stichting
mathematisch
centrum

MC

AFDELING INFORMATICA (DEPARTMENT OF COMPUTER SCIENCE) IW 163/81 APRIL

A. DE BRUIN

ON THE EXISTENCE OF COOK SEMANTICS

Preprint

kruislaan 413 1098 SJ amsterdam

1980 Mathematics subject classification: 03D20, 68B10

ACM-Computing Reviews-category: 5.24

On the existence of Cook semantics*)

by

A.de Bruin

ABSTRACT

In the literature, for instance in Cook's paper on soundness and completeness of Hoare systems [6], one can find the following technique of defining an operational semantics of a programming language: a function *Comp* is introduced which takes a program R and a state σ and yields a, possibly infinite, row of intermediate states as a result. This row is meant to be the trace resulting from executing program R starting in state σ.

The function *Comp* is characterized by a number of equations. However, these equations are such that it is not immediately clear whether they have a solution. In the above mentioned paper Cook gives some general remarks as to how these equations should be interpreted, but these remarks are not intended as a rigorous definition.

In this paper we show for a simple language, the most sophisticated feature of which is that it has parameterless procedures, that the corresponding equations have a unique solution. We show this first in a straightforward way, and then by defining the solution through an iterative process (using fixed point techniques or a little topology). Furthermore we show that the techniques used here can also be applied to other languages described in the same way, for instance to the language in Cook's paper.

KEY WORDS & PHRASES: *Operational semantics, Cook semantics, fixed points, continuation semantics, recursive definitions, denotational semantics*

---

*) This report will be submitted for publication elsewhere.

1. INTRODUCTION

In this paper we investigate a certain way of defining operational semantics of programming languages, which has become widely known because Cook used it in his soundness and completeness paper [6]. Cook remarks that this semantics has been derived from one of the operational semantics studied in Lauer's thesis [10], and also in Hoare & Lauer [7], a paper which is a condensed version of the thesis. Later on this style of definition has also been employed by de Bakker in his book on the theory of program correctness [3].

The technique is as follows: a meaning function *Comp* is described which takes a program and an initial machine state and yields a row of states as a result. This row gives the trace left by evaluating the program starting in the initial state. A terminating computation yields a finite row, and if evaluation does not terminate then the outcome is an infinite row. The possibility of infinite rows is Cook's extension over the original idea of Lauer. It is a meaningful extension because there are programs which are intended to run forever while having a well defined meaning. One can think of various kinds of real time processes, for instance data base managers and the like. It is not feasible for the meaning function *Comp* to be undefined for such programs which is generally the case for meaning functions which are more oriented towards the final outcome of programs.

For a simple language containing declarations E of parameterless procedures, procedure calls, atomic statements A (for instance assignment statements), conditional statements and composition of statements the function *Comp* would be introduced by the following four equations.

$$\mathit{Comp}(<E|A>)(\sigma) = <\sigma>,$$

where σ is the state resulting from executing the atomic statement A in σ

$$\mathit{Comp}(<E|P>)(\sigma) = <\sigma>^\wedge\ \mathit{Comp}(<E|S>)(\sigma),$$

where the declaration $P \Leftarrow S$ occurs in E.

$$\mathit{Comp}(<E|\underline{\text{if}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2>)(\sigma) = \begin{cases} <\sigma>^\wedge\ \mathit{Comp}(<E|S_1>)(\sigma) & \text{if B is true in } \sigma \\ <\sigma>^\wedge\ \mathit{Comp}(<E|S_2>)(\sigma) & \text{otherwise} \end{cases}$$

$$\mathit{Comp}(<E|S_1;S_2>)(\sigma) = <\sigma>^\wedge\ \mathit{Comp}(<E|S_1>)(\sigma)^\wedge\ \mathit{Comp}(<E|S_2>)(\kappa(\mathit{Comp}(<E|S_1>)(\sigma)))$$

Here $^\wedge$ denotes the concatenation operator, and κ the function which takes a row and yields its last element.

Now there are some questions to be answered. Does there exist a function *Comp* with the above properties? If so, is this function total? And unique? We cannot provide the answers immediately because the above equations can be interpreted as a recursive definiton which is not inductive. That is, there is no complexity measure according to which the arguments of *Comp* in the right hand sides of the equations are simpler than the arguments in the corresponding left hand sides. This is true, because if the definition would be inductive then *Comp* would yield a finite row for all arguments for which is was defined, and this is clearly not the case (evaluate for instance the call P with declaration $P \Leftarrow P$; this yields an infinite row $<\sigma,\sigma,\sigma,\ldots>$ where σ is the initial state of the calculation).

Cook was also aware of these questions as the following quotation from [6] shows:

"The definition is recursive, in the sense that *Comp* appears on the right side of the clauses. This may appear ironic in a paper on program verification, since one of the important issues in programming language semantics is interpreting recursively defined procedures. However, one does not have to understand recursive procedures in general in order to understand this specific definition. Suffice it to say that we intend *Comp* to be evaluated by "call by name", in the sense that occurrences of *Comp* are to be replaced successively by their meanings according to the appropriate clauses in the definition".

In this paper we will show that the answer to the above questions is that there is a unique total function which satisfies the equations. We will show this in four different ways.

The first idea is to derive from the recursive definition an inductive one which defines the elements of the outcome of *Comp* one by one. This is treated in chapter 3. The other techniques are derived from the idea that a solution of the equations can be obtained by iterating an operator derived from the equations, with as initial value a meaning function which is nowhere defined. The first iteration then yields a function which is defined for atomic statements only, that is which is defined for all statements for which we need to use the equations only once to get a final result. The second iteration yields a function which gives the correct result for all atomic statements and all statements of the form $<E|A_1;A_2>$ or $<\ldots,P\Leftarrow A_1,\ldots|P>$, that is for all statements for which we can derive the final outcome by using the equations not more than twice. Repeating this process we generate a sequence of meaning functions that tends to a limit. We then have to prove that this limit is the unique total solution of the equations.

This idea of transforming recursion into iteration is standard in denotational semantics. An ordering is defined on the domains and ranges of our operators which turn these domains into cpo's, and then a variant of Tarski's fixed point theorem is used to get a solution of equations like the ones given above. However, these techniques cannot be applied here straightforwardly. This phenomenon is analyzed in chapter 4. The main obstacle is that the sequence of approximations is not monotonically increasing under the obvious ordening on the domain of rows of states.

We present three ways out of this problem and these are dealt with in chapters 5,6 and 7. The first solution is to add to the domain of rows of states the finite rows marked as "not yet completed". On the thus augmented domain we are able to define an ordering which makes the fixed point theory applicable here. The second idea is to use techniques from continuation semantics: we rewrite the equations in a more generalized form which is such that fixed point techniques can be used straightforwardly, and after that we show that the unique solution thus obtained induces a unique solution of the original equations.

The third way out is to use the fact that the domain of rows has a richer structure than the cpo structure. In fact it can be made in a natural way a complete metric topological space. We then use this structure to show that any sequence of approximations converges to the unique solution of the equations. These results can be found in chapter 7.

Finally, in the last chapter we will show how to extend the results obtained for the paradigm language defined here to definitons of bigger languages. We will indicate for what language constructs the techniques treated here can be used.

2. THE PROBLEM

In this chapter we will define the language under consideration formally, and we will also give the Cook equations for this language in their official form. But first some notational conventions.

Rows will be indicated by angular brackets. For instance we have $<x_1,\ldots,x_n>$ which denotes a finite row of n elements, and $<x_1,x_2,\ldots>$ which denotes an infinite row. The empty row is denoted by $<>$.
Function application associates to the left, that is fabc is an abbreviation of $((f(a))(b))(c)$. Correspondingly, the $\to$ -operator used in forming function domains associates to the right. The above function f should have functionality definition $f\colon A \to B \to C \to D$ which should be read as $f\colon A \to (B\to(C\to D))$.

We next describe the syntax of the language. We distinguish the following syntactic classes:

- $P \in$ *Pvar* *procedure variables*
- $A \in$ *Atst* *Atomic statements*. The structure of these statements is not specified further, but think of assignments.
- $B \in$ *Bexp* *boolean expressions*. These are also considered to be atomic building blocks.
- $R \in$ *Prog* *programs*. These have the form $<E|S>$ and must be *closed*, i.e. all procedure variables in E and S are declared in E.
- $E \in$ *Decl* *declarations*. These have the form $<P_1 \Leftarrow S_1,\ldots,P_n \Leftarrow S_n>$, where all P_i are different.
- $S \in$ *Stat* *statements*. $S ::= A|P|\underline{\text{if}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2|S_1;S_2$.

The conditional statement is included to be able to build nontrivial programs but they have no other significance, i.e. they make the language bigger but not intrinsically more difficult to handle.

We now turn to the semantics. There are the following semantic classes.

- $\sigma \in \Sigma$ *states*.The internal structure of states is not specified. Notice that Σ is a set, not a cpo. There is for instance no such thing as $\perp$ in Σ.
- $\tau \in \Sigma$ *rows of states*.We define $\Sigma^{\infty} = \Sigma^{*} \cup \Sigma^{\omega}$. Σ^{*} contains the finite sequences and the empty row and Σ^{ω} the infinite ones.

We define the following operators on rows of states.

- $\wedge$ *concatenation*, defined by the following axioms:

$$\tau_1 {}^{\wedge} \tau_2 = \tau_1 \text{ for all } \tau_1 \in \Sigma^{\omega}$$

$$< > {}^{\wedge} \tau = \tau {}^{\wedge} < > = \tau$$

$$<\sigma_1,\ldots,\sigma_n> {}^{\wedge} <\sigma_1',\ldots,\sigma_k'> = <\sigma_1,\ldots,\sigma_n,\sigma_1',\ldots,\sigma_k'>$$

$$<\sigma_1,\ldots,\sigma_n> {}^{\wedge} <\sigma_1',\ldots,> = <\sigma_1,\ldots,\sigma_n,\sigma_1',\ldots,>$$

- κ *last element extraction*, defined by

$$\kappa <\sigma_1,\ldots,\sigma_n> = \sigma_n$$

$\kappa < > = \kappa\tau = \bar{\sigma}$ for all $\tau \in \Sigma^{\omega}$, where $\bar{\sigma}$ is an arbitrary (but fixed from now on) element in Σ.

Finally we distinguish the following elementary valuations.

- A: *Atst* $\longrightarrow\Sigma\longrightarrow\Sigma$ *meaning of atomic statements*.Notice that atomic statements always terminate.
- B: *Bexp* $\rightarrow \Sigma \rightarrow \{$ tt,ff$\}$ *meaning of boolean expressions*. As the internal structure of *Atst* and *Bexp* has not been specified, we cannot do more than postulate the existence of functions A and B with functionalities as above.

We now have enough tools to formulate the Cook equations. These equations are intended to define a function *Comp*: *Prog* $\longrightarrow \Sigma \longrightarrow \Sigma^{\infty}$ and are stated below

$Comp\langle E|A\rangle\ \sigma = \langle \mathcal{A}A\sigma\rangle$

$Comp\langle E|P_i\rangle\ \sigma = \langle\sigma\rangle\ ^\wedge\ Comp\langle E|S_i\rangle\ \sigma$ with $P_i \Leftarrow S_i$ in E.

$Comp\langle E|\underline{\text{if}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\rangle\ \sigma =$

$$\begin{cases} \langle\sigma\rangle\ ^\wedge\ Comp\langle E|S_1\rangle\ \sigma, & \text{if } \mathcal{B}B\sigma = tt \\ \langle\sigma\rangle\ ^\wedge\ Comp\langle E|S_2\rangle, & \text{otherwise} \end{cases}$$

$Comp\langle E|S_1;S_2\rangle\ \sigma = \langle\sigma\rangle\ ^\wedge\ \tau\ ^\wedge\ Comp\langle E|S_2\rangle(\kappa\tau),$

where $\tau = Comp\langle E|S_1\rangle\ \sigma$.

In the sequel we will refer to this set of equations as CE, which is an abbreviation of "the Cook equations". We next formulate a lemma which gives information on all total functions satisfying CE. The lemma states that a definition through a set of equations like CE is independent of the particular way we defined $\kappa\tau$ for $\tau = \langle\ \rangle$ or $\tau \in \Sigma^\omega$. This holds because CE is such that in it κ is never applied to $\langle\ \rangle$, and if κ is applied to an element of Σ^ω, then its value is irrelevant because it will be used only to determine a row which is to be appended to an infinite row, which means that it will be neglected.

<u>LEMMA 2.1</u>. *For every total function* Φ in $\mathcal{Prog} \to \Sigma \to \Sigma^\infty$ *which satisfies* CE *the following holds.*

1. *For all* R *and* σ *we have* $\Phi R\sigma \neq \langle\ \rangle$
2. *If we constuct a set of equations* CE' *which is like* CE *except for the fact that it uses another last element extraction function* κ' *which differs from* κ *only when applied to* $\langle\ \rangle$ *or elements from* Σ^ω, *then* Φ is also *a solution of* CE'.

<u>PROOF</u>. Straightforward. □

3. A STRAIGHTFORWARD SOLUTION

The idea is the following. We define a new function C which is like $Comp$ but takes besides R and σ an extra argument, a natural number n, and which yields an element from Σ. This element should then be the n^{th} element of the row $Comp$ Rσ. Now it is possible to give an inductive definition of C.

First of all we have to introduce an extra element Ω ("undefined") because in the set-up as proposed here it is possible to ask for the third element of a row of two elements. In such cases we then deliver Ω.

We define

DEFINITION 3.1. The function $C: \mathcal{Prog} \rightarrow \Sigma \rightarrow \mathbb{N} \rightarrow \Sigma \cup \{\Omega\}$ is defined by induction on n as follows:

$$C\langle E|A\rangle\sigma n = \begin{cases} \mathcal{A}A\sigma, & \text{if } n = 1 \\ \Omega & \text{otherwise} \end{cases}$$

$$C\langle E|P_i\rangle\sigma n = \begin{cases} \sigma & \text{if } n = 1 \\ C\langle E|S_i\rangle\sigma(n-1), & \text{otherwise, where } P_i \Leftarrow S_i \text{ occurs in } E \end{cases}$$

$C\langle E|$ $\underline{\text{if}}$ B $\underline{\text{then}}$ S_1 $\underline{\text{else}}$ $S_2\rangle\sigma n$ =

$$\begin{cases} \sigma & \text{if } n = 1 \\ C\langle E|S_1\rangle\sigma(n-1), & \text{if } n \neq 1 \text{ and } \mathcal{B}B\sigma = tt \\ C\langle E|S_2\rangle\sigma(n-1), & \text{otherwise} \end{cases}$$

$C\langle E|S_1;S_2\rangle\sigma n$ =

$$\begin{cases} \sigma, & \text{if } n = 1 \\ C\langle E|S_1\rangle\sigma(n-1), & \\ C\langle E|S_2\rangle(C\langle E|S_1\rangle\sigma k)(n-k-1), & \text{if } n \neq 1 \text{ and } C\langle E|S_1\rangle\sigma(n-1) \neq \Omega \\ \quad \text{and } V := \{m | C\langle E|S_1\rangle\sigma m \neq \Omega \wedge C\langle E|S_1\rangle\sigma(m+1) = \Omega \wedge m < n\} \neq \emptyset & \\ \quad \text{where } k = \min V & \\ \Omega, \quad \text{otherwise} & \end{cases}$$

This definition is adward, especially the case $\langle E|S_1;S_2\rangle$. We can simplify this clause by applying the next two lemma's.

LEMMA 4.3. $\forall R,\sigma$: $CR\sigma 1 \neq \Omega$.

PROOF. Immediate. □

LEMMA 3.3. $\forall$ R,σ,n: $CR\sigma n = \Omega \Rightarrow \forall\ k \geq n$: $CR\sigma k = \Omega$.

PROOF. Induction on n. For instance, take $R \equiv \langle E|S_1;S_2\rangle$.

If $CR\sigma n = \Omega$ then a fortiori $C\langle E|S_1\rangle\sigma n = \Omega$. If we combine this with Lemma 3.2 we obtain that the set V as defined in the last clause of definition 3.1 is not empty. So the second case in the definition of $C\langle E|S_1;S_2\rangle\sigma n$ applies and we have $CR\sigma n = C\langle E|S_2\rangle\sigma(n-m-1)$ for some m with $1\le m<n-1$ Therefore $C\langle E|S_2\rangle\sigma(n-m-1) = \Omega$.
Now we can use the same argument to show that for all $k\ge n$ we have $CR\sigma k = C\langle E|S_2\rangle\sigma(k-m-1)$. Therefore an application of the induction hypothesis yields the desired result. □

COROLLARY 3.4 *For all* R *and* n *we have that either* $CR\sigma n \ne \Omega$ *or that there exists exactly one* k *with* $1\le k<n$ *such that* $CR\sigma k \ne \Omega$ *and* $CR\sigma(k+1) = \Omega$. *Moreover in the latter case we have* $\forall m\le k: CR\sigma m \ne \Omega$ *and* $\forall m>k: CR\sigma m = \Omega$.

LEMMA 3.5.

$$C\langle E|S_1;S_2\rangle\sigma = \begin{cases} \sigma, & \text{if } n = 1 \\ C\langle E|S_1\rangle\sigma(n-1), & \textit{if } C\langle E|S_1\rangle\sigma(n-1) \ne \Omega \textit{ and } n \ne 1 \\ C\langle E|S_2\rangle(C\langle E|S_1\rangle\sigma k)(n-k-1), & \textit{otherwise} \\ \quad \textit{where } k \textit{ is such that } C\langle E|S_1\rangle\sigma k \ne \Omega \text{ and} \\ \quad C\langle E|S_1\rangle\sigma(k+1) = \Omega \end{cases}$$

PROOF. Immediate from the above Corollary. □

Next we will use C to define a function *Comp* satisfying CE.

DEFINITION 3.6.

$$\mathit{Comp}R\sigma = \begin{cases} \langle CR\sigma k\rangle_{k=1}^{n}, & \text{if } CR\sigma n \ne \Omega \text{ and } CR\sigma(n+1) = \Omega \\ \langle CR\sigma k\rangle_{k=1}^{\infty}, & \text{otherwise} \end{cases}$$

THEOREM 3.7.*Comp as defined in* 3.6 *satisfies* CE.

PROOF. We consider only the case $\mathit{Comp}R\sigma$ for $R \equiv \langle E|S_1;S_2\rangle$. There are two subcases.

a). $\mathit{Comp}\langle E|S_1\rangle\sigma$ is infinite. In that case we have $\forall n: C\langle E|S_1\rangle\sigma n \ne \Omega$ and therefore $\forall n>1: CR\sigma n = C\langle E|S_1\rangle\sigma(n-1)$, which means $\mathit{Comp}R\sigma = \langle\sigma\rangle \wedge \mathit{Comp}\langle E|S_1\rangle\sigma$.

On the other hand, we have $<\sigma> \wedge Comp<E|S_1>\sigma \wedge$

$$Comp<E|S_2> (\kappa(Comp<E|S_1>\sigma)) = <\sigma> \wedge Comp<E|S_1>\sigma,$$

because $Comp<E|S_1>\sigma$ is finite.

b). $Comp<E|S_1>\sigma$ is finite, say with length k. Then we have $\sigma' := C<E|S_1>\sigma k \neq \Omega$ and $C<E|S_1>\sigma(k+1) = \Omega$. Moreover $\kappa(Comp<E|S_1>\sigma) = \sigma'$. We thus have

$$CR\sigma n = \begin{cases} \sigma, & \text{for } n = 1 \\ C<E|S_1>\sigma(n-1), & \text{for } 1<n\leq k+1 \\ C<E|S_2>\sigma'(n-k-1), & \text{for } n>k+1 \end{cases}$$

and therefore $CompR\sigma = <\sigma> \wedge Comp<E|S_1>\sigma \wedge Comp<E|S_2>\sigma'$. □

THEOREM 3.8. *There is exactly one total function satisfying* CE.

PROOF. For any function *Comp* satisfying CE and for any R,σ and n we can calculate, using only the clauses from CE, the n^{th} element from the row *Comp*Rσ, like we have done in definition 3.1. So we have that the equations CE determine, for every R and σ, every element from the row *Comp*Rσ, that is this row must be unique, that is *Comp* must be unique.
Note that the above reasoning would no longer be valid if we allowed partial functions in $Prog \to \Sigma \to \Sigma^\infty$ to be solutions of CE. □

4. THERE IS A PROBLEM IF WE TRY TO USE THE FIXED POINT APPROACH

It is tempting to try to use fixed point theory to answer the questions raised in chapter 1, because any solution of CE will be a fixed point of the operator $\Psi: D \to D$, where $D = Prog \to \Sigma \to \Sigma^\infty$, defined by

$$\begin{aligned} \Psi = \lambda\Phi.\lambda R.\lambda\sigma.\ & R \equiv <E|A> \to <\mathcal{A} A\sigma>, \\ & R \equiv <E|P_i> \to <\sigma> \wedge \Phi<E|S_i>\sigma, \\ & R \equiv <E|\underline{\text{if}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2> \to \\ & \quad (\mathcal{B}B\sigma = tt \to <\sigma> \wedge \Phi<E|S_1>\sigma> \wedge \Phi<E|S_2>\sigma), \\ & R \equiv <E|S_1;S_2> \to <\sigma> \wedge \Phi<E|S_2>(\kappa(\Phi<E|S_1>\sigma)) \end{aligned}$$

Now it is a well known fact from denotational semantics (see for instance [12] or [3] which both give an introduction to the subject) that Ψ has a least fixed point μΨ if this operator is continuous, and in that case $\mu\Psi$ equals the lub of the chain $\bot \sqsubseteq \Psi\bot \sqsubseteq \Psi(\Psi\bot) \sqsubseteq \Psi(\Psi\bot)) \sqsubseteq \ldots.$

So, if we manage to make D a cpo such that Ψ is continuous then we obtain the required existence result immediately. Again, it is well known that D is a cpo, if there is an ordening $\sqsubseteq$ on Σ^∞ which makes this set a cpo. Now the intuitive meaning of $\tau_1 \sqsubseteq \tau_2$ is that τ_2 contains more information than τ_1, or that τ_2 is a better approximation of some final result than τ_1. A technique for turning a set into a cpo that is often used, is to make this set a *flat cpo*, that is to add a totally undefined element $\bot$ to it which is smaller than all elements while all other elements are incomparable by definition.

However, if we investigate whether this construction is suited for our purposes, we find that this is not the case. More specifically, we arrive at a least fixed point which yields the right result for terminating processes, but which yields $\bot$ for nonterminating processes. In order to illustrate what the reason of this phenomenon is, we will evaluate some elements of the chain $\bot \sqsubset \Psi\bot \sqsubset \Psi^2\bot \sqsubset \ldots$ approximating $\mu\Psi$, applied to the program $<P \Leftarrow P|P>$:

1. $\bot <P \Leftarrow P|P> \sigma = \bot$
2. $(\Psi\bot)<P \Leftarrow P|P>\sigma = <\sigma>^\wedge \bot< P \Leftarrow P|P>\sigma = <\sigma>^\wedge \bot = \bot$

   (NB. We do not have elements of the form $<\sigma, \bot>$ in the flat cpo derived from Σ^∞, but it seems reasonable to make $<\sigma,\bot> = \bot$)
3. $(\Psi^2\bot) < P \Leftarrow P|P > \sigma = <\sigma>^\wedge(\Psi\bot) < P \Leftarrow P|P > \sigma =$

   $<\sigma>^\wedge <\sigma>^\wedge \bot< P \Leftarrow P|P>\sigma = <\sigma>^\wedge <\sigma>^\wedge \bot = \bot$
4. $(\Psi^3\bot) < P \Leftarrow P|P > \sigma = <\sigma>^\wedge(\Psi^2\bot) < P \Leftarrow P|P > \sigma =$

   $<\sigma>^\wedge<\sigma>^\wedge<\sigma>^\wedge \bot <P \Leftarrow P|P > \sigma = <\sigma>^\wedge <\sigma>^\wedge <\sigma>^\wedge \bot = \bot.$

etc.

These formulae clearly show what is wrong here. The ordering $\sqsubseteq$ is not refined enough, we were forced to make $<\sigma>^\wedge \bot$ equal to $\bot$. Notice that making $<\sigma>^\wedge \bot$ equal to $<\sigma>$ does not work either, because we would then get as results in 1,2,3 and 4 respectively $\bot, <\sigma>, <\sigma, \sigma>$ and $<\sigma,\sigma,\sigma>$, and these elements do not form a chain in the flat cpo derived from Σ^∞. In a flat cpo we have always the situation that an approximation τ_1 of a final answer $\tau(\tau_1 \sqsubseteq \tau)$ contains either all information ($\tau_1 = \tau$) or no information at all($\tau_1 = \bot$). Now because all finite approximations of an infinite row are necessarily unequal to this row we must have that all these approximations are equal to $\perp$, that is we get a chain $\perp \sqsubseteq \perp \sqsubseteq \ldots$ with lub $\perp$, and this is not what we want.

This analysis also shows a way out. What the sequence of approximation given above should do is yield longer and longer initial segments of the final outcome. That is, we should have an ordening such that $<\sigma> \sqsubseteq <\sigma,\sigma> \sqsubseteq <\sigma,\sigma,\sigma> \sqsubseteq \ldots$ is a chain with the natural lub $<\sigma,\sigma,\sigma,\ldots>$. This leads us to trying the prefix ordening on Σ^{∞}: $\tau_1 \sqsubseteq \tau_2$ iff τ_1 is a prefix of τ_2. One easily checks that Σ^{∞} with this ordening is a cpo with the empty row $< >$ as bottom element.

This ordening yields a correct approximation sequence for the program $<P \Leftarrow P|P>$ as one easily can check. However this approach does not work in general because Ψ is not continuous under this ordening. This stems from the fact that the operators κ and $^\wedge$ are not continuous, not even monotonic under the prefix ordening. For instance, $\kappa<\sigma_1> = \sigma_1$ and $\kappa<\sigma_1,\sigma_2> = \sigma_2$. Now we have $<\sigma_1> \sqsubseteq <\sigma_1,\sigma_2>$, but σ_1 and σ_2 might very well be incomparable.

We can also show that the new approach does not work in a less technical way. Let us consider the sequence $<\perp\ (= \lambda R.\lambda\sigma.<>),\ \Psi\perp, \Psi^2\perp, \ldots>$ and let us apply some of the elements thereof to the program $R \equiv <E|P;A_2>$ where $E \equiv <P \Leftarrow A_1>$ and to an initial state σ. We then get the following results.

1. $\perp R\sigma = (\lambda\sigma.< >)R\sigma = < >$
2. $(\Psi\perp)R\sigma = <\sigma>^\wedge\ (\perp<E|P>\sigma)^\wedge\ (\perp<E|A_2>\ [\kappa(\perp<E|P>\sigma)]) = <\sigma>^\wedge< >^\wedge< > = <\sigma>$
3. $(\Psi^2\perp)R\sigma = \tau^\wedge\ (\Psi\perp)<E|A_2>(\kappa\tau)$,

   where $\tau = <\sigma>^\wedge((\Psi\perp)<E|P>\sigma) = <\sigma>^\wedge<\sigma>^\wedge(\perp<E|A_1>\sigma) = <\sigma,\sigma>$

   therefore $(\Psi^2\perp)R\sigma = <\sigma,\sigma>^\wedge(\Psi\perp)<E|A_2>\sigma = <\sigma,\sigma>^\wedge<AA_2\sigma> = <\sigma,\sigma,AA_2\sigma>$
4. $(\Psi^3\perp)R\sigma = \tau^\wedge(\Psi^2\perp)<E|A_2>(\kappa\tau)$,

   where $\tau = <\sigma>^\wedge((\Psi^2\perp)<E|P>\sigma) = <\sigma>^\wedge<\sigma>^\wedge(\Psi\perp)<E|A_1>\sigma =$
   $= <\sigma>^\wedge<\sigma>^\wedge<AA_1\sigma> = <\sigma,\sigma,AA_1\sigma>$

   therefore $(\Psi^3\perp)R\sigma = <\sigma,\sigma,AA_1\sigma>^\wedge(\Psi^2\perp)<E|A_2>(AA_1\sigma) =$
   $= <\sigma,\sigma,AA_1\sigma>^\wedge<AA_2(AA_1\sigma)> =$
   $= <\sigma,\sigma,AA_1\sigma,AA_2(AA_1\sigma)>$.

Now from these calculations we see that $\Psi^2\bot \not\sqsubseteq \Psi^3\bot$ because we have $(\Psi^2\bot)R\sigma \neq (\Psi^3\bot)R\sigma$. Therefore the prefix ordering on Σ^∞ is such that the sequence $< \bot, \Psi\bot, \Psi^2\bot, \ldots >$ is not a chain, and thus Ψ cannot be continuous.

If we analyse what went wrong here, we see that in evaluating $(\Psi^2\bot)R\sigma$ we apply the last element function κ to a row of states which is not yet finished, that is we start evaluating A_2 "too early", namely in state σ which is not the final state resulting from evaluation of P. This observation suggests two solutions for the difficulty we have met. The first one is to enlarge Σ^∞ so that it contains also row of states which are marked as "not yet completed" and to let the operators $^\wedge$ and κ act in a continuous manner on these rows. Another possibility is to rewrite Ψ in such a way that it does not use the non continuous operators κ and $^\wedge$ any more. Finally, though the above approximation sequence is not a chain, we observe that the right outcome has been obtained in the end. This suggests that the function Ψ might be continuous if we would use a more subtle notion of continuity. The next three chapters will be devoted to a discussion of these possibilities.

5. ADDING UNFINISHED ROWS TO Σ^∞.

We saw above that in Σ^∞ finished and unfinished rows of states must be distinguished. We will arrange this as follows: a row $<\sigma_1, \ldots, \sigma_n>$ will be marked unfinished by adding the element $\bot$ to it, so that we get $<\sigma_1, \ldots, \sigma_n, \bot>$. Notice that only finite rows can possibly be unfinished; infinite rows, which model nonterminating computations, cannot contain more information than they already do. All this leads to the following definition.

<u>DEFINITION 5.1</u> $\Sigma^\sim = \Sigma^* \cup \Sigma^{*\bot} \cup \Sigma^\infty$, where Σ^* and Σ^∞ are as before, and where $\Sigma^{*\bot}$ is the set of all rows consisting of zero or more states followed by the symbol $\bot$.

Thinking in terms of the analysis in the preceding chapter, we see that the following ordering is natural.

DEFINITION 5.2. For $\tau_1, \tau_2 \in \Sigma^\sim$ we define $\tau_1 \sqsubseteq \tau_2$ iff either $\tau_1 = \tau_2$ or $\tau_1 = <\sigma_1, \ldots, \sigma_n, \perp> \in \Sigma^{*\perp}$ and $<\sigma_1, \ldots, \sigma_n>$ is a prefix of τ_2.

LEMMA 5.3. *$\Sigma^\sim$ with the above ordening is a cpo.*

PROOF. One easily checks reflexivity, anti-symmetry and transitivity of $\sqsubseteq$. One also checks immediately that $<\perp>$ is the smallest element in $\Sigma^\sim$, and lastly one can show that every chain in $\Sigma^\sim$ has a lub by noting that all nontrivial chains in $\Sigma^\sim$ must have the form $\tau_1{}^\wedge <\perp> \sqsubseteq \tau_2{}^\wedge<\perp> \sqsubseteq \tau_3{}^\wedge<\perp> \sqsubseteq \ldots$ where τ_i is a prefix of τ_{i+1} (a nontrivial chain is a chain $<\tau_i>_i$ for which $\forall i\ \exists k : \tau_i \neq \tau_{i+k}$). But a chain of this form has an obvious lub in Σ^ω, namely the infinite row which has every element of the chain as a prefix. □

The next thing to do is to define κ and $^\wedge$ on this new domain. The analysis in chapter 4 indicates that the operator $^\wedge$ should disregard its second argument, if its first argument is a row that is not yet completed: $\tau_1 {}^\wedge \tau_2 = \tau_1$ for τ_1 in $\Sigma^{*\perp}$. Because we want $\kappa: \Sigma^\sim \to \Sigma$ to be continuous we first have to make Σ a cpo.

DEFINITION 5.4.

1. $\Sigma_\perp = \Sigma \cup \{\perp\}$, the flat cpo derived from Σ.
2. $\kappa: \Sigma^\sim \to \Sigma_\perp$ is defined by

$$\kappa(\tau) = \begin{cases} \perp, & \text{if } \tau \in \Sigma^\infty,\ \tau \in \Sigma^{*\perp} \text{ or } \tau = <> \\ \sigma_n, & \text{if } \tau = <\sigma_1, \ldots, \sigma_n> \in \Sigma^* \setminus \{<>\} \end{cases}$$

3. $^\wedge : \Sigma^\sim \times \Sigma^\sim \to \Sigma^\sim$ is defined by

$$\tau_1 {}^\wedge \tau_2 = \begin{cases} \tau_1, & \text{if } \tau_1 \in \Sigma^\infty \cup \Sigma^{*\perp} \\ <\sigma_1, \ldots, \sigma_n, \sigma_1', \ldots (,\sigma_k')>, & \text{if } \tau_1 \in \Sigma^* \end{cases}$$

LEMMA 5.5. *κ and $^\wedge$ as defined in 5.4. are continuous.*

PROOF. Straightforward, use the remark in the proof of 5.3.1 about non-trivial chains. Note that in order to make κ continuous we cannot define $\kappa(\tau)$ to be arbitrary for $\tau \in \Sigma^\infty$ or $\tau = <>$, as we did in chapter 2. □

Now that we have added the element $\perp$ to Σ we have to change the definition of Ψ a little bit.

DEFINITION 5.6. $\Psi: D \to D$, where $D = \mathit{Prog} \to \Sigma_\perp \to_s \tilde{\Sigma}$ is defined by

$$\begin{aligned}\Psi = \lambda\Phi.\lambda R.\lambda\sigma.\ & \sigma = \perp \to \langle\perp\rangle,\\ & R \equiv \langle E|A\rangle \to \langle \mathcal{A}A\sigma\rangle\\ & R \equiv \langle E|P_i\rangle \to \langle\sigma\rangle \mathbin{\hat{}}\ \Phi\langle E|S_i\rangle\sigma,\\ & R \equiv \langle E|\ \underline{\text{if}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\rangle \to\\ & \quad (\mathcal{B}B\sigma = tt \to \langle\sigma\rangle \mathbin{\hat{}}\ \Phi\langle E|S_1\rangle\sigma,\ \langle\sigma\rangle \mathbin{\hat{}}\ \Phi\ \langle E|S_2\rangle\sigma),\\ & R \equiv \langle E|S_1;S_2\rangle \to \langle\sigma\rangle \mathbin{\hat{}}\ \Phi\langle E|S_1\rangle\sigma \mathbin{\hat{}}\ \Phi\langle E|S_2\rangle\ (\kappa(\Phi\langle E|S_1\rangle\sigma)).\end{aligned}$$

REMARKS.

1. The expression $\Sigma_\perp \to_s \tilde{\Sigma}$ denotes the cpo of all *strict* functions from $\Sigma_\perp$ to $\tilde{\Sigma}$, that is all functions f for which $f\perp = \langle\perp\rangle$. This precaution is needed because otherwise Ψ would not be continuous.
2. One easily checks that the operator Ψ has the functionality as announced, that is $\forall\ \Phi \in \mathit{Prog} \to \Sigma_\perp \to_s \tilde{\Sigma}$, $R \in \mathit{Prog}$ and $\sigma \in \Sigma_\perp$, we have that $\Psi\Phi R\sigma \in \tilde{\Sigma}$ (e.g. only the last element might be $\perp$), and also $\forall\Phi\in D, R\in\mathit{Prog}$: $\Psi\Phi R\perp = \langle\perp\rangle$ (i.e. $\Psi\Phi R$ is strict again).

We will now prove that firstly Ψ as defined above is continuous, and secondly that for all R and $\sigma \neq \perp$ we have that $(\mu\Psi)R\sigma \in \Sigma^\infty$. This second fact then implies that $\mu\Psi$ restricted to the proper domain is a solution of CE, because the operators $\hat{}$ and κ, as defined in 5.4, restricted to Σ^∞ are the same as those defined in chapter 2. The second result will also be used to show that $\mu\Psi$ is the only solution of CE.

LEMMA 5.7. *Ψ is a continuous operator in* $D \to D$, *where* D *is as in definition* 5.6.

PROOF. Straightforward. □

The next fact to check is that for all R and $\sigma \neq \perp$ we have that $(\mu\Psi)R\ \sigma \in \Sigma^\infty$

The proof proceeds as follows. Suppose the assertion is not true. We then would have some R and $\sigma \neq \bot$ for which $(\mu\Psi)R\sigma \in \Sigma^{*\bot}$. Now $(\mu\Psi)R\sigma =$ $= \bigsqcup_i((\Psi^i\bot)R\sigma)$ and therefore we would have that for all i: $\tau_i := (\Psi^i\bot)R\sigma \in \Sigma^{*\bot}$. Now intuitively $\tau_i \in \Sigma^{*\bot}$ means that this approximation of evaluation of R in σ is not good enough, because this row is not yet completed. This suggests that there is a better approximation in the chain $<(\Psi^i\bot)R\sigma>_i$, and in fact this holds already for the next element in the chain: we have $\tau_i \in \Sigma^{*\bot} \Rightarrow \tau_{i+1} \neq \tau_i$. This is Lemma 5.8.

Now if Lemma 5.8 holds we then would have the following situation $(\mu\Psi)R\sigma$ is the lub of an apparently nontrivial chain, $\bot R\sigma \sqsubseteq (\Psi\bot)R\sigma \sqsubseteq \ldots$ with all $(\Psi^k\bot)R\sigma \in \Sigma^{*\bot}$. And now we have reached a contradiction, for such a chain will have a lub in Σ^∞.

<u>LEMMA 5.8</u>. *Let* $\mu\Psi = \bigsqcup_i \Phi_i$ *with* $\Phi_i = \Psi^i\bot$. *For all* R *and all* $\sigma \neq \bot$ *we have*:

$$\Phi_i R\sigma \in \Sigma^{*\bot} \Rightarrow \Phi_{i+1}R\sigma \neq \Phi_i R\sigma.$$

<u>PROOF</u>. Straightforward by induction on i. □

<u>LEMMA 5.9</u>. $\forall$ R,$\sigma \neq \bot$: $(\mu\Psi)R\sigma \in \Sigma^\infty$

<u>PROOF</u>. Cf. the remarks preceding Lemma 5.8 □

<u>THEOREM 5.10</u>. $\mu\Psi$, *restricted to the domain* *Prog* $\to \Sigma \to \Sigma^\infty$, *is a solution of* CE.

<u>PROOF</u>. Notice that we cannot state that $\mu\Psi$ is a solution of CE, because $\mu\Psi$ is an element of *Prog* $\to \Sigma_\bot \to \Sigma^\sim$ and as such it can never be a solution of CE. Notice also that we can restrict $\mu\Psi$ to the domain *Prog* $\to \Sigma \to \Sigma^\infty$ only by virtue of Lemma 5.9.

Now, to prove the theorem, we first compare the definition of κ and $^\wedge$ from chapter 2 with the ones in Definition 5.4 and we find that the restriction of $^\wedge$(according to 5.4.3) to $\Sigma^\infty \times \Sigma^\infty$ is the same operator as the one in chapter 2, while the restriction of κ to Σ^∞ is almost the same, the only differences being the cases $\kappa\tau$ where $\tau \in \Sigma^\omega$ or $\tau = <>$. If these operators would be the same then we were finished, because in Definition

5.6 κ and $\wedge$ are applied only to arguments of the form $(\mu\Psi)R\sigma$ and these are in Σ^{∞} by Lemma 5.9.

However the fixed points of Ψ have the same properties as the ones given by Lemma 2.1 for the solutions of CE. □

THEOREM 5.11. $\mu\Psi$, *restricted to* $\mathit{Prog} \to \Sigma \to \Sigma^{\infty}$, *is the only solution of* CE.

PROOF. We first prove that $\mu\Psi$ is the only fixed point of Ψ. Suppose not, then there would be a bigger fixed point Φ, that is there would be an R and σ such that $(\mu\Psi)R\sigma \sqsubsetneq \Phi R\sigma$. This is impossible however because by Lemma 5.9 $(\mu\Psi)R\sigma \in \Sigma^{\infty}$ which means that $(\mu\Psi)R\sigma$ is a maximal element in $\Sigma^{\sim}$ (there is no τ in $\Sigma^{\sim}$ which is properly bigger than $(\mu\Psi)R\sigma$).

Now suppose there would be another function C: $\mathit{Prog} \to \Sigma \to \Sigma^{\infty}$ satisfying CE. We can extend this function C to a function C': $\mathit{Prog} \to \Sigma_{\perp} \to_{s} \Sigma^{\sim}$ by defining $C'R\sigma = CR\sigma$ if $\sigma \in \Sigma$ and $\langle\perp\rangle$ if $\sigma = \perp$. One easily checks that C' is a fixed point of Ψ, but then $C' = \mu\Psi$. Contradiction. □

6. THE CONTINUATION APPROACH

In chapter 4 we remarked that the direct fixed point approach failed due to the fact that the operators κ and $\wedge$ are not continuous. In this chapter we will try to find a way out of this problem by restructuring CE in such a way that these operators are not used any more, or at least not in a non continuous way. The problem stems from the clause on constructs of the form $\langle E|S_1;S_2\rangle$. The idea that we will pursue is to use continuation semantics instead of direct semantics.

Direct semantics defines the meaning of a construct in terms of the rows of states that correspond to evaluation of the construct. Therefore the operators κ and $\wedge$ have to be used: the meaning of $\langle E|S_1;S_2\rangle$ is obtained by concatenating the rows of states corresponding to the meanings of $\langle E|S_1\rangle$ and $\langle E|S_2\rangle$. Construction semantics uses another idea: the meaning of a construct is the row of states which is the result of evaluating the construct itself followed by evaluation of the rest of the program of which the construct is supposed to be a part. Of course, the effect of evaluation of the rest of the program cannot be obtained from the construct itself, so we have to give the meaning function *Comp* another argument, a *continuation* which will be a function from states to rows of states describing the effect of the rest of the program. One can view this continuation as a coding of the row of statements which are to be evaluated once the statement under consideration has been worked through.

In this set up we do not have to concatenate two rows any more while defining the meaning of $<E|S_1;S_2>$, because the effect of evaluating S_2 can be caught by changing the continuation which describes what will happen once the whole construct has been evaluated into a continuation which describes the effect of first evaluating S_2 and then applying the original continuation. Then this new formed continuation is given as an argument to $\mathit{Comp}<E|S_1>$. All this leads to the following four equations, denoted by "CE_{cont}", which are intended to define a meaning function Comp: $\mathit{Prog} \to \Theta \to \Theta$, with $\Theta = \Sigma \to \Sigma^{\infty}$.

$$\mathit{Comp}<E|A>\theta\sigma = <\mathcal{A}A\sigma> \wedge \theta(\mathcal{A}A\sigma)$$

$$\mathit{Comp}<E|P_i>\theta\sigma = <\sigma> \wedge \mathit{Comp}<E|S_i>\theta\sigma$$

$$\mathit{Comp}<E|\underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2>\theta\sigma =$$

$$\begin{cases} <\sigma> \wedge \mathit{Comp}<E|S_1>\theta\sigma, & \text{if } \mathcal{B}B\sigma = tt \\ <\sigma> \wedge \mathit{Comp}<E|S_2>\theta\sigma, & \text{if otherwise} \end{cases}$$

$$\mathit{Comp}<E|S_1;S_2>\theta\sigma = <\sigma> \wedge \mathit{Comp}<E|S_1>\left\{\mathit{Comp}<E|S_2>\theta\right\}\sigma$$

Notice that the operator κ is not used any more. We do use the concatenation operator, but only in a continuous way (with respect to the prefix ordering on Σ^{∞}): $\lambda\sigma.\lambda\tau.<\sigma>^{\wedge}\tau$ is continuous. The fourth clause in the above equation can be phrased as follows: evaluating $<E|S_1;S_2>$ followed by evaluation according to θ amounts to evaluation of $<E|S_1>$ followed by [evaluation of $<E|S_2>$ followed by evaluating according to θ].

The next thing to do is to derive from CE_{cont} an operator Ψ, show that this operator is continuous so that the existence of a fixed point is guaranteed. Straightforward reasoning would lead to define the functionality of Ψ as $\Psi: D \longrightarrow D$, where $D = \mathit{Prog} \longrightarrow \Theta \longrightarrow \Theta$, with $\Theta = \Sigma \longrightarrow \Sigma^{\infty}$. However we must take a precaution here: we have to make the domain D equal to the domain of all functions from *Prog* to all *continuous* functions from Θ to Θ: $D = \mathit{Prog} \longrightarrow [\Theta \longrightarrow \Theta]$. This is needed because we can only prove continuity of Ψ if the functions Φ involved are continuous in their continuations θ.

Having remarked all this, we are now ready for the definition of Ψ.

DEFINITION 6.1. The operator $\Psi : D \to D$, with $D = \mathit{Prog} \to [\Theta \to \Theta]$ and $\Theta = \Sigma \to \Sigma^{\infty}$ is defined by

$$\begin{aligned}\Psi = \lambda\Phi.\lambda R.\lambda\theta.\lambda\sigma.\ & R \equiv \langle E|A\rangle \to \langle \mathcal{A}A\sigma\rangle^{\wedge}\ \theta(\mathcal{A}A\sigma),\\ & R \equiv \langle E|P_i\rangle \to \langle\sigma\rangle^{\wedge}\ \Phi\langle E|S_i\rangle\theta\sigma,\\ & R \equiv \langle E|\underline{\text{if}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\rangle \to\\ & \qquad (\mathcal{B}B\sigma = tt \to \langle\sigma\rangle^{\wedge}\ \Phi\langle E|S_1\rangle\theta\sigma, \langle\sigma\rangle^{\wedge}\ \Phi\langle E|S_2\rangle\theta\sigma),\\ & R \equiv \langle E|S_1;S_2\rangle \to \langle\sigma\rangle^{\wedge}\ \Phi\langle E|S_1\rangle[\Phi\langle E|S_2\rangle\theta]\sigma\end{aligned}$$

LEMMA 6.2.

1. *Ψ is well defined, in the sense that for all $\Phi \in D$ we have $\Psi\Phi \in D$, or in other words:* $\forall\Phi\in D\ \forall R\in\mathit{Prog}\ \forall\theta_1 \sqsubseteq \theta_2 \sqsubseteq \ldots : \Psi\ \Phi\ R(\bigsqcup_i \theta_i) = \bigsqcup_i \Psi\ \Phi\ R\ \theta_i$
2. *Ψ is continuous.*

PROOF. Straightforward. □

We will also need the following Lemma.

LEMMA 6.3. *Let $\Phi \in \mathit{Prog} \to [\Theta \to \Theta]$ be a fixed point of Ψ. Then for all* R, *and σ we have that* $\Phi R\theta\sigma \neq \langle\,\rangle$

PROOF. Immediate. □

By way of an example we will show that the counterexample given in chapter 4 is now handled correctly. We again apply the first four approximations of $\mu\Psi$ to the program $R \equiv \langle P \Leftarrow A_1 | P;A_2\rangle$ starting in state σ. Because we are interested in evaluation of this program only, we give $(\mu\Psi)R$ as a continuation argument the empty continuation $\lambda\sigma.\langle\,\rangle$.

1. $\bot\ R\{\lambda\sigma.\langle\,\rangle\}\sigma = \langle\,\rangle \qquad (\bot = \lambda R.\lambda\theta.\lambda\sigma.\langle\,\rangle)$
2. $(\Psi\bot)R\{\lambda\sigma.\langle\,\rangle\} = \langle\sigma\rangle\ ^{\wedge}\ \bot\ \langle E|P\rangle\{\bot\langle E|A_2\rangle\{\lambda\sigma.\langle\,\rangle\}\}\sigma = \langle\sigma\rangle^{\wedge}\ \langle\,\rangle\ = \langle\sigma\rangle$
3. $(\Psi^2\bot)R\{\lambda\sigma.\langle\,\rangle\}\sigma\ = \langle\sigma\rangle\ ^{\wedge}\ (\Psi\bot)\ \langle\ E|P\rangle\{(\Psi\bot)\langle E|A_2\rangle\{\lambda\sigma.\langle\,\rangle\}\}\sigma =$
   $= \langle\sigma\rangle\ ^{\wedge}\ \langle\sigma\rangle\ ^{\wedge}\ \bot\ \langle E|A_1\rangle\{(\Psi\bot)\langle E|A_2\rangle\{\lambda\sigma.\langle\,\rangle\}\}\sigma =$
   $= \langle\sigma\rangle\ ^{\wedge}\ \langle\sigma\rangle\ ^{\wedge}\ \langle\,\rangle\ = \langle\sigma,\sigma\rangle.$
4. $(\Psi^3\bot)R\{\lambda\sigma.\langle\,\rangle\}\sigma = \langle\sigma\rangle\ ^{\wedge}\ (\Psi^2\bot)\langle E|P\rangle\{(\Psi^2\bot)\langle E|A_2\rangle\{\lambda\sigma.\langle\,\rangle\}\}\sigma =$
   $= \langle\sigma\rangle\ ^{\wedge}\ \langle\sigma\rangle\ ^{\wedge}\ (\Psi\bot)\langle E|A_1\rangle\{(\Psi^2\bot)\langle E|A_2\rangle\{\lambda\sigma.\langle\,\rangle\}\}\sigma$
   $= \langle\sigma\rangle\ ^{\wedge}\ \langle\sigma\rangle\ ^{\wedge}\ \langle\mathcal{A}A_1\sigma\rangle\ ^{\wedge}\ (\Psi^2\bot)\langle E|A_2\rangle\{\lambda\sigma.\langle\,\rangle\}(\mathcal{A}A_1\sigma) =$

$$= \langle\sigma\rangle \,^\wedge \langle\sigma\rangle \,^\wedge \langle \mathcal{A} A_1\sigma\rangle \,^\wedge \langle \mathcal{A} A_2(\mathcal{A} A_1\ \sigma)\rangle \,^\wedge \{\lambda\sigma.\langle\ \rangle\}(\mathcal{A}A_2\ (\mathcal{A} A_1\ \sigma)) =$$
$$= \langle\sigma\rangle \,^\wedge \langle\sigma\rangle \,^\wedge \langle \mathcal{A} A_1\sigma\rangle \,^\wedge \langle \mathcal{A} A_2\ (\mathcal{A} A_1\ \sigma)\rangle \,^\wedge \langle\ \rangle =$$
$$= \langle\sigma,\sigma,\mathcal{A} A_1\sigma,\mathcal{A} A_2(\mathcal{A} A_1\sigma)\rangle.$$

We can now define *Comp* as $\mu\Psi$ applied to $\lambda\sigma.\langle\ \rangle$ as standard continuation parameter.

DEFINITION 6.4. $\mathit{Comp} = \lambda R.\lambda\sigma.(\mu\Psi)R\{\lambda\sigma.\langle\ \rangle\}\sigma$

The next thing to prove is that the function *Comp* thus defined is a solution of CE. The proof is by cases, and the cases that *Comp* is applied to an atomic statement, a conditional statement, or a procedure call are straightforward. The interesting case is to prove that

$$\mathit{Comp}\ \langle E|S_1;S_2\rangle\ \sigma = \langle\sigma\rangle \,^\wedge \mathit{Comp}\ \langle E|S_1\rangle\sigma \,^\wedge \mathit{Comp}\ \langle E|S_2\rangle\ \sigma' \qquad \ldots(+)$$

with σ'as usual.

Now $\mathit{Comp}\ \langle E|S_1;S_2\rangle\ \sigma = (\mu\Psi)\langle E|S_1;S_2\rangle\ \{\lambda\sigma.\langle\ \rangle\}\sigma =$
$= \langle\sigma\rangle^\wedge\ (\mu\Psi)\ \langle E|S_1\rangle\{(\mu\Psi)\langle E|S_2\rangle\{\lambda\sigma.\langle\ \rangle\}\}\sigma,$

and the right hand side of (+) equals

$$\langle\sigma\rangle^\wedge(\mu\Psi)\langle E|S_1\rangle\{\lambda\sigma.\langle\ \rangle\}\sigma\,^\wedge(\mu\Psi)\langle E|S_2\rangle\{\lambda\sigma.\langle\ \rangle\}\sigma',$$

where $\sigma' = \kappa((\mu\Psi)\langle E|S_1\rangle\{\lambda\sigma.\langle\ \rangle\}\sigma)$.

We thus have to establish a correspondence between the old definition of composition which used κ and $^\wedge$, and the new one which uses continuations. This correspondence is phrased in the next "continuation removal" lemma, which must be clear if the idea behind continuations has been well understood.

LEMMA 6.5. *Let* $\Phi \in D = \mathit{Prog} \to [\Theta \to \Theta]$ *be a fixed point of* Ψ.*For all* R,θ *and* σ *we have that* $\Phi R\theta\sigma = \tau\ ^\wedge\ \theta(\kappa\tau)$, *where* $\tau = \Phi R\{\lambda\sigma.\langle\ \rangle\}\sigma$.

PROOF.(The function κ used here is the κ as defined in chapter 2). The first fact to be remarked is that we have for all R, θ and σ that $\Phi R\{\lambda\sigma.\langle\ \rangle\}\sigma \sqsubseteq \Phi R\theta\sigma$. This holds because $\Phi \in D$ implies that ΦR is a monotonic function. From this observation we can immediately deduce the lemma for the case that $\tau := \Phi R\{\lambda\sigma.\langle\ \rangle\}\sigma$ is infinite, because in that case we have that $\tau = \Phi R\theta\sigma$, due to the fact that τ, being an infinite row, is maximal in Σ^∞. On the other hand $\tau = \tau\ ^\wedge\ \theta(\kappa\tau)$, due to the definition of $^\wedge$ and we are ready.

Next, suppose that $\tau := \Phi\,\{\lambda\sigma.<>\}\sigma$ is finite (notice that this does not imply that $\Phi R\theta\sigma$ be finite). We now prove the lemma by induction on the length of τ. As always, the cases that R is an atomic statement, a conditional statement or a procedure call are straightforward. So suppose $R \equiv <E|S_1;S_2>$. We have

$$\text{lhs} := \Phi R\theta\sigma = <\sigma> \wedge \Phi<E|S_1>\{\Phi<E|S_2>\}\sigma$$

$$\text{rhs} := \tau \wedge \theta(\kappa\tau), \text{ where } \tau = <\sigma> \wedge \Phi<E|S_1>\{\Phi<E|S_2>\{\lambda\sigma.<>\}\}\sigma.$$

Now $\Phi<E|S_1>\{\lambda\sigma.<>\}\sigma$ cannot be infinite, because the observation at the beginning of this proof would then imply that τ would be infinite. For the same reason the length of $\Phi<E|S_1>\{\lambda\sigma.<>\}\sigma$ is smaller than or equal to the length of $\Phi<E|S_1>\{\Phi<E|S_2>\{\lambda\sigma.<>\}\}\sigma$, which is smaller than the length of τ (notice that Lemma 6.3 is used here). Thus we can apply the induction hypothesis twice, and this yields

$$\text{lhs} = <\sigma> \wedge \tau_1 \wedge \Phi<E|S_2>\theta(\kappa\tau),$$

where $\tau_1 = \Phi<E|S_1>\{\lambda\sigma.<>\}\sigma$, and

$$\Phi<E|S_1>\{\Phi<E|S_2>\{\lambda\sigma.<>\}\}\sigma = \tau_1 \wedge \tau_2,$$

with τ_1 as above, and $\tau_2 = \Phi<E|S_2>\{\lambda\sigma.<>\}(\kappa\tau)$.

This last equality yields

$$\text{rhs} = <\sigma> \wedge \tau_1 \wedge \tau_2 \wedge \theta(\kappa\tau),$$

so there remains to be proved

$$\Phi<E|S_2>(\kappa\tau_1) = \tau_2 \wedge \theta(\kappa\tau_2)$$

This is another instance of the lemma. Now τ_2 cannot be infinite, because we have already derived that $\tau = <\sigma> \wedge \tau_1 \wedge \tau_2$, and we assumed that τ was finite. So τ_2 is finite with length smaller than the length of τ, and we can thus use the induction hypothesis once more, leading to the desired result. □

THEOREM 6.6. *The function* Comp *as defined in Definition* 6.4 *is a solution of* CE.

PROOF. See the remarks preceding Lemma 6.5. □

The last thing to be proved is that the function *Comp* as defined in 6.4 is the only solution of CE. This will be done in the same way as in the proof of theorem 5.11. We first prove that Ψ has only one fixed point.

LEMMA 6.7. *Ψ has exactly one fixed point.*

PROOF. Suppose there are more, then there is a $\Phi \in \mathit{Prog} \to [\Theta \to \Theta]$ with $\mu\Psi \sqsubseteq \Phi$. We will prove that for all R,θ and σ we have $(\mu\Psi)R\theta\sigma = \Phi R\theta\sigma$ and thus we reach a contradiction.

1. If $(\mu\Psi)R\theta\sigma$ is infinite then it is maximal Σ^∞. The desired equality then follows from $(\mu\Psi)R\theta\sigma \sqsubseteq \Phi R\theta\sigma$.
2. We now prove for all finite $(\mu\Psi)R\theta\sigma$ the desired equality, and we do this by induction on its length.

Because again the other cases are straightforward we restrict ourselves to the case $R \equiv \langle E|S_1;S_2\rangle$. We have

$$\text{lhs} := (\mu\Psi)\langle E|S_1;S_2\rangle\theta\sigma = \langle\sigma\rangle^\wedge(\mu\Psi)\langle E|S_1\rangle\{(\mu\Psi)\langle E|S_2\rangle\theta\}\sigma$$

and

$$\text{rhs} := \Phi\langle E|S_1;S_2\rangle\theta\sigma = \langle\sigma\rangle^\wedge\Phi\langle E|S_1\rangle\{\Phi\langle E|S_2\rangle\theta\}\sigma.$$

Using Lemma 6.5 we get lhs $= \langle\sigma\rangle^\wedge\tau_1{}^\wedge\tau_2$, where $\tau_1 = (\mu\Psi)\langle E|S_1\rangle\{\lambda\sigma.\langle\,\rangle\}\sigma$ and $\tau_2 = (\mu\Psi)\langle E|S_2\rangle\theta(\phi\tau_1)$, and rhs $= \langle\sigma\rangle^\wedge\tau_2'$, where $\tau_1' = \Phi\langle E|S_1\rangle\{\lambda\sigma.\langle\,\rangle\}\sigma$ and $\tau_2' = \Phi\langle E|S_2\rangle\theta(\kappa\tau_1')$. Now by applying the induction hypothesis we get first that $\tau_1 = \tau_1'$ and thereafter that $\tau_2 = \tau_2'$. □

THEOREM 6.8. *CE has exactly one solution.*

PROOF. We first show how to transform a solution C of CE into a fixed point αC of Ψ. This is done in a way which uses in a sense Lemma 6.4:
$\alpha C := \lambda R\ \lambda\theta.\lambda\sigma.CR\sigma^\wedge\theta(\kappa(CR\sigma))$. We then have the following facts.

1. For all $C \in \mathit{Prog} \to \Sigma \to \Sigma^\infty$ we have that $\alpha C \in \mathit{Prog} \to [\Theta \to \Theta]$, i.e αC is continuous in its continuation parameter.
2. If C is a solution of CE, then αC is a fixed point of Ψ. Notice that it is needed that αC be continuous in its continuation parameter because otherwise it could not be a fixed point of Ψ (Ψ has functionality $D \to D$, where $D = \mathit{Prog} \to [\Theta \to \Theta]$).

   We present the hard case in the proof of fact 2, by proving that for all θ and σ we have lhs $:= (\alpha C)\langle E|S_1;S_2\rangle\theta\sigma =$
   $= \langle\sigma\rangle^\wedge(\alpha C)\langle E|S_1\rangle\{(\alpha C)\langle E|S_2\rangle\theta\}\sigma =:$ rhs .

By definition we have lhs $= C\langle E|S_1;S_2\rangle\sigma^\wedge\theta(\kappa(C\langle E|S_1;S_2\rangle))$ and rhs $= \langle\sigma\rangle^\wedge\tau_1{}^\wedge(\alpha C)\langle E|S_2\rangle\theta(\kappa\tau_1) = \langle\sigma\rangle^\wedge\tau_1{}^\wedge\tau_2{}^\wedge\theta(\kappa\tau_2)$, where $\tau_1 = C\langle E|S_1\rangle\sigma$ and $\tau_2 = C\langle E|S_2\rangle(\kappa\tau_1)$.

Because C is a solution of CE we get lhs $= \langle\sigma\rangle^\wedge\tau_1'{}^\wedge\tau_2'{}^\wedge\theta(\kappa\tau_1')$, where $\tau_1' = C\langle E|S_1\rangle\sigma$ and $\tau_2' = C\langle E|S_2\rangle(\kappa\tau_1')$. Thus $\tau_1 = \tau_1'$ and $\tau_2 = \tau_2'$ and therefore lhs = rhs.

3. Suppose CE has more than one solution, say C and C'. Then there exist R and σ such that $CR\sigma \neq C'R\sigma$. This leads to the fact that αC and $\alpha C'$ are both fixed points of Ψ, and also $(\alpha C)R\theta\sigma \neq (\alpha C')R\theta\sigma$, which contradicts Lemma 6.7. □

7. Σ^∞ AS A METRIC TOPOLOGICAL SPACE

For convenience we first of all repeat the definition of the operator $\Psi:\mathit{Prog} \to \Sigma \to \Sigma^\infty$ derived from CE

$$\begin{aligned}\Psi = \lambda\phi.\lambda R.\lambda\sigma.\ & R \equiv \langle E|A\rangle \to \langle \mathcal{A}A\sigma\rangle,\\ & R \equiv \langle E|P_i\rangle \to \langle\sigma\rangle^\wedge\Phi\langle E|S_i\rangle\sigma,\\ & R \equiv \langle E|\underline{\text{if}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\rangle \to\\ & \quad(\mathcal{B}B\sigma \to \langle\sigma\rangle^\wedge\Phi\langle E|S_1\rangle\sigma,\ \langle\sigma\rangle^\wedge\Phi\langle E|S_2\rangle\sigma),\\ & R \equiv \langle E|S_1;S_2\rangle \to \langle\sigma\rangle^\wedge\Phi\langle E|S_1\rangle\sigma^\wedge\Phi\langle E|S_2\rangle(\kappa(\Phi\langle E|S_1\rangle\sigma))\end{aligned}$$

Let us consider for a moment the approximation sequence from chapter 4, $\bot R\sigma$, $(\Psi\bot)R\sigma$, $(\Psi^2\bot)R\sigma$, $(\Psi^3\bot)R\sigma$, with $R \equiv \langle P \Leftarrow A_1|P;A_2\rangle$. We found that this sequence was not a chain. However it does converge (in some sense) to the correct value. This phenomenon also holds for nonterminating computations like the evaluation of $\langle P \Leftarrow A_1;P|P;A_2\rangle$ in some σ. If one evaluates the sequence $(\Psi^k\bot)R\sigma$ one again observes that this sequence converges to the right result $\langle\sigma,\sigma,\mathcal{A}A_1\sigma,\mathcal{A}A_1\sigma,(\mathcal{A}A_1)^2\sigma,\ldots\rangle$ though it is not a chain.

In fact we can prove that the above observations hold in general: we have for all R and σ a sequence of approximations $(\Psi^i\bot)R\sigma$ which converges to a limit $\lim(\Psi^i\bot)R\sigma$. We thus have to define what the limit is of a converging sequence which is not a chain. Moreover we also need to prove that this limit process yields a function which is a fixed point of Ψ, i.e. $\forall R,\sigma:\Psi(\lim(\Psi^i\bot))R\sigma = (\lim(\Psi^i\bot))R\sigma$. Now this is equivalent to continuity in the topological sense: f is continuous iff for all converging sequences $<x_i>_i$ we have $\lim(fx_i) = f(\lim x_i)$.

In the preceding chapters we dealt with chains which are monotonic sequences, and the limit of such a sequence could conveniently be defined as its least upper bound. In this chapter we will reason along analogous lines as in the cpo approach, but now using a more powerful notion of limit. This notion can be obtained by defining a distance function d on Σ^∞ which makes Σ^∞ a metric topological space.

This approach is inspired by an endeavour to apply Nivat's results, see for example [11], to the problem treated in this paper. We saw no way to attain this, but the basic facts about Σ^∞ that he gave were very useful. In fact, the whole treatment given in this chapter is much in the style of Nivat's.

First a notation: we denote, for $\tau \in \Sigma^\infty$, by $\tau[n]$ the prefix of τ consisting of the first n elements of τ, or τ itself if its length is smaller than n. We then define the following distance function d on Σ^∞:

$$d(\tau_1,\tau_2) = \begin{cases} 2^{-n}, & \text{if } \tau_1[n-1] = \tau_2[n-1] \text{ and } \tau_1[n] \neq \tau_2[n] \\ 0, & \text{otherwise} \end{cases}$$

One easily checks that d is a metric, i.e. we have the familiar properties

$$\begin{aligned} d(\tau_1,\tau_2) &= 0 \text{ iff } \tau_1 = \tau_2 \\ d(\tau_1,\tau_2) &= d(\tau_2,\tau_1) \\ d(\tau_1,\tau_2) &\leq d(\tau_1,\tau_3) + d(\tau_2,\tau_3) \end{aligned}$$

Now the metric space (Σ^∞,d) is complete:

<u>LEMMA 7.1</u>. *A sequence in Σ^∞ converges iff it is a Cauchy sequence.*

<u>PROOF</u>. The only if-part is immediate, because Σ^∞ is a metric space. We next prove the if-part. Suppose $<\tau_i>_i$ is a Cauchy sequence. We construct a τ which is the limit of τ_i in the following way [11]. We have the Cauchy property $\forall n \exists N$: $p,q \geq N \Rightarrow d(\tau_p,\tau_q) < 2^{-n}$. Now let for all n, N_n be the smallest number N with the above property. We then have for all $p \geq N_n$

that

$$\tau_{N_n}[n] = \tau_p[n]. \qquad \ldots(1)$$

Also the sequence $<\tau_{N_n}[n]>_n$ is a $\sqsubseteq$-chain with lub $\tau := \bigsqcup_n \tau_{N_n}[n]$. By (1) $<\tau_i>_i$ converges to τ. □

It has been remarked earlier that one of the facts to prove is that Ψ is continuous. So we will first have a look at how continuity should be defined in this setting. It should be something like "for all converging sequences $<\Phi_k>_k$: $\lim(\Psi\Phi_k) = \Psi(\lim\Phi_k)$", which is the analogon of the cpo-continuity "for all chains $<\Phi_k>_k$: $\bigsqcup_k(\Psi\Phi_k) = \Psi(\bigsqcup_k\Phi_k)$". We therefore have to define what it means that a sequence of functions $<\Phi_k>_k$ converges. In the cpo theory a sequence $<\Phi_k>_k$ was called a chain if it formed a chain taken pointwise, i.e. if $<\Phi_k R\sigma>_k$ was a chain for all R and σ. In an analogous way we can define convergence of the sequence $<\Phi_k>_k$ as pointwise convergence, namely be demanding that for all R and σ the sequence $<\Phi_k R\sigma>_k$ converges.

However, the approximation sequences $<\Psi^k\bot>_k$ that we will investigate have the pleasant property that they converge uniformly in R and σ. This allows us to use a stronger notion of limit, namely the one that corresponds to the following distance function on $\mathit{Prog} \to \Sigma \to \Sigma^\infty$: $d'(\Phi_1,\Phi_2)$ is the lub, taken over all R and σ, of $d(\Phi_1R\sigma,\Phi_2R\sigma)$. One easily checks that a sequence $<\Phi_k>_k$ converges according to this distance iff it is uniformly convergent in R and σ.

We can set up the theory in this smoother version because of the fact that every right hand side in CE has the form $<\sigma'>$ or $<\sigma'>^\frown\ldots$, where σ' is completely determined by R and σ. This fact is crucial in the proof of Lemma 7.4. Compare also the remarks following Theorem 7.6.

We thus have the following metric on $\mathit{Prog} \to \Sigma \to \Sigma^\infty$

$$d'(\Phi_1,\Phi_2) = \mathrm{lub}\{d(\Phi_1R\sigma,\Phi_2R\sigma)) \mid R \in \mathit{Prog},\ \sigma \in \Sigma\},$$

and this distance can be characterized as follows.

<u>LEMMA 7.2</u>. $d'(\Phi_1,\Phi_2) \le \varepsilon$ *for* $0 < \varepsilon < 1$, *iff for all* R *and* σ, $d(\Phi_1R\sigma,\Phi_2R\sigma) \le 2^{-n}$ (*that is,* $\forall R,\sigma{:}\Phi_1R\sigma$ *and* $\Phi_2R\sigma$ *agree on at least their first*

$n-1$ *elements), where* $n = \min\{k \mid 2^{-k} \leq \varepsilon\}$.

PROOF. The if-part is trivial. In order to prove the only if-part, suppose there exists on R and σ with $d(\Phi_1 R\sigma, \Phi_2 R\sigma) > 2^{-n}$. Then we have that these elements of Σ^∞ do not agree on their first $n-1$ elements, that is their distance is greater than or equal to 2^{-n+1} which is greater than ε. This implies however that $d'(\Phi_1, \Phi_2) > \varepsilon$. Contradiction. □

We also have that the metric space $\langle \mathit{Prog} \to \Sigma \to \Sigma^\infty, d' \rangle$ is complete.

LEMMA 7.3. *Every Cauchy sequence* $\langle \Phi_k \rangle_k$ *in* $\mathit{Prog} \to \Sigma \to \Sigma^\infty$ *converges, and its limit is* $\lambda R.\lambda\sigma.\lim \Phi_k R\sigma$.

PROOF. Choose ε. We have to find an N such that $d'(\lambda R.\lambda\sigma.\ \lim \Phi_k R\sigma, \Phi_i) \leq \varepsilon$ for all $i > N$, or equivalently we have to find an N such that for all $i > N$ and for all R,σ we have

$$d(\lim \Phi_k R\sigma, \Phi_i R\sigma) \leq \varepsilon \qquad \ldots(1)$$

Now there exists an N such that for all $i,j > N$: $d'(\Phi_i, \Phi_j) \leq \varepsilon$, that is there exists an N such that for all $i,j > N$ and for all σ,R

$$d(\Phi_i R\sigma, \Phi_j R\sigma) \leq \varepsilon \qquad \ldots(2)$$

For the N determined in this way we will show that (1) holds. So choose R,σ and $i > N$. By (2) we have $\forall j > N: d(\Phi_i R\sigma, \Phi_j R\sigma) \leq \varepsilon$. From this we conclude (1) by the following argument.

Suppose (1) does not hold. Then there is a δ such that $d(\lim \Phi_k R\sigma, \Phi_i R\sigma) = \delta + \varepsilon > \varepsilon$. There exists a j such that $d(\Phi_j R\sigma, \lim \Phi_k R\sigma) < \frac{1}{2}\delta$, and without loss of generality $j > N$. But now we have a contradiction:

$$d(\Phi_i R\sigma, \Phi_j R\sigma) \geq d(\lim \Phi_k R\sigma, \Phi_i R\sigma) - d(\Phi_j R\sigma, \lim \Phi_k R\sigma) > \varepsilon + \delta - \tfrac{1}{2}\delta > \varepsilon$$

which contradicts (2). □

The next Lemma states a basic property which we need to prove continuity of ψ.

LEMMA 7.4. *If for all* $R,\sigma: d(\Phi_1 R\sigma, \Phi_2 R\sigma) \le 2^{-n}$, *then for all* $R,\sigma: d((\Psi\Phi_1)R\sigma, (\Psi\Phi_2)R\sigma) \le 2^{n-1}$.

PROOF. We show the Lemma for the case $R \equiv \langle E|S_1;S_2\rangle$. Suppose $d(\phi_1 R\sigma, \Phi_2 R\sigma) \le 2^{-n}$ for all R and σ, that is, for all R and σ we have that $\Phi_1 R\sigma$ and $\Phi_2 R\sigma$ agree on at least their first n-1 elements. We now investigate $(\Psi\Phi_i)R\sigma$ $(i = 1,2)$ and we have that

$$(\Psi\Phi_i)\langle E|S_1;S_2\rangle\sigma = \langle\sigma\rangle\hat{}\,\Phi_i\langle E|S_1\rangle\sigma\hat{}\,\Phi_i\langle E|S_2\rangle(\kappa(\Phi_i\langle E|S_1\rangle\sigma)).$$

We distinguish two cases:

1. $\Phi_1\langle E|S_1\rangle$ has length smaller than n-1. Then by hypothesis $\tau = \Phi_1\langle E|S_1\rangle\sigma = \Phi_2\langle E|S_1\rangle\sigma$ because they both agree on at least n-1 elements. Furthermore the hypothesis implies that $\Phi_1\langle E|S_1\rangle(\kappa\tau)$ and $\Phi_2\langle E|S_2\rangle(\kappa\tau)$ agree on at least their first n-1 elements. Therefore the $(\Psi\Phi_i)\langle E|S_1;S_2\rangle\sigma$ agree on at least n elements, that is $d((\Psi\Phi_1)R\sigma, (\Psi\Phi_2)R\sigma) \le 2^{-n-1}$.
2. If $\Phi_1\langle E|S_1\rangle\sigma$ has length greater than or equal to n-1 then also $\Phi_2\langle E|S_1\rangle\sigma$ must have more than n-1 elements and furthermore at least the first n-1 elements of these rows are equal. But this means that already $\langle\sigma\rangle\hat{}\,\Phi_i\langle E|S_1\rangle\sigma$ agree on n elements (i=1,2) and the Lemma follows. □

Notice that from this Lemma we can infer immediately that $\langle\Psi^k\bot\rangle_k R\sigma$ forms a converging sequence for all R and σ. So now the informal remarks at the beginning of this chapter are formally justified.

We next prove a property of Ψ which implies that Ψ is continuous, and also that for every initial value Φ the sequence $\langle\Psi^k\Phi\rangle_k$ converges. Notice that these two properties were also needed in the cpo theory in order for Ψ to have a fixed point: Ψ had to be continuous and (the particular case) $\langle\Psi^k\bot\rangle_k$ had to be a chain.

LEMMA 7.5. Ψ *is a contraction, in particular* $\forall\Phi_1,\Phi_2$: $d'(\Psi\Phi_1,\Psi\Phi_2) \le \frac{1}{2}d'(\Phi_1,\Phi_2)$.

PROOF. Suppose $d'(\Phi_1,\Phi_2) = \varepsilon$. Let $n = \min\{k|2^{-k} \le \varepsilon\}$. We have that $d(\Phi_1 R\sigma, \Phi_2 R\sigma) \le 2^{-n}$ for all R and σ by Lemma 7.2. Now Lemma 7.4 implies

$$d((\Psi\Phi_1)R\sigma,\ (\Psi\Phi_2)R\sigma) \le 2^{-n-1} = \tfrac{1}{2}.2^{-n} \le \tfrac{1}{2}\varepsilon \text{ for all } R \text{ and } \sigma. \quad \Box$$

We can now deduce the theorem we were up to, using standard techniques from topology.

THEOREM 7.6. Ψ *has exactly one fixed point.*

PROOF. 1. Ψ, being a contraction is continuous. We can straightforwardly prove that for all converging sequences $<\Phi_k>_k$ we have $\Psi(\lim \Phi_k) = \lim (\Psi\Phi_k)$, or $\forall\varepsilon > 0\ \exists N\ \forall n > N$: $d'(\Psi(\lim \Phi_k), \Psi\Phi_n) < \varepsilon$.

2. For every $\Phi \in$ *Prog* $\rightarrow \Sigma \rightarrow \Sigma^\infty$ the sequence $<\Psi^k\Phi>_k$ is a (uniform convergent) Cauchy sequence. For we have

$$\begin{aligned}
&d'(\Psi^k\Phi,\Psi^i\Phi) = d'(\Psi^k\Phi,\Psi^k\Psi^{i-k}\Phi) \le \\
&(\tfrac{1}{2})^k d'(\Phi,\Psi^{i-k}\Phi) \le \\
&(\tfrac{1}{2})^k[d'(\Phi,\Psi\Phi) + d'(\Psi\Phi,\Psi^2\Phi)+\ldots+d'(\Psi^{i-k-1}\Phi,\Psi^{i-k}\Phi)] \le \\
&(\tfrac{1}{2})^k d'(\Phi,\Psi\Phi)[1+\tfrac{1}{2}+\ldots+\tfrac{1}{2}^{i-k-1}] \le (\tfrac{1}{2})^{k-1}d'(\Phi,\Psi\Phi).
\end{aligned}$$

3. Choose a Φ in *Prog* $\rightarrow \Sigma \rightarrow \Sigma^\infty$. The limit $\eta\Psi := \lim \Psi^k\Phi$ is a fixed point of Ψ.

For we have
$$\begin{aligned}
\Psi(\eta\Psi) &= \Psi(\lim \Psi^k\Phi) = \lim \Psi(\Psi^k\Phi) \qquad \text{(by continuity)} \\
&= \lim \Psi^{k+1}\Phi = \eta\Psi.
\end{aligned}$$

4. Ψ has only one fixed point. For, suppose there would be another one Φ. Then $d'(\eta\Psi,\Phi) \neq 0$. Because both are fixed points we have $d'(\eta\Psi,\Phi) = d'(\Psi(\eta\Psi),\Psi\Phi)$. On the other hand, by Lemma 7.5, we have $d'(\Psi(\eta\Psi),\Psi\Phi) \le \tfrac{1}{2}d'(\eta\Psi,\Phi)$. This implies $d'(\eta\Psi,\Phi) = 0$, and we have a contradiction. $\Box$

We close this chapter with an investigation of the consequences of a little change in CE, namily that the fourth clause is changed into

$$\mathit{Comp} <E|S_1;S_2>\sigma = \mathit{Comp}<E|S_1>\sigma\,\hat{}\ \mathit{Comp}<E|S_2>(\kappa(\mathit{Comp}<E|S_1>\sigma)).$$

We still have that CE has exactly one solution but this fact is now harder to prove. In particular Lemma 7.4 is no longer true as one easily observes. Also it is not any more the case that the sequences $<\Psi^k\Phi>_k$ converge uniformly (for arbitrary Φ) in R and σ.

We have to handle the problem differently, we cannot use the lub-distance on $\mathit{Prog} \to \Sigma \to \Sigma^\infty$ any more, but we have to use the pointwise extensions of convergence in Σ^∞, quite analogously to how theory was set up for cpo structures. In the sequel we give a brief sketch of how Theorem 7.6 has to be deduced under these new circumstances.

1. DEFINITION. $<\Phi_k>_k$ converges iff $\forall R,\sigma$: $<\Phi_k R\sigma>$ converges. In that case we define lim Φ_k as $\lambda R.\lambda\sigma.$ (lim $\Phi_k R\sigma$).

2. LEMMA. *Ψ is continuous, i.e. for all converging sequences $<\Phi_k>_k$ we have* lim $\Psi\Phi_k = \Psi($lim $\Phi_k)$.

This is to be proved by cases, the most complex one being $R \equiv <E|S_1;S_2>$. In that case one has to prove lhs:= lim$[\Phi_k<E|S_1>\sigma\,\hat{}\,\Phi_k<E|S_2>(\kappa(\Phi_k<E|S_1>\sigma))]$ = (lim $\Phi_k)<E|S_1>\sigma\,\hat{}\,($lim $\Phi_k)<E|S_2>[\kappa(($lim $\Phi_k)<E|S_1>\sigma]$ =: rhs. We distinguish two cases.

a. (lim $\Phi_k)<E|S_1>\sigma$ is a finite row. In that case the sequence $<\Phi_k<E|S_1>\sigma>$ must be constant from a certain index k, and equal to its limit. We then can use the fact that for all τ_0 the function $\lambda\tau.\tau_0\,\hat{}\,\tau$ is continuous to prove the lemma.
b. (lim $\Phi_k)<E|S_1>\sigma$ is infinite. In that case lhs = lim$(\Phi_k<E|S_1>\sigma)$, to be proved using an ε-N-argument and the lemma follows. □

3. LEMMA. $\forall R,\sigma,n\ \exists N$: $k \geq N \Rightarrow \forall\Phi_1,\Phi_2$:$d((\Psi^k\Phi_1)R\sigma,\ (\Psi^k\Phi_2)R\sigma) \leq 2^{-n}$.

This is a useful lemma, in some sense the anologon of Lemma 7.4. Note that the N in the lemma is in general dependent on R and σ.

The lemma has to be proved by induction on the entity <n, length(R)>. This lemma has the following useful consequences (4 and 5).

4. LEMMA. *For all Φ we have that $<\Psi^k\Phi>_k$ converges.*

5. LEMMA. *The limit of $<\Psi^k\Phi>_k$ is independent of the initial value Φ.*

6. THEOREM. *The changed Cook equations have exactly one solution.*

PROOF. There is a fixed point (for instance lim $(\Psi^k\bot) =: \mu\Psi$), by result 2 and 4. If there would be another fixed point Φ_0, then we would have that $\mu\Psi$ = lim $\Psi^k\Phi_0$ = lim $<\Phi_0,\Phi_0,\ldots>$ = Φ_0 (the first equality holds by result 5). □

8. CONCLUDING REMARKS

In a certain sense we have worked in a direction opposite to the one Scott took when he devised his theory of computing. He wanted to use notions from topology such as limit and continuity, and therefore he introduced cpo's because the domains on which programs compute are in general not of a topological kind. We found in chapter 4 that Σ^{∞} considered as a cpo, did not have enough structure to prove the desired result. However by using the inherent typology on Σ^{∞} we were able to derive this result in an elegant manner (chapter 7).

The above results have been derived for a rather simple paradigm language, but the techniques used can readily be applied to more sophisticated languages. All results derived here could in fact have been obtained in a more abstract setting. We will outline briefly how this can be done.

Let *Prog* be the programming language under consideration, and suppose we want to specify a function $\mathit{Comp}\colon \mathit{Prog} \to \Gamma \to \Delta^{\infty}$, using a set of Cook equations. Here Δ can be considered as a set of trace elements which are descriptions of machine states in some general sense. Furthermore Γ is meant to be a set of machine configurations which contains enough information to start a computation. Γ can contain the initial state, but possibly also other things like the currently valid procedure declarations E. We can have $\Gamma = \Delta$ but this need not necessarily be the case.
Let *Prog* partitioned in k mutually disjoint subclasses $\mathit{Prog}_1,\ldots,\mathit{Prog}_k$.
T Then the Cook equations will in general have the following structure

$$\mathit{Comp}R\gamma = \mathit{Comp}_1 R\gamma, \quad \text{for } R \in \mathit{Prog}_1$$

$$\vdots$$

$$\mathit{Comp}R\gamma = \mathit{Comp}_k R\gamma, \quad \text{for } R \in \mathit{Prog}_k.$$

The expressions $\mathit{Comp}_i R\gamma$ will either have the form $<\delta>$ where δ depends only on R and γ, or it has the form $<\delta>^{\wedge}\mathrm{CMP}[R,\gamma]$ with again δ dependent only on R and γ, and where $\mathrm{CMP}[R,\gamma]$ is defined by the following parametrized BNF rules:

$$\mathrm{CMP}[R,\gamma] ::= \mathit{Comp}R'\gamma' \mid p \to \mathrm{CMP}[R',\gamma'], \mathrm{CMP}[R'',\gamma''] \mid$$
$$\mathit{Comp}R'\gamma'^{\wedge}\mathrm{CMP}[R'',\kappa(\mathit{Comp}\ R'\gamma')],$$

where p, γ', γ'', R' and R'' are expressions dependent on R and γ only.

Notice that according to this definition all right hand sides in the Cook equations begin with a constant one element row $<\delta>$, which ensures the uniformity property discussed in chapter 7.

Now all techniques presented in this paper can be applied to equations which are built up as above. Furthermore, a wide variety of programming languages and concepts can be described by such equation. We give a few examples.

1. The language treated in this paper has $\Delta = \Gamma = \Sigma$, and the equation CE are of the above form, as can easily be seen.
2. Cook has in his paper $\Gamma = S \times D \times P$, where D contains the functions from variables to registers (adresses), S contains the states: functions from registers to values and P contains the functions from procedure variables to pairs consisting of a procedure body and a list of formal parameters. He furthermore uses $\Delta = S$. In this set-up declarations can be handled as follows

   $$\mathit{Comp}(\underline{\text{begin}}\ \underline{\text{new}}\ x;\ D;S\ \underline{\text{end}})<s,\delta,\pi> =$$
   $$<s>^\wedge \mathit{Comp}(\underline{\text{begin}}\ D;S\ \underline{\text{end}})<s,\delta',\pi>,$$
   where $\delta'(y) = \delta(y)$ if $y \neq x$, and $= X_{k+1}$ if $y \equiv x$,
   where X_k is the highest indexed register used in δ.

   In the above clause D stands for the language construct which consists of a row of declarations separated by semicolons.
3. The while statement in Cook's paper:

   $$\mathit{Comp}(\underline{\text{while}}\ B\ \underline{\text{do}}\ S)<s,\delta,\pi> =$$
   $$\mathcal{B}B<s,\delta> \rightarrow \mathit{Comp}(S)<s,\delta,\pi>^\wedge \mathit{Comp}\ (\underline{\text{while}}\ b\ \underline{\text{do}}\ S)<\kappa(\mathit{Comp}(S)<s,\delta,\pi>),\delta,\pi>,$$
   $$<s>$$

   Notice that for this clause the uniformity property does not hold: the right hand side of this rule does not begin with a one element list for all cases. This can be remedied easily though.
4. Backtracking can be handled as follows. Suppose the language contains statements of the form $\underline{\text{try}}\ S_1\,\underline{\text{or}}\ S_2$, establishing a choice point, and an atomic statement $\underline{\text{fail}}$ which causes control to return to the latest encountered choice point and to proceed from there with evaluation of the other alternative (if this has not yet been used of course), while

returning the state to the situation it was in before evaluation of the first alternative started.

The corresponding Cook equation could then be

$$Comp(\underline{try}\ S_1\ \underline{or}\ S_2)\sigma = \langle\sigma\rangle^\wedge[\kappa(CompS_1\sigma) = FAIL \rightarrow CompS_1\sigma^\wedge CompS_2\sigma,\ Comp\,S_1\sigma]$$

$$Comp(\underline{fail})\sigma = \langle FAIL\rangle$$

(Notice that his approach does not work as it stands now, if we allow a <u>try</u> statement to be composed with another one like in constructs <u>try</u> S_1<u>or</u> $S_2;S_3$ because failure in S_3 should cause backtracking too. This can be handled though, see [5, chapter 6])

5. In [4] and [3, chapter 10] we gave a Cook semantics for the <u>goto</u>-statement. Again, the equation used there fit in the general scheme given above.

Notice that if in a set of Cook equations the last clause in the definition of CMP[R,γ] is not used (that is, if we do not use k, nor $^\wedge$ in a non continuous way), that then we can prove that the operator derived from the Cook equations is continuous, not only in the topological sense (chapter 7), but also in the cpo sense, because we are not bothered any more by the objections from chapter 4.

The semantics for the <u>goto</u>-statement that we gave in [4] and [3] obeys this restriction. This is often the case when Cook equation use continuations, or syntactic continuations, which are a variant thereof. In the latter case the function *Comp* does not have a continuation parameter which is a function, but *Comp* uses a list of statements which are to be executed after the one which is evaluated at the moment. In [4] we gave the following equations

$$Comp((S_1;S_2);S_3)\sigma = \langle\sigma\rangle^\wedge Comp(S_1;(S_2;S_3))\sigma$$
$$Comp(A;S)\sigma = \langle AA\sigma\rangle^\wedge CompS(AA\sigma)$$

Observe that the list of statements is stored here implicitly namely as part of the program (that part of the first parameter of *Comp* which lies to the right of the semicolon). We could also have stored this list more explicitly and keep it in γ. Γ would then be equal to $\Sigma \times$ {lists of statements} and the Cook equations would be like

$$Comp(S_1;S_2)\langle\sigma,L\rangle = \langle\sigma\rangle^\wedge CompS_1\langle\sigma,\langle S_2\rangle^\wedge L\rangle$$

$$CompA\langle\sigma,L\rangle = L=\langle\rangle \rightarrow \langle AA\sigma\rangle, \langle AA\sigma\rangle^\wedge Comp(hdL)\langle\sigma,t\ell L\rangle$$

(compare the SECD-machien semantics [9]).

So in general the use of continuation semantics leads to Cook equations which induce lub-continuous operators. This shows that, in some sense, continuation semantics is more elegant than direct semantics. The fact that direct semantics leads to discontinuity has to do with the following. The effect of executing $S_1;S_2$ can be caught only by applying the function derived from S_2 to the result of evaluation of S_1. Now in continuation semantics we also associate a function with S_2 but this function is used more subtly, we do not only have the option to apply this function, but we might also update it (as in the clause $Comp\ (S_1;S_2)\theta\sigma = \langle\sigma\rangle^\wedge CompS_1\{CompS_2\theta\}\sigma$), or we can, as in the case of the goto-statement disregard it completely.

In direct semantics we are forced to apply the function $Comp\ S_2$ to the result of evaluation of S_1, that is to $\kappa(CompS_{1\sigma})$. The discontinuity which creeps in here stems from the fact that $CompS_1\sigma$ has too much information: as far as $CompS_2$ is concerned only the last element of this row matters.

The opposite can also occur. For instance, if we would try to model the goto-statement with direct semantics, and we would insist that we have a Cook equation like $Comp(S_1;S_2)\sigma = \langle\sigma\rangle^\wedge CompS_1\sigma^\wedge CompS_2(\kappa(CompS_1\sigma))$, then $CompS_1\sigma$ does not contain enough information about the evaluation of S_1. We must know whether evaluation of S_1 terminated because a goto-statement leading out of S_1 has been executed, for in that case $CompS_2$ should not be applied to $\kappa(CompS_1\sigma)$.

We can obtain this effect by taking as trace elements in $\tilde{\Delta}$ and Γ not σ's, but pairs from $\Sigma \times \mathbb{N}$ instead. Now a result $\langle\sigma,n\rangle$ would mean that σ is the result of evaluating a statement during which n goto-statements have been processed. If we add to the statements the "declarations" of the labels occurring in it ($E \equiv \langle L_i \Leftarrow S_i\rangle_i$, where S_i is that part of the program that textually follows L_i) then we get programs $\langle E|S\rangle$ for which we would have Cook equations like the following.

$$Comp\langle E|\underline{goto}\ L_i\rangle\langle\sigma,n\rangle = \langle\sigma,n+1\rangle^\wedge Comp\langle E|S_i\rangle\langle\sigma,n+1\rangle$$

$$\begin{aligned}&Comp\langle E|S_1;S_2\rangle\langle\sigma,n\rangle = \\ &\quad \kappa(Comp\langle E|S_1\rangle\langle\sigma,n\rangle) = \langle\sigma',n\rangle \rightarrow \\ &\qquad \langle\sigma,n\rangle^\wedge\ Comp\langle E|S_1\rangle\langle\sigma,n\rangle^\wedge\ Comp\langle E|S_2\rangle\langle\sigma',n\rangle, \\ &\qquad \langle\sigma,n\rangle^\wedge\ Comp\langle E|S_1\rangle\langle\sigma,n\rangle,\end{aligned}$$

that is S_2 should only be evaluated if the number of goto's processed has not increased due to evaluation of S_1.

So we see here that we could make direct semantics work only by adding extra information. All this can be contrasted with the mechanism used in continuation semantics. In evaluating $\mathit{Comp}(S_1;S_2)\theta\sigma = \langle\sigma\rangle^\wedge \mathit{Comp}S_1 \{\mathit{Comp}S_2\theta\}\sigma$, the continuation $\mathit{Comp}S_2\theta$ is applied only if S_1 is an atomic statement, that is if evaluation of S_1 yields a simple result (a row of one element) and we can be certain that there are no complicating side effects. In all other cases the continuation $\mathit{Comp}S_2\theta$ is updated, and not applied. The fact that continuation semantics leads to lub-continuous operators is due to this more cautious approach.

The theory as it stands now cannot be applied to nondeterministic programs, and as a consequence of this neither to parallel programs. This is due to the fact that nondeterministic programs generate trees and not rows. However, it seems that the techniques presented here can be extended to trees as well. Part of this extension is reported on in [8].

The central theorem that we have proved four times in this paper holds also if the Cook equations do not have expressions in their right-hand sides which start with a constant row. Notice however that we have to be careful here. We could not for instance leave out the $\langle\sigma\rangle$ in the second clause on procedure calls in CE (chapter 2), because if we would have done so, then $\mathit{Comp}\langle P \Leftarrow P|P\rangle\sigma$ would not yield an infinite row which it should do because $\langle P \Leftarrow P|P\rangle$ specifies a nonterminating computation.

However, the central theorem of this paper would be much harder to prove, as has been remarked already in chapter 7. In that case Ψ does not converge uniformly, and we had to work out in more detail how Ψ behaves in order to prove the theorem. The same phenomenon can be observed in other chapters. For instance, Definition 3.1, must now be by induction on $\langle n, \mathrm{length}(R)\rangle$ instead of n, and the same holds for induction arguments in some other proofs (for instance Lemma 6.5). Furthermore, Lemma 5.8 is no longer true, as the counterexample $R \equiv A_1;A_2$ and $i = 0$ shows. A weaker version of the lemma holds though: $\forall R,\sigma \neq \bot\ \exists k{:}\Phi_i R\sigma \in \Sigma^{*\bot} \Rightarrow \Phi_{i+k}R\sigma \neq \Phi_i R\sigma$. So it pays off to demand that the Cook equations are all of the standard form described earlier in this chapter.

There are also other reasons to do so. The operational semantics yields a row of states (or δ's) which is intended as the trace left by execution of the program under consideration. Now the execution of $S_1;S_2$ can be divided into three parts, namely first determining that the statement is a composition of two other statements, secondly evaluating the first statement, and lastly evaluating the second one. It is plausible to demand that all three stages must have an effect on the trace, so in particular this must hold for the first one. It is therefore reasonable that every clause in the Cook equations adds an element to the trace, because every clause of the equations corresponds to some action, or to a decomposition of the statement being evaluated.

RELATED WORK AND ACKNOWLEDGEMENTS

In a letter to Cook [1], Krzysztof Apt suggested a method to compute *Comp* which is related to the techniques of chapter 3: he proposes to define by induction on k the row *Comp*'Rσk which should consist of the first k elements of *Comp*Rσ. Having defined *Comp*' he then defines *Comp*Rσ = τ iff ∃k: *Comp*Rσn = τ for all n ≥ k. He therefore defines *Comp* only for finite rows.

The same holds for the results of Jeff Zucker in the appendix of [3]. He defines the function *Comp* as a fixed point of a set of equations derived from CE. He does this by using the recursion theorem.

The technique in chapter 5 of adding the bottom element ⊥ to mark a row as not yet completed has been used by Ralph Back in his analysis of unbounded nondeterminism [2].

The results in chapter 7 where inspired by the reading of Nivat's and others work on infinite computations, as reported on for instance in [11]. The topology on Σ^∞ was presented there, and also the proof of Lemma 7.1 can be found there.

I acknowledge with pleasure the assistence of the following persons: the members of the Dutch working group on semantics where an earlier version of this work has been presented, and in particular Ruurd Kuiper with whom I have had frequent and stimulating discussions on the material presented here. I would like to thank Jaco de Bakker and Ruurd Kuiper who read the manuscript and came to me with useful comments and finally, I would also like to thank Nizethe Kemmink and Susan Carolan for doing such a good typing job on a rather disjointed manuscript.

REFERENCES

[1] APT, K.R., *personal communication*

[2] BACK, R.J., *Semantics of unbounded nondeterminism,* in: Proc. 7th coll. automata, languages and programming (J.W. de Bakker and J. van Leeuven, eds.), pp.51-63, Lecture Notes in Computer Science 85, Springer (1980).

[3] BAKKER, J.W. DE, *Mathematical theory of program correctness,* Prentice Hall Int. (1980).

[4] BRUIN, A. DE, *Goto statements: semantics and deduction systems,* Report IW 74/79, Mathematisch Centrum (1979).

[5] BRUIN, A. DE, *Operational and denotational semantics describing the matching process in SNOBOL4,* Report IW 151/80, Mathematisch Centrum (1980).

[6] COOK, S.A., *Soundness and completeness of an axiom system for program verification,* SIAM J. on Computing, Vol. 7, Nr. 1, pp 70-90 (1978).

[7] HOARE, C.A.R. and LAUER, P.E., *Consistent and complementary formal theories of the semantics of programming languages,* Acta Informatica, Vol. 3, pp. 135-153 (1974).

[8] KUIPER, R., *An operational semantics for nondeterminism equivalent to the mathematical one,* Mathematical Centre Report, to appear.

[9] LANDIN, P.J., *The mechanical evaluation of expressions,* Computer J., Vol. 6, nr.4, pp. 308-320 (1964).

[10] LAUER, P.E., *Consistent formal theories of the semantics of programming languages,* IBM Laboratory Vienna, Techn. Report TR 25-121 (1971).

[11] NIVAT, M., *Infinite words, infinite trees, infinite computations,* in: Foundations of computer science III, part 2: languages, logic, semantics (J.W. de Bakker and J. van Leeuwen, eds.), pp.1-52, Mathematical Centre Tracts 109.

[12] STOY, J.E., *Denotational semantics - the Scott-Strachey approach to programming language theory,* M.I.T. Press, Cambridge, Mass. (1977).