Contractions in Comparing Concurrency Semantics

J.N. Kok & J.J.M.M. Rutten
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

We define for a simple concurrent imperative language both operational and denotational semantic models as fixed points of contractions on complete metric spaces. Next, we develop a general method for comparing different semantic models by relating their defining contractions and exploiting the fact that contractions have a unique fixed point.

0. Introduction

We present a study of a simple concurrent imperative language, called L_0. We shall define an *operational semantics* $\mathcal{O}_0$ and a *denotational semantics* $\mathcal{D}_0$ for it and give a comparison of the two models. (We shall use the terms *semantics* and *semantic model* as synonyms.) This *comparison* is the main subject of our paper, rather than the specific nature of the language itself or the particular properties of its semantics.

The language L_0 has been defined and studied already in much detail in [BMOZ1,2] and [BKMOZ], on which we rely heavily. It belongs to the wide class of *concurrent* (*parallel*) *imperative* programming languages. We shall discuss parallel execution through interleaving (*shuffle*) of elementary actions. Further, L_0 contains constructs for sequential composition, local nondeterminacy, and recursion.

For our semantic definitions we shall use *metric* structures, rather than order-theoretic domains. The metric approach is particularly felicitous for problems where histories, computational traces and tree-like structures of some kind are essential. Moreover, it allows for the definition of the notion of *contraction*, which we discuss in more detail in a moment. Our operational model $\mathcal{O}_0$ is based on the transition system technique of Hennessy and Plotkin [HP] and Plotkin [Pl2, Pl3]. It is closely related to the one defined in [BKMOZ], but there are some differences. Our denotational model D_0 is almost exactly the same as in [BKMOZ]. It is defined compositionally, giving the meaning of a compound statement in terms of the meaning of its components, and tackling recursion with the help of fixed points.

Although the semantic models presented here are (roughly) the same as in [BKMOZ], there is one major difference, being the way in which they are defined. In this paper we define both the operational and denotational models as fixed points of contractions.

A *contraction* $f:M\to M$ on a complete metric space M has the useful property that there exists one and only one element $x\in M$ satisfying $f(x)=x$. This elementary fact is known as *Banach's fixed point theorem*. Such a fixed point x is entirely determined by the definition of f:

The research of J.N. Kok is partially supported by the Netherlands Organisation for Research (NWO), grant 125-20-04. The research of J.N. Kok and J.J.M.M. Rutten is partially supported by ESPRIT project 415: Parallel Architectures and Languages for Advanced Information Processing — a VLSI-directed approach.

any other element $y \in M$ satisfying the same properties as x, that is, satisfying $f(y)=y$, is equal to x. The contractions Φ we use in this paper are always of type $\Phi:(M_1 \to M_2) \to (M_1 \to M_2)$, that is, they are defined on a complete metric function space $M_1 \to M_2$. Then the fixed point of Φ is a function from M_1 to M_2.

The fact that our denotational model can be obtained as fixed point of suitable contraction is not very surprising, fixed points playing traditionally an important role in denotational semantics. It is interesting, however, to observe that the same method applies to the definition of the *operational* model. One might wonder whether a model thus obtained still deserves to be called operational. That this is the case follows from the fact (not proved here) that it equals the operational model defined in the usual manner, without the use of a contraction.

The main advantage of this style of defining semantic models as fixed points is that it enables us to compare them more easily. This brings us to the discussion of what has been announced above to be the main subject of this paper: the *comparison* of operational and denotational semantic models, which we shall also call the study of their *semantic equivalence*. About the question why this would be an interesting problem we want to be brief. Different semantic models of a given language can be regarded as different views of the same object. So they are in some way related. Their precise relationship we want to capture in some formal statement.

Let us now sketch the way we use contractions in our study of semantic equivalence. Let L be a language. Suppose an operational model $\mathcal{O}$ for L is given as the fixed point of a contraction

$$\Phi:(L \to M) \to (L \to M),$$

where M is a complete metric space. Suppose furthermore that we have a denotational model $\mathcal{D}$ for L of the same type as $\mathcal{O}$, that is, with $\mathcal{D}:L \to M$, for which we can prove $\Phi(\mathcal{D})=\mathcal{D}$. Then it follows from the uniqueness of the fixed point of Φ that $\mathcal{O}=\mathcal{D}$.

In the context of *complete partial ordering structures* similar approaches exist (see, e.g., [HP] and [AP]). There, the operational semantics $\mathcal{O}$ can be characterised as the (with respect to the pointwise ordering) smallest function $\mathcal{F}$ satisfying $\Phi(\mathcal{F})=\mathcal{F}$, for some continuous function Φ. Then it follows from $\Phi(\mathcal{D})=\mathcal{D}$ that $\mathcal{O}$ is smaller than $\mathcal{D}$. In order to establish $\mathcal{O}=\mathcal{D}$ it is proved that $\mathcal{O}$ satisfies the defining equations for $\mathcal{D}$, from which it follows that $\mathcal{D}$ is smaller than $\mathcal{O}$. Please note that within the metric setting we can omit the second part of the proof.

In general $\mathcal{O}$ and $\mathcal{D}$ have different types, that is, they are mappings from L to different mathematical domains. In the language we consider, this difference is caused by the fact that recursion is treated in the denotational and operational semantics *with* and *without* the use of so-called environments, respectively. Therefore, $\mathcal{O}$ and $\mathcal{D}$ cannot be fixed points of the same contraction. Now suppose $\mathcal{O}$ and $\mathcal{D}$ are defined as fixed points of $\Phi:(L \to M_1) \to (L \to M_1)$ and $\Psi:(L \to M_2) \to (L \to M_2)$, respectively, where M_1 and M_2 are different complete metric spaces. Then we can relate $\mathcal{O}$ and $\mathcal{D}$ by defining an intermediate semantic model for L as the fixed point of a contraction $\Phi':(L \to M') \to (L \to M')$, and by relating Φ , Φ' and Ψ as follows. If we define

$$f_1:(L \to M_1) \to (L \to M'), \quad \text{and} \quad f_2:(L \to M_2) \to (L \to M'),$$

and we next succeed in proving the commutativity (indicated by *) of the next diagram:

$$\begin{array}{ccc} L \to M_1 & \overset{\Phi}{\to} & L \to M_1 \\ f_1 \downarrow & *_1 & \downarrow f_1 \\ L \to M' & \overset{\Phi'}{\to} & L \to M' \\ f_2 \uparrow & *_2 & \uparrow f_2 \\ L \to M_2 & \overset{\Psi}{\to} & L \to M_2 \end{array}$$

then we will be able to deduce the following relation between $\mathcal{O}$ and $\mathcal{D}$:

$$f_2(\mathcal{D})=f_1(\mathcal{O}).$$

It is straightforward from $*_1$ and $*_2$, and the fact that Φ, Φ', and Ψ are contractions.

This will be the procedure we follow for the models $\mathcal{O}_0$ and $\mathcal{D}_0$ of L_0 in section 1. There f_1 and f_2 are such that for closed statements (i.e., containing no free statement variables) $s \in L_0$, we have: $\mathcal{O}_0(s)=\mathcal{D}_0(s)$. This result is not new: It was already proved in [BMOZ1,2] and [BKMOZ]. However, the proofs given there are quite complicated and not so easy to understand. Furthermore, it seems to be difficult to extend and generalise them.

Given the definitions of $\mathcal{O}_0$ and $\mathcal{D}_0$, it is intuitively obvious that they yield the same values for closed statements. In other words, the result that $\mathcal{O}_0=\mathcal{D}_0$ is not very surprising, neither. In that respect, the method applied in this paper for deriving it might seem disproportionally complex. Therefore, we would like to stress what we consider to be the main contribution of this paper: We have developed a method which can be easily generalised for proving the semantic equivalence of operational and denotational semantics for languages much more complicated than L_0. In section 2, we shall illustrate this by briefly describing some languages for which we have proved semantic equivalence in this manner (references to some corresponding papers will be given there).

In section 3, some conclusions and remarks about future research are formulated. Section 4 gives the references. For the basic definitions of metric topology we refer to [Du] and [En]. Most of what we need is repeated in [BKMOZ].

This paper is in fact an extended abstract of [KR], to which the reader is referred for a more detailed description of our ideas. (In [KR] all the proofs can be found that are omitted here.)

ACKNOWLEDGEMENTS: We are much indebted to Jaco de Bakker, John-Jules Meijer, Ernst-Rudiger Olderog, and Jeffrey Zucker, authors and co-authors of the papers [BMOZ1,2] and [BKMOZ], respectively, on which we have relied heavily. We are also grateful to Jaco de Bakker for his many comments and suggestions made during our work on this subject. We thank Pierre America for pointing out an error in the definition of guardedness (which caused us considerable trouble and therefore increased our insights). We acknowledge fruitful discussions on our work in the Amsterdam concurrency group, including Jaco de Bakker, Frank de Boer, Arie de Bruin, John-Jules Meijer, and Erik de Vink. Finally, we express our thanks to Dini Verloop, who has expertly typed this document.

1. A SIMPLE LANGUAGE (L_0)

1.1 *Syntax*

For the definition of the language studied in this paper we need two sets of basic elements. Let A, with typical elements $a,b, \ldots$, be the set of *elementary actions*. For A we take an arbitrary, possibly infinite, set. Further, let *Stmv*, with typical elements $x,y, \ldots$, be the set of statement variables. For *Stmv* we take some infinite set of symbols.

DEFINITION 1.1 (Syntax for L_0): We define the set of *statements* L_0, with typical elements $s,t, \ldots$, by the following syntax:

$$s ::= a \mid s_1;s_2 \mid s_1 \cup s_2 \mid s_1 \| s_2 \mid x \mid \mu x[t]$$

where $t \in L_0^x$, the set of statements which are *guarded for x*, to be defined below.

A statement s is of one of the following six forms: (1) an *elementary action* a; (2) the *sequential composition* $s_1;s_2$ of statements s_1 and s_2; (3) the *nondeterministic choice* $s_1 \cup s_2$, also

known as *local nondeterminism* [FHLR]: $s_1 \cup s_2$ is executed by executing either s_1 or s_2 chosen nondeterministically; (4) the *concurrent execution* $s_1 \| s_2$, modeled by the arbitrary interleaving *(shuffle)* of the elementary actions of s_1 and s_2; (5) a statement variable x, which is (normally) used in: (6) the recursive construct $\mu x[t]$: its execution amounts to execution of t where occurrences of x in t are executed by (recursively) executing $\mu x[t]$. For example, with the definition to be proposed presently, the intended meaning of $\mu x[(a;x) \cup b]$ is the set $a^* \cdot b \cup \{a^\omega\}$.

An important restriction of our language is that we consider only recursive constructs $\mu x[t]$ for which t is guarded for x: $t \in L_0^x$. Intuitively, a statement t is guarded for x when all occurrences of x in t are preceded by some statement. More formally:

DEFINITION 1.2 (Syntax for L_0^x): The set L_0^x of statements which are *guarded for* x is given by

$$t ::= \quad a \mid t;s \mid t_1 \cup t_2 \mid t_1 \| t_2 \mid y \mid \mu x[t] \mid \mu y[t']$$

where $s \in L_0$, $y \neq x$, and $t' \in L_0^x \cap L_0^y$.

It appears to be useful to have the languages under consideration contain a special element, denoted by E, which can be regarded as the empty statement. From now on E is considered to be an element of L_0, and L_0^x. We shall still write L_0 for $L_0 \cup \{E\}$ and L_0^x for $L_0^x \cup \{E\}$. Please note that syntactic constructs like $s;E$ or $E \| s$ are *not* in L_0.

Now that we have formulated the notion of guardedness for x, we can easily generalise this:

DEFINITION 1.3 (Guarded statements): The set L_0^g of *guarded* statements (guarded for all x) is defined as $L_0^g = \cap_{x \in Stmv} L_0^x$.

As L_0 and L_0^x, also L_0^g has a simple inductive structure:

LEMMA 1.4: *The set L_0^g can be given by the following syntax:*

$$t ::= a \mid t;s \mid t_1 \cup t_2 \mid t_1 \| t_2 \mid \mu x[t]$$

where $s \in L_0$.

We have the usual notion of *free variables,* that is, variables that are not bound by any operator μ. A statement containing no free variables is called *closed.* We define for $L = L_0$, L_0^x, and L_0^g: $L^{cl} = \{s \mid s \in L \mid \text{closed } (s)\}$. We have: $(L_0)^{cl} = (L_0^x)^{cl} = (L_0^g)^{cl}$.

We expect that the reader may benefit from a few examples. First, we observe that $L_0^g \subseteq L_0^x \subseteq L_0$. Further, we have that

$$x \in L_0,\ x \notin L_0^x,\ y;x \in L_0^x,\ y;x \notin L_0^y$$

$$\mu x[y;x] \in L_0,\ \mu y[y;x] \notin L_0,\ a;\mu x[y;x] \in L_0^x \cap L_0^y,\ \mu y[a;\mu x[y;x]] \in L_0.$$

1.2 *Operational semantics*

We first introduce a semantic universe for both the operational and the denotational semantics for L_0.

DEFINITION 1.5 (Semantic universe P_0): Let A^∞, the set of finite and infinite words over A, be given by $A^\infty = A^* \cup A^\omega$. For the empty word we use the special symbol ϵ. Let d_{A^∞} denote the usual metric on A^∞. We define $P_0 = \mathcal{P}_{nc}(A^\infty)$, with typical elements $p, q, \ldots$, the set of all non-empty, compact subsets of A^∞. As a metric on P_0 we take $d_{P_0} = (d_{A^\infty})_H$, the Hausdorff distance induced by d_{A^∞}. We have that P_0 together with the metric d_{P_0} is a complete metric space.

The operational semantics for L_0 is based on the notion of a *transition relation:*

DEFINITION 1.6 (Transition relation for L_0^g): We define a transition relation $\rightarrow \subseteq L_0^g \times A \times L_0$ (writing $s \xrightarrow{a} s'$ for $(s,a,s') \in \rightarrow$) as the smallest relation satisfying

(i) $a \xrightarrow{a} E$ (for all $a \in A$)

(ii) for all $a \in A$, $s,t \in L_0^g$, $s',\bar{s} \in L_0$: if $s' \not\equiv E$, then:

$$s \xrightarrow{a} s' \Rightarrow (s;\bar{s} \xrightarrow{a} s';\bar{s} \ \wedge\ s \cup t \xrightarrow{a} s' \ \wedge\ t \cup s \xrightarrow{a} s'$$
$$\wedge\ s \| t \xrightarrow{a} s' \| t \ \wedge\ t \| s \xrightarrow{a} t \| s' \ \wedge\ \mu x[s] \xrightarrow{a} s'[\mu x[s]/x]),$$

where the latter statement is obtained by replacing all free occurrences of x in s by $\mu x[s]$; and if $s' = E$, then:

$$s \xrightarrow{a} E \Rightarrow (s;\bar{s} \xrightarrow{a} \bar{s} \ \wedge\ s \cup t \xrightarrow{a} E \ \wedge\ t \cup s \xrightarrow{a} E$$
$$\wedge\ s \| t \xrightarrow{a} t \ \wedge\ t \| s \xrightarrow{a} t \ \wedge\ \mu x[s] \xrightarrow{a} E).$$

Intuitively, $s \xrightarrow{a} s'$ tells us that s can do the elementary action a as a first step, resulting in the statement s'. We now give the definition of $\mathcal{O}_0$, the operational semantics for L_0^{cl}. (It is defined on *closed* statements only, because we do not want to give an operational meaning to, e.g., $a;x$: what should it be?) It will be the fixed point of the following contraction.

DEFINITION 1.7 (Φ_0): Let $\Phi_0 : (L_0^{cl} \rightarrow P_0) \rightarrow (L_0^{cl} \rightarrow P_0)$ be given, for $F \in L_0^{cl} \rightarrow P_0$ and $s \in L_0^{cl}$, by

$$\Phi_0(F)(s) = \begin{cases} \{\epsilon\} & \text{if } s = E \\ \bigcup \{a \cdot F(s') \mid s' \in L_0^{cl} \wedge a \in A \wedge s \xrightarrow{a} s'\} & \text{if } s \not\equiv E. \end{cases}$$

It is straightforward to prove that Φ_0 is contracting. (As a metric on $L_0^{cl} \rightarrow P_0$ we take $d(F_1,F_2) = \sup_{s \in L_0^{cl}} \{d_{P_0}(F_1(s),F_2(s))\}$.)

DEFINITION 1.8 : $\mathcal{O}_0 = \text{Fixed Point}(\Phi_0)$

We give yet another characterisation of $\mathcal{O}_0$. It is based on the following definition and will be the one we use in proving semantic equivalence.

DEFINITION 1.9 (Initial steps): We define a function $I : L_0^g \rightarrow \mathcal{P}_{fin}(A \times L_0)$ (where $\mathcal{P}_{fin}(X) = \{Y \mid Y \subseteq X \wedge \text{finite } (Y)\}$) by induction on L_0^g:

(i) $I(E) = \varnothing$, and $I(a) = \{(a, E)\}$

(ii) Suppose $I(s) = \{(a_i, s_i)\}$, $I(t) = \{(b_j, t_j)\}$ for $s,t \in L_0^g$, $a_i,b_j \in A$, $s_i,t_j \in L_0$. (The variables i and j range over some finite sets of indices, which we have omitted.) Then

$$I(s;\bar{s}) = \{(a_i, s_i;\bar{s})\} \text{ (for } \bar{s} \in L_0), \quad I(s \cup t) = I(s) \cup I(t)$$
$$I(s \| t) = \{(a_i, s_i \| t)\} \cup \{(b_j, s \| t_j)\}, \quad I(\mu x[s]) = \{(a_i, s_i[\mu x[s]/x])\}.$$

This definition is motivated by the following lemma, which can be easily proved.

LEMMA 1.10: $\forall a \in A \forall s \in L_0^g \forall s' \in L_0 \ [s \xrightarrow{a} s' \Leftrightarrow (a, s') \in I(s)]$

COROLLARY 1.11: $\Phi_0(F)(s) = \bigcup \{a \cdot F(s') \mid (a,s') \in I(s)\}$, for $F: L_0^{cl} \to P_0$, $s \in L_0^{cl} \setminus \{E\}$.

1.3 *Denotational semantics*

The second semantic function we define for L_0 will be *denotational:* We call a semantic function $F: L_0 \to M$ (where M is some mathematical domain) denotational if it is compositionally defined and tackles recursion with the help of fixed points. The first condition is satisfied if for every syntactic operator *op* in L_0 we can define a corresponding semantic operator $\tilde{op}: M \times M \to M$ (assuming *op* to be binary) such that

$$F(s_1 op s_2) = F(s_1) \tilde{op} F(s_2).$$

As semantic domain for the denotational semantics of L_0 we take again P_0. The semantic operators corresponding with ; , $\cup$ and $\|$, the syntactic operators in L_0, will be of type $P_0 \times P_0 \to P_0$.

DEFINITION 1.12 (Semantic operators)
The operators $\tilde{;}$, $\tilde{\cup}$, $\tilde{\|}$: $P_0 \times P_0 \to P_0$ are defined as follows. Let $p, q \in P_0$, then

$$(i) \quad p \tilde{;} q = \begin{cases} q & \text{if } p = \{\epsilon\} \\ \bigcup \{a \cdot (p_a \tilde{;} q) \mid p_a \neq \varnothing\} & \text{otherwise} \end{cases}$$

$$(ii) \quad p \tilde{\cup} q = p \cup q \quad \text{(set-theoretic union)}$$

$$(iii) \quad p \tilde{\|} q = \begin{cases} p & \text{if } q = \{\epsilon\} \\ q & \text{if } p = \{\epsilon\} \\ \bigcup \{a \cdot (p_a \tilde{\|} q) \mid p_a \neq \varnothing\} \cup \bigcup \{a \cdot (p \tilde{\|} q_a) \mid q_a \neq \varnothing\} & \text{otherwise,} \end{cases}$$

where for every $p \in P_0$ and $a \in A$ we define:

$$p_a = \{w \mid w \in A^\infty \wedge a \cdot w \in p\}.$$

(We often write *op* rather than $\tilde{op}$ if no confusion is possible.)

REMARKS 1.13

(1) Definitions (i) and (iii) are self-referential and need some justification. We shall give it for $\tilde{;}$ and leave the case of $\tilde{\|}$ to the reader. We define a mapping: $\Phi: (P_0 \times P_0 \to P_0) \to (P_0 \times P_0 \to P_0)$ by

$$\Phi(F)(p,q) = \begin{cases} q & \text{if } p = \{\epsilon\} \\ \bigcup \{a \cdot F(p_a, q) \mid p_a \neq \varnothing\} & \text{otherwise.} \end{cases}$$

It is not difficult to show that Φ is contracting. Then we define: $\tilde{;} = \text{Fixed Point}(\Phi)$, which satisfies the equation of definition 1.16 above.

(2) If we define the *left-merge* operator $\lfloor\!\lfloor$ by

$$p \lfloor\!\lfloor q = \begin{cases} \varnothing & \text{if } p = \{\epsilon\} \\ \bigcup \{a \cdot (p_a \| q) \mid p_a \neq \varnothing\} & \text{otherwise,} \end{cases}$$

then we will have that $p \| q = p \lfloor\!\lfloor q \cup q \lfloor\!\lfloor p$ (using the fact that $p' \| q' = q' \| p'$, for all p' and q'). This abbreviation will be helpful in some future proofs.

We shall treat recursion with the help of environments, which are used to store and retrieve meanings of statement variables. They are defined in

DEFINITION 1.14 (Semantic environments)
The set Γ of *semantic environments*, with typical elements γ, is given by $\Gamma = Stmv \rightarrow^{fin} P_0$. We write $\gamma\{p/x\}$ for a *variant* of γ which is like γ but with $\gamma\{p/x\}(x)=p$.

Now we have defined everything we need to introduce the denotational semantics for L_0.

DEFINITION 1.15 (Ψ_0, D_0)
We shall define D_0 as the fixed point of $\Psi_0:(L_0 \rightarrow \Gamma \rightarrow {}^1P_0) \rightarrow (L_0 \rightarrow \Gamma \rightarrow {}^1P_0)$, which is given by induction on L_0. (Here $\Gamma \rightarrow {}^1P_0$ denotes the set of non-distance-increasing functions from Γ to P_0.) Let $F \in L_0 \rightarrow \Gamma \rightarrow {}^1P_0$, then:

$$(i) \quad \Psi_0(F)(a)(\gamma)=\{a\},\ \Psi_0(F)(x)(\gamma)=\gamma(x),\ \Psi_0(F)(E)(\gamma)=\{\epsilon\}$$

$$(ii) \quad \Psi_0(F)(s\ op\ t)(\gamma) = \Psi_0(F)(s)(\gamma)\ \tilde{op}\ \Psi_0(F)(t)(\gamma)$$

$$(iii) \quad \Psi_0(F)(\mu x[s])(\gamma) = \Psi_0(F)(s)(\gamma\{F(\mu x[s])(\gamma)/x\}) \text{ for } s \in L_0^x,$$

for $op = ;,\ \cup,\ \|$, and $\tilde{op}$ as in definition 1.12. (We define $\Psi_0(F)$ only for those s and γ, such that $FV(s) \subseteq dom(\gamma)$.) Now we set D_0 = Fixed Point(Ψ_0).

We have: $D_0[\![\mu x[s]]\!](\gamma) = D_0[\![s]]\!](\gamma\{D_0[\![\mu x[s]]\!](\gamma)/x\})$.
The fact that we consider only *guarded* recursion is essential for the proof (omitted here) that Ψ_0 is contracting.

1.4 *Semantic equivalence of* $\mathcal{O}_0$ **and** $\mathcal{D}_0$

An important difference between $\mathcal{D}_0$ and $\mathcal{O}_0$ is that recursion is treated with and without semantic environments, respectively. We have

$$\mathcal{D}_0[\![\mu x[s]]\!](\gamma) = \mathcal{D}_0[\![s]]\!](\gamma\{\mathcal{D}_0[\![\mu x[s]]\!](\gamma)/x\})$$

and

$$\mathcal{O}_0[\![\mu x[s]]\!] = \mathcal{O}_0[\![s[\mu x[s]/x]]\!].$$

In the latter case the statement $\mu x[s]$ is *syntactically* substituted for all free statement variables x in s, whereas in the first case the environment γ is changed by setting x to the *semantic* value of $\mu x[s]$. We shall compare $\mathcal{O}_0$ and $\mathcal{D}_0$ by relating both to an intermediate semantic function $\mathcal{O}_0'$, which takes *syntactic* instead of *semantic* environments as arguments. It will be defined such that for syntactic environments δ:

$$\mathcal{O}_0'[\![\mu x[s]]\!](\delta) = \mathcal{O}_0'[\![s]]\!](\delta\{\mu x[s]/x\}).$$

Here δ is changed, the new value of x is the statement $\mu x[s]$. By first comparing $\mathcal{O}_0$ and $\mathcal{O}_0'$ and next $\mathcal{O}_0'$ and $\mathcal{D}_0$ we are able to prove the main result of this section: $\mathcal{O}_0[\![s]]\!] = D_0[\![s]]\!](\gamma)$, for all $s \in L_0^{cl}$ and arbitrary $\gamma \in \Gamma$. For the definition of $\mathcal{O}_0'$, we need

DEFINITION 1.16 (Syntactic environments)
The set Δ of *syntactic* environments, with typical elements δ, is defined by

$$\Delta = \{\delta \mid \delta \in (Stmv \rightarrow^{\text{fin}} L_0) \wedge (\delta \text{ is normal})\},$$

where the notion of *normal* environments is given in:

DEFINITION 1.17 (Normal environments)
A syntactic environment δ is called *normal* whenever

(i) $\forall x \in dom(\delta)\ [\delta(x) \in L_0^x]$

(ii) $\forall s \in L_0\ [FV(s) \subseteq dom(\delta) \Rightarrow \exists k \geqslant 0\ [s[\delta]^k \in L_0^{cl}]]$,

where $s[\delta]^0 = s$, $s[\delta]^1 = s[\delta(x_1)/x_1, \ldots, \delta(x_n)/x_n]$ (with $FV(s) = \{x_1, \ldots, x_n\}$) and $s[\delta]^{n+1} = (s[\delta])[\delta]^n$. For δ normal and $s \in L_0$, with $FV(s) \subseteq dom(\delta)$, we define $s\langle\delta\rangle = s[\delta]^k$, where $k = \min\{m \mid s[\delta]^m \in L_0^{cl}\}$.

REMARKS 1.18

(1) From now on we shall assume whenever we consider $s \in L_0$ and $\delta \in \Delta$ together (as two arguments for a function, or as a pair) that $FV(s) \subseteq dom(\delta)$.

(2) Let $\delta \in Stmv \to^{fin} L_0$ be such that for $x, y \in Stmv$: $\delta(x) = y$ and $\delta(y) = x$. Such an environment is *not* normal. It does not give us any useful information about the values of x and y.

(3) It would be too restrictive to require for all $\delta \in Stmv \to^{fin} L_0$ that $\forall x \in dom(\delta)\ [x[\delta] \in L_0^{cl}]$. An example may illustrate this. Let δ be defined such that $dom(\delta) = \{x, y\}$, and

$$\delta(y) = \mu y[b; x; y], \quad \delta(x) = \mu x[a; \mu y[b; x; y]].$$

Such an environment we shall encounter when computing $\mathcal{O}_0'[\![\mu x[a; \mu y[b; x; y]]]\!]$. Now $y[\delta] = \delta(y) \notin L_0^{cl}$, but $y[\delta]^2 \in L_0^{cl}$.

Now that we have introduced syntactic environments, we can formulate a principle of induction for the set $L_0 \times \Delta$, which we shall heavily use in the sequel.

THEOREM 1.19 (*Induction principle for* $L_0 \times \Delta$): *Let* $\Xi \subseteq L_0 \times \Delta$. *If:*

(1) $A \times \Delta \subseteq \Xi$

(2) $\{s, t\} \times \Delta \subseteq \Xi \Rightarrow \{s; \bar{s}, s \cup t, s \| t\} \times \Delta \subseteq \Xi$ *for* $s, t, \bar{s} \in L_0$

(3) $\{s\} \times \Delta \subseteq \Xi \Rightarrow \{\mu x[s]\} \times \Delta \subseteq \Xi$ *for* $s \in L_0^{\times}$

(4) $(\delta(x), \delta) \in \Xi \Rightarrow (x, \delta) \in \Xi$ *for* $x \in Stmv$ *and* $\delta \in \Delta$,

then:

$$\Xi = L_0 \times \Delta.$$

We cannot reason about a free statement variable x unless we know what statement it is bound to. Therefore, we consider non-closed statements together with syntactic environments, which give information about the free variables they contain. This explains why we have formulated an induction principle for $L_0 \times \Delta$ instead of L_0 only.

Now let $\Xi \subseteq L_0 \times \Delta$. The first three conditions of the principle suffice to prove that $L_0^g \times \Delta \subseteq \Xi$, since they express exactly the syntactic structure of L_0^g (see lemma 1.4). (We have chosen L_0^g here instead of L_0^{cl}, because the latter set has no simple inductive structure.) Thus also $L_0^{cl} \times \Delta$ ($\subseteq L_0^g \times \Delta$) $\subseteq \Xi$. Adding condition (4) enables us to prove $L_0 \times \Delta \subseteq \Xi$. This may be motivated by the following. For every statement $s \in L_0$ and normal environment $\delta \in \Delta$ there exists an $l \in \mathbb{N}$ such that $s[\delta]^l \in L_0^{cl} \subseteq L_0^g$. Let us call $k \in \mathbb{N}$ with $k = \min\{l \mid s[\delta]^l \in L_0^{cl}\}$ the *degree of closedness* of s with respect to δ. Please note that every $s \in L_0^{cl}$ has degree 0, and arbitrary $s \in L_0$ has, for arbitrary δ, a finite degree. Therefore, this degree can be used as a measure for the complexity of statements. Our induction principle is indeed a principle of induction on the degree of closedness. Conditions (1), (2), and (3) are sufficient to prove Ξ for all (s, δ) with degree 0. They form, so to speak, the basis of the principle. Condition (4) expresses the "step part": if Ξ holds for $(\delta(x), \delta)$, which has degree k, say, then Ξ holds for (x, δ), which then has degree $k + 1$.

We now proceed with the definition of $\mathcal{O}_0'$. It will be of type $\mathcal{O}_0' : L_0 \to \Delta \to P_0$, which could be called intermediate between $\mathcal{O}_0 : L_0^{cl} \to P_0$ and $D_0 : L_0 \to \Gamma \to P_0$. Instead of basing the definition of $\mathcal{O}_0'$ on some transition relation (as in definition 1.7) we use a variant of the initial step function (definition 1.9).

DEFINITION 1.20 (Initial steps with syntactic environments): We define a function $I' : L_0 \to \Delta \to \mathcal{P}_{fin}(A \times L_0 \times \Delta)$, using the induction principle for $L_0 \times \Delta$. The predicate $\Xi \subseteq L_0 \times \Delta$ we use is defined as:

$$\Xi(s,\delta) \equiv I'(s)(\delta) \text{ is defined.}$$

We shall define I' such that Ξ satisfies the induction conditions. Thus we ensure that I' is defined for every $s \in L_0$ and $\delta \in \Delta$ (with $FV(s) \subseteq dom(\delta)$).

(1) $I'(E)(\delta) = \varnothing$, and $I'(a)(\delta) = \{(a, E, \delta)\}$, for all $a \in A$, $\delta \in \Delta$.

(2) Suppose $I'(s) = \lambda\delta \cdot \{(a_i, s_i, \delta_i)\}$, $I'(t) = \lambda\delta \cdot \{(b_j, t_j, \delta_j)\}$ for $s, t, s_i, t_j \in L_0$, $a_i, b_j \in A$, $\delta_i, \delta_j \in \Delta$. (The variables i and j range over some finite sets of indices, which are omitted.) Then:

$$I'(s;\overline{s})(\delta) = \{(a_i, s_i;\overline{s}, \delta_i)\} \quad (\textit{for } \overline{s} \in L_0)$$

$$I'(s \cup t)(\delta) = I'(s)(\delta) \cup I'(t)(\delta)$$

$$I'(s \| t)(\delta) = \{(a_i, s_i \| t, \delta_i)\} \cup \{(b_j, s \| t_j, \delta_j)\}$$

(3) For the definition of $I'(\mu x[s])$ we have to consider possible clashes of variables. Therefore, we distinguish between two cases (supposing that $I'(s)$ has already been defined):

$$I'(\mu x[s])(\delta) = \begin{cases} I'(s)(\delta\{\mu x[s]/x\}) & \text{if } x \notin dom(\delta) \\ I'(\overline{s})(\delta\{\mu\overline{x}[\overline{s}]/\overline{x}\}) & \text{if } x \in dom(\delta), \end{cases}$$

where $\overline{x}$ is some fresh variable with $\overline{x} \notin dom(\delta)$ and $\overline{s} = s[\overline{x}/x]$.

(4) Suppose $I'(\delta(x))(\delta)$ has already been defined. We set:

$$I'(x)(\delta) = I'(\delta(x))(\delta).$$

Note that if $I'(s)(\delta) = \{(a_i, s_i, \delta_i)\}$, then for all i and $x \in Stmv$: if $x \in dom(\delta) \cap dom(\delta_i)$, then $\delta(x) = \delta_i(x)$.

DEFINITION 1.21 (Φ_0'): We define $\Phi_0' : (L_0 \to \Delta \to P_0) \to (L_0 \to \Delta \to P_0)$ by

$$\Phi_0'(F)(s)(\delta) = \begin{cases} \{\epsilon\} & \text{if } s = E \\ \bigcup \{a \cdot F(s')(\delta') \mid (a, s', \delta') \in I'(s)(\delta)\} & \text{otherwise} \end{cases}$$

for $F \in L_0 \to \Delta \to P_0$, $s \in L_0$, and $\delta \in \Delta$ with $FV(s) \subseteq dom(\delta)$.

DEFINITION 1.22: $\mathcal{O}_0' = \text{Fixed Point}(\Phi_0')$

Next, we compare $\mathcal{O}_0$ and $\mathcal{O}_0'$. We can do this by relating I and I', since we have:

$$\mathcal{O}_0[\![s]\!] = \bigcup \{a \cdot \mathcal{O}_0[\![s']\!] \mid (a, s') \in I(s)\}, \text{ for } s \in L_0^{cl},\ s \neq E$$

$$\mathcal{O}_0'[\![s]\!](\delta) = \bigcup \{a \cdot \mathcal{O}_0'[\![s']\!](\delta') \mid (a, s', \delta') \in I'(s)(\delta)\}, \text{ for } s \in L_0,\ s \neq E,\ \delta \in \Delta.$$

THEOREM 1.23 (*Relating I and I'*): *For all $s \in L_0$ and $\delta \in \Delta$, with $FV(s) \subseteq dom(\delta)$, we have:*

$$\forall a \in A \forall s' \in L_0 \forall \delta' \in \Delta\ [(a, s', \delta') \in I'(s)(\delta) \Leftrightarrow (a, s' <\delta'>) \in I(s <\delta>)].$$

(For the definition of $s <\delta>$ see definition 1.17.)

The proof should generalise the intuitively obvious fact that, for s with x occurring freely in s: $(a, s', \delta') \in I'(s)(\delta\{\mu x[s]/x\}) \Leftrightarrow (a, s' <\delta'>) \in I(s[\mu x[s]/x])$.

We formulate the relation of $\mathcal{O}_0$ and $\mathcal{O}_0'$ in terms of their defining contractions Φ_0 and Φ_0'. This can be elegantly done using the following

DEFINITION 1.24: We define $<>:(L_0^{cl} \to P_0) \to (L_0 \to \Delta \to P_0)$, for every $F \in L_0^{cl} \to P_0$, by

$$<>(F) = F^{<>} \text{ (notation)} = \lambda s \in L_0 \cdot \lambda \delta \in \Delta \cdot F(s<\delta>).$$

This mapping links two kinds of semantic functions, of which the first uses syntactic environments whereas the second does not. If $F \in L_0^{cl} \to P_0$, then $F^{<>}$ is a in a sense *extended* version of F: it can take as an argument also statements $s \in L_0$ that are *not* closed, provided it is supplied with a syntactic environment, which is to give the (syntactic) values for the free variables in s.

THEOREM 1.25 *(Relating Φ_0 and Φ_0')*: $\forall F \in L_0^{cl} \to P_0\ [\Phi_0'(F^{<>}) = (\Phi_0(F))^{<>}]$

PROOF: The theorem is an immediate consequence of theorem 1.23. Let $F \in L_0^{cl} \to P_0$, let $s \in L_0$, $s \neq E$.

$$\begin{aligned}
\Phi_0'(F^{<>})(s)(\delta) &= \bigcup \{a \cdot F^{<>}(s')(\delta') \mid (a,s',\delta') \in I'(s)(\delta)\} \\
&= \bigcup \{a \cdot F(s'<\delta'>) \mid (a,s',\delta') \in I'(s)(\delta)\} \\
&= [\text{theorem 1.23}]\ \bigcup \{a \cdot F(s'<\delta'>) \mid (a,s'<\delta'>) \in I(s<\delta>)\} \\
&= \Phi_0(F)(s<\delta>) = (\Phi_0(F))^{<>}(s)(\delta).
\end{aligned}$$

Because Φ_0 and Φ_0' are contractions with $\mathcal{O}_0$ and $\mathcal{O}_0'$ as their respective fixed points, we have:

COROLLARY 1.26 ($\mathcal{O}_0' = \mathcal{O}_0^{<>}$): $\forall s \in L_0\ \forall \delta \in \Delta\ [\mathcal{O}_0'[\![s]\!](\delta) = \mathcal{O}_0[\![s<\delta>]\!]]$.

Finally we relate $\mathcal{O}_0' : L_0 \to \Delta \to P_0$ and $\mathcal{D}_0 : L_0 \to \Gamma \to P_0$. For this purpose we define the following mapping:

DEFINITION 1.27: We define $\sim : (L_0 \to \Gamma \to P_0) \to (L_0 \to \Delta \to P_0)$ by:

$$\sim(F) = \tilde{F} \text{ (notation)} = \lambda s \in L_0 \cdot \lambda \delta \in \Delta \cdot F(s)(\tilde{\delta}^F)$$

for $F \in L_0 \to \Gamma \to P_0$, where $\tilde{\delta}^F$ is given by $\tilde{\delta}^F = \lambda x \in dom(\delta) \cdot F(\delta(x))(\tilde{\delta}^F)$. (We often write $\tilde{\delta}$ rather than $\tilde{\delta}^F$ if from the context it is clear which F should be taken.)

We have to justify the self-referential definition of $\tilde{\delta}$. For this purpose we define

$$\Xi(s,\delta) \equiv \forall x \in FV(s)\ [s \notin L_0^x \to (\tilde{\delta}(x) \text{ is well defined})],$$

for $s \in L_0$ and $\delta \in \Delta$, and use the induction principle to prove: $\Xi = L_0 \times \Delta$. Then it follows for all $x \in Stmv$ with $x \in dom(\delta)$ that $\tilde{\delta}(x)$ is well defined. Conditions (1) through (3) of the induction principle are trivially fulfilled. We prove condition (4). Suppose $(\delta(x),\delta) \in \Xi$. Thus $\tilde{\delta}(y)$ is well defined for all $y \in FV(\delta(x))$. This implies that $\tilde{\delta}(x)$ is well defined, since $\tilde{\delta}(x) = F(\delta(x))(\delta)$.

In the same way as $<>$, also $\sim$ links two different kinds of semantic functions, one using *syntactic*, and the other using *semantic* environments. Again $\tilde{F}$ is an extended version of F in the sense that it takes syntactic environments as an argument instead of semantic ones. In the definition above, a syntactic environment $\delta \in \Delta$ is changed into a *semantic version* (according to the semantic function F) $\tilde{\delta}$ of it, which then is supplied as an argument to F.

Next, we come to the main theorem of this chapter. It relates the denotational semantics $\mathcal{D}_0$ and the operational semantics $\mathcal{O}'_0$, which is a fixed point of Φ'_0, by stating that also $\tilde{\mathcal{D}}_0$ is a fixed point of Φ'_0. From this it follows that $\mathcal{O}_0' = \tilde{\mathcal{D}}_0$.

THEOREM 1.28: $\Phi_0'(\tilde{\mathcal{D}}_0) = \tilde{\mathcal{D}}_0$

PROOF: Let $\Xi \subseteq L_0 \times \Delta$ be defined by

$$\Xi(s,\delta) \equiv \Phi_0'(\tilde{\mathcal{D}}_0)(s)(\delta) = \tilde{\mathcal{D}}_0(s)(\delta)$$

for $(s,\delta) \in L_0 \times \Delta$. We use the induction principle for $L_0 \times \Delta$ to show that $\Xi = L_0 \times \Delta$. Let $\delta \in \Delta$.

(1) For $a \in A$ we have $\Phi_0'(\tilde{\mathcal{D}}_0)(a)(\delta) = \{a\} = \tilde{\mathcal{D}}_0(a)(\delta)$, so $A \times \Delta \subseteq \Xi$.

(2) Let $s,t \in L_0$ and suppose $\Xi(s,\delta)$ and $\Xi(t,\delta)$. We show: $\Xi(s \| t, \delta)$.

$$\begin{aligned}
\Phi_0'(\tilde{\mathcal{D}}_0)(s\|t)(\delta) &= \text{[definition } \Phi_0' \text{ and } I'(s\|t)\text{]} \\
&\quad \bigcup \{a' \cdot \tilde{\mathcal{D}}_0(s'\|t)(\delta') \mid (a',s',\delta') \in I'(s)(\delta)\} \cup \\
&\quad \bigcup \{b' \cdot \tilde{\mathcal{D}}_0(s\|t')(\delta') \mid (b',t',\delta') \in I'(t)(\delta)\} \\
&= \bigcup \{a' \cdot (\tilde{\mathcal{D}}_0(s')(\delta') \| \tilde{\mathcal{D}}_0(t)(\delta')) \mid (a',s',\delta') \in I'(s)(\delta)\} \cup \\
&\quad \bigcup \{b' \cdot (\tilde{\mathcal{D}}_0(s)(\delta') \| \tilde{\mathcal{D}}_0(t')(\delta')) \mid (b',t',\delta') \in I'(t)(\delta)\} \\
&= \text{[see remark after definition 1.20]} \\
&\quad \bigcup \{a' \cdot (\tilde{\mathcal{D}}_0(s')(\delta') \| \tilde{\mathcal{D}}_0(t)(\delta)) \mid (a',s',\delta') \in I'(s)(\delta)\} \cup \\
&\quad \bigcup \{b' \cdot (\tilde{\mathcal{D}}_0(s)(\delta) \| \tilde{\mathcal{D}}_0(t')(\delta')) \mid (b',t',\delta') \in I'(t)(\delta)\} \\
&= \text{[definition } \mathbin{\mathbb{L}} \text{ (see remark 1.13(2))]} \\
&\quad ((\bigcup \{a' \cdot \tilde{\mathcal{D}}_0(s')(\delta') \mid (a',s',\delta') \in I'(s)(\delta)\}) \mathbin{\mathbb{L}} \tilde{\mathcal{D}}_0(t)(\delta)) \cup \\
&\quad ((\bigcup \{b' \cdot \tilde{\mathcal{D}}_0(t')(\delta') \mid (b',t',\delta') \in I'(t)(\delta)\}) \mathbin{\mathbb{L}} \tilde{\mathcal{D}}_0(s)(\delta)) \\
&= \text{[definition } \Phi_0'\text{]} \\
&\quad (\Phi_0'(\tilde{\mathcal{D}}_0)(s)(\delta) \mathbin{\mathbb{L}} \tilde{\mathcal{D}}_0(t)(\delta)) \cup (\Phi_0'(\tilde{\mathcal{D}}_0)(t)(\delta) \mathbin{\mathbb{L}} \tilde{\mathcal{D}}_0(s)(\delta)) \\
&= \text{[we have } \Xi(s,\delta) \text{ and } \Xi(t,\delta)\text{]} \\
&\quad (\tilde{\mathcal{D}}_0(s)(\delta) \mathbin{\mathbb{L}} \tilde{\mathcal{D}}_0(t)(\delta)) \cup (\tilde{\mathcal{D}}_0(t)(\delta) \mathbin{\mathbb{L}} \tilde{\mathcal{D}}_0(s)(\delta)) \\
&= \tilde{\mathcal{D}}_0(s)(\delta) \| \tilde{\mathcal{D}}_0(t)(\delta) = \tilde{\mathcal{D}}_0(s\|t)(\delta).
\end{aligned}$$

This proves $\Xi(s \| t, \delta)$. The cases $\Xi(s;\bar{s},\delta)$ and $\Xi(s \cup t, \delta)$ are very similar.

(3) Let $s \in L_0^x$ and suppose $\{s\} \times \Delta \subseteq \Xi$. We show: $\Xi(\mu x[s],\delta)$. Assume (without loss of generality) that $x \notin dom(\delta)$. Then

$$\begin{aligned}
\Phi_0'(\tilde{\mathcal{D}}_0)(\mu x[s])(\delta) &= \text{[definition } \Phi_0' \text{ and } I'(\mu x[s])(\delta)\text{; let } \delta' = \delta\{\mu x[s]/x\}\text{]} \\
&\quad \bigcup \{a' \cdot \tilde{\mathcal{D}}_0(s')(\delta') \mid (a',s',\delta') \in I'(s)(\delta')\} \\
&= \Phi_0'(\tilde{\mathcal{D}}_0)(s)(\delta') \\
&= \text{[we have } \Xi(s,\delta')\text{]} \quad \tilde{\mathcal{D}}_0(s)(\delta') = \mathcal{D}_0[\![s]\!](\tilde{\delta}') \\
&= \text{[definition } \tilde{\delta}'\text{]} \quad \mathcal{D}_0[\![s]\!](\tilde{\delta}\{\mathcal{D}_0[\![\mu x[s]]\!](\tilde{\delta})/x\}) \\
&= \text{[definition } \mathcal{D}_0\text{]} \quad \mathcal{D}_0[\![\mu x[s]]\!](\tilde{\delta}) = \tilde{\mathcal{D}}_0(\mu x[s])(\delta)
\end{aligned}$$

This proves $\Xi(\mu x[s],\delta)$.

(4) Let $x \in Stmv$, suppose $\Xi(\delta(x),\delta)$. Now

$$\begin{aligned}
\Phi_0'(\tilde{\mathcal{D}}_0)(x)(\delta) &= \text{[definition } \Phi_0' \text{ and } I'(x)(\delta)\text{]} \quad \Phi_0'(\tilde{\mathcal{D}}_0)(\delta(x))(\delta) \\
&= \text{[because } \Xi(\delta(x),\delta)\text{]} \quad \tilde{\mathcal{D}}_0(\delta(x))(\delta) = \mathcal{D}_0[\![\delta(x)]\!](\tilde{\delta}) \\
&= \text{[definition } \tilde{\delta}\text{]} \quad \tilde{\delta}(x) = \mathcal{D}_0[\![x]\!](\tilde{\delta}) = \tilde{\mathcal{D}}_0(x)(\delta).
\end{aligned}$$

Thus $\Xi(x,\delta)$.
The induction principle now implies: $\Xi = L_0 \times \Delta$. □

As an immediate consequence of this theorem, we have

COROLLARY 1.29 ($\mathcal{O}_0' = \tilde{\mathcal{D}}_0$): $\forall s \in L_0\ \forall \delta \in \Delta\ [\mathcal{O}_0'[\![s]\!](\delta) = \mathcal{D}_0[\![s]\!](\tilde{\delta})]$.

The combination of corollaries 1.26 and 1.29 yields the main theorem of this section.

THEOREM 1.30 ($\mathcal{O}_0^{<>} = \tilde{\mathcal{D}}_0$): $\forall s \in L_0\ \forall \delta \in \Delta\ [\mathcal{O}_0[\![s<\delta>]\!] = \mathcal{D}_0[\![s]\!](\tilde{\delta})]$.

COROLLARY 1.31: *For all* $s \in L_0^{cl}$, *and arbitrary* $\gamma \in \Gamma$: $\mathcal{O}_0[\![s]\!] = \mathcal{D}_0[\![s]\!](\gamma)$.

1.5 *Summary of section* 1

It may be useful to give a short overview of this section. We have defined an operational semantics $\mathcal{O}_0$ for L_0 as the fixed point of Φ_0, and a denotational semantics $\mathcal{D}_0$ as the fixed point of Ψ_0. We have related $\mathcal{O}_0$ and $\mathcal{D}_0$ via an intermediate semantic function $\mathcal{O}_0'$, defined as the fixed point of Φ_0'. To be more precise, we have related Φ_0, Ψ_0, and Φ_0' using mappings $<>$ and $\sim$, for which we have proved some properties, schematically represented by the following diagram:

$$\begin{array}{ccc}
 & \overset{\Phi_0}{\rightarrow} & \\
L_0^{cl} \rightarrow P_0 & \rightarrow & L_0^{cl} \rightarrow P_0 \\
<>\downarrow & * & \downarrow<> \\
 & \overset{\Phi_0'}{\rightarrow} & \\
L_0 \rightarrow \Delta \rightarrow P_0 & \rightarrow & L_0 \rightarrow \Delta \rightarrow P_0 \\
\sim\uparrow & *_{fix} & \uparrow\sim \\
 & \overset{\Psi_0}{\rightarrow} & \\
L_0 \rightarrow \Gamma \rightarrow P_0 & \rightarrow & L_0 \rightarrow \Gamma \rightarrow P_0
\end{array}$$

The * in the upper rectangle indicates that it commutes, the symbol $*_{fix}$ in the lower rectangle indicates that it commutes only for the fixed point of Ψ_0 (that is, $\mathcal{D}_0$). Please note that * has been formulated as theorem 1.25, and $*_{fix}$ as theorem 1.28. The main result of section 1 (theorem 1.30) follows from this diagram, because * implies: $\mathcal{O}_0^{<>} = \mathcal{O}_0'$ and $*_{fix}$ implies: $\mathcal{O}_0' = \tilde{\mathcal{D}}_0$.

2. SEMANTIC EQUIVALENCE FOR OTHER LANGUAGES

In [KR], the full paper of which this is an extended abstract, the method of proving semantic equivalence defined in the previous section is applied to two other languages, L_1 and L_2, which we shall briefly describe here. Finally, we shall mention two other parallel languages (POOL and Concurrent Prolog) to which this method has been successfully applied.

For L_1 we introduce some structure to the (possibly infinite) alphabet A of elementary actions. Let $C \subseteq A$ be a subset of so-called *communications*. From now on let c range over C and a,b over A. Similarly to CCS [Mi] or CSP [Ho], we stipulate a bijection $^- : C \rightarrow C$ with $^- \circ ^- = id_C$. It yields for every $c \in C$ a *matching* communication $\overline{c}$, which will be denoted by $\overline{c}$. In $A \setminus C$ we have a special element τ denoting a successful communication. Let *Stmv*, with typical elements $x,y, \ldots$, be again the set of statement variables.

DEFINITION 2.1 (Syntax for L_1): The set L_1, with typical elements $s,t, \ldots$, is given by

$$s ::= a \mid s_1;s_2 \mid s_1 + s_2 \mid s_1 \| s_2 \mid x \mid \mu x[t]$$

where t is guarded for x. Note that $a \in A \supseteq C$.

In [KR], an operational semantics $\mathcal{O}_1 : L_1 \to P_1$ and a denotational semantics $\mathcal{D}_1 : L_1 \to \Gamma \to \overline{P}_1$ are defined along the lines of the previous section. An importance difference is the use of a so-called *reflexive domain* $\overline{P}_1$ for the denotational semantics of L_1 (as will be the case for L_2, described below), being a solution of some domain equation in the style of Plotkin ([Pl1]) and Scott ([Sc]). In a metric setting these domain equations have been treated in [BZ] and [AR]. Then the method of section 1 is straightforwardly generalised. A slight complication in proving the semantic equivalence of $\mathcal{O}_1$ and $\mathcal{D}_1$ is the difference between the domains P_1 and $\overline{P}_1$: the first contains sets of *sequences*, whereas the latter has tree-like structures for its elements. (This difference is sometimes characterised by the terms *linear time* semantics versus *branching time* semantics.) As a consequence of this, we compare $\mathcal{O}_1$ and $\mathcal{D}_1$ via *two* intermediate models instead of just one. It is shown that the following diagram commutes:

$$\begin{array}{lcl}
L_1^{cl} \to P_1 & \overset{\Phi_1}{\to} & L_1^{cl} \to P_1 \\
<> \downarrow & * & \downarrow <> \\
L_1 \to \Delta \to P_1 & \overset{\Phi_1'}{\to} & L_1 \to \Delta \to P_1 \\
\alpha \uparrow & * & \uparrow \alpha \\
L_1 \to \Delta \to \overline{P}_1 & \overset{\Psi_1'}{\to} & L_1 \to \Delta \to \overline{P}_1 \\
\sim \uparrow & *_{fix} & \uparrow \sim \\
L_1 \to \Gamma \to \overline{P}_1 & \overset{\Psi_1}{\to} & L_1 \to \Gamma \to \overline{P}_1
\end{array}$$

where (as in subsection 1.5) * indicates commutativity and $*_{fix}$ indicates commutativity with respect to the fixed point of Ψ_1 (that is, $\mathcal{D}_1$); Φ, Φ', Ψ and Ψ' are suitably defined contractions; and $\alpha : \overline{P}_1 \to P_1$ is an abstraction operation, mapping branching structures onto sets of sequences.

From this diagram it follows that for all $s \in L_1^{cl}$ and arbitrary $\gamma \in \Gamma$:

$$\mathcal{O}_1[\![s]\!] = \alpha(\mathcal{D}_1[\![s]\!](\gamma)).$$

The third language discussed in [KR] is a *nonuniform* language. Elementary actions are no longer uninterpreted but taken as either assignments or tests. Communication actions c and $\overline{c}$ are refined to actions $c?v$ and $c!e$ (with v variable and e an expression), and successful communication now involves two effects: both *synchronisation* (as in the language L_1), and *value passing*: the (current) value of e is assigned to v. Thus, we have here the synchronous handshaking variety of message passing in the sense of CCS or CSP. In the definition of the syntax of L_2, we need three new syntactic categories, viz.: the set *Var*, with elements v,w, of *individual variables*; the set *Exp*, with elements e, of *expressions*; and the set *Bexp*, with elements b, of *boolean expressions*. We shall not specify a syntax for *Exp* and *Bexp*. We assume that (boolean) expressions are of an elementary kind; in particular, they have no side effects and their evaluation always terminates. Statement variables $x,y, \ldots$ are as before, as are the communications $c \in C$. The latter now appear syntactically as part of value passing communication actions $c?v$ or $c!e$.

DEFINITION 2.2 (Syntax for L_2)

$$s ::= v := e \mid b \mid c?v \mid c!e \mid s_1;s_2 \mid s_1 + s_2 \mid s_1 \| s_2 \mid x \mid \mu x[t],$$

where t is guarded for x.

The semantic domains for the operational semantics $\mathcal{O}_2$ and the denotational semantics $\mathcal{D}_2$ of L_2 are somewhat more intricate than the semantics of L_1: they now involve the notion of *state*. But this turns out to be irrelevant for the proof of the semantic equivalence of $\mathcal{O}_2$ and $\mathcal{D}_2$ (defined similarly to $\mathcal{O}_1$ and $\mathcal{D}_1$), which is established in exactly the same way as for L_1.

We would like to conclude this section by mentioning two other examples. Along the lines of this paper, we have proved the semantic equivalence of an operational and a denotational semantics (defined in [ABKR1] and [ABKR2], respectively) for POOL, which is an acronym for parallel object-oriented language (defined in [Am]). At first sight, these two semantics seem to be quite different; a major problem is the fact that the denotational semantics uses *continuations*, whereas the operational semantics does not. Moreover, the denotational semantics has for its semantic domain a rather intricate version of Plotkin's domain of resumptions ([Pl1]). Nevertheless, also this language fits smoothly into our approach (see [Ru2]). Finally, it is also possible to apply the method in the domain of (concurrent) logic programming. In [Ko1], a compositional semantics for Concurrent Prolog is defined. The main idea is to describe the meaning of a program with the help of *substitution* transforming processes rather than state transforming processes. This semantics can be related to an operational one, which is based on the use of transition systems, by the method described above. There are some complications due to the atomic execution of guards, but the skeleton of the proof remains the same. The result will be described in [Ko2].

3. Conclusions

We have developed a uniform method of comparing different semantic models for imperative concurrent programming languages. We have defined operational and denotational semantic models for such languages as fixed points of contractions on complete metric spaces, and have related them by relating their corresponding contractions. Here, we benefit from the metric structure of the underlying mathematical domains, which ensures the uniqueness of the fixed point of such contractions (Banach's theorem). It turns out that once this method has been applied to a certain (simple) language (L_0), it can be easily generalised to more complex languages (L_1 and L_2). This we consider to be the strength of our approach. Recently, we have investigated possible extensions of this method to deal with yet other languages, containing, e.g., program constructs for process creation. This has resulted (in [Ru2]) in an equivalence proof for POOL, a parallel object-oriented language defined in [Am]. An equivalence proof for Concurrent Prolog will be presented in [Ko2]. In [BM], a number of concurrent languages, containing constructs for simultaneous recursion, is presented for which equivalence proofs are given along the lines of this paper.

Our investigations are related to the question of *full abstraction*, which at the same time is a topic for further research. If L is a language with semantics $\mathcal{O}$ and $\mathcal{D}$, then we call $\mathcal{D}$ *fully abstract with respect to* $\mathcal{O}$ if

$$\forall s \in L \forall t \in L\ [\mathcal{D}[\![s]\!] = \mathcal{D}[\![t]\!] \Leftrightarrow \forall C(\cdot)\ [\mathcal{O}[\![C(s)]\!] = \mathcal{O}[\![C(t)]\!]],$$

where $C(\cdot)$ ranges over the set of *contexts* for L, that is, the set of statements in L containing one or more holes. An example would be $s;(\cdot)$, where $(\cdot)$ denotes the hole. Given such a context $C(\cdot)$ and a statement s, the statement $C(s)$ is obtained by substituting s for all the holes in $C(\cdot)$. The issue of full abstraction is mostly raised with respect to a model $\mathcal{O}$ that is *operational*, expressing a notion of observability, and a model $\mathcal{D}$ that is *compositional*. Then it follows from a relation between $\mathcal{O}$ and $\mathcal{D}$ of the form $\mathcal{O} = \alpha \circ \mathcal{D}$ that for all s and $t \in L$:

$$\mathcal{D}[\![s]\!] = \mathcal{D}[\![t]\!] \Rightarrow \forall C(\cdot)\ [\mathcal{O}[\![C(s)]\!] = \mathcal{O}[\![C(t)]\!]].$$

(This property is sometimes called: *correctness* of $\mathfrak{D}$ with respect to $\mathfrak{O}$.) Thus, our result of proving $\mathfrak{O}=\alpha\circ\mathfrak{D}$ partly solves the problem of full abstraction. In [Ru1], a semantics for a simple language like L_0 is defined with the use of *failure* sets (introduced in [BHR]), which is shown to be fully abstract with respect to a given operational semantics.

4. References

[Am] P. America, *Definition of the programming language POOL-T,* Esprit project 415, Doc. No. 91, Philips Research Laboratories, Eindhoven, September 1985.

[ABKR1] P. America, J. de Bakker, J. Kok, J.J.M.M. Rutten, *Operational semantics of a parallel object-oriented language,* Conference record of the 13th Symposium on Principles of Programming Languages, St. Petersburg, Florida, January 1986, pp.194-208.

[ABKR2] P. America, J. de Bakker, J. Kok, J.J.M.M. Rutten, *Denotational semantics of a parallel object-oriented language,* Technical Report (CS-R8626), Centre for Mathematics and Computer Science, Amsterdam, 1986. (To appear in: Information and Computation.)

[AP] K. Apt, G. Plotkin, *Countable nondeterminism and random assignment,* Journal of the Association for Computing Machinery, Vol. 33, No. 4, October 1986, pp. 724-767.

[AR] P. America, J.J.M.M. Rutten, *Solving reflexive domain equations in a category of complete metric spaces,* Report CS-R8709, Centre for Mathematics and Computer Science, Amsterdam, February 1987. (To appear in: Proceedings of the Third Workshop on Mathematical Foundations of Programming Language Semantics, Springer-Verlag, Lecture Notes in Computer Science, 1988.)

[BHR] S. Brookes, C. Hoare, W. Roscoe, *A theory of communicating sequential processes,* J. Assoc. Comput. Mach. 31, No. 3, 1984, pp. 560-599.

[BM] J.W. de Bakker, J.-J. Ch. Meyer, *Metric semantics for concurrency,* Report CS-R8803, Centre for Mathematics and Computer Science, Amsterdam, 1988.

[BKMOZ] J.W. de Bakker, J.N. Kok, J.-J. Ch. Meyer, E.-R. Olderog, J.I. Zucker, *Contrasting themes in the semantics of imperative concurrency,* in: Current Trends in Concurrency (J.W. de Bakker, W.P. de Roever, G. Rozenberg, eds.), Lecture Notes in Computer Science 224, Springer-Verlag, 1986, pp. 51-121.

[BMOZ1] J.W. de Bakker, J.-J. Ch. Meyer, E.-R. Olderog, J.I. Zucker, *Transition systems, infinitary languages and the semantics of uniform concurrency,* in: Proceedings 17th ACM STOC, Providence, R.I. (1985), pp. 252-262.

[BMOZ2] J.W. de Bakker, J.-J. Ch. Meyer, E.-R. Olderog, J.I. Zucker, *Transition systems, metric spaces and ready sets in the semantics of uniform concurrency,* Report CS-R8601, Centre for Mathematics and Computer Science, Amsterdam, January 1986. (To appear in: Journal of Computer and System Sciences.)

[BZ] J.W. de Bakker, J.I. Zucker, *Processes and the denotational semantics of concurrency,* Information and Control 54 (1982), pp. 70-120.

[Du] J. Dugundji, *Topology,* Allen and Bacon, Rockleigh, N.J., 1966.

[En] E. Engelking, *General topology,* Polish Scientific Publishers, 1977.

[FHLR] N. Francez, C.A.R. Hoare, D.J. Lehmann, W.P. de Roever, *Semantics of nondeterminism, concurrency and communication,* J. CSS 19 (1979), pp. 290-308.

[HP] M. Hennessy, G.D. Plotkin, *Full abstraction for a simple parallel programming language,* in: Proceedings 8th MFCS (J. Bečvař ed.), Lecture Notes in Computer Science 74 Springer-Verlag (1979), pp. 108-120.

[Ho] C.A.R. Hoare, *Communicating sequential processes,* Prentice Hall International, 1985.

[Ko1] J.N. KOK, *A compositional semantics for Concurrent Prolog,* in: Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, Lecture Notes in Computer Science, Springer Verlag, 1988, pp. 373-388.

[Ko2] J.N. KOK, *Semantic equivalence for Concurrent Prolog,* to appear.

[KR] J.N. KOK, J.J.M.M. RUTTEN, *Contractions in comparing concurrency semantics* (full version), Report CS-R8755, Centre for Mathematics and Computer Science, Amsterdam, November 1987.

[Mi] R. MILNER, *A Calculus of communicating systems,* Lecture Notes in Computer Science 92, Springer-Verlag, 1980.

[Pl1] G.D. PLOTKIN, *A powerdomain construction,* SIAM J. Comp. 5 (1976), pp. 452-487.

[Pl2] G.D. PLOTKIN, *A structural approach to operational semantics,* Report DAIMI FN-19, Comp. Sci. Dept., Aarhus Univ. 1981.

[Pl3] G.D. PLOTKIN, *An operational semantics for CSP,* in: Formal Description of Programming Concepts II (D. Bjørner ed.) North-Holland, Amsterdam (1983), pp. 199-223.

[Ru1] J.J.M.M. RUTTEN, *Correctness and full abstraction of metric semantics for concurrency,* to appear in: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency (J.W. de Bakker, W.P. de Roever, G. Rozenberg, eds.), Proc. REX Workshop 1988, Lecture Notes in Computer Science, Springer-Verlag.

[Ru2] J.J.M.M. RUTTEN, *Semantic equivalence for a parallel object-oriented language,* to appear.

[Sc] D.S. SCOTT, *Domains for denotational semantics,* Proc. 9th ICALP (M. Nielsen, E.M. Schmidt, eds.), Lecture Notes in Computer Science 140, Springer-Verlag, 1982, pp. 577-613.