) \subseteq B^*(d)$ & $\underset{\smile}{\hat{\chi}}(d) \Rightarrow \underset{\smile}{\hat{\chi}}(c)$. Both the image of a process in a model and the truth of predicates "$\underset{\smile}{\hat{\chi}}$" about it are determined completely by its reduced behaviour set.

Before we get involved in maps to the non-deterministic model it is perhaps wise to see how we might use the machinery we have set up to construct and analyse maps from P,Q-spaces to the deterministic model P. There is really only one map worth considering, namely that induced by the following function "θ" of H:

$$\theta(\langle\rangle) = \langle\rangle; \quad \text{if } \underline{a} \text{ is finite then}$$
$$\theta(\langle(X,\delta)\rangle\underline{a}) = \theta(\underline{a}) \text{ if } \delta \in \{\cdot, -, *\} \text{ and } = \langle\delta\rangle\theta(\underline{a}) \text{ if } \delta \in \Sigma;$$
$$\theta(\underline{a}) = \langle\rangle \quad \text{if } \underline{a} \text{ is infinite.}$$

It is easy to show from our postulates that the "Φ" induced by $\hat{\theta}$ satisfies the following, when regarded as a function from a P,Q-space C to $\wp(\Sigma^*)$.

(i) $\Phi(c)$ is non-empty for all $c \in C$

(ii) $\Phi(c)$ is prefix closed for all $c \in C$.

To establish the finality of "$\checkmark$" we would have to make some additional postulate of our spaces such as

(D1) $(\sigma, X) \xrightarrow{\checkmark} \rho \Rightarrow \neg((\rho, Y) \xrightarrow{b} \tau)$.

It is easy to see that D1, when regarded as a property of P,Q-spaces, satisfies the hypothesis of 8.24. There is thus no difficulty in constructing universal spaces for P,Q-spaces which additionally satisfy D1. Let us call a P,Q-space which satisfies D1 a D1-space. It is easy to show that when C is a D1-space we have

(iii) $w\langle\checkmark\rangle v \in \Phi(c) \Rightarrow v = \langle\rangle$ for all $c \in C$.

Thus Φ is a well-defined map from every D1-space to the deterministic model P.

P can itself be regarded as a D1-space. Transitions are defined: $A \xrightarrow{a} B$ iff $B = A$ after$\langle a \rangle$; no internal transitions. It is a simple matter to prove $\Phi(A) = A$ for all $A \in P$. One easy consequence of this fact is that Φ is a surjective function from the universal finite branching D1-space to P. For the time being let us adopt this universal space as the "C" which we are trying to model by P.

The first thing which we must investigate, when studying the relationship between a "real" system and a model, is the way in which predicates which we wish to prove of the real system transfer to the model. Intuitively one might suspect that the map Φ is adequate for expressing many partial correctness conditions (those which demand that anything which a process actually does is correct), but is poor when it comes to total correctness conditions (which demand that a process must actually be willing to do things).

Because of the universal nature of the space C it is not hard to show that given any $c \in C$ there exists some (unique) element d of C whose only transitions are $d \xrightarrow{\cdot} c$ and $d \xrightarrow{\cdot} e$, where e is a process with no transitions. It is easy to see that $\Phi(c) = \Phi(d)$ (because they have the same possible sequences of external actions). However it is also easy to see that $B^*(d) \supseteq B^*(f)$, where f is the element of C with a unique transition $f \overset{\cdot}{\rightarrow} e$. Suppose now that $\mathcal{X}$ is any predicate of H such that $(\hat{\mathcal{X}})'$ (the weakest predicate Π of P s.t. $\forall c \in C. \Pi(\Phi(c)) \Rightarrow \hat{\mathcal{X}}(c)$) is satisfiable. Suppose A is chosen so that $(\mathcal{X})'(A)$ holds, and that $c \in C$ is such that $\Phi(c) = A$ (such a "c" exists since Φ is surjective). Now construct "d" as above. Since $\Phi(c) = \Phi(d)$ holds we have $(\hat{\mathcal{X}})'(\Phi(d))$; this implies $\hat{\mathcal{X}}(d)$, which in turn implies $\hat{\mathcal{X}}(f)$ (because $B^*(d) \supseteq B^*(f)$).

We are thus forced to conclude that whenever "$\hat{\mathcal{X}}$" is a predicate of C sufficiently weak to allow it to be deduced of any process from the process' image in P, it must itself be satisfied by the process "f" which can only perform one (internal) action before deadlocking. This confirms the suspicion which we developed in chapter one, that the model P is not adequate for telling us anything reliable about potentially non-deterministic processes. Some indications were given in chapter one about the type of process which we felt was adequately described by P. Without going into any more detail on the modelling of the above system C let us now try to restrict our object space to processes which we can model accurately over P.

We wish to axiomatize "deterministic" behaviour. We might expect the chief sources of non-determinism to be firstly internal actions (which can "resolve" non-determinism) and secondly cases where one state has more than one external action with a particular name. The postulate which expresses the proscription of these types of behaviour is the following:

$$\text{(D2)} \quad \neg((\sigma,X) \overset{\cdot}{\rightarrow} \rho) \quad \& \quad (((\sigma,X) \overset{a}{\rightarrow} \rho) \;\&\; ((\sigma,X) \overset{a}{\rightarrow} \tau) \quad \Rightarrow \quad \rho = \tau)$$

This is another property which satisfies the hypotheses of 8.24; thus so is the joint condition D1 & D2. We may thus deduce the existence of a universal D-space, where a D-space is defined to be a P,Q-space which satisfies D1 and D2. (The existence of a completely universal space follows from the fact that any space which satisfies D2 is automatically finite-branching). It is possible to weaken condition D2 slightly to allow a little internal behaviour. We can reasonably allow a deterministic process to have internal actions if they are all single (for each σ there is at most one ρ such that $\sigma \stackrel{\cdot}{\rightarrow} \rho$) and inevitable (if $\sigma \stackrel{\cdot}{\rightarrow} \rho$ is possible then $\sigma \stackrel{a}{\rightarrow} \tau$ is not). A modified D2 which expresses this is

$$(D2') \quad (((\sigma,X) \stackrel{\delta}{\rightarrow} \rho) \ \& \ ((\sigma,X) \stackrel{\delta}{\rightarrow} \tau) \ \Rightarrow \ \rho = \tau)$$
$$\& \ \neg(((\sigma,X) \stackrel{\cdot}{\rightarrow} \rho) \ \& \ ((\sigma,X) \stackrel{a}{\rightarrow} \tau)).$$

D2' is also a condition which satisfies the hypotheses of 8.24, so in a similar manner to the above we may deduce the existence of a universal D'-space (a P,Q-space which satisfies D1 and D2'). Note that every D-space is a D'-space.

Clearly P, when made into a D1-space as before, is a D-space. We can deduce from this that Φ is a surjective function to to P from each of the universal D-space and the universal D'-space.

It is left as an exercise for the interested reader to verify that in either of the above types of space the set of predicates we can reasonably expect to prove by reference to the model is much larger and more useful than in the earlier case. It is worthwhile to make two remarks however. Firstly it is not hard to show that the universal D-space is isomorphic to P (when P is regarded as a D-space in the usual way), and that Φ is a morphism from any D-space to P. Thus (recalling 8.1) it is not surprizing that predicates should transfer well between the two systems. Secondly, in the D'-space case, it is interesting to note that the function Φ identifies the processes "e", which has no actions, and "d", which has a single action $d \stackrel{\cdot}{\rightarrow} d$. ($\Phi$ also identifies any pair of processes whose structure is the same except for the substitution of "e" for "d" at some points, or vice-versa.) It thus identifies "divergence" or "infinite internal chatter" with "computed termination". Thus the absence of divergence is not expressible as a predicate of P which does not also imply freedom from deadlock. This issue (the difference between divergence and computed deadlock) will be more important later, when we come to consider the non-deterministic model.

Another interesting point which arises from the study of the relationship between these "real" systems and the model

P, is that P is very much a class 2 model for these systems. The most obvious way of seeing this is to note that if P were thought of as a class 1 model, this would mean that we only allowed ourselves monotonic predicates. We would then find ourselves in very much the same boat which we were in with the D1-space, for every satisfiable predicate which we allowed ourselves would be satisfied by abort. The processes "e" and "d" (as defined in the last paragraph) would (as constant functions) be correct implementations of every operator.

The basic reasons for this arise from the structure of the function "Φ" in the following way. The function "Φ" is based on a function "$\hat{\theta}$" of behaviours which is essentially one-sided in that it only reflects one of the two aspects of behaviour which are essential to most total correctness predicates. It is based purely on the "positive" aspects of behaviour (triples of the form (σ,X,a) for $a\in\Sigma$), and not the equally important "negative" aspects (triples of the form $(\sigma,X,*)$ and infinite sequences of internal actions). The restrictive conditions which we have placed on D- and D'-spaces enable us to discover enough about the negative aspects of behaviour which are possible by studying the positive aspects. To do this we have to exploit the relationship which states that (in D- and D'-spaces) the more positive behaviours there are possible, the more "negative" behaviours can be deduced impossible. Thus, if we wish to check that an undesirable "negative behaviour" is impossible in a process by studying its image in P, it will often be necessary to check that some "positive" behaviour is possible. For example to ensure that a process c cannot deadlock on its first step it is necessary to check that $\Phi(c)^{O}\neq\emptyset$. It is because of this "upside-down exclusion" of negative behaviours that we cannot expect monotonic predicates to be sufficiently expressive, since by removing elements from an element of P we are adding possibly incorrect negative behaviours to its pre-image.

Having decided that our model is to be regarded as class 2 it is necessary that our implementations of the various operators and constructs of our language be exact. So far we have been too involved with the construction and interpretation of our "real" systems to consider the problem of how we might seek to implement our language in them. What we would like is an operational semantics for our language. We have not got space here to go into this subject in very much detail; we will therefore quickly survey the various options open to us, draw a few general conclusions, and pass on to the study of the non-deterministic model.

There are several ways in which one might seek to give an operational semantics to our language; some of these are more abstract than others. One approach would be to define exactly what was meant by the "state" of a process: how it stores and recalls the values of its various variables, and how it decides which actions to take (with what influence on itself). If one did this it would be necessary to check that the space of states which resulted satisfied whatever postulates were required of them. Without going into technical detail it is only possible to make a few general remarks about this approach.

(i) We cannot expect an operator to be able to see any more about its operands than an environment could. It is not reasonable to expect an operator to be able to predict what its operands will do after actions which have not yet been completed (nor to anticipate divergence). One useful way to think of an operator is as a "black box" which is placed around its operands and which has certain powers over them, for example:

(a) An operator can "switch on" or "switch off" its operands. On switching an operand on for the first time the operator must initialize all its variables. Only "on" operands may perform any action. The act of switching will itself be an internal action of the total state.

(b) Any internal action performed by an operand is an internal action of the total state, and uncontrollable by the operand.

(c) The operator can act as environment to its operands and communicate with them without telling its own environment. Any such communications are internal actions of the total state.

(d) An operator can communicate with its environment without reference to its operands.

(e) An operator can regulate communication between the environment and its operand(s). For example it might transform the alphabet in some way, synchronize several of its operands, or become completely transparent.

(If we tried to implement our existing operators as "black boxes", we might expect the above features to appear in the following in important ways:

(a) $[]$, a →, x:T →, recursion

(c) /X, ;

(d) a →, x:T →

(e) a., ‖ .)

The above approach, while it might be considered to leave something to be desired, is a valuable aid to the intuition when considering the "reasonableness" of operators defined over universal spaces in abstract ways.

(ii) We would expect the values stored in process variables to be processes "switched off" (unevaluated code?) with the property that, when activated, they ignore any values assigned to their variables (except recursion parameters "λ") and adopt values stored with them in some way. There should be no problem in showing recursion to be well-defined, for all operators are in some sense "non-destructive" of actions because of principle (b) above, etc., and the act of making a recursive call is constructive because it involves "switching-on", which is an internal action.

In general each of the principles seems largely consistent with the idea that an operator (k-place) is a function from C'^k to C' (where $C' = C^{\Theta} \times S \to C$), as in the first part of this chapter.

Alternatively we might choose to define a semantics in a more abstract way. This could be done by direct reference to the structure of universal spaces (8.16 - 8.24). This should be done bearing the above principles carefully in mind with a view to later implementation by the more practical methods above. This is the most obvious approach to adopt when we are using P,Q-spaces as idealized models of other spaces. Because of the nature of universal spaces it is possible to define elements uniquely by their transitions. Examples of the ways one might wish to do this are the state "$a \to \sigma$", which is defined to have a single transition, namely "a" after which it becomes σ, and "$\sigma;\rho$" which has its transitions recursively defined $\sigma \xrightarrow{\delta} \tau \Rightarrow (\sigma;\rho) \xrightarrow{\delta} (\tau;\rho) \quad (\delta \neq \surd)$

$$\sigma \xrightarrow{\surd} \upsilon \Rightarrow (\sigma;\rho) \xrightarrow{\cdot} \rho \quad .$$

The first type of operator is easy to define: if (as is common) the universal space in use is the subspace U of some T_λ (λ a limit ordinal) defined through 8.24 by some property X, then we can define "$a \to \sigma$" as follows. We take it to be the element ρ of T_λ such that $\rho_{\gamma+1} = \{(a,\sigma_\gamma)\}$ for all $\gamma \in \lambda$ (the other components can be deduced from these). If U is a proper subspace of T_λ then we are obliged to show that the ρ so defined is an element of U. This can be done in this case by checking that the space U', which is U with an extra element "ρ'" adjoined with the single transition $\rho' \xrightarrow{a} \sigma$, has the defining property X. If this is the case then there is a morphism from U' to U, and the image of ρ' is ρ.

The second type of operator requires a little more work. To define "$\sigma;\rho$" explicitly we have to appeal to transfinite recursion. Let us once again suppose that U is a subspace of T_λ. We define $(\sigma;\rho)_0 = \emptyset$

$$(\sigma;\rho)_{\gamma+1} = \{(\delta,(\tau;\rho)_\gamma) \mid \sigma \xrightarrow{\delta} \tau \ \& \ \delta \neq \surd\}$$
$$\cup \{(\cdot,\rho_\gamma) \mid \sigma \xrightarrow{\surd} \upsilon\}$$
$$((\sigma;\rho)_{\lambda'})_\gamma = (\sigma;\rho)_\gamma \qquad (\gamma \in \lambda') \quad .$$

Once again to show that ";" is a well-defined operator on U (if it is) we take a space U' which includes a copy of U and a disjoint copy of U×U. The transitions of the elements of the copy of U are those inherited from U. A pair (σ,ρ) has transitions $(\sigma,\rho) \xrightarrow{\delta} (\tau,\rho)$ if $\delta \neq \surd$ and $\sigma \xrightarrow{\delta} \tau$ in U

$$(\sigma,\rho) \xrightarrow{\cdot} \rho \quad \text{if } \sigma \xrightarrow{\surd} \upsilon \text{ in U.}$$

One must show that U' has the defining properties of U, so that there is a morphism from U' to U. If this is so the pair (ρ,τ) can be shown to map under this morphism to $\rho;\tau$, the element of T_λ defined above.

Note that provided that the space U is closed under each of the above operators it is easy to prove the relations

$$a \to (\rho;\tau) = (a \to \rho;\tau) \quad \text{and} \quad ((\sigma;\rho);\tau) = (\sigma;(\rho;\tau)).$$

Other operators can be defined in very much the same way. This should always be done in a way which does not violate the principles derived from our "black box" discussion. The first of these is that at every stage the transitions of the result of applying an operator should depend only on the available transitions of the operands (in the states in which they currently find themselves), and past history; if a transition is executed which depends on the existence of some transition in one of the operands then the operand must execute that transition. The second principle is that every transition executed by an operand must be represented by a corresponding transition in the "finished product". These principles are guaranteed by stipulating that every operator must have some defining equation of the form below. A zero-place operator is a constant.

A one-place operator "op" must have exactly the transitions

$$\begin{array}{lll} op(\sigma) \xrightarrow{\delta} op_\alpha(\sigma) & (\delta,\alpha) \in D & \\ op(\sigma) \xrightarrow{\delta} op'_\alpha(\rho) & \sigma \xrightarrow{\delta'} \rho \quad \& \quad (\delta,\delta',\alpha) \in D' \end{array}$$

(The first line corresponds to the operator carrying out some action without reference to its operand, the second line corresponds to the operator transforming some action of its operand. In each case the operator may transform itself (non-deterministically) into another, dependent on the action which occurs.) One might wish to strengthen the above to ensure that internal actions of operands can neither become external actions of the finished product nor influence the composition of the operator. This can easily be done by editing the second line above.

A two-place operator "op" must have exactly the transitions

$$\begin{array}{ll} op(\sigma,\rho) \xrightarrow{\delta} op_\alpha(\sigma,\rho) & (\delta,\alpha) \in D \\ op(\sigma,\rho) \xrightarrow{\delta} op'_\alpha(\tau,\rho) & \sigma \xrightarrow{\delta'} \tau \quad \& \quad (\delta,\delta',\alpha) \in D' \\ op(\sigma,\rho) \xrightarrow{\delta} op''_\alpha(\sigma,\tau) & \rho \xrightarrow{\delta'} \tau \quad \& \quad (\delta,\delta',\alpha) \in D'' \\ op(\sigma,\rho) \xrightarrow{\delta} op^*_\alpha(\tau,\upsilon) & \sigma \xrightarrow{\delta'} \tau \quad \& \quad \rho \xrightarrow{\delta''} \upsilon \quad \& \quad (\delta,\delta',\delta'',\alpha) \in D^*. \end{array}$$

(The four lines here correspond to the operator carrying out some action without reference to its operands, transforming some action of its first operand, transforming some action of its second operand, and transforming and co-ordinating a pair of actions respectively.) Once again one might choose to tighten up on the last three lines to ensure that internal actions of operands cannot have undue influence.

Three and higher place operators can be constructed by combining two-place operators.

(Note of clarification: in the above definitions it is the relations D, D' etc. which define the operators, together with the operators op_α, etc. which are assumed to be defined in the same way. "op_α" can vary with α, which is assumed to range over some indexing set.)

As an example a hiding operator might be defined by the scheme

$(\sigma/X) \dot{\to} (\rho/X)$ if $\sigma \dot{\to} \rho$ or $\sigma \xrightarrow{a} \rho$ for some $a \in X$;

$(\sigma/X) \xrightarrow{a} (\rho/X)$ if $\sigma \xrightarrow{a} \rho$ and $a \notin X$.

This and both our earlier schemes can easily translate to the form described formally above.

Note that we cannot expect the above hiding operator to be well-defined on any universal space U which does not reliably model all branching smaller than the smallest infinite cardinal greater than $\|X\|$.

The theory of operators defined in this way is quite interesting, but we do not have space to go into it in any depth. We merely quote the next result, which helps to formally justify our assertion that "properly defined" operators are in some sense non-destructive.

8.25 Lemma

Suppose that op is a k-place operator on a space U which is a subspace of some T_λ, and that op and all the other operators on which it depends in its definition are defined by the methods set out above, then if $\underline{\sigma}_i$ & $\underline{\sigma}'_i$ are two sets of elements of U such that $\forall i \in \{1,..,k\}.(\underline{\sigma}_i)_\gamma = (\underline{\sigma}'_i)_\gamma$ $(\gamma \in \lambda)$ we have $op(\underline{\sigma}_1,...,\underline{\sigma}_k)_\gamma = op(\underline{\sigma}'_1,...,\underline{\sigma}'_k)_\gamma$.

We need to be able to extend operators defined, as above, on a space U, to the space $U' = U^\Theta \times S \to U$. This can be done in very much the same way as before. Suppose that elements of U' are denoted e*, f*,.. and that elements of $U^\Theta \times S$ (states) are denoted π, o ,... . If we have a k-place operator op over

U then op can be re-defined as an operator $\underline{op}$ over U' by

$$\underline{op}(e_1^*,\ldots,e_k^*)(\pi) = op(e_1^*(\pi),\ldots,e_k^*(\pi)).$$

We also need operators defined over U' which are not simply extensions of operators over U. There are basically two categories of these: recursions and "others". Non-recursive operators should be defined by transition schemes similar to those used above. These should observe the general principle that nothing is observable of the contents of the U^Θ component of a "state" π without switching on any processes we wish to observe and treating them as "normal" operands. For most practical purposes one can get away with using zero and one-place operators of this type. We will therefore stipulate that any non-recursion operator not of the above form must be of one of the two forms set out below.

We will write an element π of $U^\Theta \times S$ as (π_1,π_2), π_1 being the U^Θ-component and π_2 being the S-component.

A zero-place operator $\underline{e}^*$ (constant element of U') must be defined:

$\underline{e}^*(\pi_1,\pi_2) = \rho$, where ρ has exactly the transitions

$$\rho \xrightarrow{\delta} \underline{f}_\alpha^*(\pi_1,\pi_2') \qquad (\delta,\pi_2,\pi_2',\alpha) \in D$$

$$\rho \dot{\rightarrow} \pi_1(\theta) \qquad (\theta,\pi_2) \in D'$$

($\underline{f}_\alpha^*$ is assumed to be another zero-place operator, similarly defined.)

A one-place operator $\underline{op}$ must be defined:

$\underline{op}(e^*)(\pi_1,\pi_2) = \rho$, where ρ has exactly the transitions

$$\rho \xrightarrow{\delta} \underline{op}_\alpha(e^*)(\pi_1,\pi_2') \qquad (\delta,\pi_2,\pi_2',\alpha) \in D$$

($\underline{op}_\alpha$ is assumed to be another one-place operator defined in the same way or as an extension of a U-operator.)

As examples of these we can define the one-place operator "x:T → " and the zero-place operator "B" (call of the process variable B).

$(x:T \rightarrow e^*)(\pi) = \rho$, where ρ has exactly the transitions

$$\rho \xrightarrow{a} e^*(\pi[a/x]) \qquad a \in T(\pi_2).$$

(We assume that the set T may be a function of one or more non-process variables. The "$\underline{op}_\alpha$" used is the extension to U' of the identity function on U.)

$B(\pi) = \rho$, where ρ has the single transition

$\rho \dot{\rightarrow} \pi(B)$.

Existence proofs for all operators defined by any of our "transition scheme" methods can be carried out by extending the methods used for ";" and "a → " earlier. This will involve the setting up of a P,Q-space of syntactic/state objects, possibly including a copy of the space U as a sub-space, then showing that the space one has set up satisfies enough for there to exist a (unique) morphism into U. If a definition of such an operator is required in the form given for "a →" and ";" (exact definition of the components of the result of an operator regarded as an element of T_λ) this can be done recursively.

If $U \subseteq T_\lambda$ then there is an obvious way in which one can define the projection π_γ of a "state" π into the space $(T_\gamma^\Theta) \times S$. The pay-off of all our careful definitions above is that, so long as e* is an element of U' defined only by operators of the types we allow, it can be proved that whenever π and π' are two "states" such that $\pi_\gamma = \pi'_\gamma$ we have $e^*(\pi)_{\gamma+1} = e^*(\pi')_{\gamma+1}$ for all $\gamma \in \lambda$. This is easily shown to imply that each recursive fixed-point equation has at most one solution. The existence of solutions to these equations can be proved without too much difficulty so long as the space U we are using satisfies some simple and natural closure conditions. As an example of a recursion one might use, to define "recB.e*" (e* defined using only permitted operators) we would say $(\text{recB.e}^*)(\pi) = \rho$ s.t. $\rho = e^*(\pi[\rho/B])$. The value of $\text{recB.e}^*(\pi)$ is defined as follows. We define a function f from $\lambda+1$ to U as follows:

f(o) is chosen from U at random

$f(\gamma+1) = e^*(\pi[f(\gamma)/B])$

$f(\lambda')$ is chosen (by closure conditions) to be such that $f(\lambda')_\gamma = f(\gamma)_\gamma$ for all $\gamma \in \lambda'$.

The value of $\text{recB.e}^*(\pi)$ is $f(\lambda)$.

Other, more complex, recursive operators can be defined similarly.

The use of nested recursions (i.e. recursions within recursions) can also be justified without too much difficulty.

Conclusions for the deterministic model

We do not have space here to launch into any attempts to formally implement our operators over P (nor will we when we later examine the non-deterministic model). What we can do is to get a good impression of what is, and what it not, possible.

It is clear that unless we relax substantially our conditions upon operators there is little prospect of our being able to implement all our operators over D-spaces (where internal actions are banned). Operators which appear to be impossible because of their dependence on internal actions are ";", "/X", calls of recursive variables and meaningful recursion. Difficulty arises in a less expected way with the operator "$\Box$", when applied to processes whose initials are not disjoint. The obvious implementation scheme

$$\sigma \Box \rho \xrightarrow{a} \tau \quad \text{if } \sigma \xrightarrow{a} \tau \quad (\& \ \neg\exists v.\ \rho \xrightarrow{a} v)$$
$$\sigma \Box \rho \xrightarrow{a} \tau \quad \text{if } \rho \xrightarrow{a} \tau \quad (\& \ \neg\exists v.\ \sigma \xrightarrow{a} v)$$
$$\sigma \Box \rho \xrightarrow{a} \tau \Box v \quad \text{if } \sigma \xrightarrow{a} \tau \ \& \ \rho \xrightarrow{a} v$$

suffers from the drawback that the bracketed terms are not permissible within the rules which we set out earlier (essentially they would require the environment to be able to detect more about its operands than we have thought correct hitherto). If these offending terms were withdrawn then the operator would not preserve the postulate D2, for it would introduce multiple branching.

One might hope that D'-spaces, with their weaker postulates, might give us an easier ride. This is in some senses true, since it is now possible to correctly implement each of ";", "/X", process variables and recursion so long as syntactic rules similar to those set out in chapter one are observed. Problems still arise with the "$\Box$" operator, however, and of a more serious nature than before. This results from the fact that the map "Φ" identifies deadlock with divergence. Consider for a moment the situation which will arise when we try to compose a simply diverging process with a process which can perform some action, "a" say. Since the operator is unable to detect that its operand is diverging (it cannot know that it will not decide to perform some visible action or halt at some future point) it must allow it to perform its successive internal actions. These will be reflected in internal actions of the resultant process, so it must itself be able to diverge. It is easily seen that this is impossible in any element of a D'-space which can execute the transition "a". We must conclude that some strong syntactic condition is required to ensure that "▯" is implementable. Such a condition is the requirement that "▯" only be used in the context "(a → *) ▯ (b → *)" where $a \neq b$. The (very desirable) use of more general guards on the two sides of "▯", such as "x" (alphabet variable) or "x:T" (input) would create problems because of the requirement that guards should be distinct.

It is possible to invent a third type of "deterministic" P,Q-space where many problems disappear. Unfortunately it is not quite so natural as the other two. One postulates that branching is finite, that divergence is absent, and that while multiple branching and internal actions may occur they may not influence the external actions.

(D2") $\forall\sigma.\forall\delta.\forall X.\ \{\rho \mid (\sigma,X) \xrightarrow{\delta} \rho\}$ is finite

$\&\ \forall\delta.\forall\sigma.\forall X.\forall\rho,\tau.\ (\sigma,X) \xrightarrow{\delta} \rho\ \&\ (\sigma,X) \xrightarrow{\delta} \tau \Rightarrow \Phi(\rho) = \Phi(\tau)$

$\&\ \forall\sigma.\forall\rho.\forall X.\ (\sigma,X) \dot{\rightarrow} \rho \Rightarrow \Phi(\sigma) = \Phi(\rho)$

$\&\ \neg\exists\sigma_0,\sigma_1,\ldots\ .\ \exists X.\forall i.\ (\sigma_i,X) \dot{\rightarrow} \sigma_{i+1}$

D2" is a postulate which satisfies the conditions of 8.24, so as before we can deduce the existence of a universal D"-space (P,Q-space which satisfies D1 and D2"). Over this space U it is possible to implement each of our operators correctly with the limitations described below:

(i) Non-constructive recursions are simply not defined when they give rise to divergence. This is the main weakness of this type of space.

(ii) Hiding is not defined where it would give rise to non-determinism.

(iii) The operators ";" and "▯", while they can be fully defined, are very inefficient unless the rules set out in chapter one are followed, because backtracking is required if this is not done.

All one does when one tries to implement the deterministic model is to confirm the prejudices we developed in chapter one. It is now time to examine the non-deterministic model.

<u>The non-deterministic model: first attempt</u>

Recall that we interpret the value N(c) in the non-deterministic model M of a process "c" as being the set of sequences of external actions possible for "c" paired with the sets of symbols which "c" can refuse after accepting them. What is basically at issue here is the notion of "refusal"; this is closely linked with the observability of internal actions.

Let us first examine the implications of an assumption that an experimenter cannot observe what is going on inside any process. What must his criterion for deducing refusal be? Such an experimenter cannot tell the difference between a process which is deadlocked and one which is engaged in internal communication (whether or not this internal activity will eventually cease). There is no period after which he can deduce that any non-empty set he is offering is refused (i.e. no element of it has been or will be accepted). This is because there may, at any finite time, still be activity going on internally which will later result in acceptance. If we let c_n be a process which can (and must) perform n internal actions before becoming able to perform the external action "a", then any finite deduction of refusal by our experimenter would be incorrect (if he were offering the set $\{a\}$ to one of the processes c_n) for sufficiently large n. Thus an experimenter can only deduce refusal when it is too late: when a set has been offered for an infinite time without response.

Bearing in mind that we wish to construct a map to the non-deterministic model based on the observed behaviour of processes, our next step must be to see how we can extract the observable parts of behaviour from the "full" behaviours of a process. From the point of view of an experimenter who cannot observe any internal behaviour any experiment will consist of finite applications of sets, either with or without observable response from the process, and possibly a (final) infinite application of a set without visible response. A very plausible procedure for the translation of our existing behaviours to "observations" is the following:

(i) Delete all the state components of the triples (i.e. project into H).

(ii) Replace all "·"s and non-final "*"s by "-".

(iii) Replace any final infinite sequence of the form $\langle(X_1,-)(X_2,-)\ldots\rangle$ by $(Z,*)$, where $Z = \bigcup_{i=1}^{\infty}(\bigcap_{j=i}^{\infty} X_j)$.

(iv) Delete any term of the form (X,-) which is followed by one of the form (X,-) or (X,a) (same X).

To illustrate this procedure let us apply it to the behaviour $\langle(\sigma_0,X_0,\cdot)(\sigma_1,X_0,*)(\sigma_1,X_1,a)(\sigma_2,X_2,\cdot)(\sigma_3,X_3,-)\ldots\rangle$ where all subsequent terms (infinitely many of them) have one of the forms $(\sigma_3,X_i,-)$ and $(\sigma_3,X_i,-)$.

The first and second steps translate this to $\langle(X_0,-)(X_0,-)(X_1,a)(X_2,-)(X_3,-)\ldots\rangle$, all subsequent terms having the form $(X_i,-)$. These steps relect the facts that the experimenter cannot see the structure of the state, and that he cannot detect internal actions or long intervals without them.

The third step translates this to $\langle(X_0,-)(X_0,-)(X_1,a)(Z,*)\rangle$ where Z is the liminf of all the X_is occurring in the final sequence. This step says that if the experimenter applies an infinite sequence of sets without response he can infer the refusal of **all** symbols applied continuously for an infinite period.

The fourth and final step reduces this to $\langle(X_0,-)(X_1,a)(Z,*)\rangle$. This step has the effect of collapsing contiguous applications of the same set into one application.

One point in the above procedure which seems a little suspect is the retention of final "*"s, since we interpret these (usually) as being infinite or sufficiently long finite waits without response. We cannot be sure that "*" represents an infinite wait (though we can be sure that it does not if it is not final). This complaint is rather academic however, since postulate Q8 ensures that final sets of "observations" of processes are the same whether or not we translate such "*"s as "-".

Let us define (for an element c of P,Q-space C) the set Obs(c) which results from the translation of each of the elements of

B(c). There are several results which one can prove of Obs(c) from our postulates. In the following we denote the sequences of pairs which constitute observations by $\underline{s}$, $\underline{t}$, ...

8.26 Lemma

If C is a P,Q-space and $c \in C$ then we have:

a) $\langle\rangle \in \mathrm{Obs}(c)$

b) $\underline{s}.\underline{t} \Rightarrow \underline{s}'\langle(\emptyset,*)\rangle \in \mathrm{Obs}(c)$ & $\underline{s}''\langle(X,-)\rangle \in \mathrm{Obs}(c)$
where $\underline{s}'$ is $\underline{s}$ stripped of any final "(Y,*)" or "(Y,-)"s
and $\underline{s}''$ is $\underline{s}$ stripped of any final (X,-)

c) $\underline{s}\langle(X,a)\rangle\underline{t} \in \mathrm{Obs}(c) \Leftrightarrow a \in X$ & $\underline{s}\langle(\{a\},a)\rangle\underline{t} \in \mathrm{Obs}(c)$
provided $\underline{s}$ has neither of the forms $\underline{s}''\langle(\{a\},-)\rangle$ and $\underline{s}''\langle(X,-)\rangle$

d) $\underline{s}\langle(X,\delta)\rangle\underline{t} \in \mathrm{Obs}(c)$ & $X \neq Y$ & $(\neg\exists\underline{r}.\ \underline{s} = \underline{r}\langle(Y,-)\rangle)$ & $\delta \neq *$
$\Leftrightarrow \underline{s}\langle(Y,-)(X,\delta)\rangle\underline{t} \in \mathrm{Obs}(c)$

e) $\underline{s}\langle(X,-)(X,\delta)\rangle\underline{t} \notin \mathrm{Obs}(c)$; $\underline{s}\langle(X,-)(Y,*)\rangle \notin \mathrm{Obs}(c)$

f) $\underline{s}\langle(X,*)\rangle\underline{t} \in \mathrm{Obs}(c) \Rightarrow \underline{t} = \langle\rangle$

g) $\underline{s}\langle(X,*)\rangle \in \mathrm{Obs}(c)$ & $Y \subseteq X \Rightarrow \underline{s}\langle(Y,*)\rangle \in \mathrm{Obs}(c)$

h) $\underline{s}\langle(X,*)\rangle$ Obs(c) & $(\forall a \in Y.\ \underline{s}\langle(\{a\},a)(\emptyset,*)\rangle \notin \mathrm{Obs}(c))$
$\Rightarrow \underline{s}\langle(X \cup Y,*)\rangle \in \mathrm{Obs}(c)$

The above tells us that we can effectively deduce nothing from the components of observations which have the form (X,-), so we might as well ignore them. In fact it tells us that the set of finite observations (i.e. observations with only finitely many components) can be deduced from a knowledge of which observations of the form $\langle(\{a\},a)\ldots(\{d\},d)(X,*)\rangle$ are in the set Obs(c) (i.e. finite sequences of single symbols offered and accepted, followed by a set infinitely refused). This clearly is closely related to the non-deterministic model. Define a map "ζ" from observations to $(\Sigma^* \times \wp(\Sigma))$ by $\zeta(\underline{s}) = (\langle\rangle,\emptyset)$ if $\underline{s}$ does not have the above "canonical" form; $\zeta(\underline{s}) = (\langle a\ldots d\rangle, X)$ if $\underline{s} = \langle(\{a\},a)\ldots(\{d\},d)(X,*)\rangle$. Let us call the function from H to observations represented by steps (ii) - (iv) of the earlier translation procedure by the name "η". Define $\theta = \zeta \circ \eta$. Define a map "$\Psi$" from C (a P,Q-space) to $\wp(\Sigma^* \times \wp(\Sigma))$ by $\Psi(c) = \{\hat{\theta}(\underline{a}) \mid \underline{a} \in B(c)\}$. From the above discussion we can deduce several things about this function.

Firstly $\Psi(c)$ is almost an element of the non-deterministic model M. $\Psi(c)$ is non-empty, has a prefix-closed domain, and satisfies the two conditions $(s,X) \in \Psi(c)$ & $Y \subseteq X \Rightarrow (s,Y) \in \Psi(c)$ and $(s,X) \in \Psi(c)$ & $(\forall a \in Y.(s\langle a\rangle,\emptyset) \notin \Psi(c)) \Rightarrow (s,X \cup Y) \in \Psi(c)$.

Secondly we can deduce from $\Psi(c)$ exactly what the finite elements of Obs(c) are. Furthermore, if c and d are two processes such that $\Psi(c) \subseteq \Psi(d)$, then every finite element of Obs(c) is also an element of Obs(d). This means that every predicate of C which depends only on the observed response of a process to finite sequences of sets can be exactly determined from the process' image under Ψ. Any predicate of the form "each finite observation is correct" can be translated to a predicate "χ" of H for which $\hat{\chi}^{+}(\Psi(c)) \Leftrightarrow \hat{\chi}(c)$. If we restrict ourselves to predicates of this form (of which more later) we can regard the image of Ψ as a class 1 model for C.

There is no necessity that $\Psi(c)$ should satisfy the directed closure condition which we imposed on the non-deterministic model. To see this we simply have to consider the following case, where it is assumed that $N \subseteq \Sigma$ (natural numbers).

Define a P,Q-space C to contain the elements σ (which has no transitions), ρ_n (for each $n \in N$) which has exactly the transitions $\rho_n \overset{m}{\rightarrow} \sigma$ $(m \geqslant n)$, and finally τ which has exactly the transitions $\tau \dot{\rightarrow} \rho_n$ $(n \in N)$. A little thought reveals that $(\langle\rangle,X) \in \Psi(\tau)$ for each finite $X \subset N$, but that $(\langle\rangle,N) \notin \Psi(\tau)$.

There are essentially two ways of putting this right: either one closes up under the rule (i.e. modifying the function Ψ to include all pairs (s,X) implied by directed closure) or one restricts consideration to spaces C which satisfy it naturally. The simpler of these two options is the second, and it is this one which we shall follow here. There are two alternative conditions we can adopt to ensure directed closure: either Σ must be finite or the P,Q-space we study must be finite branching. In the first case directed closure is trivial; in the second case it follows from Konig's lemma.

Let us consider now the expressive power of the type of predicates described above. A predicate of the form "every finite observation is correct" is clearly the same as one which says that "every incorrect finite observation is impossible". Thus any predicate of behaviour with the property that all infringements of it are both observable and detectable after finitely many external actions, is of the correct form for reliable and monotonic determination from $\Psi(c)$. As an example consider the implications of the predicate Buff (first introduced in chapter five) when true of $\Psi(c)$. (Note that Buff is a monotonic predicate.)

Buff($\Psi(c)$) implies several facts about Obs(c), which can informally be written as follows.

(i) "c" is a partially correct buffer, in that its output is at all times a prefix of its input.

(ii) When "c"s output is the same as its input (is "empty") it will not infinitely resist communicating with any experimenter who persists in offering it some set of input symbols.

(iii) When "c" has output less than it has input it will not infinitely resist communicating with an experimenter who persists in offering it (at least) the symbol which it next ought to output.

Note one fact which is illustrated by this example, namely that our insistence that certain infinite (in time) observations be absent implies the presence of certain finite (in time) observations (with certainty, rather than possibility, of occurrence). It is because of this influence on the set of observations which we can reliably expect to occur in finite time that we need to consider the possible infinite refusals. The other type of infinite observation, namely cases where infinite sequences of external actions occur, has no such influence.

We have established that the non-deterministic model M is reasonably thought of as a class 1 model for U, the universal finite branching P,Q-space (relative to the function Ψ and whichever alphabet we care to use). The first difference which one notes between this case and our study of the deterministic model is the fact that M, when regarded as a P,Q-space in the natural way, is not finite branching and so is not naturally "modelled by itself". We would expect to think of M as a P,Q-space by the law $N_1 \overset{a}{\to} N_2$ if $N_2 \subseteq N_1$ <u>after</u>$\langle a \rangle$. Even if Σ is finite this gives rise to infinite branching. If Σ is infinite the situation is irredeemable since there are elements of M which cannot be the image under Ψ of any element of U. An example of such a process is given by $\{(\langle\rangle,X) \mid \forall i.\{2i,2i+1\}\cap X \neq \emptyset\}\cup\{(\langle i\rangle,X) \mid i \in N\}$ (once again we assume $N\subseteq\Sigma$). If Σ is finite it is possible to make M into a finite branching P,Q-space on which the map Ψ is the identity. One way of doing this is with the transitions

$$a \in N \;\Rightarrow\; N \xrightarrow{a} (N \text{ after} \langle a\rangle)$$

$$(\langle\rangle,X) \in N \;\&\; (\Sigma - X) \subsetneq N^{O} \;\Rightarrow\; N \dot{\rightarrow} (\{(\langle\rangle,Y) \mid Y \subseteq X\}\cup\{(\langle a\rangle s,Y) \in N \mid a \notin X\}).$$

We can thus conclude that the map $\Psi:U \to M$ is onto if and only if Σ is finite.

It is now time to consider the question of implementation. We have a class 1 model so our requirements are not on the face of it so stringent as in the case of the deterministic, class 2, model. We know from long experience that each of our operators over the model M is monotonic, which is all that is required of them for the class 1 model theory to work. Recall our definitions: $e^* \in U'$ is a correct implementation of $e \in M'$ if for all "states" $\pi \in U^{\Theta}\times S$ we have (adopting the same notation as in our study of the deterministic model) $e(\Psi(\pi_1),\pi_2) \supseteq \Psi(e^*(\pi))$. If $\underline{op}:M'^{k} \to M'$ is a k-place operator over M then $\underline{op}^*:U'^{k} \to U'$ is said to implement op ($\underline{op}^*\,\underline{imp}\,\underline{op}$) if for all $e_1,\ldots,e_k \in M'$ and $e_1^*,\ldots,e_k^* \in U'$ such that e_i^* implements e_i correctly for all i, we have that $\underline{op}^*(e_1^*,\ldots,e_k^*)$ is a correct implementation of $\underline{op}(e_1,\ldots,e_k)$. In the case of operators $\underline{op}$ and $\underline{op}^*$ which are just the natural extensions to M' and U' of operators over M and U (op and op*, say) it is easy to see that it is enough to prove that $op(\Psi(c_1),\ldots,\Psi(c_k)) \supseteq \Psi(op^*(c_1,\ldots,c_k))$ for all $c_1,\ldots,c_k \in U$. (This remains true in a simple way even when $\underline{op}^*$ is the composition of more than one "extended" operators, since the extension of a composition is the same as the composition of extensions.) Thus in attempting to implement the majority of our operators over M' which are extensions of operators over M it is sufficient to consider the implementation of the M-operators by U-operators.

Notice the fact that infinite hiding is impossible to define properly over the model M, and also impossible to implement properly over U in any obvious way (because of the creation of infinite branching). This emphasizes the link between these two systems, and goes at least part of the way towards explaining the difficulties which arise with infinite hiding over the non-deterministic model.

Some of the operators, notably a →, ;, /X (finite X) are easy to define correctly. Indeed it is not too hard to show that the versions of these three operators introduced earlier (in the section on operators over universal spaces) are all well-defined over U and correct implementations of the corresponding operators over M. Also the operators over U' we defined to represent "x:T →" and recursion are not hard to justify. An interesting example is the case of recursion. Firstly the existence of fixed points of "constructive" functions of U is easy to prove because of the comparatively simple structure of the space U. Suppose that e* is an element of U' defined using operators of the types described earlier which correctly implements some element e of M'. If B is any process variable (element of Θ) then for each "state" π the equation $\sigma = e^*(\pi[\sigma/B])$ has a unique solution which we will call σ. To show that the recursion operator over U' correctly implements that over M' it is sufficient to show that $\Psi(\sigma) \subseteq \bigsqcup_{n=0}^{\infty} F^n(\text{CHAOS})$, where $F:M \to M$ is defined $F(A) = e(\pi'[A/B])$ $(\pi' = \Psi(\pi))$. To do this it is sufficient to show that $\Psi(\sigma) \subseteq F^n(\text{CHAOS})$ for each n; we will do this by induction.

$$\Psi(\sigma) \subseteq F^0(\text{CHAOS}) = \text{CHAOS} \quad \text{trivially.}$$

Suppose $\Psi(\sigma) \subseteq F^n(\text{CHAOS})$,

then $e(\pi'[\Psi(\sigma)/B]) \subseteq e(\pi'[F^n(\text{CHAOS})/B]) = F^{n+1}(\text{CHAOS})$

as e is monotonic; also $\pi'[\Psi(\sigma)/B] = \Psi(\pi[\sigma/B])$ so

$e(\pi'[\Psi(\sigma)/B]) \supseteq \Psi(e^*\pi[\sigma/B]))$ as e* correctly implements e.

Putting these facts together we get $F^{n+1}(\text{CHAOS}) \supseteq \Psi(e^*\pi[\sigma/B]))$, which is what we wanted to show since $\sigma = e^*\pi[\sigma/B])$.

The analysis of mutual recursions is no more difficult.

Problems arise however when one tries to implement the two important operators "□" and "||". In each case the problem occurs because of the necessity of being able to cope with one diverging and one non-diverging operand. It is not hard to see from the non-destructive nature of our operators and the finite-branching nature of U, that each of these operators, when faced with an initially diverging operand, must itself have the possibility of diverging before performing any external actions. This is because they must be able to cope with processes which execute n internal actions before doing any external action for each $n \in N$, and so can perform arbitrarily long initial sequences of internal actions; we can then deduce the existence of an infinite sequence by Konig's lemma.

With "$\square$" this problem is avoided if we restrict the use of "$\square$" to the case where each side is guarded in some way (whether or not the guards are disjoint).

However with "$\|$" the problems are more serious. There seems to be no way of implementing "$\|$" correctly without appealing to some kind of "fairness". One would require that neither operand could perform infinitely many actions while there was some action continuously possible for the other. While this certainly does not seem an unreasonable assumption it is not possible to make such a stipulation in systems satisfying the postulate Q7. An example of the problem we face is given below.

Let σ be the element of U with the single transition $\sigma \xrightarrow{\cdot} \sigma$ and let ρ be the element of U with the single transition $\rho \xrightarrow{a} \tau$, where τ has no transitions ($a \in \Sigma$). The values assigned by Ψ to σ and ρ are respectively <u>abort</u> and a → <u>abort</u>. Thus $(\Psi(\sigma) {}_{\{b\}}\|_{\{a\}} \Psi(\rho)) =$ a → <u>abort</u>. Thus whenever op* is a correct implementation of "${}_{\{b\}}\|_{\{a\}}$" the process op*(σ,ρ) cannot initially diverge (as $\Psi(\upsilon)$ contains $(\langle\rangle,\Sigma)$ whenever υ can initially diverge).

The introduction of fairness would require the alteration of Q7. This would rather complicate matters. Firstly our work concerning morphisms and universal spaces would no longer be valid because 8.10 would no longer hold. Secondly allowing fairness would mean that the image $\Psi(\sigma)$ of a finite-branching process was not necessarily directed-closed (because Konig's lemma would no longer be applicable). While it is likely that these problems could be overcome to some extent and some sort of consistent theory produced we have not got space here to investigate this topic further. In any case it is perhaps better not to assume fairness unless we have to, and we will see shortly that we can do without it. It might also be remembered that the map Ψ identifies divergence with deadlock, something which does not seem consistent with the philosophy expressed in the introductory booklet (ℯ). This fact is brought out further by the observation that, with respect to the map Ψ, the hiding operator "/X" which we earlier defined over U is a correct implementation of the (non-continuous) alternative hiding operator "\X" defined over M by

$$A\backslash X = \{(s/X,Y) \mid (s,X\cup Y)\in A\}$$
$$\cup \{(s/X,Y) \mid \{s' \mid s'/X = s/X\} \text{ is infinite}\} .$$

All this seems to imply that the map "Ψ" is not quite the one we want. We will thus give up this attempt and try again.

The non-deterministic model: second attempt

In the last section we made the assumption that our modelling function was based on the observations of an experimenter who cannot detect anything about what goes on inside processes. We have previously mooted the possibility that an experimenter might be able to detect the presence of internal activity by means of a light on the side of the machine or suchlike. The chief consequence of assuming this would be the finite detectability of deadlock by the experimenter. If a set of symbols is offered to a process while it is inactive (the light is out) for a sufficiently long time the experimenter may (correctly) deduce that no further action (internal or external) can occur while he persists in offering the same set (or a subset of it).

There is thus often no need to wait infinitely to detect refusal of a set. Indeed the need to wait infinitely would be rather unfortunate, implying both an infinite consumption of energy by the process and infinite patience on the part of the experimenter. It is important to discriminate between the notions of finite refusal (brought about by the process coming into some "stable" state) and infinite refusal (brought about by divergence, an infinite sequence of internal actions). Because our experimenter now has the ability to detect refusal finitely, and because divergence is inherently undesirable, we must expect that he will desire that all refusals will be finite (i.e. that processes are free from divergence).

Let us assume that our experimenter is chiefly interested in the behaviour of processes over finite intervals. We will thus examine the behaviours which can result from the application of a finite sequence of sets to a process. There are several things which the experimenter might see when he applies a set:

finite inconclusive wait	(denoted by -)
finitely observed refusal	(*)
communication of some $a \in \Sigma$	(a)
divergence (infinite wait without response)	(?)

We will assume that the experimenter is not confident enough to record any other details about internal activity than that which is implied in the above.

We will assume that he bases all correctness conditions upon a process' observed reactions to such experiments. Note that any observation of this type which does not include divergence is completed in a finite time, and that divergence can only occur at the end of an observation. This is of course a very good reason for defining divergent observations to be incorrect.

The following is a translation procedure designed to extract from the behaviour set of a process those behaviours which can result from the application of a finite sequence of sets, and then extract the observable features of these (from the point of view of the experimenter described above).

(i) Delete the state components of a behaviour (i.e. project into H).

(ii) If the resulting sequence has one of the following "inadmissible" types replace it by $\langle\rangle$.

a) Sequence with infinitely many external actions, "-"s and "*"s.

b) Any sequence with an infinite tail of $(X,\cdot)$s in which the "X"s are not eventually constant.

(iii) Replace all "$\cdot$"s by "-".

(iv) Replace any infinite tail of "(X,-)"s (with constant X) by (X,?).

(v) Delete all "(X,-)"s which are followed by some (X,δ) (same X).

The explanation of these steps is as follows:

(i) The internal state is not observable.

(ii) These types of sequences cannot result from behaviours which occur when the experimenter applies a finite sequence of **sets**. Sequences of type (a) are impossible because the end of each pair of any of the forms (X,-), (X,a) or (X,*) represents the end of an application of a set. This is not so for pairs of the form $(X,\cdot)$ because individual internal actions are not observable. The only infinite sequences which are left after (a) are those with an infinite tail of the form $\langle (X_1,\cdot)(X_2,\cdot)\ldots\rangle$; clause (b) eliminates all of these which cannot occur when a finite sequence of sets is applied.

(iii) Individual internal actions are not observable.

(iv) Any infinite tail of the form $\langle (X,-)(X,-)\ldots\rangle$ must have resulted from an infinite sequence of internal actions.

(v) Any (X,-) which is followed by another application of the same set can be ignored.

Define ξ to be the function of behaviours implied by steps (ii) - (v) of the above procedure; define Obs'(c) to be $\{\hat{\xi}(\underline{a}) \mid \underline{a} \in B(c)\}$ for any process c. Obs'(c) is the set of observations which our experimenter can make of c. The following are all easy consequences of our postulates.

8.27 Lemma

If C is a P,Q-space and $c \in C$ then

a) $\langle\rangle \in Obs'(c)$

b) $\underline{s}.\underline{t} \in Obs'(c) \Rightarrow (\underline{s}'(\emptyset,*) \in Obs'(c) \vee \underline{s}'(\emptyset,?) \in Obs'(c))$
& $\underline{s}''(X,-) \in Obs'(c)$

where $\underline{s}'$ & $\underline{s}''$ result from stripping $\underline{s}$ of any final $(\emptyset,-)$ and (X,-) respectively after removing any final (Y,?)

c) $\underline{s}\langle(X,a)\rangle\underline{t} \in Obs'(c) \Leftrightarrow a \in X \;\&\; \underline{s}\langle(\{a\},a)\rangle\underline{t} \in Obs'(c)$
if $\underline{s}$ has neither of the forms $\underline{s}'\langle(X,-)\rangle$ and $\underline{s}'\langle(\{a\},-)\rangle$

d) $\underline{s}\langle(X,\delta)\rangle\underline{t} \in Obs'(c) \;\&\; X \neq Y \;\&\; \neg\exists\underline{r}.\, \underline{s} = \underline{r}\langle(Y,-)\rangle$
$\Leftrightarrow \underline{s}\langle(Y,-)(X,\delta)\rangle\underline{t} \in Obs'(c)$

e) $\underline{s}\langle(X,?)\rangle\underline{t} \in Obs'(c) \Rightarrow \underline{t} = \langle\rangle$ (f) $\underline{s}\langle(X,*)(Y,?)\rangle \notin Obs'(c)$

g) $\underline{s}\langle(X,*)(Y,a)\rangle\underline{t} \in Obs'(c) \Rightarrow a \notin X$

h) $\underline{s}\langle(X,*)\rangle\underline{t} \in Obs'(c)$ & $Y \subseteq X \Rightarrow \underline{s}'\langle(Y,*)\rangle\underline{t} \in Obs'(c)$
where $\underline{s}' = \underline{s}$ unless $\underline{s} = \underline{s}''\langle(Y,-)\rangle$ in which case $\underline{s}' = \underline{s}''$

i) $\underline{s}\langle(X,*)\rangle\underline{t} \in Obs'(c)$ & $(\forall a \in Y . s\langle(X,*)(\{a\},a)(\emptyset,-)\rangle \notin Obs'(c))$
$\Rightarrow \underline{s}'\langle(X \cup Y,*)\rangle\underline{t} \in Obs'(c)$
where $\underline{s}' = \underline{s}$ unless $\underline{s} = \underline{s}''\langle(X \cup Y,-)\rangle$ in which case $\underline{s} = \underline{s}''$

j) $\underline{s}\langle(X,*)\rangle\underline{t} \in Obs'(c) \Rightarrow \underline{s}'\underline{t} \in Obs'(c)$
where $\underline{s}' = \underline{s}$ unless there are some $\underline{s}''$, $\underline{t}'$, Y and δ such that
$\underline{s} = \underline{s}''\langle(Y,-)\rangle$ and $\underline{t} = \langle(Y,\delta)\rangle\underline{t}'$ in which case $\underline{s}' = \underline{s}''$

k) $\underline{s}\langle(X,?)\rangle \in Obs'(c) \Rightarrow \underline{s}'(Y,?) \in Obs'(c)$
where $\underline{s}' = \underline{s}$ unless $\underline{s} = \underline{s}''\langle(Y,-)\rangle$ in which case $\underline{s}' = \underline{s}''$

l) $\underline{s}\langle(X,-)\rangle$ $Obs'(c) \Rightarrow \underline{s}\langle(X,*)\rangle \in Obs'(c) \vee \underline{s}\langle(X,?)\rangle \in Obs'(c)$
$\exists a \in X.\ \underline{s}\langle(X,a)(\emptyset,-)\rangle \in Obs'(c)$

m) $\underline{s}\langle(X,*)(Y,*)\rangle\underline{t} \in Obs'(c) \Leftrightarrow \underline{s}\langle(X \cup Y,*)\rangle\underline{t} \in Obs'(c)$
provided $\underline{s}$ has neither of the forms $\underline{s}'(X,-)$ or $\underline{s}'(X \cup Y,-)$

Once again, by d, we can deduce nothing from the components of observations with the form (X,-) so we might as well ignore them (from the above the only purpose they seem to serve is to complicate matters). Once again it is possible to deduce the exact form of Obs'(c) from a knowledge of which of a class of "canonical" elements it contains. These are the ones which are of the form $\{(\{a\},a),(X,*),(\emptyset,?) \mid a \in \Sigma, X \subseteq \Sigma\}^*$ with $(\emptyset,?)$ only occuring at the end of sequences and components of the form (X,*) being separated by at least one of the form $(\{a\},a)$.

Thus every predicate of processes of the form "each observation of behaviour is correct" can be re-written in the form "each canonical observation is correct".

Because of the various rules of 8.27, notably (b), (f), (g), (j), (k) and (m) a very expressive subset of the canonical observations are those of the two forms

$\langle(\{a\},a)\ldots(\{d\},d)(X,*)\rangle$ (+)

$\langle(\{a\},a)\ldots(\{d\},d)(\emptyset,?)\rangle$ (++).

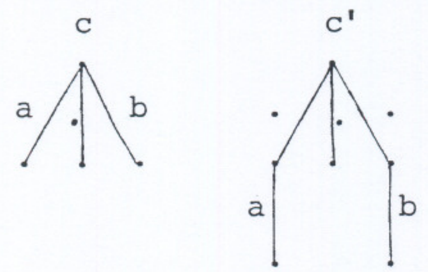
From a knowledge of which observations of these two types are present in Obs'(c) one cannot deduce the whole shape of Obs'(c) (see below) but one can answer such questions as

"Can c perform the string $\langle a\ldots d\rangle$ and then finitely refuse the set X?"

"Can c perform the string $\langle a\ldots d\rangle$ and then diverge when offered X?"

accurately. (For example, if $\langle\langle(\{a\},a)\ldots(\{d\},d)(X,*)\rangle\rangle \in Obs'(c)$ then there is no element of Obs'(c) in which the symbols a..d are accepted from any sets followed by the finite refusal of X, whether or not there are any "(Y,-)"s and/or "(Y,*)"s between these events.)

There do exist processes with identical sets of observations of the forms (+) and (++) but different sets Obs'. As an example of this consider the two processes illustrated below.



$(\Sigma = \{a,b\})$

The processes c and c' have the same observations of the forms (+) and (++), but $\langle\langle(\{a\},*)(\{b\},b)(\emptyset,*)\rangle\rangle$ is an element of Obs'(c') without being one of Obs'(c).

The minor differences between processes with identical sets of (+) and (++) observations are usually unimportant from a correctness point of view, however. The use of observations of this simplified form has the advantages of simplicity and the production of a natural map into the non-deterministic model M.

Let us suppose that our experimenter, aware of the expressive power of observations of types (+) and (++), contents himself with checking to see which if them are possible. It is sufficient for him to check those of type (+), since any possible observation of type (++) will become apparent as he does this. To check the observation $\langle\langle(\{a\},a)\ldots(\{d\},d)(X,*)\rangle\rangle$ (which we abbreviate as $(\langle a..d\rangle,X)$ he will apply the sets $\{a\},..,\{d\}$, X and $\emptyset$ in turn, with each set except the last waiting until he gets some definite response. There are essentially three possible outcomes to such an experiment.

(i) It may succeed: i.e. the process may accept a,..,d in turn and then refuse X finitely.

(ii) It may fail finitely: i.e. it may finitely refuse one of the sets $\{a\},\ldots,\{d\}$ or it may accept some element of X.

(iii) It may diverge: i.e. the process may not give any definite response at some stage where one is required.

Define the three sets E_1^c, E_2^c and E_3^c to be the sets of possible passes, failures and divergences amongst the experiments he might carry out on the process c. (We will assume that the elements of these sets are written in the abbreviated form, so that E_1^c, E_2^c and E_3^c are all subsets of $\Sigma^* \times \wp(\Sigma)$.) We can deduce E_1^c, E_2^c and E_3^c from Obs'(c) as follows:

$$E_1^c = \{(\langle a..d\rangle, X) \mid \langle\langle(\{a\},a)\ldots(\{d\},d)(X,*)\rangle\rangle \in \text{Obs}'(c)\}$$

$$E_2^c = \{(\langle a..d\rangle, X) \mid \langle\langle(\{a\},*)\rangle\rangle \in \text{Obs}'(c) \vee \ldots \vee \langle\langle(\{a\},a)..(\{d\},*)\rangle\rangle \in \text{Obs}'(c) \vee \exists b.\, \langle\langle(\{a\},a)\ldots(\{d\},d)(X,b)(\emptyset,-)\rangle\rangle \in \text{Obs}'(c)\}$$

$$E_3^c = \{(\langle a..d\rangle, X) \mid \langle\langle(\{a\},?)\rangle\rangle \in \text{Obs}'(c) \vee \ldots \vee \langle\langle(\{a\},a)..(\{d\},?)\rangle\rangle \in \text{Obs}'(c) \vee \langle\langle(\{a\},a)\ldots(\{d\},d)(X,?)\rangle\rangle \in \text{Obs}'(c)\}$$

(We might note at this point that the three sets each depend monotonically on the set Obs'(c).)

These sets satisfy some simple laws, which are easily provable from 8.27 and their definitions.

8.28 <u>Lemma</u>

(i) $(s,X) \in E_3^c \Rightarrow (st,Y) \in E_3^c$

(ii) $(s,X) \in E_1^c \;\&\; Y \subseteq X = (s,Y) \in E_1^c$

(iii) $(s,X) \in E_1^c \;\&\; (\forall a \in Y.(s\langle a\rangle,\emptyset) \notin E_1^c \cup E_3^c) \Rightarrow (s,X \cup Y) \in E_1^c$

(iv) $(st,X) \in E_1^c \cup E_3^c \Rightarrow (s,\emptyset) \in E_1^c \cup E_3^c$

(v) $E_1^c \cup E_2^c \cup E_3^c = \Sigma^* \times \wp(\Sigma)$

(vi) $(\langle\rangle,\emptyset) \in E_1^c \cup E_3^c$

The set $E_1^c \cup E_3^c$ represents the set of experiments which need not necessarily fail finitely, and as such is very important. If we desire that every observation of type (+) which occurs is correct in some sense then this can be checked by trying out all incorrect ones and expecting them to fail finitely. By the above this must occur if each element of $E_1^c \cup E_3^c$ is correct. Because of (i) above there is also a simple way to exclude undesirable divergence (observations of type (++))

from consideration of the set $E_1^c \cup E_3^c$. In particular the set Obs'(c) is completely free from divergence if $E_1^c \cup E_3^c$ satisfies the condition

$$\forall s. \forall X. (s,X) \in E_1^c \cup E_3^c \Rightarrow \exists t,Y. \ (st,Y) \notin E_1^c \cup E_3^c$$

since this implies that $E_3^c = \emptyset$. When E_3^c is empty every finite experiment (whether or not of the form used to determine E_1^c, E_2^c and E_3^c) must terminate finitely. When E_3^c is empty it is not hard to see that E_2^c depends monotonically on E_1^c (i.e. the more experiments there are which can pass, the more there are which can fail).

Thus $E_1^c \cup E_3^c$ is a very useful set. It also satisfies all the laws of the non-deterministic model except directed closure. When the P,Q-space in question is finite-branching we once again have directed closure. We will therefore once again assume that the space we are seeking to model is U, the universal finite branching P,Q-space.

We are, in $E_1^c \cup E_3^c$, provided with a natural and expressive map from U to the non-deterministic model M. We have not however defined it in the usual fashion (a function $\hat{\theta}$ of behaviours). It can in fact be written in a correct form, but we need to invoke our right to use a relation rather than a function. We already have the function $\hat{\xi}$ for producing Obs'(c) from B(c). Define a relation μ on the sequences making up Obs'(c) as follows.

$$\begin{aligned}
\mu(\underline{s}) &= \emptyset \quad \text{if } \underline{s} \text{ has neither of the forms (+) and (++)} \\
&= \{(\langle a..d\rangle, X)\} \quad \text{if } \underline{s} = \langle(\{a\},a)..(\{d\},d)(X,*)\rangle \\
&= \{(\langle a..d\rangle t, X) \mid t \in \Sigma^* \ \& \ X \subseteq \Sigma\} \text{ if} \\
&\qquad\qquad \underline{s} = \langle(\{a\},a)..(\{d\},d)(\emptyset,?)\rangle
\end{aligned}$$

If we now let $\rho = \mu \circ \hat{\xi}$ it is clear that $E_1^c \cup E_3^c = \bigcup\{\rho(\underline{a}) \mid \underline{a} \in B(c)\}$.

Let us define $X(c) = E_1^c \cup E_3^c$. Because of the expressive power of the set $E_1^c \cup E_3^c$ it is fair to regard M as a class 1 model for U relative to the map X.

This map seems far more satisfactory than Ψ from an aesthetic point of view: the discrimination between divergence and deadlock appears to correspond far better to our earlier expectations. Despite this the problems of implementing our standard operators are worse rather than better.

"a →", "x:T →", "a.x:T →", "x →" , "a.x →" and "<u>or</u>" present no difficulty, and neither do hiding and recursion. (Note that hiding and recursion are the two operators most closely associated with divergence.) All the other operators seem to give rise to unsurmountable problems. This is because of the ways in which they deal with the representations of diverging processes. The value given by X to any process which can diverge without communicating externally is CHAOS. If any of the operators "a.", "□" and "‖" is presented with such a process in any argument or if ";" is presented with one in its first argument then, by the same arguments which we used in the last section for "‖", the result must also be able to diverge without communicating externally (and so have value CHAOS assigned to it by X). This is inconsistent with the following observations:

a.CHAOS ≠ CHAOS

CHAOS;<u>abort</u> ≠ CHAOS

CHAOS □ (a → <u>skip</u>) ≠ CHAOS

$(\text{CHAOS}_X\|_Y\underline{\text{abort}}) \neq \text{CHAOS}$ unless $X = \Sigma$ and $Y = \emptyset$.

(This time there is no hope of mending "‖" by an assumption of fairness.)

The basic problem here is that, though in the initial construction of our model we were careful to identify the "bad" processes we created with CHAOS (witness the correctness of hiding and recursion), we did not follow through our arguments to see how these "bad" processes would behave when operated upon. In short it seems to be the operators which we defined over M which are at fault here rather than the modelling function: they do not appear to be reasonable (in the sense defined earlier). All these failings can be remedied by adjusting the definitions of the operators: making them "strict" in some sense. The following are definitions of more acceptable operators "□'", ";'", "a.'" and "‖'".

$A \,\square'\, B = A \,\square\, B$ if $A \neq \text{CHAOS}$ and $B \neq \text{CHAOS}$

$\qquad = \text{CHAOS}$ otherwise

$A;'B = A;B \cup \{(s,X) \mid A \ \underline{\text{after}}\ s = \text{CHAOS}\}$

$a.'B = a.B \cup \{((a.s)t,X) \mid B \ \underline{\text{after}}\ s = \text{CHAOS}\}$

$(A_X\|'_Y B) = (A_X\|_Y B) \cup \{(st,X) \mid s \in (X \cup Y)^* \ \&\ s\restriction X \in \text{dom}(A) \ \&$
$\qquad s\restriction Y \in \text{dom}(B) \ \&\ \text{CHAOS} \in \{A \ \underline{\text{after}}\ s\restriction X,\ B \ \underline{\text{after}}\ s\restriction Y\}\}$

Each of these operators seems to be implementable with respect to X. There is however a price to pay. Some of the theory which we developed for the old operators no longer holds. A notable example of this is that the operator $\|'$ is not associative: suppose X,Y is a non-trivial partition of Σ (i.e. $X \neq \emptyset$ & $Y \neq \emptyset$).

Let $A = \{(s,S) \mid s \in X^*\}$

$B = \{(s,S) \mid s \in Y^*\}$

Observe that $((A_X \|'_Y B)_\Sigma \|_\Sigma' \underline{abort}) = CHAOS$

$(A_X \|'_\Sigma (B_Y \|'_\Sigma \underline{abort})) = \underline{abort}$

It is ironic that lemma 5.35 now holds in general (i.e. we no longer need to assume the absence of infinite internal chatter). These two lemmas (5.35 and associativity of $\|$) were both critical in proving the associativity of "$\gg$", the non-universality of which was one of the more paradoxical properties of our old operators. 5.35 holds in our new system generally because the parallel operator is no longer allowed to "hide" divergence from the environment.

Most of the troubles with the new operators seem to arise from the special way in which CHAOS is treated, the identification of a diverging process with one which can do anything but always terminates finitely. In the next section we will see how this problem can be removed through an adjustment to the model.

The main part of our earlier theory which is hit by the new operators is recursion through the parallel operator. For a discussion of how this is affected see the next section.

An improved model

This section is an extension of the last, so we will use the same notation.

The obvious way round the problems which arise from the confusion over CHAOS is to separate the notions of diverging and passing experiments more. One way of doing this would be to adopt the pair (E_1^C, E_3^C) as our representation of a process. This would have the advantage of being much more expressive; it has the disadvantage that the underlying model is not nearly so elegant or mathematically versatile as the old model M.

We can arrive at a satisfactory compromise in the following manner. Firstly we re-state our view that divergence of an experiment is worse than anything else. This time we will therefore keep a separate record of all experiments which may diverge. However we note again that when an experiment diverges there is no way in which an experimenter can finitely detect that it will not pass. We therefore adopt as our new representation of a process

$$\Omega(c) = E_1^c \cup E_3^c \cup \{(s,?) \mid (s,\emptyset) \in E_3^c\}.$$

We take as our new model the subset Q of $\wp(\Sigma^* \times (\wp(\Sigma) \cup \{?\}))$, defined to be those elements N which satisfy the following conditions. (We use the same notation as before, except that α will now conventionally represent an element of $\wp(\Sigma) \cup \{?\}$.)

(i) dom(N) is non-empty and prefix closed

(ii) $(s,?) \in N \Rightarrow (st,X) \in N$ & $(st,?) \in N$ for all t,X

(iii) $(s,X) \in N$ & $Y \subseteq X \Rightarrow (s,Y) \in N$

(iv) $(s,X) \in N$ & $(\forall a \in Y.(s\langle a\rangle,\emptyset) \notin N) \Rightarrow (s,X \cup Y) \in N$

(v) $\{X \mid (s,X) \in N\}$ is directed closed

Ω is easily seen to be a well-defined map from U to Q, and can easily be shown to be generated by a relation on behaviours in a very similar way to X.

The new model Q seems to possess all of the useful properties of M and a few more besides. There is a natural order on Q in exactly the same way as on M ($A \sqsubseteq B$ if $B \subseteq A$) and Q is a complete semilattice with respect to "$\sqsubseteq$". Q has minimum element $\Sigma^* \times (\wp(\Sigma) \cup \{?\})$ which we will call <u>CHAOS</u> to distinguish it from CHAOS ($= \Sigma^* \times \wp(\Sigma)$) which is not minimal in Q, though it is the minimal element free from divergence. Q has the same maximum elements as M. We should note that if c is an element of U free from divergence then $\Omega(c) = X(c) = \Psi(c)$. The map Ω makes more distinctions than either of the others, guaranteeing that if c is a process free from divergence and c' is one which can diverge then $\Omega(c) \neq \Omega(c')$.

The operators we define over this model should correspond to our original operators over M for processes free from divergence and take heed of our observations in the last section when their arguments can diverge. The following is a list of the operators we adopt over Q, together with a few remarks on how one might expect them to be implemented.

$a \to A = \{(\langle\rangle,X) \mid a \notin X\} \cup \{(\langle a\rangle s,\alpha) \mid (s,\alpha) \in A\}$

(We use the same scheme as before, switching "A" on after "a".)

$$\begin{aligned} A;B = {} & \{(s,X) \mid s \in (\Sigma^{-})^* \;\&\; \surd \notin X \;\&\; (s,X) \in A\} \\ & \cup \{(st,\alpha) \mid s \in (\Sigma^{-})^* \;\&\; (s\langle\surd\rangle,\emptyset) \in A \;\&\; (t,\alpha) \in B\} \\ & \cup \{(s,\alpha) \mid (s,?) \in A\} \end{aligned}$$

(We use the scheme defined in the section on operators. A is switched on initially, and when A communicates a hidden "√" it is switched off and B is switched on.)

$$\begin{aligned} A \,\Box\, B = {} & \{(\langle\rangle,X\cap Y) \mid (\langle\rangle,X) \in A \;\&\; (\langle\rangle,Y) \in B\} \\ & \cup \{(s,\alpha) \mid (s,\alpha) \in A\cup B \;\&\; s \neq \langle\rangle\} \\ & \cup \{(s,\alpha) \mid (\langle\rangle,?) \in A\cup B\} \end{aligned}$$

(Initially both are switched on and are allowed to perform any action. As soon as one performs an internal action the other is switched off.)

A <u>or</u> B = $A \cup B$

(This is trivial to implement.)

$$\begin{aligned} (A_X\|_Y B) = {} & \{(s,(X\cap V)\cup(Y\cap W)\cup T) \mid s \in (X\cup Y)^* \;\&\; (s\restriction X,V) \in A \;\&\; \\ & \qquad (s\restriction Y,W) \in B \;\&\; T\cap(X\cup Y) = \emptyset\} \\ & \cup \{(st,\alpha) \mid s \in (X\cup Y)^* \;\&\; (s\restriction X,\emptyset)\in A \;\&\; (s\restriction Y,\emptyset)\in B \;\&\; \\ & \qquad (s\restriction X,?) \in A \vee (s\restriction Y,?) \in B)\} \end{aligned}$$

(Both processes are always switched on and their external communications co-ordinated in the obvious way.)

$$\begin{aligned} a.A = {} & \{(a.s,a.X\cup Y) \mid (s,X) \in A \;\&\; Y\cap a.\Sigma = \emptyset\} \\ & \cup \{((a.s)t,\alpha) \mid (s,?) \in A\} \end{aligned}$$

$$\begin{aligned} A/X = {} & \{(s/X,Y) \mid (s,X\cup Y) \in A\} \cup \{((s/X)t,\alpha) \mid (s,?) \in A\} \\ & \cup \{((s/X)t,\alpha) \mid \{s' \in \mathrm{dom}(A) \mid s'/X = s/X\} \text{ is infinite}\} \end{aligned}$$

(These operators are implemented by operating on the external communications of "A" in the obvious ways.)

The various operators over Q' which we need (x:T →, recursion etc.) are defined in the obvious ways. Each of the above is a monotonic and continuous function of its operands.

In the above we thus seem to have definitions of our operators over Q which are implementable and which do not suffer from the artificial strictness we needed with *X*. Because we do not need this strictness we recover all the pleasant properties of old operators with the addition of the universal truth of (h) below (Lemma 5.35).

8.29 Theorem

The operators we have defined over Q satisfy the following.

a) $A \square A = A$

b) $A \square B = B \square A$

c) $(A \square B) \square C = A \square (B \square C)$

d) $(A;B);C = A;(B;C)$

e) $(a \to A);B = a \to (A;B)$

f) $(A \square B);C = (A;C) \square (B;C)$ if $\surd \notin A^{o} \cup B^{o}$

g) $((A_X\|_Y B)_{X\cup Y}\|_Z C) = (A_X\|_{Y\cup Z}(B_Y\|_Z C))$

h) $(A/X_Y\|_Z C) = (A_{X\cup Y}\|_Z C)/X$ if $X \cap Z = \emptyset$

i) each of "a →", ";", "□", "||", "a." and "/X" distributes over "or"

etc., etc.

Because we have both (g) and (h) we now have that "»" (if defined in the obvious way with a suitably defined "strip" operator) is in general associative on processes whose domains are contained in $(?T \cup !T)^*$.

Most of the theory of chapter five appears to go through practically unaltered. Operators remain non-destructive (in the obviously defined sense) in the same circumstances as before, and "a →", "x:T →" etc. are constructive. The parallel operator is never constructive over Q except in trivial circumstances; we do not use any constructive properties of "»" in chapter 5 and its non-destructiveness remains over Q in the circumstances of 5.30.

Circumstances are different in chapter six, however. Here we extensively use constructiveness properties of (A//a::B), an operator which is defined using "||". If this operator were defined in the same way over Q then it could never be constructive in its second argument when A was not CHAOS. This would be a very serious problem. It can be avoided by defining the operator seperately (i.e. without using the parallel operator we defined over Q). To do this we must look at what this operator represents. It is fundamentally different from the ordinary parallel operator in that it that it implies the dominance of one operand over the other. Let us imagine that the dominant operand is in effect the "black box" of our earlier discussion with the power to switch the other one on or off. The "master" might only switch on the "slave" when it had any need of it (i.e. when it had the ability to communicate with it). There is thus reason to believe that the operator

$$\begin{aligned}(A \| a::B) = \; & \{(s/a.\Gamma, X) \mid \exists Y, Z.(s,Y) \in A \;\&\; (Y \cup a.(\mathrm{swap}?!(Z)) = X \cup a.\Gamma \\ & \qquad \& \; (\mathrm{swap}?!(\mathrm{stripa}(s \upharpoonright a.\Gamma)), Z) \in B\} \\ & \cup \{((s/a.\Gamma)t, \alpha) \mid (s,?) \in A \;\&\; (\mathrm{swap}?!(\mathrm{stripa}(s \upharpoonright a.\Gamma)), \emptyset) \in B\} \\ & \cup \{((s/a.\Gamma)t, \alpha) \mid (s,\emptyset) \in A \;\&\; (\mathrm{swap}?!(\mathrm{stripa}(s \upharpoonright a.\Gamma)), \emptyset) \in B \\ & \qquad \& \; \exists s' \leqslant s, \exists b \in a.\Gamma, s\langle b\rangle \in \mathrm{dom}(A)\} \\ & \cup \{((s/a.\Gamma)t, \alpha) \mid \{s' \in \mathrm{dom}(A) \mid s/a.\Gamma = s/a.\Gamma \\ & \qquad \& \; \mathrm{swap}?!(\mathrm{stripa}(s \upharpoonright a.\Gamma) \in \mathrm{dom}(B)\} \text{ is infinite}\}\end{aligned}$$

is implementable (it seems likely that an implementation might resolve some of the non-determinism inherent in the above). This operator can be shown to satisfy the relation $a \neq b \Rightarrow ((A \| a::B) \| b::C) = ((A \| b::C) \| a::B)$ (the failure of this over M was a consequence of the untruth of 5.35 when infinite internal chatter was present).

One pleasing aspect of the above definition is that it resolves the problems we had in chapter six with "network chatter" because the function $F(B) = (A \| a::B)$ is no longer necessarily constructive if A satisfies the condition C_o^a (though it is with C_1^a). To see this consider the first example we quoted there, namely the process defined

$A \Leftarrow (X \| a::A)$

$X \Leftarrow ?x \rightarrow a!x \rightarrow \underline{\text{abort}}$

The progress of the two sequences of approximations (over M and Q) is as follows.

	over M	over Q
$F^o(\bot)$	CHAOS	CHAOS
$F^1(\bot)$	?x → abort	?x → CHAOS
$F^2(\bot)$	?x → abort	?x → CHAOS
. . .	. . .	. . .

The value assigned to this process in Q is far more satisfactory.

The buffer and stack examples of chapter six should go through virtually unaltered (because their recursions are both C_1^a). The sort examples will need more care. We will need to apply the analysis previously used to prove them free of network chatter to prove them well-defined and correct (there seems little doubt that this could be done with a little care).

It is to be expected that all of the theory of chapter seven will apply equally to the new model, with the rider that we should no longer have to make assumptions on freedom from infinite internal chatter for "↔" to be associative, etc. The various conditions we develop here and in earlier chapters for the avoidance of infinite chatter are of course still of considerable use, as they will now imply freedom from divergence (when divergence-free processes are combined).

Thus the revised model Q seems to have considerable advantages over the old one, as it appears to remove several of the less satisfactory features of that model. There is of course much further work to be done in formally transferring the results of M to Q, and a final verdict must await that together with more rigorous analysis of implementation. All we can say at this point is that there is considerable circumstan ial evidence pointing towards Q on both counts.

Conclusion

This has been a very long chapter, and yet it has left a lot of loose ends to be tidied up. There is more work to be done at several points. We have however developed at least the skeleton of a theory for comparing "real" systems with abstract systems and operational semantics with abstract semantics.

Conclusion

It is now time to look back over the work in this thesis, in order both to identify the points where further work is desirable and to make a few comparisons with similar work elsewhere. We will first go through the topics covered roughly in the order in which they have been presented, and later make a few general remarks.

Chapter one was of an introductory nature. The language it introduces is of a rather abstract type, though as stated there is no reason why more conventional languages should not be given semantics in the model introduced (or in the other models used later). It is possible to define congruent semantics for the same language over different domains. Indeed the present author has done this in two ways over Scott domains, both without continuations (presented in (j)) and with continuations. The first of these two was implemented by S.D. Brookes using Mosses' S.I.S. system, confirming amongst other things the awfulness of the "palindromes" example (1.19(iii)). The formal semantic techniques used in this chapter are useful but by no means essential for the abstract language used in this thesis, but are almost indispensable when dealing with more conventional languages with more advanced use of variables and perhaps jumps. We will return to the subject of such languages shortly.

The proof rules introduced in chapter two, amplified in chapters three and five, and used throughout this thesis seem to have certain advantages over the more common, and very similar type which requires us to prove a (possibly vacuous) predicate R_o of fix(F) and the relation "$R_i(A) \Rightarrow R_{i+1}(F(A))$" to be able to deduce $\forall i.R_i(\text{fix}(F))$. In the cases of our simpler rules, when used with respect to restriction operators, most of the advantages are aesthetic: proofs tend to be more elegant and there is no need to break up a predicate R into infinitely many R_i. The more advanced forms, for example those described in 2.21, 2.29, 2.33 and 2.34, seem to arise more naturally from the type of rule adopted here. The more abstract cases (where strong and extra continuity are used) do not appear to translate at all into the other type.

A further advantage of this type of rule is the form of the conditions required of predicates for them to be amenable to proof. The considerable sharpening of our insight gained in chapter three is a direct consequence of this. As in the case of chapter two this chapter and its appendix seem to be a fairly comprehensive treatment of its subject. The questions raised at the end of the appendix, while interesting from a mathematical point of view, can have little bearing on any practical work. The topology generated by extra-continuity (which coincides with strong continuity over countable alphabets) is of a type which has found several other uses in computer science; it is for example practically the same as the "Cantor topology" of Plotkin (i).

Chapter four is a summary of some of the author's contributions to the foundations of the non-deterministic model. Because this was a joint project much material desirable for a proper understanding of this model is missing. The set-theoretic principle proposed in 4.10 and proved in 4.15 is clearly the main result of this chapter. In addition to the proof of propositional compactness which forms part of 4.15 there are several other cases where 4.10 can be substituted for (the strictly stronger) Zorn's lemma in proofs of standard results in a natural way. Examples of this are the ultrafilter lemma and the classic paradoxical dissection of the unit sphere. (Of course the fact that we have the double implication in 4.15 places it exactly in the known hierarchy, but it is nevertheless pleasing that our result proves other results in natural ways.) The results of this chapter will apply equally to the revised model suggested at the end of chapter eight.

The proof rules introduced in chapter five are of course the same as those of chapter two, so the same comments apply. The main thing which is missing is a topological study of the non-deterministic model in the spirit of chapter three. If this were done it seems likely that the results obtained would be very similar to those of chapter three, with the exception that in a domain without a "top" it is impossible to obtain any of the proof rules derived from strong and extra continuity. It is clear from chapter eight that monotonic predicates play a special (though by no means exclusive) role over both the model of chapter four and the similar model of the last section of chapter eight. The study of these will certainly generate further (weaker) topologies over our spaces and may well allow us to develop further proof rules. (The topology of continuous monotonic predicates will be very similar to that of 3.21.)

The discussion of buffers in chapter five is more or less self contained. The techniques developed for the study of buffers are likely to have much wider applications, however. It should not require more than a few adjustments to transfer most of the work of chapter five to the revised model Q suggested in chapter eight. As suggested earlier the transfer of the work in chapter six will require a little more care because of the reduced constructiveness of $(A \| a::B)$ in its second argument. It would be interesting to compare the work done in chapter six on the elimination of network chatter over M with conditions required to prove the absence of divergence over Q. Hopefully it can be proved that the two are more or less equivalent: the implication of absence of divergence by the absence of network chatter would be a justification of our definition of network chatter. There is also the point that if divergence is excluded by conditions such as E_1^a then any fixpoint is greater than (divergence-free) CHAOS; the coincidence of the two systems of operators over the region $\{A \mid A \sqsupseteq \text{CHAOS}\}$ could then be used to justify the old analysis of such processes as Quicksort (6.9) over the new model.

The work in the first half of chapter seven requires few comments. Translation to the revised model should bring a few improvements. Since the "matrix" operator is not obviously suited to recursive use it is likely that its theory and the proof techniques for processes defined using it will be more akin to those used with "$\gg$" than those of "$(A \| a::B)$". The advantage of our non-deterministic models M and Q when deadlock is studied is the extremely simple way in which it manifests itself (i.e. $\Sigma \in A(s)$). There is clearly room for much work in extending the results and techniques developed in this section. It is also possible to study other, similar predicates such as "liveness", the predicate demanding that it is always possible for a process to return to its initial state. (The author has developed several methods for proving this predicate, which is rather less easy to prove than the absence of deadlock.)

Chapter eight is a summary of some of the author's most recent work, linked by the common theme of the study of the relationships between "real" systems and their models. It is of a less complete nature than the other chapters, several of its sections requiring further work. Despite this the author feels the conclusions reached are too important to leave out. Further formal analysis is required especially in the sections on the connection between weak-postulate spaces and P,Q-spaces, operators over universal spaces and the analysis of particular models. It is of course still necessary to formally **analyze** the revised model Q in the contexts of the earlier chapters. It would also be interesting to compare our relational "universal spaces" with other objects such as powerdomains (Plotkin (i), Smyth (1)) used to model non-determinism. Several comparative studies might be possible, such as an examination of the operational semantics of Cousot (b), and with CCS. It will be especially interesting to compare the treatments of diverging processes. (The author believes that Brookes (a) will contain some analysis of the connections between the model M and CCS.) There must also be comparisons between our study of processes through the tests they pass (E_1^c, E_2^c and E_3^c) and the work of Kennaway (g).

In addition to the specific points mentioned in the above paragraphs where further work is required, there are several wider fields where further work is desirable. The first of these is the application of our various methods to more "realistic" versions of C.S.P., with assignment to variables, "if.." statements, less rich recursion (and perhaps jumps?). These could probably be tackled best by a denotational semantics over the model Q in the style of chapter one.

Secondly it might be desirable to attempt to formalize some of the rather ad hoc proof techniques into more systematic methods. A good example of such methods is the technique developed in 6.16 - 6.18 for proving E_i^a and C_i^a. Some interesting formal rules over the deterministic models have been produced by Zhou (n).

Thirdly it would be interesting and probably instructive to attempt to apply our methods to some larger examples than those which we have used as illustrative examples. These might include the specification and justification of communications protocols, and the proof of correctness of a model operating system.

References

(a) S.D. Brookes: D.Phil thesis, to appear.

(b) P. Cousot and R. Cousot: Semantic analysis of communicating sequential processes; ICALP 80 proceedings Springer-Verlag.

(c) C.A.R. Hoare: Communicating sequential processes; Comm. ACM 21, 8(1978).

(d) C.A.R. Hoare: Communicating sequential processes, in "Construction of programs" Ed. McKeag & MacNaughton, C.U.P. 1980.

(e) C.A.R.Hoare, S.D. Brookes and A.W. Roscoe: A theory of communicating sequential processes; Technical monograph PRG 16, May 1981 (Also to appear in an extended form in J.A.C.M.).

(f) M. Hennessy and R. Milner: On observing nondeterminism and concurrency; ICALP 80 proceedings, Springer-Verlag.

(g) J.R. Kennaway and C.A.R. Hoare: A theory of nondeterminism, ICALP 80 proceedings, Springer-Verlag.

(h) C. Mead and L. Conway: Introduction to VLSI systems, Addison-Wesley 1980.

(i) G.D. Plotkin: A powerdomain construction, SIAM Journal on computing 5, Vol.3, pp.452-487, 1976.

(j) A.W. Roscoe: D.Phil. qualifying dissertation, 1979.

(k) D.S. Scott: Data types as lattices, SIAM Journal on computing 5 1976, pp.522-587.

(l) M.B. Smyth: Power domains; J. Comp. Syst. Sci. 16, (1978).

(m) S. Willard: General topology, Addison-Wesley 1968.

(n) Zhou Chou Chen and C.A.R. Hoare: Partial correctness of communicating sequential processes, Proc. Int. Conf. on distributed computing, April 1981.

In addition to the above works, which were specifically referred in the text, there are several others which have had a significant influence. Amongst these are the following.

(1) J.H. Conway: Regular algebra and finite machines, Chapman and Hall, 1971.

(2) E.W. Dijkstra: A discipline of programming, Prentice-Hall 1976.

(3) H.B. Enderton: Elements of set theory, Academic Press, 1977.

(4) K. Kuratowski: Introduction to set theory and topology, Permagon Press. (Especially chapter XIX, but beware the false theorem 2.4.)

(5) R. Milner: A calculus of communication systems, Springer Verlag 1980.

(6) D.S. Scott: Mathematical theory of computation, Oxford University lecture notes, Michaelmas term 1980.

(7) J.E. Stoy: Denotational Semantics, M.I.T. press 1977.

There are two further categories of works which should be mentioned. The first of these are numerous locally circulated documents, particularly those of C.A.R. Hoare and S.D. Brookes. The second category comprises the sources from which the author has borrowed the ideas for many of his worked examples.