FUNDAMENTAL STUDY

CONTRACTIONS IN COMPARING CONCURRENCY SEMANTICS *

Joost N. KOK and Jan J.M.M. RUTTEN
Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Communicated by M. Nivat
Received October 1988

Abstract. We define for a number of concurrent imperative languages 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.

Contents

0. Introduction ..................................................................... 180
1. A simple language \(L_0\) ................................................................. 183
   1.1. Syntax ........................................................................ 183
   1.2. Operational semantics .................................................. 185
   1.3. Denotational semantics .................................................. 188
   1.4. Semantic equivalence of \(\mathcal{O}_0\) and \(\mathcal{D}_0\) ....................... 191
   1.5. Summary of Section 1 .................................................. 199
2. A language with communication and global nondeterminism \(L_1\) ..... 200
   2.1. Syntax ........................................................................ 200
   2.2. Operational semantics .................................................. 203
   2.3. Denotational semantics .................................................. 203
   2.4. Semantic equivalence of \(\mathcal{O}_1\) and \(\mathcal{D}_1\) ....................... 204
   2.5. Summary of Section 2 .................................................. 209
3. A nonuniform language with value passing \(L_2\) ......................... 209
   3.1. Syntax ........................................................................ 210
   3.2. Operational semantics .................................................. 211
   3.3. Denotational semantics .................................................. 214
   3.4. Semantic equivalence of \(\mathcal{O}_2\) and \(\mathcal{D}_2\) ....................... 215
4. Conclusions .................................................................... 218
Appendix A ........................................................................ 218
Acknowledgment .................................................................. 221
References ........................................................................ 221

* This work was carried out in the context of ESPRIT project 415: Parallel Architectures and Languages for Advanced Information Processing—a VLSI-directed approach.
0. Introduction

We present a study of three concurrent imperative languages, called $L_0$, $L_1$, and $L_2$. For each of them we shall define an operational semantics $\mathcal{O}_i$, and a denotational semantics $D_i$, for $i = 0, 1, 2$, 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 languages themselves, or the particular properties of their semantics.

The languages $L_i$ have been defined and studied in much detail in [4, 5, 3]. We rely heavily on these papers, using many definitions taken from them literally, and others in an adapted version. (The languages $L_0$, $L_1$, and $L_2$ used here are called $L_0$, $L_2$, and $L_3$ in the previous papers.)

Let us try to characterize in a few words the languages under consideration. They all belong to the wide class of concurrent (parallel) imperative programming languages. We shall discuss parallel execution through interleaving (shuffle) of elementary actions (in $L_0$), together with synchronization and communication (in $L_1$) and extended with (an elementary form of) message passing (in $L_2$). Imperative concurrency is further characterized by an explicit operator for parallel composition on top of the usual imperative constructs, such as elementary action and sequential composition. Herein it differs from another widely used style, so-called applicative concurrency, where the parallelism is implicit. Further, $L_0$ and $L_1$ are uniform and $L_2$ is nonuniform. In $L_0$ and $L_1$ the elementary actions are left atomic, whereas in $L_2$ an interpretation of these actions is supplied. They consist of assignments, test and send and receive actions. Another important feature is the presence of local nondeterminacy (in $L_0$) and global nondeterminacy (in $L_1$ and $L_2$). (Sometimes this is called internal and external nondeterminacy.) The difference between the two has major implications for the different semantic models. (For an extensive discussion of this matter see, e.g., the introduction of [3]).

For our semantic definitions we shall use metric structures, rather than order-theoretic domains, following the approach of Nivat [14] and De Bakker and Zucker [6]. 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 models $\mathcal{O}_i$ are based on the transition system technique in [10, 16, 17]. They are closely related to the ones defined in [2], but there are some differences. We use labeled transitions and (in $\mathcal{O}_1$ and $\mathcal{O}_2$) communication is treated somewhat differently. Our denotational models $D_i$ are almost exactly the same as in [3]. They are 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. For $D_1$ and $D_2$ we use a reflexive domain, being a solution of some domain equation in the style of [15, 19]. We shall not give the details of solving this type of equation in a metric setting, but refer the reader to [6], where a solution was presented first, and [1], where this metric approach is reformulated and extended in a category-theoretic setting.
Although the semantic models presented here are (roughly) the same as in [3], there is one major difference, i.e., 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 fixed point \( x \in M \) (satisfying \( f(x) = x \)). This elementary fact is known as Banach’s fixed point theorem (see Proposition A.5(b)). Such a fixed point \( x \) is entirely determined by the definition of \( f \): any other element \( y \in M \) satisfying the same properties as \( x \), that is, satisfying \( f(y) = y \), is equal to \( x \).

The contractions \( \Phi \) 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 \( \Phi \) is a function from \( M_1 \) to \( M_2 \).

The fact that our denotational models can be obtained as fixed points of suitable contractions is not very surprising, fixed points traditionally playing an important role in denotational semantics. It is interesting, however, to observe that the same method applies to the definition of operational models. One might wonder whether the models thus obtained still deserve to be called operational. That this is the case, follows from the fact that they equal the models defined in the usual manner, without the use of fixed points (see Lemma 1.16).

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 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. We want to capture their precise relationship in some formal statement.

Let us now sketch the way we use contractions in our study of semantic equivalences. 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 \( \Phi \) that \( \mathcal{O} = \mathcal{D} \).

In the context of complete partial ordering structures similar approaches exist (see, e.g., [10, 2]). There, the operational semantics \( \mathcal{O} \) can be characterized as the (with respect to the pointwise ordering) smallest function \( \mathcal{F} \) satisfying \( \Phi(\mathcal{F}) = \mathcal{F} \), for some continuous function \( \Psi \). 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 languages 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

$$
\Phi : (L \to M_1) \to (L \to M_1) \quad \text{and} \quad \Psi : (L \to M_2) \to (L \to M_2)
$$

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 $\Phi$, $\Phi'$ and $\Psi$ as follows. If we define

$$
\phi_1 : (L \to M_1) \to (L \to M') \quad \text{and} \quad \phi_2 : (L \to M_2) \to (L \to M'),
$$

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

$$
\begin{array}{ccc}
L \to M_1 & \xrightarrow{\phi} & L \to M_1 \\
\phi_1 \downarrow & \star_1 & \phi_1 \downarrow \\
L \to M' & \xrightarrow{\phi'} & L \to M' \\
\phi_2 \uparrow & \star_2 & \phi_2 \uparrow \\
L \to M_2 & \xrightarrow{\psi} & L \to M_2,
\end{array}
$$

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

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

It is straightforward from $\star_1$ and $\star_2$, and the fact that $\Phi$, $\Phi'$, and $\Psi$ 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. $\phi_1$ and $\phi_2$ are such that for closed statements (i.e., containing no free statement variables) $s \in L_0$, we have $\mathcal{O}(s) = \mathcal{D}(s)$. Once this result has been achieved for $L_0$, it is straightforward to adapt the definitions, lemmas and theorems involved to deduce a similar result for $L_1$ and $L_2$. (For the latter languages there is one slight complication. It appears to be convenient to relate $L \to M_1$ and $L \to M_2$ via two intermediate types, $L \to M'$ and $L \to M''$. In [4, 5, 3] proofs for the semantic equivalence of operational and denotational models for $L_0$ and $L$, have been given.
These proofs, however, are quite complicated and not so easy to understand. Furthermore, the proof of $L_1$ is much more complex than that for $L_0$, involving an intermediate ready-set domain.

The method of proving semantic equivalence as described above is general in the sense that it is applicable to very different languages, such as $L_0$, $L_1$, and $L_2$.

This paper has four sections. This introduction is followed by the treatment of $L_0$, $L_1$, and $L_2$ in Sections 1, 2, and 3, respectively. Then, in Section 4, some conclusions and remarks about future research are formulated. In addition, Appendix A gives the basic definitions of metric topology.

1. A simple language ($L_0$)

1.1. Syntax

For the definition of the first 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.

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

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

where $t \in \mathcal{T}_x^n$, the set of statements which are guarded for $x$, to be defined below.

A statement $s$ is of one of the following six forms:

- an elementary action $a$.
- the sequential composition $s_1 \cdot s_2$ of statements $s_1$ and $s_2$.
- the nondeterministic choice $s_1 \cup s_2$, also known as local nondeterminism [9]: $s_1 \cup s_2$ is executed by executing either $s_1$ or $s_2$ chosen nondeterministically.
- the concurrent execution $s_1 \| s_2$, modeled by the arbitrary interleaving (shuffle) of the elementary actions of $s_1$ and $s_2$.
- a statement variable $x$, which is (normally) used in
- 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 $a^* \cdot b \cup \{a^w\}$.

An important restriction of our language is that we consider only recursive constructs $\mu x[t]$, for which $t$ is guarded for $x$: $t \in L_0^n$. Intuitively, a statement $t$ is guarded for $x$ when all occurrences of $x$ in $t$ are preceded by some statement. More formally:
1.2. Definition (Syntax for $L^x_0$). The set $L_0^x$ of statements which are guarded for $x$ is given by

$$t ::= a$$

$$\mid t; s, \text{ for } s \in L_0$$

$$\mid t_1 \cup t_2, t_1 \parallel t_2$$

$$\mid y, \text{ for } y \neq x$$

$$\mid \mu x[t]$$

$$\mid \mu y[t'], \text{ for } y \neq x, t' \in L_0^x \cap L_0^y.$$

1.3. Remark. In order to avoid possible confusion about the definitions of $L_0$ and $L_0^x$, let us give a more extensive definition, for which the ones given above are shorthand. We define $L_0$ and, for every $x \in \text{Stmv}$, $L_0^x$ simultaneously and in stages:

Stage 0:

$$L_0(0) = A \cup \text{Stmv}, \quad L_0^x(0) = A \cup (\text{Stmv}\setminus\{x\}).$$

Stage $(n+1)$:

$$L_0(n+1) = L_0(n) \cup \{s_1; s_2 | s_1, s_2 \in L_0(n)\}$$

$$\cup \{s_1 \cup s_2 | s_1, s_2 \in L_0(n)\}$$

$$\cup \{s_1 \parallel s_2 | s_1, s_2 \in L_0(n)\}$$

$$\cup \{\mu x[t] | t \in L_0^x(n)\}.$$

$$L_0^x(n+1) = L_0^x(n) \cup \{t; s | t \in L_0^x(n), s \in L_0(n)\}$$

$$\cup \{t_1 \cup t_2 | t_1, t_2 \in L_0^x(n)\}$$

$$\cup \{t_1 \parallel t_2 | t_1, t_2 \in L_0^x(n)\}$$

$$\cup \{\mu x[t] | t \in L_0^x(n)\}$$

$$\cup \{\mu y[t'] | y \neq x, t' \in L_0^x(n) \cap L_0^y(n)\}.$$

We define

$$L_0 = \bigcup_{n \in \mathbb{N}} L_0(n), \quad L_0^x = \bigcup_{n \in \mathbb{N}} L_0^x(n).$$

1.4. Remark (Empty statement). It turns out 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 $E \parallel s$ are not in $L_0$.

Now that we have formulated the notion of guardedness for $x$, we can easily generalize this.
1.5. Definition (Guarded statements). The set $L_0^g$ of guarded statements (guarded for all $x$) is defined as
\[ L_0^g = \bigcap_{x \in \text{Stmt}} L_0^x. \]
As $L_0$ and $L_0^x$, also $L_0^g$ has a simple inductive structure.

1.6. Lemma. The set $L_0^g$ can be given by the following syntax:
\[ t ::= a \mid t_1 ; t_2 \mid t_1 \cup t_2 \mid t_1 \| t_2 \mid \mu x[t] \]
where $s \in L_0$.

We need yet another notion of syntactic nature, that is, the notion of closedness.

1.7. Definition (Free variables, closed statements). For every statement $s \in L_0$ we define the set $FV(s)$ of all statement variables that occur freely in $s$ as usual:
\[ FV(a) = \emptyset, \quad FV(x) = \{x\}, \quad FV(\mu x[s]) = FV(s) \setminus \{x\}, \]
\[ FV(s_1 \circ s_2) = FV(s_1) \cup FV(s_2), \quad \text{for} \ op = \; ; \cup \| \].
We call a statement $s$ closed (notation closed$(s)$), whenever $FV(s) = \emptyset$. Finally, 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.

1.8. Examples. First we observe that $L_0^g \subseteq L_0 \subseteq L_0^g$. Further we have that
\[ x \in L_0, \quad x \notin L_0^x, \quad y ; x \in L_0^x, \quad y ; x \notin L_0^g, \]
\[ \mu x[y ; x] \in L_0, \quad \mu y[y ; x] \notin L_0, \]
\[ a ; \mu x[y ; x] \in L_0^x \cap L_0^g, \quad \mu y[a ; \mu x[y ; x]] \in L_0. \]

1.2. Operational semantics

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

1.9. Definition (Semantic universe $P_0$). Let $A^x$, the set of finite and infinite words over $A$, be given by $A^x = A^* \cup A^\omega$. For the empty word we use the special symbol $\epsilon$. Let $d_{A^x}$ denote the usual metric on $A^x$ (see Example A.2). We define $P_0 = P_{nc}(A^x)$, with typical elements $p, q, \ldots$, the set of all non-empty, compact subsets of $A^x$. As a metric on $P_0$ we take $d_{P_0} = (d_{A^x})_H$, the Hausdorff distance induced by $d_{A^x}$. According to Proposition A.8 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.

1.10. Definition (Transition relation for $L_0^x$). We define a transition relation $\rightarrow \subseteq L_0^x \times A \times L_0$ (writing $s \rightarrow^a s'$ for $(s, a, s') \in \rightarrow$) as the smallest relation satisfying

(i) $a \rightarrow^a E$ (for all $a \in A$),
(ii) for all $a \in A$, $s, t \in L_0^x$, $s', s \in L_0$. If $s' \neq E$, then

\[ s \rightarrow^a s' \Rightarrow (s; \bar{a} \rightarrow^a s'; \bar{s}) \]

\[ \land s \cup t \rightarrow^a s' \land t \cup s \rightarrow^a s' \]

\[ \land s \parallel t \rightarrow^a s' \parallel t \rightarrow^a t \parallel s' \]

\[ \land \mu x[s] \rightarrow^a s'[\mu x[s]/x], \]

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

\[ s \rightarrow^a E \Rightarrow (s; \bar{s} \rightarrow^a \bar{s}) \]

\[ \land s \cup t \rightarrow^a E \land t \cup s \rightarrow^a E \]

\[ \land s \parallel t \rightarrow^a t \rightarrow^a t \]

\[ \land \mu x[s] \rightarrow^a E). \]

Intuitively, $s \rightarrow^a s'$ 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 $C_0$, the operational semantics for $L_0^{\text{cl}}$. (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.

1.11. Definition ($\Phi_0$). Let $\Phi_0 : (L_0^{\text{cl}} \rightarrow P_0) \rightarrow (L_0^{\text{cl}} \rightarrow P_0)$ be given by

\[ \Phi_0(F)(s) = \begin{cases} \{e\} & \text{if } s = E, \\ \bigcup \{a \cdot F(s') | s' \in L_0^{\text{cl}} \land a \in A \land s \rightarrow^a s\} & \text{if } s \neq E. \end{cases} \]

for $F \in L_0^{\text{cl}} \rightarrow P_0$ and $s \in L_0^{\text{cl}}$.

1.12. Remarks. (1) It is straightforward to prove that $\Phi_0$ is contracting.

(2) Please note that $\text{closed}(s)$ and $s \rightarrow^a s'$ imply $\text{closed}(s')$.

(3) We have that $\Phi_0(F)(s)$ is a non-empty, compact subset of $A^\ast$, because

\[ \{a | \exists s' \in L_0^x[s \rightarrow^a s']\} \] is finite and non-empty (this follows from Lemma 1.6) and

$F(s')$ is compact for every $s' \in L_0^{\text{cl}}$. This implies that $\Phi_0(F) \in L_0^{\text{cl}} \rightarrow P_0$. 

1.13. Definition \((\mathcal{O}_0)\). \(\mathcal{O}_0 = \text{FixedPoint}(\Phi_0)\).

1.14. Remark. We use open brackets to denote application of \(\mathcal{O}_0\) to an argument \(s\): \(\mathcal{O}_0[s]\).

In [3] another, seemingly more operational, definition of \(\mathcal{O}_0\) is given. We shall repeat a slightly different version of it here and show that it is equivalent to this fixed-point definition.

1.15. Definition \((\mathcal{O}^*_0)\). Let \(s \in L_0^\ell\), \(s \neq E\). We define \(\mathcal{O}^*_0 : L_0^\ell \to P_0\) by putting \(w \in A^\infty\) in \(\mathcal{O}^*_0[s]\) if and only if one of the following two conditions is satisfied:

1. \(s \rightarrow a_1 \rightarrow s_1 \rightarrow a_2 \rightarrow s_2 \rightarrow \cdots \rightarrow a_n \rightarrow s_n \wedge s_n = E \wedge w = a_1 \cdots a_n\),

2. \(s \rightarrow a_1 \rightarrow s_1 \rightarrow a_2 \rightarrow s_2 \rightarrow \cdots \rightarrow a_n \rightarrow s_n \rightarrow a_{n+1} \rightarrow \cdots \wedge w = a_1 \cdots a_n a_{n+1} \cdots\)

(Where \(s \rightarrow a_i s' \rightarrow a_j s''\) abbreviates \(s \rightarrow a_i \wedge s' \rightarrow a_j s''\)). If \(s = E\), then \(\mathcal{O}^*_0[E] = \{e\}\).

1.16. Lemma. \(\mathcal{O}_0 = \mathcal{O}^*_0\).

Proof. Let \(w \in A^\infty\), \(s \in L_0^\ell\), with \(s \neq E\). We have

\[
\begin{align*}
\mathcal{O}^*_0(s) &\Rightarrow \exists a \in A \exists s' \in L_0^\ell \exists w' \in A^\infty \quad [s \rightarrow a \wedge w = a \cdot w' \wedge w' \in \mathcal{O}^*_0(s)] \\
&\Rightarrow w \in \Phi_0(\mathcal{O}^*_0)(s) \quad \text{(definition } \Phi_0)\).
\end{align*}
\]

Since \(\mathcal{O}^*_0 : L_0^\ell \to P_0\), it follows that \(\mathcal{O}^*_0 = \Phi(\mathcal{O}^*_0)\). Thus, \(\mathcal{O}^*_0 = \mathcal{O}_0\). \(\square\)

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

1.17. Definition (Initial steps). We define a function

\[I : L_0^\ell \to \mathcal{P}_{\text{fin}}(A \times L_0)\]

(Where \(\mathcal{P}_{\text{fin}}(X) = \{Y \mid Y \subseteq X \wedge \text{finite } (Y)\}\)) by induction on \(L_0^\ell\):

1. \(I(E) = \emptyset\), and \(I(a) = \{(a, E)\}\);

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

\[
\begin{align*}
I(s; s) &= \{(a_i, s_i; s)\} \quad \text{(for } s \in L_0), \\
I(s \cup t) &= I(s) \cup I(t), \\
I(s \parallel t) &= \{(a_i, s_i \parallel t)\} \cup \{(b_j, s \parallel t_j)\}, \\
I(\mu x[s]) &= \{(a_i, s_i[\mu x[s]/x])\}.
\end{align*}
\]
1.18. Remark. Please note that for all $s \not\in E$ the set $I(s)$ is finite and non-empty.

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

1.19. Lemma. $\forall a \in A \forall s \in L_0^s \forall s' \in L_0 [s \rightarrow^a s' \iff (a, s') \in I(s)]$.

1.20. Corollary.

$$\Phi_i(F)(s) = \bigcup \{ a \cdot F(s') \mid (a, s') \in I(s) \},$$

for $F : L_0^d \rightarrow P_0$, $s \in L_0^d \setminus \{E\}$.

1.3. Denotational semantics

The second semantic function we define for $L_0$ will be denotational. We call a semantic function $F : L_0 \rightarrow M$ (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 $\overline{\overline{op}} : M \times M \rightarrow M$ (assuming $op$ to be binary) such that

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

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

1.21. Definition (Semantic operators). The operators $\tilde{\;}$, $\tilde{\cup}$, $\tilde{\parallel} : P_0 \times P_0 \rightarrow P_0$ are defined as follows. Let $p, q \in P_0$, then

(i) $p; q = \begin{cases} q & \text{if } p = \{e\}, \\ \bigcup \{ a \cdot (p_a; q) \mid p_a \neq \emptyset \} & \text{otherwise}. \end{cases}$

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

(ii) $p \tilde{\parallel} q = \begin{cases} q & \text{if } q = \{e\}, \\ \bigcup \{ a \cdot (p_a \parallel q) \mid p_a \neq \emptyset \} \cup \bigcup \{ a \cdot \parallel (p \parallel q) \mid q_a \neq \emptyset \} & \text{otherwise}, \end{cases}$

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

$$p_a = \{ w \mid x \in A^x \land a \cdot w \in p \}.$$  

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

1.22. Remarks. (1) This definition is self-referential and needs some justification. We shall give it for $\tilde{\;}$ and leave the case of $\tilde{\parallel}$ to the reader. We define a mapping $\Phi : (P_0 \times P_0 \rightarrow P_0) \rightarrow (P_0 \times P_0 \rightarrow P_0)$ by

$$\Phi(F)(p, q) = \begin{cases} q & \text{if } p = \{e\}, \\ \bigcup \{ a \cdot F(p_a, q) \mid p \neq \emptyset \} & \text{otherwise}. \end{cases}$$
It is not difficult to show that \( \Phi \) is contracting. Then we define \( \tilde{\tau} = \text{FixedPoint}(\Phi) \),
which satisfies the equation of Definition 1.21.

(2) If we define the \textit{left-merge} operator \( \|= \)
by
\[
p \|= q = \begin{cases} 
\emptyset & \text{if } p = \{\epsilon\}, \\
\bigcup \{ a \cdot (p_a \|= q) \mid p_a \neq \emptyset \} & \text{otherwise}, 
\end{cases}
\]
then we have that \[ p \|= q = p \|= q \cup q \|= p \]
(using the fact that \( p' \|= q' = q' \|= p' \), for all \( p' \) and \( q' \)). This abbreviation will be helpful in some future proofs.

We need the following properties, which are easily verified.

1.23. \textbf{Lemma.} (a) For \( op = \tilde{\tau}, \, \land, \) and \( \|= \) we have
\[ \forall p, p', q, q' \in P_0 \quad [d_{p_0}(p \, op \, q, p' \, op \, q') \leq \max\{d_{p_0}(p, p'), d_{p_0}(q, q')\}] \]
(b) For \( p, p' \in P_0 \) with \( \epsilon \notin p, \epsilon \notin p' \), and \( q, q' \in P_0 \) we have
\[ d_{p_0}(p; q, p; q') = \max\{d_{p_0}(p, p'), \frac{1}{2} \cdot d_{p_0}(q, q')\} \]
(c) The operators \( \tilde{\tau}, \land, \) and \( \|= \) preserve compactness.

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

1.24. \textbf{Definition (Semantic environments).} The set \( \Gamma \) of \textit{semantic environments}, with
typical elements \( \gamma \), is given by
\[ \Gamma = \text{Stm}_\omega \rightarrow^{\text{fn}} P_0. \]
We write \( \gamma\{p/x\} \) for a \textit{variant} of \( \gamma \) which is like \( \gamma \) but with \( \gamma\{p/x\}(x) = p \).

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

1.25. \textbf{Definition (\( \Psi_0, D_0 \)).} We shall define \( D_0 \) as the fixed point of
\[
\Psi_0::(L_0 \rightarrow \Gamma \rightarrow^{-1} P_0) \rightarrow (L_0 \rightarrow \Gamma \rightarrow^{-1} P_0)
\]
which is given by induction on \( L_0 \). (Here \( \Gamma \rightarrow^{-1} P_0 \) denotes the set of
non-distance-increasing functions (see Definition A.4(c)).) Let \( F \in L_0 \rightarrow \Gamma \rightarrow^{-1} P_0 \), then
(i) \[ \Psi_0(F)(a)(\gamma) = \{a\}, \Psi_0(F)(x)(\gamma) = \gamma(x), \Psi_0(F)(E)(\gamma) = \{\epsilon\}, \]
(ii) \[ \Psi_0(F)(s \, op \, t)(\gamma) = \Psi_0(F)(s)(\gamma) \tilde{op} \Psi_0(F)(t)(\gamma), \]
(iii) \[ \Psi_0(F)(\mu x[s])(\gamma) = \Psi_0(F)(s)(\gamma\{F(\mu x[s](\gamma)/x)\}) \text{ for } s \in L_0, \]
for \( op = \cdot, \lor, \|, \) and \( \tilde{op} \) as in Definition 1.21. (We define \( \Psi_0(F) \) only for those \( s \)
and \( \gamma \), such that \( FV(s) \subseteq \text{dom}(\gamma) \).) Now we set \( D_0 = \text{FixedPoint}(\Psi_0) \).
1.26. Remark. We have $D_{\mu} s(x)[y] = D_{\mu} s(x)\{y[D_{\mu} s(x)][y]/x\}$. (As for $O_0$, we also use open brackets for $O_0$.)

It is not obvious that $\Psi_0$ is contracting. The fact that we consider only guarded recursion is essential for proving it.

1.27. Lemma. (a) If $F \in I_0 \to \Gamma \to P_0$, then $\Psi_0(F) \in L_0 \to \Gamma \to P_0$.
(b) If $F \in I_0 \to \Gamma \to P_0$, then for all $y_1$, $y_2 \in \Gamma$, $s \in L_0$

\[ \forall y \in Stm \quad [s \not\in L_0 \Rightarrow \gamma_1(y) = \gamma_2(y)] \Rightarrow \]

\[ (**) \quad d_{\mu}(\Psi_0(F)(s)(\gamma_1), \Psi_0(F)(s)(\gamma_2)) \leq \frac{1}{2} \cdot d_r(\gamma_1, \gamma_2). \]

(c) $\Psi_0$ is contracting on $L_0 \to \Gamma \to P_0$.

Proof. (a) The proof of (a) goes along the lines of (b), which is more interesting.
(b) Let $F \in L_0 \to P_0$, let $\gamma_1$, $\gamma_2 \in \Gamma$. We use induction on $L_0$.

(i) For $s = a$ we have $d_\mu(\Psi_0(F)(a)(\gamma_1), \Psi_0(F)(a)(\gamma_2)) = 0$. Let $s = x$, with $x \in Stm$. Suppose (*) holds for $x$. Then

\[ d_{\mu}(\Psi_0(F)(s)(\gamma_1), \Psi_0(F)(s)(\gamma_2)) = d_{\mu}(\gamma_1, \gamma_2) \]

\[ = 0 \quad \text{because of (*)}. \]

(ii) We only treat sequential composition and recursion. Let $s = s_1 \cdot s_2$, with $s_1$, $s_2 \in I_0$. Suppose (b) holds for $s_1$ and $s_2$. Suppose (*) holds for $s_1 \cdot s_2$. This implies that (*) holds for $s_1$. Thus, we have (**) for $s_1$. Now:

\[ d_{\mu}(\Psi_0(F)(s_1 \cdot s_2)(\gamma_1), \Psi_0(F)(s_1 \cdot s_2)(\gamma_2)) \]

\[ = d_{\mu}(\Psi_0(F)(s_1)(\gamma_1), \Psi_0(F)(s_2)(\gamma_2)) \]

\[ \leq \max \{d_{\mu}(\Psi_0(F)(s_1)(\gamma_1), \Psi_0(F)(s_2)(\gamma_2)), \frac{1}{2} \cdot d_{\mu}(\gamma_1, \gamma_2)\} \]

\[ \text{for all } s \in I_0 \setminus \{F\}, \text{ and } \gamma \text{ we have } \epsilon \not\in \Psi_0(F)(s)(\gamma); \]

\[ \text{thus Lemma 1.23(b) applies} \]

\[ \leq \max \{\frac{1}{2} \cdot d_r(\gamma_1, \gamma_2), \frac{1}{2} \cdot d_r(\gamma_1, \gamma_2)\} \quad (**) \text{ for } s_1 \text{; (a) for } s_2 \]

\[ = \frac{1}{2} \cdot d_r(\gamma_1, \gamma_2). \]

(The proof for $s_1 \cup s_2$ and $s_1 \parallel s_3$ is similar.) Next we treat recursion. Let $s_1 \in L_0$ and suppose that $\mu x[s_1]$ satisfies (*). Then $s_1$ satisfies it. Thus, we have (**) for $s_1$. Now

\[ d_{\mu}(\Psi_0(F)(\mu x[s_1])(\gamma_1), \Psi_0(F)(\mu x[s_1])(\gamma_2)) \]

\[ = d_{\mu}(\Psi_0(F)(s)(\gamma_1)[F(\mu x[s_1])(\gamma_1)/x]), \]

\[ \Psi_0(F)(s)(\gamma_2[F(\mu x[s_1])(\gamma_2)/x]) \]
Contractions in comparing concurrency semantics

\[ \leq \frac{1}{2} \cdot d_r(\gamma_1[F(\mu x[s_1])(\gamma)/x], \gamma_2[F(\mu x[s_1])(\gamma_2)/x]) \quad (*) \text{holds for } s_1, \]
also w.r.t. \( \gamma_i[F(\mu x[s_1])(\gamma_i)/x] \), for \( i = 1, 2 \), thus so does \((***)\)
\[ \leq \frac{1}{2} \cdot \max\{d_r(\gamma_1, \gamma_2), d_{r_0}(F(\mu x[s_1])(\gamma_1), F(\mu x[s_1])(\gamma_2))\} \]
\[ \leq \frac{1}{2} \cdot d_r(\gamma_1, \gamma_2), \quad (a) \text{for } \mu x[s_1]. \]

(c) Let \( F_1, F_2 \in L_0 \rightarrow I \rightarrow P_0 \). We only treat recursion. Suppose
\[ d_{r_0}(\Psi_0(F_1)(s)(\gamma), \Psi_0(F_2)(s)(\gamma)) \leq \frac{1}{2} \cdot d(F_1, F_2), \]
for some \( s \in L_0^t \), all \( \gamma \in I \). Then
\[ d_{r_0}(\Psi_0(F_1)(\mu x[s])(\gamma), \Psi_0(F_2)(\mu x[s])(\gamma)) \]
\[ = d_{r_0}(\Psi_0(F_1)(s)(\gamma_1), \Psi_0(F_2)(s)(\gamma_2)), \quad \gamma_1 = \gamma[F(\mu x[s])(\gamma)/x], \quad i = 1, 2 \]
\[ \leq \max\{d_{r_0}(\Psi_0(F_1)(s)(\gamma_1), \Psi_0(F_2)(s)(\gamma_1)), d_{r_0}(\Psi_0(F_2)(s)(\gamma_1), \Psi_0(F_2)(s)(\gamma_2))\} \]
\[ \leq \max\{\frac{1}{2} \cdot d(F_1, F_2), \frac{1}{2} \cdot d_r(\gamma_1, \gamma_2)\} \quad \text{(induction, (b))} \]
\[ = \max\{\frac{1}{2} \cdot d(F_1, F_2), \frac{1}{2} \cdot d_{r_0}(F_1(\mu x[s])(\gamma), F_2(\mu x[s])(\gamma))\} \]
\[ = \frac{1}{2} \cdot d(F_1, F_2). \quad \square \]

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 \( \gamma \) 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 \( \delta \),
\[ \mathcal{O}_0'[\mu x[s]](\delta) = \mathcal{O}_0'[s](\delta[\mu x[s]/x]). \]
Here \( \delta \) is changed, the new \( \gamma \) 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:
\[ \mathcal{O}_0[s] = D_0[s](\gamma), \quad \text{for all } s \in L_0^t \text{ and arbitrary } \gamma \in I. \]
For the definition of \( \mathcal{O}_0' \), we need the following.

1.28. Definition (Syntactic environments). The set \( \Delta \) of syntactic environments, with typical elements \( \delta \), is defined by
\[ \Delta = \{ \delta \mid \delta \in (Stmu \rightarrow L_0) \wedge (\delta \text{ is normal}) \}, \]
where the notion of normal environments is given in the following.
1.29. Definition (Normal environments). A syntactic environment is called normal whenever

(i) \( \forall x \in \text{dom}(\delta) \left[ \delta(x) \in L_0^a \right] \),

(ii) \( \forall s \in L_0 \left[ \text{FV}(s) \subseteq \text{dom}(\delta) \Rightarrow \exists k \geq 0 \left[ s[\delta]^k \in L_0^{ci} \right] \right] \),

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

1.30. Remarks. (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 \( \text{FV}(s) \subseteq \text{dom}(\delta) \).

(2) Let \( \delta \in \text{Simv} \to L_0^a \) be such that for \( x, y \in \text{Simv} \), \( \delta(x) = y \) 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 \( \delta \in \text{Simv} \to L_0^a \) that \( \forall x \in \text{dom}(\delta) \left[ \delta(x) \in L_0^{ci} \right] \). An example may illustrate this. Let \( \delta \) be defined such that \( \text{dom}(\delta) = \{x, y\} \), and

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

We shall encounter such an environment when computing \( \sigma_0[\mu x[a; \mu y[b; x; y]]] \). Now \( x[\delta] = \delta(y) \in L_0^a \), but \( y[\delta]^2 \in L_0^{ci} \).

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

1.31. Theorem (Induction principle for \( L_0 \times \Delta \)). Let \( \Xi \subseteq L_0 \times \Delta \). If

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

(2) \( \{s, t\} \times \Delta \subseteq \Xi \Rightarrow \{s, s \cup t, s \uplus t\} \times \Delta \subseteq \Xi \) for \( s, t, \tilde{s} \in L_0 \),

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

(4) \( (\delta(x), \delta) \in \Xi \Rightarrow (x, \delta) \in \Xi \) for \( x \in \text{Simv} \) and \( \delta \in \Delta \),

then \( \Xi = L_0 \times \Delta \).

Proof. Let \( \Xi \subseteq L_0 \times \Delta \), suppose \( \Xi \) satisfies (1)-(4). We first prove fact (a) and fact (b) given below, and next show that (a) and (b) imply \( \Xi = L_0 \times \Delta \). So we have

fact (a) \( L_0^a \times \Delta \subseteq \Xi \),

fact (b) \( \forall S \subseteq L_0 \times \Delta \left[ S \subseteq \Xi \Rightarrow S' \subseteq \Xi \right] \),

where \( S' = \{(s, \delta) \mid (s, \delta) \in L_0 \times \Delta \land \forall x \in \text{FV}(s) \left[ s \notin L_0^a \Rightarrow (\delta(x), \delta) \in S \right] \} \).

To show that (a) holds, we use (1), (2), and (3), and induction on the structure of \( L_0^a \). We proceed with (b). Let \( S \subseteq L_0 \times \Delta \) and suppose \( S \subseteq \Xi \). Let \( S' \) be as above. We use (1)-(4) and induction on \( L_0 \) to show that \( S' \subseteq \Xi \). Let \( (s, \delta) \in S' \), for \( s \in L_0 \), \( \delta \in \Delta \).

(i) \( s \equiv a \cdot (a, \delta) \in \Xi \), because (1).

(ii) \( s \equiv s_i \cdot (a, \delta) \): Suppose that if \( (s_i, \delta) \in S' \), then \( (s_i, \delta) \in \Xi \), for \( i = 1, 2 \). If \( (s, \delta) \in S' \), then also \( (s_1, \delta) \) and \( (s_2, \delta) \in S' \). Thus \( (s_1, \delta), (s_2, \delta) \in \Xi \). By (2) we have \( (s_1 \cdot (a, \delta), (s_2, \delta) \in \Xi \).
Contractions in comparing concurrency semantics

(iii) \( s \equiv \mu x[s_1], \) for \( s_1 \in \mathcal{L}_0^\delta: \) Suppose that \((s_1, \delta) \in S'\) implies \((s_1, \delta) \in \Xi.\) Because \(s_1 \in \mathcal{L}_0^\delta\) we have: \((s_1, \delta) \in S' \iff (\mu x[s_1], \delta) \in S'.\) Because \((\mu x[s_1], \delta) \in S'\) we have \((s_1, \delta) \in \Xi.\) Thus, using (3), we have \((\mu x[s_1], \delta) \in \Xi.\)

(iv) \( s = x: \) if \((x, \delta) \in S', \) then \((\delta(x), \delta) \in S,\) thus (because \(S \subseteq \Xi\)) \((\delta(x), \delta) \in \Xi.\) Because of (4), we then have that \((x, \delta) \in \Xi.\) Thus facts (a) and (b) hold. Next we show that \( \Xi = \mathcal{L}_0 \times \Delta. \) For this purpose we define, for all \( n \in \mathbb{N}. \)

\[
\forall s \in \mathcal{L}_0 \forall \delta \in \Delta \exists n \in \mathbb{N} \quad [s[\delta]^n \in \mathcal{L}_0^\delta \Rightarrow (s, \delta) \in \mathcal{V}_n].
\]

Thus we have

\[
\forall s \in \mathcal{L}_0 \forall \delta \in \Delta \exists n \in \mathbb{N} \quad [s[\delta]^n \in \mathcal{L}_0^\delta \Rightarrow (s, \delta) \in \mathcal{V}_n].
\]

which we prove with induction on \( n \in \mathbb{N}. \) Let \( s \in \mathcal{L}_0 \) and \( \delta \in \Delta. \) If \( s[\delta]^n \in \mathcal{L}_0^\delta, \) then \( s \in \mathcal{L}_0 \subseteq \mathcal{L}_0^\delta.\) Thus \((s, \delta) \in \mathcal{V}_0.\) Now suppose \((*)\) holds for \( n \in \mathbb{N}, \) and suppose \( s[\delta]^n \in \mathcal{L}_0^\delta.\) Then \((s[\delta]^n)[\delta]^n \in \mathcal{L}_0^\delta,\) thus by induction \((s[\delta], \delta) \in \mathcal{V}_n.\) This implies \((s, \delta) \in \mathcal{V}_{n+1},\) which proves \((*)\) for \( n + 1. \) Because all \( \delta \in \Delta \) are normal we have

\[
\forall(s, \delta) \in \mathcal{L}_0 \times \Delta \forall n \in \mathbb{N} \quad [s[\delta]^n \in \mathcal{L}_0^\delta].
\]

Together with \((*)\) this implies

\[
\forall(s, \delta) \in \mathcal{L}_0 \times \Delta \exists n \in \mathbb{N} \quad [(s, \delta) \in \mathcal{V}_n].
\]

Since \( \forall n \in \mathbb{N}, \) it follows that \( \mathcal{L}_0 \times \Delta = \bigcup_{n \in \mathbb{N}} \mathcal{V}_n. \) Now \( \mathcal{V}_n \subseteq \Xi \) because of (a), and \( \mathcal{V}_n \subseteq \Xi \Rightarrow \mathcal{V}_{n+1} \subseteq \Xi \) because of (b), so we conclude: \( \Xi = \mathcal{L}_0 \times \Delta. \)

1.32. Remark. 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 \( \mathcal{L}_0 \times \Delta \) instead of \( \mathcal{L}_0 \) only.

Now let \( \Xi \subseteq \mathcal{L}_0 \times \Delta. \) The first three conditions of the principle suffice to prove that \( \mathcal{L}_0^\delta \times \Delta \subseteq \Xi, \) since they express exactly the syntactic structure of \( \mathcal{L}_0^\delta \) (see Lemma 1.6). (We have chosen \( \mathcal{L}_0^\delta \) here instead of \( \mathcal{L}_0^\Xi, \) because the latter set has no simple inductive structure.) Thus, also \( \mathcal{L}_0^\delta \times \Delta \subseteq \Xi. \) Adding condition (4) enables us to prove \( \mathcal{L}_0 \times \Delta \subseteq \Xi. \) This may be motivated by the following. For every statement \( s \in \mathcal{L}_0 \) and normal environment \( \delta \in \Delta \) there exists an \( l \in \mathbb{N} \) such that \( s[\delta]^l \in \mathcal{L}_0^\delta \subseteq \mathcal{L}_0^\delta. \) Let us call \( k \in \mathbb{N} \) with \( k = \min\{|l| \mid s[\delta]^l \in \mathcal{L}_0^\delta\} \) the degree of closedness of \( s \) with respect to \( \delta. \) Please note that every \( s \in \mathcal{L}_0^\delta \) has degree 0, and arbitrary \( s \in \mathcal{L}_0 \) has, for arbitrary \( \delta, \) 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 \( \Xi \) for all \((s, \delta)\) with degree 0. They form, so to speak, the basis of the principle. Condition
expresses the “step part”: if \( \exists \) holds for \((\delta(x), \delta)\), which has degree \(k\), say, then \( \exists \) holds for \((x, \delta)\), which then has degree \(k + 1\).

We now proceed with the definition of \( \mathcal{C}'_0 \). It will be of type \( \mathcal{C}'_0 : L_0 \to \Delta \to P_0 \), which could be called intermediate between \( \mathcal{C}_0 : L_0^\text{cl} \to P_0 \), and \( D_0 : L_0 \to I' \to P_0 \). Instead of basing the definition of \( \mathcal{C}'_0 \) on some transition relation (as in Definition 1.11), we use a variant of the initial step function (Definition 1.17).

1.33. Definition (Initial steps with syntactic environments). We define a function

\[ I' : L_0 \to \Delta \to \mathcal{P}(\Delta \times L_0 \times \Delta), \]

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 \( \Xi \) satisfies the induction conditions. Thus, we ensure that \( I' \) is defined for every \( s \in L_0 \) and \( \delta \in \Delta \) (with \( \text{FV}(s) \subseteq \text{dom}(\delta) \)).

1. Suppose \( I'(E)(S) = \emptyset \) and \( I'(a)(S) = \{(a, E, \delta)\} \), 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 \( s, t, s_i, t_j \in L_0 \), \( a_i, b_j \in A \), and \( \delta_i, \delta_j \in \Delta \). (The variables \( i \) and \( j \) range over some finite sets of indices, which are omitted.) Then

\[ I'(s \cup t)(\delta) = I'(s)(\delta) \cup I'(t)(\delta), \]

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

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

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

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

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

\[ I'(x)(\delta) = I'(\delta(x))(\delta). \]

1.34. Remarks. (1) We have, if \( I'(s)(\delta) = \{(a_i, s_i, \delta_i)\} \), then \( \text{normal}(\delta_i) \), and thus \( \delta_i \in \Delta \), for all \( i \).

2. The definition of \( I'(\mu x[s])(\delta) \), with \( x \in \text{dom}(\delta) \), is correct, because \( s \) and \( \bar{s} \) have the same complexity.

3. If \( I'(s)(\delta) = \{(a_i, s_i, \delta_i)\} \) then for all \( i \):

\[ \forall x \in \text{Stmv}[x \in \text{dom}(\delta) \cap \text{dom}(\delta_i) \Rightarrow \delta(x) = \delta_i(x)]. \]

1.35. Definition (\( \Phi'_0 \)). We define \( \Phi'_0 : (L_0 \to \Delta \to P_0) \to (L_0 \to \Delta \to P_0) \) by

\[ \Phi'_0(F)(s)(\delta) = \begin{cases} 
\{s\} & \text{if } s = E, \\
\bigcup \{a \cdot F(s')(\delta') | (a, s', \delta') \in I'(s)(\delta)\} & \text{otherwise,} 
\end{cases} \]

for \( F \in L_0 \to \Delta \to P_0 \), \( s \in L_0 \), and \( \delta \in \Delta \) with \( \text{FV}(s) \subseteq \text{dom}(\delta) \).
1.36. Definition $\mathcal{O}_0 = \text{FixedPoint}(\Phi_0)$.\\

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

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

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

1.37. Theorem (Relating $I$ and $I'$). For all $s \in L_0$ and $\delta \in \Delta$, with $FV(s) \subseteq \text{dom}(\delta)$, we have

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

For the definition of $s(\delta)$, see Definition 1.29.

Proof. We define

\[
\Xi(s, \delta) = \forall a \in A \forall s' \in L_0 \forall \delta' \in \Delta \quad [(a, s', \delta') \in I'(s)(\delta) \Leftrightarrow (a, s'(\delta')) \in I(s(\delta))]
\]

and use the induction principle for $L_0 \times \Delta$ to show that $\Xi = L_0 \times \Delta$. We only treat the case of recursion. Suppose $s \in L_0^l$ such that $\{s\} \times \Delta \subseteq \Xi$. We have to show that $\{\mu x[s]\} \times \Delta \subseteq \Xi$. Let $\delta \in \Delta$ and assume (without loss of generality) that $x \notin \text{dom}(\delta)$. Then

\[
I'(\mu x[s])(\delta) = I'(s)(\delta')
\]

where $\delta' = \delta{\mu x[s]/x}$ (by the definition of $I'$). On the other hand, we have

\[
I(\mu x[s])(\delta)) = I(\mu x[s(\delta)]), \quad x \in \text{dom}(\delta)
\]

\[
-I(s(\delta)[\mu x[s(\delta)]/x])
\]

(the latter equality following from

\[
\forall r \in L_0^l \quad [I(\mu x[r]) = I(r[\mu x[r]/x])].
\]

We take a quick (but deep) breath and proceed as follows:

\[
s(\delta)[\mu x[s(\delta)]/x] = s[\delta](\delta)[\mu x[s(\delta)]/x] \quad \text{(definition $s(\delta)$)}
\]

\[
= s[\delta][\mu x[s(\delta)]/x](\delta),
\]

\[
x \notin \text{dom}(\delta), \forall y \in \text{dom}(\delta)[x \notin FV(\delta(y))]
\]

\[
= s[\delta][\mu x[s]/x](\delta)
\]

\[
= s[\delta'](\delta), \quad \delta' = \delta{\mu x[s]/x}
\]

\[
= s[\delta'](\delta'), \quad x \notin FV(s[\delta'])
\]

\[
= s(\delta').
\]
Thus, we have \( I(\mu x[s]s, \delta) = I(s, \delta') \). Combining this with \( I'(\mu x[s])(\delta) = I'(s)(\delta') \), which we saw above, yields
\[
\Xi(\mu x[s], \delta) \iff \Xi(s, \delta').
\]
Because \( \{s\} \times \Delta \subseteq \Xi \) we may conclude \( \Xi(\mu x[s], \delta) \). \( \square \)

We formulate the relation of \( C_0 \) and \( C'_0 \) in terms of their defining contractions \( \Phi_0 \) and \( \Phi'_0 \). This can be elegantly done using the following.

1.38. Definition. We define \( \langle \rangle : (L_0^l \rightarrow P_0) \rightarrow (L_0 \rightarrow \Delta \rightarrow P_0) \), for every \( F \in L_0^l \rightarrow P_0 \), by
\[
\langle \rangle(F) = F^{(1)} \quad (\text{notation})
\]
\[
= \ \lambda s \in L_0 \cdot \lambda \delta \in \Delta \cdot F(s(\delta)).
\]

1.39. Remark. This mapping links two kinds of semantic functions, one using syntactic environments, and the other one not using environments. If \( F \in L_0^l \rightarrow P_0 \), then \( F^{(1)} \) is an in a sense extended version of \( F \): it can also take as an argument statements \( s \in L_0 \) that are not closed, provided it is supplicated with a syntactic environment, which is to give the (syntactic) values for the free variables in \( s \).

1.40. Theorem (Relating \( \Phi_0 \) and \( \Phi'_0 \)). \( \forall F \in L_0^l \rightarrow P_0 \) \( \Phi'_0(F^{(1)}) = (\Phi_0(F))^{(1)} \).

Proof. The theorem is an immediate consequence of Theorem 1.37. Let \( F \in L_0^l \rightarrow P_0 \), let \( s \in L_0 \), \( s \neq E \).
\[
\Phi'_0(F^{(1)})(s)(\delta) = \bigcup \{ a \cdot F^{(1)}(s')(\delta') \mid (a, s', \delta') \in I'(s)(\delta) \}
\]
\[
= \bigcup \{ a \cdot F(s'(\delta')) \mid (a, s', \delta') \in I'(s)(\delta) \}
\]
\[
= \bigcup \{ a \cdot F(s'(\delta)) \mid (a, s'(\delta)) \in I(s(\delta)) \} \quad \text{(Theorem 1.37)}
\]
\[
= \Phi_0(F)(s(\delta)) = (\Phi_0(F))^{(1)}(s)(\delta). \quad \square
\]

Because \( \Phi_0 \) and \( \Phi'_0 \) are contractions with \( C_0 \) and \( C'_0 \) as their respective fixed points, we have the following.

1.41. Corollary \( (C'_0 = C_0^{(1)}) \). \( \forall s \in L_0 \forall \delta \in \Delta \quad [C'_0[s](\delta) = C_0[s](\delta)] \).

Finally we relate \( C'_0 : L_0 \rightarrow \Delta \rightarrow P_0 \) and \( \Phi_0 : L_0 \rightarrow \Gamma \rightarrow P_0 \).

For this purpose we define the following mapping.

1.42. Definition. We define \( \sim : (L_0 \rightarrow \Gamma \rightarrow P_0) \rightarrow (L_0 \rightarrow \Delta \rightarrow P_0) \) by
\[
\sim(F) = \tilde{F} \quad \text{(notation)}
\]
\[
= \lambda s \in L_0 \cdot \lambda \delta \in \Delta \cdot F(s(\delta^F))
\]
for \( F \in L_0 \rightarrow \Gamma \rightarrow P_0 \), where \( \delta^F \) is given by \( \delta^F = \lambda x \in \text{dom} (\delta) \cdot F(\delta(x))(\delta^F) \). (We often write \( \tilde{\delta} \) rather than \( \delta^F \) if from the context it is clear which \( F \) should be taken.)
1.43. Remarks. (1) We have to justify the self-referential definition of $\tilde{\delta}^F$. For this purpose we define

$$\Xi(s, \delta) = \forall x \subset FV(s) \ [s \not\in L_0^* \rightarrow (\tilde{\delta}^F(x) \text{ is well defined})],$$

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 $x \in \text{Simv}$ with $x \in \text{dom}(\delta)$ that $\tilde{\delta}^F(x)$ is well defined. Conditions (1)-(3) of the induction principle are trivially fulfilled. We prove condition (4). Suppose $(\delta(x), \delta) \in \Xi$. Thus $\tilde{\delta}^F(y)$ is well defined for all $y \in FV(\delta(x))$. This implies that $\tilde{\delta}^F(x)$ is well defined, since

$$\tilde{\delta}^F(x) = F(\delta(x))(\tilde{\delta}^F).$$

(2) In the same way as $\langle \rangle$, 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 $F$) $\tilde{\delta}^F$ of it, which then is supplied as an argument to $F$.

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

1.44. Theorem. $\Phi'_0(\mathcal{D}_0) = \mathcal{D}_0$.

Proof. Let $\Xi \subseteq L_0 \times \Delta$ be defined by

$$\Xi(s, \delta) = \Phi'_0(\mathcal{D}_0)(s)(\delta) = \mathcal{D}_0(s)(\delta)$$

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 $\Phi'_0(\mathcal{D}_0)(a)(\delta) = \{a\} = \mathcal{D}_0(a)(\delta)$, so $A \times \Delta \subseteq \Xi$.

(2) Let $s, \bar{s} \in L_0$ and suppose $\Xi(s, \delta)$. We show $\Xi(s; \bar{s}, \delta)$.

$$\Phi_0(\mathcal{D}_0)(s; \bar{s}; \delta)(\delta) = \bigcup \{a' \cdot \mathcal{D}_0(s'; \bar{s})(\delta') | (a', s', \delta') \in I'(s)(\delta) \}$$

(definition $\Phi_0$ and $I'(s; \bar{s})$)

$$= \bigcup \{a' \cdot (\mathcal{D}_0(s'; \bar{s})(\delta')); \mathcal{D}_0(\delta)(\delta') | (a', s', \delta') \in I'(s)(\delta) \}$$

(see Remark 1.34(3))

$$= (\bigcup \{a' \cdot \mathcal{D}_0(s')(\delta') | (a', s', \delta') \in I'(s)(\delta) \}; \mathcal{D}_0(\delta)(\delta)$$

(definition $\cdot$)

$$= \Phi_0(\mathcal{D}_0)(s)(\delta); \mathcal{D}_0(\delta)(\delta) \quad \text{(definition $\Phi_0$)}$$

$$= \mathcal{D}_0(s)(\delta); \mathcal{D}_0(\delta)(\delta) \quad \text{(definition $\mathcal{D}_0$)}$$

$$= \mathcal{D}_0(s; \bar{s})(\delta).$$
This proves $\Xi(s; \xi, \delta)$. Now let $s, t \in L_0$ and suppose $\Xi(s, \delta)$ and $\Xi(t, \delta)$. We show $\Xi(s \parallel t, \delta)$.

\[
\Phi'(\tilde{D}_0)(s \parallel t)(\delta)
\]

\[
= \bigcup \{ a' \cdot \tilde{D}_0(s \parallel t)(\delta') | (a', s', \delta') \in I'(s)(\delta) \}
\]

\[
\cup \bigcup \{ b' \cdot \tilde{D}_0(s \parallel t)(\delta') | (b', t', \delta') \in I'(t)(\delta) \} \quad \text{(definition $\Phi'$ and $I'(s \parallel t)$)}
\]

\[
= \bigcup \{ a' \cdot (\tilde{D}_0(s')(\delta') \parallel \tilde{D}_0(t')(\delta')) | (a', s', \delta') \in I'(s)(\delta) \}
\]

\[
\cup \bigcup \{ b' \cdot (\tilde{D}_0(s)(\delta') \parallel \tilde{D}_0(t')(\delta')) | (b', t', \delta') \in I'(t)(\delta) \}
\]

\[
= \bigcup \{ a' \cdot (\tilde{D}_0(s')(\delta') \parallel \tilde{D}_0(t')(\delta')) | (a', s', \delta') \in I'(s)(\delta) \}
\]

\[
\cup \bigcup \{ b' \cdot (\tilde{D}_0(s)(\delta') \parallel \tilde{D}_0(t')(\delta')) | (b', t', \delta') \in I'(t)(\delta) \}
\]

(see Remark 1.34(3))

\[
= \left( \bigcup \{ a' \cdot \tilde{D}_0(s')(\delta') | (a', s', \delta') \in I'(s)(\delta) \} \right) \parallel \tilde{D}_0(t)(\delta)
\]

\[
\cup \left( \bigcup \{ b' \cdot (\tilde{D}_0(t')(\delta') | (b', t', \delta') \in I'(t)(\delta) \} \right) \parallel \tilde{D}_0(s)(\delta)
\]

\[
= \left( \bigcup \{ a' \cdot \tilde{D}_0(s')(\delta') | (a', s', \delta') \in I'(s)(\delta) \} \right) \parallel \tilde{D}_0(t)(\delta)
\]

\[
\cup \left( \bigcup \{ b' \cdot (\tilde{D}_0(t')(\delta') | (b', t', \delta') \in I'(t)(\delta) \} \right) \parallel \tilde{D}_0(s)(\delta)
\]

\[
= \left( \Phi'(\tilde{D}_0)(s)(\delta) \parallel \tilde{D}_0(t)(\delta) \right)
\]

\[
\cup \left( \Phi'(\tilde{D}_0)(t)(\delta) \parallel \tilde{D}_0(s)(\delta) \right) \quad \text{(definition $\Phi'$)}
\]

\[
= (\tilde{D}_0(s)(\delta) \parallel \tilde{D}_0(t)(\delta))
\]

\[
\cup (\tilde{D}_0(t)(\delta) \parallel \tilde{D}_0(s)(\delta)) \quad \text{(we have $\Xi(s, \delta)$ and $\Xi(t, \delta)$)}
\]

\[
= \tilde{D}_0(s)(\delta) \parallel \tilde{D}_0(t)(\delta) = \tilde{D}_0(s \parallel t)(\delta).
\]

This proves $\Xi(s \parallel t, \delta)$. The case $\Xi(s \cup t, \delta)$ is simple.

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

\[
\Phi_0(\tilde{D}_0)(\mu x[s])(\delta) = \left( \bigcup \{ a' \cdot \tilde{D}_0(s')(\delta') | (a', s', \delta') \in I'(s)(\delta') \} \right) \quad \text{(definition $\Phi_0$ and $I'(\mu x[s])(\delta)$; let $\delta' = \delta \{ \mu x[s]/x \}$)}
\]

\[
= \Phi_0(\tilde{D}_0)(s)(\delta')
\]

\[
= \tilde{D}_0(s)(\delta') \quad \text{(we have $\Xi(s, \delta')$)}
\]

\[
= \tilde{D}_0(\delta)
\]

\[
= \tilde{D}_0(\mu x[s])(\delta') \quad \text{(definition $\tilde{D}_0$)}
\]

\[
= \tilde{D}_0(\mu x[s])(\delta).
\]
This proves $\Xi(\mu x[s], \delta)$.

(4) Let $x \in \text{Stmt}$, suppose $\Xi(\delta(x), \delta)$. Now

$$
\Phi'_0(\mathcal{D}_0)(x)(\delta) = \Phi'_0(\mathcal{D}_0)(\delta(x))(\delta) \quad \text{(definition $\Phi'_0$ and $I'(x)(\delta)$)}
$$

$$
= \mathcal{D}_0(\delta(x))(\delta) \quad \text{(because $\Xi(\delta(x), \delta)$)}
$$

$$
= \mathcal{D}_0[\delta(x)](\delta)
$$

$$
= \tilde{\delta}(x) \quad \text{(definition $\tilde{\delta}$)}
$$

$$
= \mathcal{D}_0[x](\tilde{\delta}) = \mathcal{D}_0(x)(\delta).
$$

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

As an immediate consequence of this theorem, we have the following.

1.45. Corollary ($\mathcal{C}_0' = \mathcal{D}_0$). $\forall s \in L_0 \forall \delta \in \Delta [\mathcal{C}_0'[s](\delta) = \mathcal{D}_0[s](\tilde{\delta})]$.

Now combining Corollaries 1.41 and 1.45 yields the main theorem of this section.

1.46. Theorem ($\mathcal{C}_0' = \mathcal{D}_0$). $\forall s \in L_0 \forall \delta \in \Delta [\mathcal{C}_0'[s](\delta) = \mathcal{D}_0[s](\tilde{\delta})]$.

1.47. Corollary. For all $s \in L_0^c$, and arbitrary $\gamma \in \Gamma$, $\mathcal{C}_0'[s] = \mathcal{D}_0[s](\gamma)$.

1.5. Summary of Section 1

It may be useful to give a short overview of this section because we shall follow the same approach of proving semantic equivalence in the next sections. We have defined an operational semantics $\mathcal{C}_0$ for $L_0$ as the fixed point of $\Phi_0$, and a denotational semantics $\mathcal{D}_0$ as the fixed point of $\Psi_0$. We have related $\mathcal{C}_0$ and $\mathcal{D}_0$ via an intermediate semantic function $\mathcal{C}_0'$, defined as the fixed point of $\Phi'_0$. To be more precise, we have related $\Phi_0$, $\Psi_0$, and $\Phi'_0$ using mappings $())$ and $\sim$, for which we have proved some properties, schematically represented by the following diagram:

$$
\begin{align*}
L_0^c &\rightarrow P_0 \xrightarrow{\Phi_0} L_0^c \rightarrow P_0 \\
&\downarrow () \swarrow \downarrow () \\
L_0 &\rightarrow \Delta \rightarrow P_0 \xrightarrow{\Phi'_0} L_0 &\rightarrow \Delta \rightarrow P_0 \\
&\downarrow \sim \swarrow \downarrow \sim \\
L_0 &\rightarrow \Gamma \rightarrow P_0 \xrightarrow{\Psi_0} L_0 \rightarrow \Gamma \rightarrow P_0.
\end{align*}
$$
The * in the upper rectangle indicates that it commutes, the symbol \(*_{\text{fix}}\) in the lower rectangle indicates that it commutes only for the fixed point of \(\Psi_0\) (that is, \(\mathcal{D}_0\)). Please note that * has been formulated as Theorem 1.40, and \(*_{\text{fix}}\) as Theorem 1.44. The main result of Section 1 (Theorem 1.46) follows from this diagram, because * implies \(\mathcal{C}_0^{(1)} = \mathcal{C}_0\) and \(*_{\text{fix}}\) implies \(\mathcal{C}_0^{(1)} = \mathcal{D}_0\).

1.48. Remark. The lower rectangle does not commute for arbitrary \(F \in L_0 \rightarrow \Gamma \rightarrow P_0\). As an example take \(F = \lambda s \cdot \lambda y \cdot \{\varepsilon\}\). Then, for given \(a, b \in A\) and \(\delta \in \Delta\):

\[
\Phi_0(F)(a;b)(\delta) = \Psi_0(F)(a;b)(\delta_{\Psi_0(F)}) = \Psi_0(F)(a)(\delta_{\Psi_0(F)})\gamma \Psi_0(F)(b)(\delta_{\Psi_0(F)}) = \{a\} \cdot \{b\} = \{ab\},
\]

whereas

\[
\Phi_0(F')(a;b)(\delta) = \{a \cdot F'(b)(\delta)\} = \{a \cdot F(b)(\delta')\} = \{a\}.
\]

2. A language with communication and global nondeterminism \((L_1)\)

2.1. Syntax

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 [13] or CSP [11] we stipulate a bijection \(\prec: C \rightarrow C\) with \(\prec \circ \prec = \text{id}_C\). It yields for every \(c \in C\) a matching communication \(\prec(c)\), which will be denoted by \(\tilde{c}\). In \(A\setminus C\) we have a special element \(\tau\) denoting a successful communication. Let \(\text{Stmu}\), with typical elements \(x, y, \ldots\), again be the set of statement variables.

2.1. Definition (Syntax for \(L_1\)). The set \(L_1\), with typical elements \(s, t, \ldots\), is given by

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

where \(t \in L_1^s\), which is defined below. Please note that \(a \in A \supseteq C\).

2.2. Definition (Syntax for \(L_1^s\)). The set \(L_1^s\) of statements which are guarded for \(x\) is given by

\[
t ::= a \\
| t; s, \text{ for } s \in L_1 \\
| t_1 + t_2 | t_1 || t_2 \\
| y, \text{ for } y \neq x \\
| \mu x[t] \\
| \mu y[t'], \text{ for } y \neq x, t' \in L_1^s \cap L_1^s.
\]
2.3. Definition (Syntax for $L_1^\xi$). The set $L_1^\xi$ of statements which are guarded for all $x \in Stmv$ is defined by

$$t ::= a|t; s|t_1 + t_2|t_2||t|x[t],$$

where $s \in L_1$.

2.4. Remark. We extend $L_1$, $L_1^\xi$, and $L_1^\mu$ with the empty statement $E$ (see Remark 1.3).

The definitions of $FV(s)$ (free variables of $s$) and of (syntactically) closed statements are as in Section 1. The language $L_1$ differs from $L_0$ in two respects. First, the presence of communication actions entails a more sophisticated interpretation of $s_1 || s_2$. Second, the operators of global nondeterminism $s_1 + s_2$ and of local nondeterminism $s_1 \cup s_2$ of $L_0$ are differently interpreted. For an extensive discussion of $L_1$ we refer the reader to [3] (where, for obvious reasons, it is called $L_2$). After we have defined an operational semantics for $L_1$, we shall briefly discuss the intuitive meaning of $L_1$.

2.2. Operational semantics

2.5. Definition (Semantic universe $P_1$). Let, as in Definition 1.10, the set $A^\times$ be defined as $A^\times = A^* \cup A^\omega$. We extend this set by allowing as the last element of a finite sequence a special element $\varnothing$, which will be used to denote deadlock:

$$A^\times = A^* \cup A^* \cup \{\varnothing\} \cup A^\omega.$$

Now we define a complete metric space $P_1$, with typical elements $p, q, \ldots$, as $P_1 = \mathcal{P}_{\text{ne}}(A_1^\times)$, the set of all non-empty, compact subsets of $A_1^\times$. As a metric on $P_1$ we take $(d_{\lambda_\gamma})_\lambda$ (see Definition A.7(d)). We shall use $P_1$ as the semantic universe for the operational semantics of $L_1$, which will again (as for $L_0$) be based on a transition relation.

2.6. Definition (Transition relation for $L_1^\xi$). We define a transition relation

$$\rightarrow \subseteq L_1^\xi \times A \times L_1$$

as the smallest relation satisfying

(i) $a \rightarrow aE$, for $a \in A$ (please note that it is also possible that $a \in C$!),

(ii) for all $a \in A$, $s, t \in L_1^\xi$ and $s', \bar{s} \in L_1$, if $s' \neq E$, then

$$s \rightarrow s' \Rightarrow (s; \bar{s} \rightarrow s'; \bar{s})$$

$$\wedge s + t \rightarrow aE \wedge s + s \rightarrow aE$$

$$\wedge s\langle t \rightarrow aE \wedge s \rightarrow t\rangle$$

$$\wedge \mu x[s] \rightarrow s'[\mu x[s]/x]),$$
and if $s' = E$, then

$$s \xrightarrow{a} E \Rightarrow (s; s' \xrightarrow{a} s')$$

$$\land s + t \xrightarrow{a} E \land t + s \xrightarrow{a} E$$

$$\land s \parallel t \xrightarrow{a} t \land t \parallel s \xrightarrow{a} t$$

$$t \mu x[s] \xrightarrow{a} E).$$

(iii) for all $c \in C$, $s, t \in L^E_1$, $s', t' \in L_1$, if $s' \neq E \neq t'$, then

$$(s \xrightarrow{c} s' \land t \xrightarrow{c} t') \Rightarrow s \parallel t \xrightarrow{c} s' \parallel t',$$

and if $s' = E$, then

$$(s \xrightarrow{c} E \land t \xrightarrow{c} t') \Rightarrow s \parallel t \xrightarrow{c} t'.$$

2.7. Definition ($\Phi_1$). Let $\Phi_1 : (L^c_1 \rightarrow P_1) \rightarrow (L^c_1 \rightarrow P_1)$ be given by

$$\Phi_1(F)(s) = \begin{cases} 
\{a\} & \text{if } s = E, \\
\{\emptyset\} & \text{if } \{a \mid \exists s'[s \xrightarrow{a} s'] \land a \notin C\} = \emptyset, \\
\bigcup \{a \cdot F(s') \mid s \xrightarrow{a} s' \land a \notin C\} & \text{otherwise}
\end{cases}$$

for $F \in L^c_1 \rightarrow P_1$ and $s \in L^c_1$.

2.8. Definition $\mathcal{O}_1 = \text{FixedPoint}(\Phi_1)$.

2.9. Examples. The following examples illustrate the intended meaning of $L_1$:

$$\mathcal{O}_1\{x\} = \{\emptyset\}, \quad \mathcal{O}_1\{c\\parallel c\} = \{\tau\}.$$

$$\mathcal{O}_1\{(a; c)\parallel (b; c)\} = \{abr, bar\}.$$

$$\mathcal{O}_1\{(a; b) + (a; c)\} = \{ab, a\emptyset\},$$

$$\mathcal{O}_1\{a; (b + c)\} = \{ab\}. \quad \text{for } c \in C, a, b \in A \setminus C.$$

Thus, with global nondeterminacy $+$, the statements $s_1 = (a; b) + (a; c)$ and $s_2 = a; (b + c)$ have different meanings under $\mathcal{O}_1$. This difference can be understood as follows. If $s_1$ performs the elementary action $a$, the remaining statement is either the elementary action $b$ or the communication $c$. In case of $c$, a deadlock occurs since no matching communication is available. However, if $s_2$ performs $a$, the remaining statement is $b + c$, which cannot deadlock because the action $b$ is possible. Thus, communications $c$ create deadlock only if neither a matching communication $c$ nor an alternative elementary action $b$ is available.

We again characterize the operational semantics by defining for each statement $s$ a set of pairs of which the first element denotes a possible first step of $s$. 
2.10. Definition (Initial steps). We define a function \( I : L^I \rightarrow \mathcal{P}_{\text{fin}}(A \times L) \) by induction on \( L^I \).

(i) \( I(E) = \emptyset \) and \( I(a) = \{(a, E)\} \).

(ii) Suppose \( I(s) = \{(a_i, s_i)\}, I(t) = \{(b_j, t_j)\} \) for \( s, t \in L^I \), \( a_i, b_j \in A \), and \( s_i, t_j \in L_1 \). (The variables \( i \) and \( j \) range over some finite sets of indices, which we have omitted.) Then

\[
\begin{align*}
I(s; s) &= \{(a_i, s_j; s)\} \quad \text{(for } s \in L_1),
I(s + t) &= I(s) \cup I(t),
I(s \parallel t) &= \{(a_i, s_i \parallel t)\} \cup \{(b_j, s \parallel t_j)\} \cup \{(a_i, s_i \parallel t_j) \mid a_i = b_j\},
I(\mu x[s]) &= \{(a_i, s_i[\mu x[s]/x])\}.
\end{align*}
\]

2.11. Lemma. \( \forall a \in A \forall s \in L^I \forall s' \in L_1 \ [s \rightarrow_a s' \iff (a, s') \in I(s)] \).

2.12. Corollary. For \( F \in L^I \rightarrow P_1 \) and \( s \in L^I \), such that \( \{a \mid \exists s'[s \rightarrow_a s'] \land a \notin C\} \neq \emptyset \), we have

\[
\Phi_1(F)(s) = \bigcup \{a \cdot F(s') \mid (a, s') \in I(s) \land a \notin C\}.
\]

2.3. Denotational semantics

We follow [33] in introducing a branching time semantics for \( L_1 \). First we have to define a suitable semantic universe. It is obtained as a solution of the following domain equation:

\[
(*) \quad \bar{P} \equiv \{p_0\} \cup \mathcal{P}_{\text{co}}(A \times \bar{P}).
\]

Such a solution we call a domain, and its elements are called processes. We can read the equation as follows: a process \( p \in \bar{P} \) is either \( p_0 \), the so-called nil process indicating termination, or it is a (compact) set \( X \) of pairs \((a, q)\), where \( a \) is the first action taken and \( q \) is the resumption, describing the rest of \( p \)'s actions. If \( X \) is the empty set, it indicates deadlock (as does \( \hat{o} \) in the operational semantics). For reasons of cardinality, \((*)\) has no solution when we take all subsets, rather than all compact subsets of \( A \times \bar{P} \). Moreover, we should be more precise about the metrics involved. We should have written \((*)\) like this.

2.13. Definition (Semantic universe \( \bar{P}_1 \)). Let \((\bar{P}_1, d)\) be a complete metric space satisfying the following reflexive domain equation

\[
\bar{P} \equiv \{p_0\} \cup \mathcal{P}_{\text{co}}(A \times \text{id}_{1/2}(\bar{P})),
\]

where, for any positive real number \( c \), \( \text{id}_c \) maps a metric space \((M, d)\) onto \((M, d')\) with \( d'(x, y) = c \cdot d(x, y) \), and \( \cup \) denotes the disjoint union (see Definition A.7). (For a formal definition of the metric on \( \bar{P} \) we refer the reader to Appendix A.) Typical elements of \( \bar{P}_1 \) are \( p \) and \( q \), and are called processes.
We shall not go into the details of solving this equation. In [6], it was first described how to solve this type of equation in a metric setting. In [1] this approach is reformulated and extended in a category-theoretic setting.

2.14. Definition (Semantic operators). The operators \( \tilde{\cdot}, \tilde{\cdot}, \tilde{\cdot} \): \( \tilde{P}_1 \times \tilde{P}_1 \rightarrow \tilde{P}_1 \) are defined as follows. Let \( p, q \in \tilde{P}_1 \), then

\[
\begin{align*}
(\text{i}) & \quad p; q = \begin{cases} q & \text{if } p = p_0, \\
\{(a, p; q) | (a, p) \in p\} & \text{otherwise.}
\end{cases} \\
(\text{ii}) & \quad p \tilde{+} q = \begin{cases} p & \text{if } q = p_0, \\
q & \text{if } p = p_0, \\
p \cup q & \text{otherwise.}
\end{cases} \\
(\text{iii}) & \quad p \tilde{\upharpoonright} q = \begin{cases} p & \text{if } q = p_0, \\
q & \text{if } p = p_0, \\
\{(a, p \tilde{\upharpoonright} q) | (a, p) \in p\} \cup \{(a, p) | (a, q) \in q\} & \text{otherwise.}
\end{cases}
\end{align*}
\]

(We often write \( op \) rather than \( \tilde{op} \) if no confusion is possible.) For a justification of these definitions see Remark 1.22.

2.15. Definition (Semantic environments). We use \( \Gamma \) to denote the set of semantic environments (as in Definition 1.24), with typical elements \( \gamma \), given by \( \Gamma = Stmv \rightarrow \tilde{P}_1 \).

2.16. Definition (\( \Psi_1, \Theta_1 \)). We define the denotational semantics \( \Theta_1 \) of \( L_1 \) as \( \Theta_1 = \text{FixedPoint}(\Psi_1) \), where \( \Psi_1 : L_1 \rightarrow \Gamma \rightarrow \tilde{P}_1 \) is defined exactly as \( \Psi_0 \) in Definition 1.25 but for the following two clauses:

\[
\Psi_1(F)(a)(\gamma) = \{(a, p_0)\}, \quad \Psi_1(F)(E)(\gamma) = p_0.
\]

We realize that it must be difficult for the reader who sees this type of denotational semantics for the first time to understand and appreciate it. Nevertheless, we consider it for our purposes preferable to refer the reader to [3], where he can find an extensive explanation. In this paper, we want to stress the technique of proving semantic equivalences, with which we now proceed.

2.4. Semantic equivalence of \( \Theta_1 \) and \( \Theta_1 \)

It is quite obvious that the result of the previous section, as formulated in Corollary 1.47, namely that

\[
\forall s \in L_1^x \forall \gamma \in \Gamma \quad \left[ \Theta(s) = \tilde{\Theta}_0(s) \right],
\]

does not hold for the semantic functions \( \Theta_1 \) and \( \Theta_1 \). The semantic universe \( P_1 \) of
Contractions in comparing concurrency semantics

\( \mathcal{C}_1 \) is a set of sets of streams, whereas \( \bar{\mathcal{D}}_1 \), the semantic universe for \( \mathcal{D}_1 \), is a set of tree-like, branching processes. Thus, when comparing the types of \( \mathcal{C}_1 : L_1 \rightarrow P_1 \) and \( \mathcal{D}_1 : L_1 \rightarrow I \rightarrow \bar{\mathcal{D}}_1 \), we observe that besides the fact that \( \mathcal{D}_1 \) takes a statement as an argument as well as an environment, which \( \mathcal{C}_1 \) does not (as is the case with \( \mathcal{D}_0 \) and \( \mathcal{C}_0 \)), there is a second difference between \( \mathcal{C}_1 \) and \( \mathcal{D}_1 \). That is, they have different co-domains, \( P_1 \neq \bar{\mathcal{D}}_1 \) (which is not the case in the previous section). The strategy we shall follow to relate \( \mathcal{C}_1 \) and \( \mathcal{D}_1 \) is to define functions \( \mathcal{C}_1' : L_1 \rightarrow \Delta \rightarrow P_1 \) (where \( \Delta \) will again be a set of syntactic environments) and \( \mathcal{D}_1' : L_1 \rightarrow \Delta \rightarrow \bar{\mathcal{D}}_1 \), and then relate \( \mathcal{C}_1 \) and \( \mathcal{C}_1' \) (similarly as with \( \mathcal{C}_0 \) and \( \mathcal{C}_0' \)), next \( \mathcal{D}_1' \) and \( \mathcal{D}_1 \) (similarly as with \( \mathcal{D}_0' \) and \( \mathcal{D}_0 \)), and finally compare \( \mathcal{C}_1' \) and \( \mathcal{D}_1' \) by using a suitable abstraction operator \( \alpha : \bar{\mathcal{D}}_1 \rightarrow P_1 \). As in the previous section, we define \( \mathcal{C}_1' \) (and \( \mathcal{D}_1' \)) as fixed point of a contraction. We start with the comparison of \( \mathcal{C}_1 \) and \( \mathcal{C}_1' \).

2.17. Definition (Syntactic environments). The set \( \Delta \) of syntactic environments, with typical elements \( \delta \), is given by

\[
\Delta = \{ \delta \mid \delta \in (Stmv \rightarrow^{\text{fin}} L_1) \land (\delta \text{ is normal}) \}.
\]

(For the notion of normal see Definition 1.29.)

We formulate an induction principle for \( L_1 \times \Delta \), as in Theorem 1.31.

2.18. Theorem (Induction principle for \( L_1 \times \Delta \)). Let \( \Xi \subseteq L_1 \times \Delta \). If

1. \( A \times \Delta \subseteq \Xi \),
2. \( \{s, t\} \times \Delta \subseteq \Xi \Rightarrow \{s; s + t, s \| t\} \times \Delta \subseteq \Xi \), for \( s, t, \bar{s} \in L_1 \),
3. \( \{s\} \times \Delta \subseteq \Xi \Rightarrow \{s \times a \} \times \Delta \subseteq \Xi \), for \( s \in L_0 \),
4. \( (\delta(x), \delta) \in \Xi \Rightarrow (x, \delta) \in \Xi \), for \( x \in Stmv \), and \( \delta \in \Delta \),

then \( \Xi = L_1 \times \Delta \).

Proof. See Theorem 1.31.

2.19. Definition (Initial steps with syntactic environments). As in Definition 1.33, we use the induction principle to define a function

\[ I' : L_1 \rightarrow \Delta \rightarrow \mathcal{P}_{\text{fin}}(A \times L_1 \times \Delta). \]

1. \( I'(E)(\delta) = \emptyset \), and \( I'(a)(\delta) = \{(a, E, \delta)\} \) for all \( a \in A, \delta \in \Delta \).
2. Suppose \( I'(s) = \lambda \delta \cdot \{(a, s, \delta_i)\} \) and \( I'(t) = \lambda \delta \cdot \{(b, t, \delta_j)\} \) for \( s, t \in L_1, a_i, b_j \in A \), and \( \delta_i, \delta_j \in \Delta \). Then

\[
I'(s, s)(\delta) = \{(a_i, s, \delta_i)\} \quad (\text{for all } s \in L_1)
\]

\[
I'(s + t)(\delta) = I'(s)(\delta) \cup I'(t)(\delta)
\]

\[
I'(s \| t)(\delta) = \{(a_i, s, \| t, \delta_i)\} \cup \{(b_j, s, \| t, \delta_j)\} \cup \{(\tau, s, \| t, \delta, \cup \delta_j) \mid \bar{a}_i = b_j\}.
\]

3. (4) As in Definition 1.33.
2.20. Remark. In the clause for $\|s\|_I$ in the above definition we take the union of
two environments, $\delta_i$ and $\delta_j$. This we can always do, if we impose the restriction
upon all $\delta_i$ and $\delta_j$ that
\[
\text{if } a_i = b_j, \quad \text{then } (\text{dom}(\delta_i) \setminus \text{dom}(\delta)) \cap (\text{dom}(\delta_j) \setminus \text{dom}(\delta)) = \emptyset.
\]
If this condition is not satisfied (and in general it is not), a suitable renaming of
variables should be applied. An example of a statement for which this should happen
is $\mu x[c;x][\mu x[c;x]]$.

2.21. Definition ($\Phi'_1$). We define $\Phi'_1: (L_1 \rightarrow \Delta \rightarrow P_1) \rightarrow (L_1 \rightarrow \Delta \rightarrow P_1)$ by
\[
\Phi'_1(F)(s)(\delta) = \begin{cases}
\{\varepsilon\} & \text{if } s = F, \\
\{\delta\} & \text{if } \{(a, s', \delta') \in I'(s)(\delta) | a \not\in C\} = \emptyset, \\
\bigcup \{a \cdot F(s')(\delta') \mid (a, s', \delta') \in I'(s)(\delta) \land a \not\in C\} & \text{otherwise},
\end{cases}
\]
for $F \in L_1 \rightarrow \Delta \rightarrow P_1$, $s \in L_1$, and $\delta \in \Delta$.

2.22. Definition $\Theta'_1 = \text{FixedPoint}(\Phi'_1)$.

2.23. Theorem (Relating $I$ and $I'$).
\[
\forall s \in L_1 \forall \delta \in \Delta \quad [I'(s)(\delta) = \{(a_i, s_i, \delta_i)\} \Rightarrow I(s(\delta)) = \{(a_i, s_i(\delta_i))\}].
\]
Proof. See Theorem 1.37. \qed

2.24. Definition. We define $\langle \rangle: (L_1^{cl} \rightarrow P_1) \rightarrow (L_1 \rightarrow \Delta \rightarrow P_1)$ by
\[
\langle \rangle F = F^{cl} = \lambda s \in L_1 \cdot \lambda \delta \in \Delta \cdot F(s(\delta))
\]
for $F \in L_1^{cl} \rightarrow P_1$.

2.25. Theorem (Relating $\Phi_1$ and $\Phi'_1$). $\forall F \in L_1^{cl} \rightarrow P_1 \ [\Phi'_1(F^{cl}) = (\Phi_1(F))^{cl}]$.
Proof. See Theorem 1.40. \qed

2.26. Corollary ($\Theta'_1 \equiv \Theta'_1$). $\forall s \in L_1 \forall \delta \in \Delta \ [\Theta'_1[\|s\|_s(\delta)] = \Theta'_1[\|s(\delta)\|]$.

Next we define $\Theta'_1: L_1 \rightarrow \Delta \rightarrow \bar{P}_1$ as the fixed point of the contraction below and
compare $\Theta_1$ and $\Theta'_1$.

2.27. Definition ($\Psi'_1$). We define $\Psi'_1: (L_1 \rightarrow \Delta \rightarrow \bar{P}_1) \rightarrow (L_1 \rightarrow \Delta \rightarrow \bar{P}_1)$ by
\[
\Psi'_1(F)(s)(\delta) = \begin{cases}
\{\varepsilon\} & \text{if } s = F, \\
\{(a, F(s')(\delta')) \mid (a, s', \delta') \in I'(s)(\delta)\} & \text{otherwise},
\end{cases}
\]
for $F \in L_1 \rightarrow \Delta \rightarrow \bar{P}_1$, $s \in L_1$, and $\delta \in \Delta$. 
2.28. Definition. $\mathcal{D}_1' = \text{FixedPoint}(\Psi_1').$

2.29. Remark. As $\mathcal{C}_1'$ also $\mathcal{D}_1'$ takes syntactic environments as arguments. Their co-domains, however, are different: $P_1 \neq \tilde{P}_1$. One could call $\mathcal{D}_1'$ a branching variant of $\mathcal{C}_1'$. Another difference is that $\mathcal{C}_1'(c)(\delta) = \{\delta\}$, whereas $\mathcal{D}_1'(c)(\delta) = \{(c, p_0)\}$, for $c \in C$ and $\delta \in \Delta$.

In order to relate $\mathcal{C}_1', \mathcal{L} \rightarrow \Delta \rightarrow \tilde{P}_1$ and $\mathcal{D}_1: L_1 \rightarrow \Delta \rightarrow \tilde{P}_1$ we use the following.

2.30. Definition. Let $\sim : (L_1 \rightarrow \Gamma \rightarrow \tilde{P}_1) \mapsto (L_1 \rightarrow \Delta \rightarrow \tilde{P}_1)$ be given by

$$\sim(F) = \tilde{F} = \lambda s \in L_1 \cdot \lambda \delta \in \Delta \cdot F(s)(\tilde{\delta}^F)$$

for $F \in L_1 \rightarrow \Gamma \rightarrow \tilde{P}_1$, where $\tilde{\delta}^F$ is defined as $\tilde{\delta}^F = \lambda x \in \text{dom}(\delta) \cdot F(\delta(x))(\tilde{\delta}^F)$. (For a justification of the definition of $\tilde{\delta}^F$ see Remark 1.43(1).)

2.31. Theorem. $\Phi_1'(\mathcal{D}_1') = \tilde{\Phi}_1$.

Proof. This theorem can be proved in essentially the same way as Theorem 1.44.

2.32. Corollary. $\mathcal{D}_1' = \tilde{\mathcal{D}}_1$.

Finally we provide the only missing link in the chain that is to connect $\mathcal{C}_1$ with $\mathcal{D}_1$: the comparison of $\mathcal{C}_1': L_1 \rightarrow \Delta \rightarrow P_1$ and $\mathcal{D}_1': L_1 \rightarrow \Delta \rightarrow \tilde{P}_1$. We relate their different semantic universes $P_1$ and $\tilde{P}_1$ in the following.

2.33. Definition (Abstraction operator $\alpha$). We define an abstraction operator $\alpha : \tilde{P}_1 \rightarrow P_1$ by $\alpha = \text{streams} \circ \text{restr}$, where $\text{restr}$ (for restriction) and $\text{streams}$ are recursively defined

(i) $\text{restr} : \tilde{P}_1 \rightarrow \tilde{P}_1$

$$p \mapsto \begin{cases} p_0 & \text{if } p = p_0, \\ \{(a, \text{restr}(p')) | (a, p') \in p \wedge a \notin C\} & \text{otherwise}. \end{cases}$$

(ii) $\text{streams} : \tilde{P}_1 \rightarrow P_1$

$$p \mapsto \begin{cases} \{\varepsilon\} & \text{if } p = p_0, \\ \{\delta\} & \text{if } p = \emptyset, \\ \{a \cdot \text{streams}(p') | (a, p') \in p\} & \text{otherwise}. \end{cases}$$

2.34. Remarks. (1) Since the definition of $\text{restr}$ and $\text{streams}$ is recursive, we have to verify that it is well formed. It suffices to note that these functions can be defined as fixed points of contracting functions (cf. Remark 1.22).
(2) The abstraction operator \( \alpha \) transforms a (branching) process \( p \in \bar{P}_1 \) into an element \( \alpha(p) \in P_1 \) in two steps. First it cuts off all branches (all subprocesses) of \( p_1 \) that are labeled with an element of \( C \): these can be regarded as failed (individual) attempts at communication. This is what \( \text{restr} \) does. Then \( \text{streams} \) takes all paths (streams) of the result of \( \text{restr}(p) \), putting a \( \delta \) symbol (denoting deadlock) at the end of all paths ending in the empty process. This can be understood as follows. When a path in \( \text{restr}(p) \) ends in the empty process this means that the operation \( \text{restr} \) has cut off everything at the end of the corresponding path in \( p \). By definition of \( \text{restr} \) only elements of \( C \) could have been present. Thus, this path in \( p \) should be interpreted as indicating a situation in which only individual communication steps can be taken. Operationally, we consider this to be a case of deadlock. Therefore, we replace this empty process by \( \delta \). This is what \( \text{streams} \) does.

Now that we have defined a mapping \( \alpha : \bar{P}_1 \to P_1 \), we extend it in the following way.

**2.35. Definition.** Let \( \alpha : (L_1 \to \Delta \to \bar{P}_1) \to (L_1 \to \Delta \to P_1) \) be defined by

\[
\alpha(F) = F'\quad \text{(notation)}
\]

for \( F \in L_1 \to \Delta \to \bar{P}_1 \). (Please note that we use again the symbol \( \alpha \). We trust that no confusion will arise from this slight abuse of language.)

**2.36. Theorem (Relating \( \psi_1 ' \) and \( \Phi_1 ' \)).** \( \forall F \in L_1 \to \Delta \to \bar{P}_1 \) \( \Phi_1 '(F') = (\psi_1 '(F'))' \).

**Proof.** Let \( F \in L_1 \to \Delta \to \bar{P}_1 \), let \( s \in L_1 \) and \( \delta \in \Delta \) be such that \( \{(a, s', \delta') \in I'(s) \times (\delta) \mid a \notin C \} \neq \emptyset \). Then

\[
\Phi_1 '(F')(s)(\delta) = \bigcup \{ a \cdot F'(s')(\delta') \mid (a, s', \delta') \in I'(s)(\delta) \land a \notin C \}
\]

\[
= \bigcup \{ a \cdot (\alpha(F(s')(\delta'))) \mid (a, s', \delta') \in I'(s)(\delta) \land a \notin C \}
\]

\[
= \text{streams}(\{(a, \text{restr}(F(s')(\delta'))) \mid (a, s', \delta') \in I'(s)(\delta) \land a \notin C \})
\]

\[
= \text{streams} \circ \text{restr}(\{(a, F(s')(\delta')) \mid (a, s', \delta') \in I'(s)(\delta) \})
\]

\[
= \alpha(\psi_1 '(F)(s)(\delta)) = (\psi_1 '(F))'(s)(\delta).
\]

If \( s \in L_1 \) and \( \delta \in \Delta \) are such that \( \{(a, s', \delta') \in I'(s)(\delta) \mid a \notin C \} = \emptyset \), then

\[
\Phi_1 '(F)(s)(\delta) = [\emptyset]
\]

\[
= \text{streams}(\emptyset)
\]

\[
= \text{streams} \circ \text{restr}(\{(a, F(s')(\delta')) \mid (a, s', \delta') \in I'(s)(\delta) \})
\]

\[
= (\psi_1 '(F))'(s)(\delta).
\]

**2.37. Corollary (\( \Phi_1 ' = \phi_1 ' \)).** \( \forall s \in L_1 \forall \delta \in \Delta \) \( \alpha(\phi_1 ' s(\delta)) = \phi_1 ' s(\delta) \).

Combining Corollaries 2.26, 2.32, and 2.37, which state

(2.26) \( \phi_1 ' = \phi_1 ' \),
now yields the main theorem of this section.

2.38. Theorem \((C_i' = (\mathcal{D}_i')^\alpha)\). \(\forall s \in L_1 \forall \delta \in \Delta \ [C_i[\llbracket s(\delta)\rrbracket] = \alpha(\mathcal{D}_i[\llbracket s(\delta)\rrbracket)]\).

2.39. Corollary. For all \(s \in L_1\) and arbitrary \(\gamma \in \Gamma\): \(C_i[\llbracket s\rrbracket] = \alpha(\mathcal{D}_i[\llbracket s\rrbracket](\gamma))\).

2.5. Summary of Section 2

We can again give a quick overview of the main theorem of this section by drawing a diagram as follows:

\[
\begin{array}{c}
L_1^{\text{cl}} \rightarrow P_1 \xrightarrow{\varphi_1} L_1^{\text{cl}} \rightarrow P_1 \\
\downarrow \gamma \quad \xrightarrow{\text{*}} \quad \downarrow \gamma \\
L_1 \rightarrow \Delta \rightarrow P_1 \xrightarrow{\varphi_1} L_1 \rightarrow \Delta \rightarrow P_1 \\
\uparrow \alpha \quad \xrightarrow{\text{*}} \quad \uparrow \alpha \\
L_1 \rightarrow \Delta \rightarrow \bar{P}_1 \xrightarrow{\psi_1} L_1 \rightarrow \Delta \rightarrow \bar{P}_1 \\
\downarrow \xrightarrow{\text{*}_{\text{fix}}} \\
L_1 \rightarrow \Gamma \rightarrow \bar{P}_1 \xrightarrow{\psi_1} L_1 \rightarrow \Gamma \rightarrow \bar{P}_1
\end{array}
\]

(Theorem 2.25) (Theorem 2.36) (Theorem 2.31)

where (as in Section 1.5) * indicates commutativity and *_{fix} indicates commutativity with respect to the fixed point of \(\Psi_1\) (that is, \(\mathcal{D}_1\)). Please note that if we could identify \(P_1\) and \(\bar{P}_1\), we could identify the second and the third horizontal lines of this diagram, leaving out the mapping \(\alpha\). This would yield a diagram of exactly the same shape as that of Section 1.5. This is just a way of rephrasing what has already been said above. The only new thing about proving semantic equivalence for \(L_1\), compared with \(L_0\), is the presence of a difference between the semantic universes \(P_1\) and \(\bar{P}_1\) of \(C_1\) and \(\mathcal{D}_1\), which made the introduction of \(\alpha\) necessary. Theorems 2.25 and 2.31 are just (slightly) modified versions of theorems already present in Section 1 (namely, Theorems 1.40 and 1.44).

3. A nonuniform language with value passing \((L_2)\)

We devote the third section of our paper to the discussion of semantic equivalence for a nonuniform language. Elementary actions are no longer uninterpreted but taken as either assignment or tests. Communication actions \(c\) and \(\bar{c}\) are refined to
actions $c?v$ and $c!e$ (with $v$ variable and $e$ an expression), and successful communication now involves two effects: both synchronization (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.

We shall introduce a language $L_2$ which embodies these features and present its operational and denotational semantics $\mathcal{Θ}_2$ and $\mathcal{D}_2$. Nonuniformity of $L_2$ calls for the notion of state in both semantic models. They now deliver sets of streams, or processes, over state transformations, not over uninterpreted actions as in the previous sections. The main goal of this section is to provide the reader with yet another example of a language to which the method for proving semantic equivalence, as developed in Sections 1 and 2, applies. Although $L_2$ will be in some sense more complex than $L_1$ and accordingly $\mathcal{Θ}_2$ and $\mathcal{D}_2$ more intricate than $\mathcal{Θ}_1$ and $\mathcal{D}_1$, the proof of the equivalence of operational and denotational semantics will essentially be the same. Because of this emphasis on proving semantic equivalence, we shall not give very much explanation when defining the semantics. For this we refer the reader again to [3], which we (roughly) follow in our definition of $\mathcal{Θ}_2$ and $\mathcal{D}_2$. Nor shall we give any proofs, because all of them can be obtained by straightforwardly modifying a corresponding one from Section 2.

3.1. Syntax

We now present the syntax of $L_2$. We use three new syntactic categories. viz.
- the set $\text{Var}$, with elements $v, w$, of individual variables
- the set $\text{Exp}$, with elements $e$, of expressions
- the set $\text{Bexp}$, with elements $b$, of boolean expressions.

We shall not specify a syntax for $\text{Exp}$ and $\text{Bexp}$. 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 $c!e$.

3.1. Definition 3.1 (Syntax for $L_2$).

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

where $t \in L_2^x$, defined in the following.

3.2. Definition (Syntax for $L_2^x$). The set $L_2^x$ of statements which are guarded for $x$ is given by

$$t ::= v := e|b|c?v|c!e$$

$$| t; s, \text{ for } s \in L_2$$

$$| t_1 + t_2| t_1||t_2$$

$$| y, \text{ for } y \neq x$$

$$| \mu x[t]$$

$$| \mu y[t'], \text{ for } y \neq x, t' \in L_2^x \cap L_2^x.$$
3.3. Definition (Syntax for $L^5_2$). The set $L^5_2$ of statements which are guarded for all $x \in Stmv$ is defined by

$$t ::= v := e | b | c ? v | c ! e | t_1 ; t_2 | t_1 || t_2 | \mu x [ t ],$$

where $s \in I_2$.

3.4. Remark. The sets $L_2$, $L^5_2$, and $L^5_2$ are extended with the empty statement $E$ (cf. Remark 1.4).

It will be useful to unite assignments $v := e$, tests $b$ and communications $c ? v$ and $c ! e$ into one set of basic steps.

3.5. Definition (Basic steps). We define the set $Bsteps$ of basic steps, with 'typical element $a$, by

$$Bstep = Comm \cup Bexp \cup Asg,$$

where the set $Comm$ of communications is defined by

$$Comm = \{ c ? v | c \in C, v \in Var \} \cup \{ c ! e | c \in C, e \in Exp \},$$

and the set $Asg$, of assignments, is defined by

$$Asg = \{ v := e | v \in Var, e \in Exp \}.$$

The sets $BSteps$ and $Comm$ can be regarded as the nonuniform equivalents of the sets of atomic actions and $C$ of communications of the previous section.

3.2. Operational semantics

3.6. Definition (Transition relation for $L^5_2$). We define $\rightarrow \subseteq L^5_2 \times Bstep \times L_2$ as the smallest relation satisfying

(i) $a \rightarrow^a E$, for all $a \in Bstep$. (Please note that it is also possible that $a \in Comm$ !)

(ii) for all $a \in Bstep$, $s$, $t \in L^5_2$ and $s'$, $\tilde{s} \in L_2$, if $s' \neq E$, then

$$a \rightarrow s' \Rightarrow (s; \tilde{s} \rightarrow s'; \tilde{s})$$

$$\land s \rightarrow t \rightarrow a s' \land t \rightarrow s'$$

$$\land s || t \rightarrow a t \leftrightarrow s'|| t \rightarrow a s'$$

$$\land \mu x [ s ] \rightarrow a s' [ \mu x [ s ] / x ]$$;
and if \( s' = E \), then
\[
\begin{align*}
\Delta s \to E \Rightarrow (s; s \to s) \\
\land s + t \to E \land t + s \to E \\
\land s \parallel t \to t \land t \parallel s \to t \\
\land \mu x[s] \to a \to E.
\end{align*}
\]

(iii) for all \( s, t \in L_2, s', t' \in L_2 \), and \( c?v, c!e \in \text{Comm} \), if \( s' = E \neq t' \), then
\[
(s \xrightarrow{c?v} s' \land t \xrightarrow{c!e} t') \Rightarrow (s \xrightarrow{V := e} s' \land t \parallel s \xrightarrow{V := e} t \parallel s'),
\]
and if \( s' = E \), then
\[
(s \xrightarrow{c!e} E \land t \xrightarrow{c?v} t') \Rightarrow (s \xrightarrow{V := e} t \land t \parallel s \xrightarrow{V := e} t \parallel s').
\]

For both operational and denotational models the notion of state is fundamental. Elements \( v, w \) in \( \text{Var} \) will have values in a set \( \text{Val} \). A state is a function that maps variables to their (current) values. Accordingly, we define the following.

3.7. Definition (States). The set \( \Sigma \) of states, with typical element \( \sigma \), is defined as
\[\Sigma = \text{Var} \rightarrow \text{Val}.\]

We shall also employ a special failure state \( \emptyset \), with \( \emptyset \notin \Sigma \), and define
\[\Sigma_\emptyset^\infty = \Sigma^* \cup \Sigma^* \cdot \{\emptyset\} \cup \Sigma_\emptyset^\infty.\]
Elements of \( \Sigma_\emptyset^\infty \) will be denoted by finite or infinite tuples \( \langle \sigma_1, \sigma_2, \ldots \rangle \). The empty tuple will be denoted by \( \emptyset \). We shall write \( \sigma \) for \( \langle \sigma \rangle \). Concatenation is defined as usual.

For expressions \( e \in \text{Exp} \) and \( b \in \text{BExp} \) we postulate a simple semantic evaluation function, details of which we \( \not\circ \) bother to provide. The values of \( e \) and \( b \) in state \( \sigma \) will be denoted simply by \( [e]_{\sigma} \in \text{Val} \) and \( [b]_{\sigma} \in \{\text{tt, ff}\} \).

3.8. Definition (Semantic universe \( P_2 \)). We define the semantic universe \( P_2 \) by \( P_2 = \Sigma \to \mathcal{P}_{nc}(\Sigma_\emptyset^\infty) \), where \( \mathcal{P}_{nc}(\Sigma_\emptyset^\infty) \) is the set of all non-empty and compact subsets of \( \Sigma_\emptyset^\infty \).

3.9. Definition (\( \Phi_2 \)). Let \( \Phi_2 : (L_2^L \to P_2) \to (L_2^L \to P_2) \) be defined by \( \Phi_2(F)(E) = \{e\} \); if \( \{a \mid \exists s' \langle s \to a s' \rangle \land (a \in \text{Asg} \lor (a \in \text{BExp} \land \parallel a \rangle_{\sigma} = \text{tt})\} = \emptyset \), then \( \Phi_2(F)(s) = \{\emptyset\} \); otherwise
\[
\Phi_2(F)(s)(\sigma) = \bigcup \{\sigma \cdot F(s')(\sigma) \mid s \to s' \land \parallel b \rangle_{\sigma} = \text{tt} \}
\]
\[
\cup \bigcup \{\sigma \cdot F(s')(\sigma) \mid s \xrightarrow{v := e} s' \},
\]
for \( F \in L^2_2 \to P_2 \) and \( s \in L_2 \), and with \( \sigma_{v \leftarrow c} = \sigma(\llbracket e \rrbracket \sigma/v) \). (The notation \( \sigma_{v \leftarrow c} \) will also be used in the sequel.)

3.10. Definition. \( \Phi_2 = \text{FixedPoint}(\Phi_2) \).

3.11. Examples

\[
\begin{align*}
\Phi_2[v := 0] &= \lambda \sigma \cdot \{ \langle \sigma(0/v) \rangle \}, \\
\Phi_2[v := 0; v := v + 1] &= \lambda \sigma \cdot \{ \langle \sigma(0/v) \rangle, \sigma(1/v), \sigma(2/v) \}, \\
\sigma(1/v), \\
\sigma(0/v), \sigma(1/v), \sigma(2/v), \\
\sigma(0/v), \sigma(1/v), \sigma(2/v), \sigma(0/v) \}, \\
\mu x[v := v + 1; x] &= \lambda \sigma \cdot \{ \langle \sigma(0/v) \rangle, \sigma(1/v), \sigma(2/v), \sigma(0/v), \ldots \}, \\
\Phi_2[v := 0; v < 0] &= \lambda \sigma \cdot \{ \langle \sigma(0/v) \rangle, \sigma \}, \\
\Phi_2[\mu x[v := x] &= \lambda \sigma \cdot \{ \langle \sigma(0/v) \rangle \}, \\
\Phi_2[\mu x[v := x]] &= \lambda \sigma \cdot \{ \langle \sigma(3/v) \rangle \}. \\
\end{align*}
\]

We can again characterize the operational model using an initial step function.

3.12. Definition (Initial steps). Let \( I : L^2_2 \to \mathcal{P}_{\text{in}}(\text{BStep} \times L_2) \) be defined by

(i) \( I(\epsilon) = \emptyset, I(a) = \{ (a, \epsilon) \} \), for \( a \in \text{BStep} \).

(ii) Suppose \( I(s) = \{ (a_i, s_i) \}, I(t) = \{ (b_j, t_j) \} \) for \( s, t \in L^2_2, a_i, b_j \in \text{BStep} \), and \( s_i, t_j \in L_2 \). Then

\[
\begin{align*}
I(s, \bar{s}) &= \{(a_i, s_i; \bar{s})\}, \text{ for } \bar{s} \in L_2, \\
I(s \uplus t) &= I(s) \cup I(t), \\
I(s \parallel t) &= \{ (a_i, s_i \parallel t) \} \cup \{ (b_j, s \parallel t_j) \} \\
&\quad \cup \{ (v := e, s_i \parallel t_j) \mid (a_i = c?v \land b_j = c!e) \lor (a_i = c!e \land b_j = c?v) \}, \\
I(\mu x[s]) &= \{(a_i, s_i[\mu x[s]/x])\}. \\
\end{align*}
\]

3.13. Lemma. \( \forall a \in \text{BStep} \forall s \in L^2_2 \forall s \in L_2 \ [s \rightarrow a s' \equiv (a, s') \in I(s)] \).

3.14. Corollary. For \( F \in L^2_2 \to P_2, \ s \in L^2_2 \) and \( \sigma \in \Sigma \) with \( \{ (a, s') \in I(s) \mid a \in \text{Asg} \lor (a \in BExp \land \llbracket a \rrbracket \sigma = \text{tt}) \} \neq \emptyset, \)

\[
\Phi_2(F)(s)(\sigma) = \bigcup \{ \sigma \cdot F(s')(\sigma) \mid (b, s') \in I(s) \land \llbracket b \rrbracket \sigma = \text{tt} \} \\
\bigcup \{ \sigma_{v \leftarrow c} \cdot F(s')(\sigma_{v \leftarrow c}) \mid (v := e, s') \in I(s) \}. 
\]
3.3. **Denotational semantics**

As in Section 2.3 we start with the definition of a suitable semantic universe. It will be a process domain that is obtained as a solution of the following domain equation:

\[ \mathcal{P} = \{ p_0 \} \cup \mathcal{P}_{\omega}(\text{SSteps} \times \mathcal{P}), \]

where the set \text{SSteps} of \textit{semantic steps}, with typical elements \( \kappa \), is given by

\[ \text{SSteps} = (\Sigma \to \Sigma) \]
\[ \cup (\Sigma \to \{tt, ff\}) \]
\[ \cup (C \times \text{Var}) \]
\[ \cup (C \times (\Sigma \to \text{Val})). \]

We can read this equation as follows: a process \( p \in \mathcal{P} \) is either \( p_0 \), the nil process, or it is a (compact) set \( X \) of semantic steps \( \kappa \in \text{SSteps} \). Each a semantic step can have one out of four forms. First it can be a state transformation. These will be used to give a semantics to assignments. Then it can be a mapping from states to the set of truth values, corresponding with boolean expressions. Next, it can be a pair \((c, v)\), corresponding with an input statement \( c?v \). And finally it can be a pair \((c, f)\), corresponding with an output statement \( c!e \). Here, \( f \) is used to denote the value of \( e \) (that is, \( [e] \in \Sigma \to \text{Val} \)).

As in Section 2.3 we should be more precise about the metrics involved. We give a formal definition below and refer the reader to Section 2.3 for further explanation and references.

3.15. **Definition (Semantic universe \( \mathcal{P}_2 \)).** Let \( (\mathcal{P}_2, d) \) be a complete metric space such that it satisfies the following domain equation:

\[ \mathcal{P} = \{ p_0 \} \cup \mathcal{P}_{\omega}(\text{SSteps} \times \text{id}_{\omega}(\mathcal{P})), \]

with \text{SSteps} as above. Typical elements of \( \mathcal{P}_2 \) will be \( p \) and \( q \).

3.16. **Definition (Semantic operators).** The operators \( \preceq, \preceq, \) and \( \|: \mathcal{P}_2 \times \mathcal{P}_2 \to \mathcal{P}_2 \) are defined as follows. Let \( p, q \in \mathcal{P}_2, \kappa \in \text{SSteps}, c \in C, v \in \text{Var}, \) and \( f \in \Sigma \to \text{Val} \). Then

\[ p^\preceq q = \begin{cases} q & \text{if } p = p_0, \\ \{ (\kappa, p^\preceq q) \mid (\kappa, p') \in p \} & \text{if } p \neq p_0. \end{cases} \]

\[ p^\preceq q = \begin{cases} p & \text{if } q = p_0, \\ q & \text{if } p = p_0, \\ p \cup q & \text{otherwise.} \end{cases} \]
(iii) If \( p = p_0 \), then \( p_0 \parallel q = q \parallel p = q \). If \( p \neq p_0 \) and \( q \neq p_0 \), then
\[
\{\langle \kappa, p_0 \parallel q \rangle | \langle \kappa, p \rangle \in p \} \\
\cup \{\langle \kappa, p \parallel q \rangle | \langle \kappa, q \rangle \in q \}
\cup \{\langle \lambda \sigma \cdot \sigma(f(\sigma)/v), p_0 \parallel q \rangle | \langle \langle \lambda (c, v), p \rangle \in p \land \langle \langle c, f \rangle, q \rangle \in q \rangle \}
\lor \{\langle \langle c, f \rangle, p \rangle \in p \land \langle \langle c, v \rangle, q \rangle \in q \} \}.
\]

For a justification of these self-referential definitions see Remark 1.22.

3.17. Definition (Semantic environments). \( \Gamma = Stmv \rightarrow \text{fin} \bar{P}_2 \) (typical elements are \( \gamma \)).

3.18. Definition (\( \Psi_2, D_2 \)). We define the denotational semantics \( D_2 \) of \( L_2 \) as \( D_2 = \text{FixedPoint}(\Psi_2) \), where \( \Psi_2 : (L_2 \rightarrow \Gamma \rightarrow \bar{P}_2) \rightarrow (L_2 \rightarrow \Gamma \rightarrow \bar{P}_2) \) is given, for \( F \in L_2 \rightarrow \Gamma \rightarrow \bar{P}_2 \), by

(i) \( \Psi_2(F)(a)(\gamma) = \{\langle \kappa_a, p_0 \rangle\} \), and \( \Psi_2(F)(E)(\gamma) = p_0 \), with

\[
\kappa_a = \begin{cases}
\lambda \sigma \cdot \sigma v := e & \text{if } a - v := e, \\
\lambda \sigma \cdot [a] \sigma & \text{if } a \in BExp, \\
\langle c, v \rangle & \text{if } a = c?v, \\
\langle c, v \cdot [v] \sigma \rangle & \text{if } a = c!e.
\end{cases}
\]

(ii) \( \Psi_2(F)(s op t)(\gamma) = \Psi_2(F)(s)(\gamma) \parallel \bar{\sigma} \Psi_2(F)(t)(\gamma) \) for \( op = :, +, \parallel \).

(iii) \( \Psi_2(F)(\mu x[s])(\gamma) = \Psi_2(F)(s)(\gamma \{F(\mu x[s])(\gamma)/x\}) \).

Similarly to Lemma 1.27 we have that \( \Psi_2 \) is contracting.

3.19. Examples

\[
D_2[v := 0](\gamma) = \{\langle \lambda \sigma \cdot \sigma \{0/v\}, p_0 \}\}
\]
\[
D_2[v := 1; v := v + 1](\gamma) = \{\langle \lambda \sigma \cdot \sigma \{1/v\}, \{\langle \lambda \sigma \cdot \sigma \{\sigma'(v) + 1/v\}, p_0 \}\} \}
\]
\[
D_2[c?v][c!3](\gamma) = \{\langle c, v \rangle, \{\langle c, \lambda \sigma \cdot 3 \rangle, p_0 \}\},
\]
\[
\langle c, \lambda \sigma \cdot 3 \rangle, \{\langle c, v \rangle, p_0 \}\},
\]
\[
\lambda \sigma \cdot \sigma \{3/v\}, p_0 \}
\]
\[
D_2[v := 0; \mu x[v := v + 1; x]] = \{\langle \lambda \sigma \cdot \sigma \{0/v\}, p \}\},
\]

where \( p \in \bar{P}_2 \) satisfies \( p = \{\langle \lambda \sigma \cdot \sigma \{\sigma(v) + 1/v\}, p \}\} \).

3.4. Semantic equivalence of \( \mathcal{O}_2 \) and \( D_2 \)

The proof of the semantic equivalence of \( \mathcal{O}_2 \) and \( D_2 \) is essentially the same as in the previous section. Therefore, we only give a brief outline of how to proceed, leaving out the details of some definitions, omitting all proofs, and stressing the
(small) differences. We define $\mathcal{O}'_\sigma = \text{FixedPoint}(\Phi'_\sigma)$ and $\mathcal{D}'_\sigma = \text{FixedPoint}(\Psi'_\sigma)$ with $\Phi'_\sigma$ and $\Psi'_\sigma$ defined as follows. Let $\Phi'_\sigma : (L_2 \to \Delta \to P_2) \to (L_2 \to \Delta \to P_2)$ be given by $\Phi'_\sigma(F)(E)(\delta) = \{e\}$; if $\{(a, s', \delta') \in \Gamma'(s)(\delta) \mid a \in \text{Asg} \land (a \in \text{BExp} \land \|\sigma\|_o = tt)\} = \emptyset$, then $\Phi'_\sigma(F)(s)(\delta) = \{\delta\}$; otherwise

$$\Phi'_\sigma(F)(s)(\delta) = \bigcup \{ (\sigma \cdot F(s')(\sigma)(\delta')(b, s', \delta') \in \Gamma'(s)(\delta) \land [b]_o = tt)$$

$$\cup \{{(\sigma_{v := e} \cdot F(s')(\delta')(\sigma_{v := e}))|(v := e, s', \delta') \in \Gamma'(s)(\delta)}\},$$

for $F \in L_2 \to \Delta \to P_2$, $s \in L_2$ and $\delta \in \Delta$ (\Delta and I' can be defined similarly to Definitions 2.6 and 2.19). Let $\Psi'_\sigma : (L_2 \to \Delta \to \bar{P}_2) \to (L_2 \to \Delta \to \bar{P}_2)$ be defined by

$$\Psi'_\sigma(F)(s)(\delta) = \begin{cases} \{\rho_0\} & \text{if } s = E, \\
\{((\kappa_a, F(s')(\delta'))|(a, s', \delta') \in \Gamma'(s)(\delta))\} & \text{otherwise}
\end{cases}$$

(with $\kappa_a$ as in Definition 3.18) for $F \in L_2 \to \Delta \to \bar{P}_2$, $s \in L_2$, and $\delta \in \Delta$.

The definitions of $\Phi'_\sigma$ and $\Psi'_\sigma$ are somewhat more involved than their counterparts from Section 2. What is different here is that a syntactic basic step does not literally coincide with the semantic step that represents its meaning. In the previous section we had elementary actions $a$ and $c$ both as syntactic and semantic entities. Here we have syntactic basic steps $v := e$, $b$, $c!e$, and $c?v$, all of which are semantically represented in a different way.

Similarly to the Definitions 2.24 and 2.30 we can define mappings

$$\langle \rangle : (L_2^{el} \to P_2) \to (L_2 \to \Delta \to P_2) \quad \text{and} \quad \sim : (L_2 \to \Gamma \to \bar{P}_2) \to (L_2 \to \Delta \to \bar{P}_2),$$

and prove

$$\mathcal{C}'_\sigma = \mathcal{D}'_\sigma \quad \text{and} \quad \mathcal{D}'_\sigma = \mathcal{D}'_\sigma.$$

Finally, we can compare $\mathcal{C}'_\sigma$ and $\mathcal{D}'_\sigma$ by recursively defining a suitable abstraction operator $\alpha : \bar{P}_2 \to P_2$ by $\alpha(p_0)(\sigma) = \{e\}$, and, for $p \neq p_0$, by

$$\alpha(p)(\sigma) = \bigcup \{ f(\sigma) \cdot \alpha(p')(f(\sigma)) \mid (f, p') \in p \land f \in \Sigma \to \Sigma \}$$

$$\cup \bigcup \{ \sigma \cdot \alpha(p')(\sigma) \mid (f, p') \in p \land (f \in \Sigma \to \{tt, \text{tt}\}) \land f(\sigma) = \text{tt} \},$$

if $\{(f, p') \mid (f, p') \in p \land (f \in \Sigma \to \Sigma \cup (f \in \Sigma \to \{tt, \text{tt}\} \land f(\sigma) = \text{tt})) \neq \emptyset$, and by

$$\alpha(p)(\sigma) = \{\delta\},$$

otherwise. (For a justification of this self-referential definition see Remark 1.22.) In $\alpha(p)(\sigma)$ all pairs $\langle k, p' \rangle \in p$ with $k \in \Sigma \to \{tt, \text{tt}\}$ and $k(\sigma) = \text{ff}$, or $k \in C \times \text{Var}$, or $k \in C \times (\Sigma \to \text{Val})$, are neglected. This corresponds with the restriction operator of Definition 2.33. A second effect of applying $\sim$ is that it transforms a (branching) process $p \in \bar{P}_2$ into a function $\alpha(p) \in P_2 \to \Sigma \to \mathcal{P}_{\text{mc}}(A^\infty)$, which yields, when supplied with an argument $\sigma$, a set of streams (in a sense the paths of $\sigma$). In this respect $\alpha$ is similar to the operator streams of Definition 2.33. Applying $\alpha$ has yet another effect. If $f \in \Sigma \to \Sigma$ and $\langle f, p' \rangle \in p$, then $f(\sigma) \cdot \alpha(p')(f(\sigma)) \in \alpha(p)(\sigma)$: the state transformation $f$ is applied to the current state $\sigma$, and the resulting state $\sigma'$ is
concatenated with $\alpha(p')(f(\sigma))$, in which $f(\sigma)$, being the new state, is passed through to $\alpha$ applied to $p'$, the resumption of $f$. In this way, the effect of different state transformations occurring subsequently in $p$ is accumulated. A simple example may illustrate this. Consider

$$p = \mathcal{D}_2[v ::= 1; v ::= v + 1]$$

$$= \{(\lambda \sigma \cdot \sigma_{v:=1}, \{\lambda \sigma' \cdot \sigma'_{v:=\sigma(v)+1}, p_0\})\}.$$

Then

$$\alpha(p)(\sigma) = \{(\sigma_{v:=1}, \alpha(\{\lambda \sigma' \cdot \sigma'_{v:=\sigma(v)+1}, p_0\})\})\}$$

$$= \{(\sigma_{v:=1}, \sigma_{v:=2}, \alpha(p_0)(\sigma_{v:=2}))\}$$

$$= \{(\sigma_{v:=1}, \sigma_{v:=2})\}.$$

Next, we extend $\alpha$ to a mapping $\alpha : (L_2 \rightarrow \Delta \rightarrow \tilde{P}_2) \rightarrow (L_2 \rightarrow \Delta \rightarrow P_2)$ by putting for $F \in L_2 \rightarrow \Delta \rightarrow \tilde{P}_2$

$$\alpha(F) = F^\alpha = \lambda s \cdot \lambda \delta \cdot \alpha(F(s)(\delta)).$$

We shall prove that

$$\forall F \in L_2 \rightarrow \Delta \rightarrow \tilde{P}_2 \hspace{1cm} [\Phi'_2(F^\alpha) = (\Psi'_2(F))^\alpha].$$

Let $F \in L_2 \rightarrow \Delta \rightarrow \tilde{P}_2$, $s \in L_2$, $\delta \in \Delta$, and $\sigma \in \Sigma$ be such that

$$\{(a, s', \delta') \in I'(s)(\delta) | a \in \text{Asg} \lor (a \in B\text{Exp} \land \|a\|_\sigma = \text{tt})\} \neq \emptyset.$$

Then

$$\Phi'_2(F^\alpha)(s)(\delta)(\sigma)$$

$$= \bigcup \{\sigma \cdot F^\alpha((s')(\delta'))(\sigma) | (b, s', \delta') \in I'(s)(\delta) \land \|b\|_\sigma = \text{tt}\}$$

$$\cup \bigcup \{\sigma_{v:=e} \cdot F^\alpha((s')(\delta'))(\sigma_{v:=e}) | (v ::= e, s', \delta') \in I'(s)(\delta)\}$$

$$= \bigcup \{\sigma \cdot (\alpha(F'(s')(\delta'))(\sigma)) | (b, s', \delta') \in I'(s)(\delta) \land \|b\|_\sigma = \text{tt}\}$$

$$\cup \bigcup \{\sigma_{v:=e} \cdot (\alpha(F'(s')(\delta'))(\sigma_{v:=e})) | (v ::= e, s', \delta') \in I'(s)(\delta)\}$$

$$= \alpha(\{(\kappa_a, F'(s')(\delta')) | (a, s', \delta') \in I'(s)(\delta')\})(\sigma) \quad (\text{with } \kappa_a \text{ as above})$$

$$= \alpha(\Psi'_2(F)(s)(\delta))(\sigma) = (\Psi'_2(F))^\alpha(s)(\delta)(\sigma).$$

The case that $\Phi'_2(F)(s)(\delta)(\sigma) = \{\delta\}$ goes similarly. This proves

$$\forall F \in L_2 \rightarrow \Delta \rightarrow \tilde{P}_2 \hspace{1cm} [\Phi'_2(F^\alpha) = (\Psi'_2(F))^\alpha].$$

Now it follows that $(\mathcal{D}_2)^\alpha = \mathcal{O}'_2$. Collecting the results from above, we see $\mathcal{O}'_2 = (\mathcal{D}_2)^\alpha$, or

$$\forall s \in L_2 \forall \delta \in \Delta \hspace{1cm} [\mathcal{O}'_2[s(\delta)] = \alpha(\mathcal{D}_2[s](\delta))],$$

with the obvious corollary, that

$$\forall s \in L_2 \forall \gamma \in \Gamma \hspace{1cm} [\mathcal{O}'_2[s] = \alpha(\mathcal{D}_2[s](\gamma))].$$
4. 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 generalized for more complex languages ($L_1$ and $L_2$). This we consider to be the strength of this approach. In [18], this idea is further explored. There a general method is designed for deriving denotational models from transition system specifications that satisfy certain syntactic constraints.

Appendix A: Mathematical definitions

A.1. Definition (Metric space). A metric space is a pair $(M, d)$ with $M$ a non-empty set and $d$ a mapping $d : M \times M \to [0, 1]$ (a metric or distance) that satisfies the following properties:

(a) $\forall x, y \in M \ [d(x, y) = 0 \iff x = y]$.

(b) $\forall x, y \in M \ [d(x, y) = d(y, x)]$.

(c) $\forall x, y, z \in M \ [d(x, y) \leq d(x, z) + d(z, y)]$.

We call $(M, d)$ and ultra-metric space if the following stronger version of property (c) is satisfied:

(c') $\forall x, y, z \in M \ [d(x, y) \leq \max\{d(x, z), d(z, y)\}]$.

Please note that we consider only metric spaces with bounded diameter: the distance between two points never exceeds 1.

A.2. Examples. (a) Let $A$ be an arbitrary set. The discrete metric $d_A$ on $A$ is defined as follows. Let $x, y \in A$, then

$$d_A(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{if } x \neq y. \end{cases}$$

(b) Let $A$ be an alphabet, and let $A^\infty = A^* \cup A^\omega$ denote the set of all finite and infinite words over $A$. Let, for $x \in A^\infty$, $x(n)$ denote the prefix of $x$ of length $n$, in case $\text{length}(x) \geq n$, and $x$ otherwise. We put

$$d(x, y) = 2^{-\sup\{n \mid x(n) = y(n)\}},$$

with the convention that $2^{-\infty} = 0$. Then $(A^\infty, d)$ is a metric space.
A.3. Definition. Let \((M, d)\) be a metric space, let \((x_i)\), be a sequence in \(M\).
(a) We say that \((x_i)\) is a Cauchy sequence whenever we have
\[
\forall \varepsilon > 0 \exists N \in \mathbb{N} \forall n, m > N \ [d(x_n, x_m) < \varepsilon].
\]
(b) Let \(x \in M\). We say that \((x_i)\), converges to \(x\) and call \(x\) the limit of \((x_i)\), whenever we have
\[
\forall \varepsilon > 0 \exists N \in \mathbb{N} \forall n > N \ [d(x, x_n) < \varepsilon].
\]
Such a sequence we call convergent. Notation \(\lim_{i \to \infty} x_i = x\).

(c) The metric space \((M, d)\) is called complete whenever each Cauchy sequence converges to an element of \(M\).

A.4. Definition. Let \((M_1, d_1), (M_2, d_2)\) be metric spaces.
(a) We say that \((M_1, d_1)\) and \((M_2, d_2)\) are isometric if there exists a bijection \(f : M_1 \to M_2\) such that \(\forall x, y \in M_1 \ [d_2(f(x), f(y)) = d_1(x, y)]\). We then write \(M_1 = M_2\). When \(f\) is not a bijection (but only an injection), we call it an isometric embedding.
(b) Let \(f : M_1 \to M_2\) be a function. We call \(f\) continuous whenever for each sequence \((x_i)\), with limit \(x\) in \(M_1\), we have that \(\lim_{i \to \infty} f(x_i) = f(x)\).
(c) Let \(A \geq 0\). With \(M_1 \to^A M_2\) we denote the set of functions \(f\) from \(M_1\) to \(M_2\) that satisfy the following property:
\[
\forall x, y \in M_1 \ [d_2(f(x), f(y)) \leq A \cdot d_1(x, y)].
\]
Functions \(f\) in \(M_1 \to^A M_2\) we call non-distance-increasing (NDI), functions \(f\) in \(M_1 \to^\varepsilon M_2\) with \(0 < \varepsilon < 1\) we call contracting.

A.5. Proposition. (a) Let \((M_1, d_1), (M_2, d_2)\) be metric spaces. For every \(A \geq 0\) and \(f \in M_1 \to^A M_2\) we have that \(f\) is continuous.
(b) (Banach's fixed-point theorem.) Let \((M, d)\) be a complete metric space and \(f : M \to M\) a contracting function. Then there exists an \(x \in M\) such that the following holds:
1. \(f(x) = x\) (\(x\) is a fixed point of \(f\)),
2. \(\forall y \in M \ [f(y) = y \Rightarrow y = x]\) (\(x\) is unique),
3. \(\forall x_0 \in M \ [\lim_{n \to \infty} f^{(n)}(x_0) = x]\), where \(f^{(n+1)}(x_0) = f(f^{(n)}(x_0))\) and \(f^{(0)}(x_0) = x_0\).

A.6. Definition (Compact subsets). A subset \(X\) of a complete metric space \((M, d)\) is called compact whenever each sequence in \(X\) has a subsequence that converges to an element of \(X\).

A.7. Definition. Let \(\mathcal{M}, (d), (M_1, d_1), \ldots, (M_n, d_n)\) be metric spaces.
(a) With \(M_1 \to M_2\) we denote the set of all continuous functions from \(M_1\) to \(M_2\).
We define a metric \( d_F \) on \( M_1 \to M_2 \) as follows. For every \( f_1, f_2 \in M_1 \to M_2 \)

\[
d_F(f_1, f_2) = \sup_{x \in M_1} \{ d_2(f_1(x), f_2(x)) \}.
\]

For \( A \geq 0 \) the set \( M_1 \to^A M_2 \) is a subset of \( M_1 \to M_2 \), and a metric on \( M_1 \to^A M_2 \) can be obtained by taking the restriction of the corresponding \( d_F \).

(b) With \( M_1 \cup \cdots \cup M_n \) we denote the disjoint union of \( M_1, \ldots, M_n \), which can be defined as \( \{1\} \times M_1 \cup \cdots \cup \{n\} \times M_n \). We define a metric \( d_U \) on \( M_1 \cup \cdots \cup M_n \) as follows. For every \( f, g \in M_1 \cup \cdots \cup M_n \)

\[
d_U(f, g) = \begin{cases} d(f, g) & \text{if } f \neq g \\
1 & \text{if } f = g \end{cases}
\]

(c) We define a metric \( d_P \) on \( M_1 \times \cdots \times M_n \) by the following clause. For every \( (x_1, \ldots, x_n), (y_1, \ldots, y_n) \in M_1 \times \cdots \times M_n \)

\[
d_P((x_1, \ldots, x_n), (y_1, \ldots, y_n)) = \max_{i \leq j \leq n} \{ d(x_i, y_i) \}.
\]

(d) Let \( \mathcal{P}_{nc}(M) = \{ X \mid X \subseteq M \land X \text{ is compact and non-empty} \} \). We define a metric \( d_H \) on \( \mathcal{P}_{nc}(M) \), called the Hausdorff distance, as follows. For every \( X, Y \in \mathcal{P}_{nc}(M) \)

\[
d_H(X, Y) = \max \{ \sup_{x \in X} d(x, Y), \sup_{y \in Y} d(y, X) \},
\]

where \( d(x, Z) = \inf_{Z \subseteq Z} \{ d(x, z) \} \) for every \( Z \subseteq M, x \in M \). In \( \mathcal{P}_{co}(M) = \{ X \mid X \subseteq M \land X \text{ is compact} \} \) we also have the empty set as an element. We define \( d_H \) on \( \mathcal{P}_{co}(M) \) as above but extended with the following case. If \( X \neq \emptyset \), then

\[
d_H(\emptyset, X) = d_H(X, \emptyset) = 1.
\]

(e) Let \( c \in [0, \infty) \). We define \( \text{id}_c(M, d) = (M, c \cdot d) \).

A.8. Proposition. Let \( (M, d), (M_1, d_1), \ldots, (M_n, d_n), d_F, d_U, d_P \) and \( d_H \) be as in Definition A.7 and suppose that \( (M, d), (M_1, d_1), \ldots, (M_n, d_n) \) are complete. We have that

(a) \( (M_1 \to M_2, d_F), (M_1 \to^A M_2, d_F) \),

(b) \( (M_1 \cup \cdots \cup M_n, d_U) \),

(c) \( (M_1 \times \cdots \times M_n, d_P) \),

(d) \( (\mathcal{P}_{nc}(M), d_H), \) and \( (\mathcal{P}_{co}(M), d_H) \)

are complete metric spaces. If \( (M, d) \) and \( (M_1, d_1) \) are all ultra-metric spaces these composed spaces are again ultra-metric. (Strictly speaking, for the completeness of \( M_1 \to M_2 \) and \( M_1 \to^A M_2 \) we do not need the completeness of \( M_1 \). The same holds for the ultra-metric property.)
The proofs of Proposition A.8(a), (b), and (c) are straightforward. Part (d) is more involved. It can be proved with the help of the following characterization of the completeness of the Hausdorff metric.

**A.9. Proposition.** Let \((\mathcal{P}_{co}(M), d_H)\) be as in Definition A.7. Let \((X_i)\), be a Cauchy sequence in \(\mathcal{P}_{co}(M)\). We have

\[
\lim_{i \to \infty} X_i = \{\lim_{i \to \infty} x_i | x_i \in X_i, (X_i) \text{ is a Cauchy sequence in } M\}.
\]

The proof of Proposition A.9 can be found in [12] as a generalization of a similar result (for closed subsets) in (7) and (8).

**Acknowledgment**

We are much indebted to Jaco de Bakker, John-Jules Meijer, Ernst-Rudiger Olderog, and Jeffrey Zucker, authors and co-authors of the paper [4, 5, 33, 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, Aric de Bruin, John-Jules Meijer, and Erik de Vink. Finally, we express our thanks to Dini Verloop, who has expertly typed this document.

**References**


