Denotational Models for Programming Languages: Applications of Banach's Fixed Point Theorem

J.W. de Bakker*† & E.P. de Vink†

Abstract

For an abstract programming language both a linear and a branching denotational semantics are developed. The main instrument for the construction of the two models and for the semantical operators involved is the classical Banach Fixed Point Theorem. Via higher-order transformations the various semantical definitions are justified by their characterization as —necessarily unique— fixed points of contractions on a complete metric space. Additionally the Banach Theorem proves itself useful in relating the two models presented.

1 Introduction

This paper presents an introduction to denotational semantics for programming languages starting from elementary metric topology. Using a simple programming language as a vehicle the use of Banach's Fixed Point Theorem for the justification of semantical definitions is shown. The method ultimately relies on the uniqueness of fixed points of contractions on a complete metric space. The techniques presented here (all exploiting Banach's result) have been applied for developing, amongst others, denotational models for a wide range of programming concepts. The monograph [BV96] provides a comprehensive account of the method illustrated below.

In the study of the semantics of programming languages, one investigates ways of assigning *meanings* to programs. Expressed in mathematical terminology, one is interested in functions

$$\mathcal{M}: \mathit{Prog} \to \mathbb{P}, \tag{1.1}$$

In equation (1.1) *Prog* is a programming language, i.e., a collection of programs.

*Department of Software Technology, CWI, P.O.Box 94079, NL–1090 GB Amsterdam, The Netherlands, e-mail: `jaco@cwi.nl`.

†Faculty of Mathematics and Computer Science, Vrije Universiteit, De Boelelaan 1081a, NL–1081 HV Amsterdam, The Netherlands, e-mail: `vink@cs.vu.nl`.

The codomain $\mathbb{P}$ is the collection of meanings, i.e., a set of mathematical entities which are chosen for the purpose of modeling (certain aspects of) the computation specified by the program. Usually, $\mathbb{P}$ is referred to as the 'semantical domain'. In general, in order to be able to express the meanings of the constructs from the language *Prog*, the domain $\mathbb{P}$ will be equipped with some mathematical structure. For example, *Prog*, as given below, includes a syntactical construction for sequential composition (';') and also features recursion. The mathematical structure should enable us to define their effect in a rigorous manner.

Thus, $\mathcal{M}$ is a mapping assigning to each program π in *Prog* its meaning $\mathcal{M}(\pi)$. Accordingly, the study of semantics centers around the development of methods to specify the functions $\mathcal{M}$ (and associated domains $\mathbb{P}$) for a wide range of languages.

We will illustrate our method for the simple rudimentary programming language *Prog* given below. Atomic programs in *Prog* are actions taken from an abstract set *Act* which is further left unspecified. Composite programs can be obtained through sequential composition ';' and choice or alternative composition '+'. The intuitive meaning of $\pi_1 ; \pi_2$ is execution of π_1 first followed by execution of π_2 (the latter only in case π_1 happens to terminate). The construction $\pi_1 + \pi_2$ introduces nondeterminism: execution of $\pi_1 + \pi_2$ can either be execution of π_1 or of π_2. Which of these possibilities is taken is not specified. So, for the modeling of $\pi_1 + \pi_2$ both options are incorporated. Additionally in *Prog* we have recursion by means of procedure variables x from some postulated set *PVar*. A global 'declaration' D for procedure variables is assumed, i.e. a mapping $D\colon PVar \to Prog$ which associates to each procedure variable x a program $D(x)$ called its procedure body or its definition. In general, the procedure variable x may occur in its definition $D(x)$. It is for this reason that infinite computations may arise. Thus, in the modeling of *Prog*, we will have to cater for sequencing, nondeterminacy and both finite and infinite behaviour.

In this paper we focus on the type of semantics which is traditionally called 'denotational.' This terminology stems from the principle that 'programs denote values.' Typically, denotational semantics adheres to the 'compositionality principle,' stating that the meaning of a program is composed from the meanings of its constituent parts. Though appealingly simple, this principle raises several subsidiary questions. Here we only mention the way to construct 'semantical' operators and to handle recursion, for which additional techniques in terms of so-called fixed point definitions are necessary. Fortunately, our basing the semantical models on metric topology pays off nicely. The classical Banach Fixed Point Theorem proves to be a powerful instrument for both the construction and comparison of semantical definitions.

Below we will provide two denotational models for a language *Prog*, which differ in the way nondeterminacy is dealt with. The models will be mappings of the form $\mathcal{D}\colon Prog \to \mathbb{P}$ where $\mathbb{P}$, the mathematical structure used as denotational domain, is a complete metric space. The principle of compositionality boils down to the equation

$$\mathcal{D}(\pi_1 * \pi_2) \quad = \quad \mathcal{D}(\pi_1) * \mathcal{D}(\pi_2)$$

for all $\pi_1, \pi_2 \in Prog$ and $* \in \{;, +\}$. Here the operator '$*$' at the left-hand side is a symbol belonging to the syntax of *Prog* whereas the operator '$*$' at the right-hand side is the corresponding semantical operator which will be given as a binary operation on $\mathbb{P}$.[‡] So, the mappings $\mathcal{D}$ can be viewed as homomorphisms when the semantical domains $\mathbb{P}$ are interpreted as algebras.

The first model $\mathcal{D}_1$ for *Prog* will be a so-called linear model. Programs are interpreted by $\mathcal{D}_1$ as sets of (finite and infinite) sequences. For the linear model the characteristic equation

$$\mathcal{D}_1(\pi_1 \,;(\pi_2 + \pi_3)) = \mathcal{D}_1((\pi_1 \,; \pi_2) + (\pi_1 \,; \pi_3)) \tag{1.2}$$

will hold. The second model $\mathcal{D}_2$ will of the 'branching' kind. Here to each program a tree-like structure is associated. Equation 1.2 does not hold for $\mathcal{D}_2$. In fact, we have

$$\mathcal{D}_2(a\,;(b + c)) \neq \mathcal{D}_2((a\,;b) + (a\,;c)).$$

The motivation for the distinction between linear versus branching models goes beyond the scope of this paper. (See the last section for some further references.)

The collection Act^∞ of finite and infinite sequences over Act will come equipped with the Baire-distance based on the length of common prefixes. The semantical domain $\mathbb{P}_1$ for the model $\mathcal{D}_1$ will be the hyperspace of (nonempty) compacta of Act^∞. For the model $\mathcal{D}_2$ the situation is not so simple. The semantical domain $\mathbb{P}_2$ will be given by a system of 'domain equations'. Both topological methods and techniques from category theory may be used to guarantee solvability of such a system. Cf. Section 5.

The remainder of the paper is organized as follows: The next section collects the various basic metric results. Section 3 presents the linear model $\mathcal{D}_1$

[‡]It is customary to use the infix notation of the syntactical operator for the semantical operator as well.

for *Prog* and the first use of higher-order transformations for defining semantical operators and mappings. Section 4 extends this to the setting of the branching domain $\mathbb{P}_2$ and also covers the relationship of the linear semantics $\mathcal{D}_1$ and the branching model $\mathcal{D}_2$. Section 5 contains a short overview of the historical perspective of the metric approach to programming language semantics and provides further references to the literature.

2 Mathematical preliminaries

The main tool from metric topology for the semantical constructions below is the classical result known as Banach's Fixed Point Theorem ([Ban22]).

Theorem 2.1 *Let (M, d) be a complete metric space, and let $f: M \to M$ be contractive. Then*

(a) There exists $x \in M$ such that $f(x) = x$.

(b) The fixed point is unique: If $f(x) = x$, $f(y) = y$, then $x = y$. We shall write $\mathrm{fix}(f)$ for the unique fixed point of f.

In the next section the Banach Theorem will be exploited in a setting of sets of sequences. Here we introduce the basic notations, definitions and results for this.

Definition 2.2 Let $\mathbb{N}$ be the set of natural numbers including 0. Let $\mathbb{N}_{\geq 1}$ be the set of positive natural numbers. A nonempty subset N of $\mathbb{N}_{\geq 1}$ is called an initial segment if $\forall n, m \in \mathbb{N}_{\geq 1}$: $n \in N \wedge m < n \Rightarrow m \in N$.

(a) Let $\mathcal{A}$ be a nonempty set. Define the collection $\mathcal{A}^\infty$ of *words* over $\mathcal{A}$ by $\mathcal{A}^\infty = \{ w: N \to \mathcal{A} \mid N \subseteq \mathbb{N}_{\geq 1} \text{ an initial segment } \}$. For $n \in \mathbb{N}$ and $w \in \mathcal{A}^\infty$, $w: N \to \mathcal{A}$ say, the word $w[n]$ is the restriction $w \restriction (N \cap \{1, \ldots, n\})$.

(b) The Baire-distance $d_B: \mathcal{A}^\infty \times \mathcal{A}^\infty \to [0, 1]$ ([Bai09]) is given by

$$d_B(w, v) = 2^{-\sup\{\, n \in \mathbb{N} \mid w[n] = v[n] \,\}}$$

for $w, v \in \mathcal{A}^\infty$. By convention $\sup(\emptyset) = 0$ and $2^{-\infty} = 0$.

We usually write $w = a_1 a_2 \cdots a_n$ if $w\colon \{1, \ldots, n\} \to \mathcal{A}$ is such that $w(i) = a_i$ $(1 \le i \le n)$ and similarly $w = a_1 a_2 \cdots$ if $w\colon \mathbb{N}_{\ge 1} \to \mathcal{A}$ satisfies $w(i) = a_i$ $(1 \le i < \infty)$. In the first case w is called a *finite* word of length n, in the second case w is called *infinite*. $\mathcal{A}^*$ and $\mathcal{A}^\omega$ will be the subsets of finite and infinite words, respectively. So $\mathcal{A}^\infty = \mathcal{A}^* \cup \mathcal{A}^\omega$. We refer to $w[n]$ as the prefix of w of length (at most) n. We distinguish $\epsilon\colon \emptyset \to \mathcal{A}$. It is called the empty word. Note $w[0] = \epsilon$ for each $w \in \mathcal{A}^\infty$. Typically we have

- $d_B(abc, abe) = \frac{1}{4}$, $d_B(abc, ab) = \frac{1}{4}$, $d_B(abc, acb) = \frac{1}{2}$, $d_B(abc, abc) = 0$
- $d_B(\epsilon, w) = 1$ for $w \in \mathcal{A}^\infty \setminus \{\epsilon\}$, since $w[1] \ne \epsilon$ for nonempty words w
- $d_B(a^n, a^\omega) = 2^{-n}$, $n \ge 0$ (employing the abbreviation a^n for the word of n consecutive a's and a^ω for the infinite word of a's).

A nonempty word w has a first letter a, say, and a possibly empty remainder w'. We write $w = a \cdot w'$ in such a situation. A basic property of the Baire distance is $d_B(a \cdot w, a \cdot v) = \frac{1}{2} d_B(w, v)$ for arbitrary $a \in \mathcal{A}$, $w, v \in \mathcal{A}^\infty$. The following result is well-known.

Theorem 2.3 *$(\mathcal{A}^\infty, d_B)$ is a complete (ultra)metric space.*

The Baire-distance on $\mathcal{A}^\infty$ is an ultrametric, i.e.

$$d_B(w, u) \le \max\{\, d_B(w, v), d_B(v, u) \,\}$$

for $w, v, u \in \mathcal{A}^\infty$. This is typical for our applications. All distances used in following sections are in fact ultrametrics. Although useful in some situations of semantic modeling, we have no occasion to profit from the strong triangle-inequality here.

Before we consider the hyperspace of nonempty compact subsets of $\mathcal{A}^\infty$, we first introduce the Hausdorff-distance on arbitrary subsets.

Definition 2.4

(a) Let $a \cdot W = \{\, a \cdot w \mid w \in W \,\}$ for $a \in \mathcal{A}$, $W \subseteq \mathcal{A}^\infty$.

(b) The Hausdorff-distance $d_H\colon \mathcal{P}(\mathcal{A}^\infty) \times \mathcal{P}(\mathcal{A}^\infty) \to [0, 1]$ ([Hau14]) is given by $d_H(W, V) =$

$$\max\{\, \sup\{\, d_B(w, V) \mid w \in W \,\}, \sup\{\, d_B(v, W) \mid v \in V \,\} \,\}$$

for all $W, V \subseteq \mathcal{A}^\infty$, where $d_B(x, Y) = \inf\{\, d_B(x, y) \mid y \in Y \,\}$.

We have

$$d_H(\bigcup_{i\in I} W_i, \bigcup_{i\in I} V_i) \leq \sup\{\, d_H(W_i, V_i) \mid i \in I \,\} \tag{2.3}$$

for arbitrary index set I and sequences $(W_i)_{i\in I}$, $(V_i)_{i\in I}$ in $\mathcal{P}(\mathcal{A}^\infty)$. Moreover, for $W, V \subseteq \mathcal{A}^\infty$ and $a \in \mathcal{A}$ we have

$$d_H(a \cdot W, a \cdot V) = \tfrac{1}{2} d_H(W, V) \tag{2.4}$$

provided W, V are *nonempty.*

Theorem 2.5 *Put* $\mathcal{P}_{nco}(\mathcal{A}^\infty) = \{\, W \subseteq \mathcal{A}^\infty \mid W$ *nonempty and compact* $\}$. *Then* $(\mathcal{P}_{nco}(\mathcal{A}^\infty), d_H)$ *is a complete metric space.*

Proof By Theorem 2.2 and a result due to Kuratowski ([Kur56]). □

Definition 2.6

(a) For a 1-bounded metric space (M, d) and any nonempty set X we use $X \to M$ to denote the function space of all mappings from X to M endowed with the distance d_F of uniform convergence, i.e.,

$$d_F(f, g) = \sup\{\, d(f(x), g(x)) \mid x \in X \,\}$$

for any $f, g{:}\, X \to M$.

(b) For two metric spaces $(M_1, d_1), (M_2, d_2)$ the metric space $(M_1 \times M_2, d_P)$ is defined as the Cartesian product of M_1 and M_2 endowed with the max-distance d_P, which is given by

$$d_P((x_1, x_2), (y_1, y_2)) = \max\{\, d_1(x_1, y_1), d_2(x_2, y_2) \,\}$$

for all $x_1, y_1 \in M_1$, $x_2, y_2 \in M_2$.

An elementary result states that the metric space $(X \to M, d_F)$ is complete whenever (M, d) is. A similar result holds for $(M_1 \times M_2, d_P)$. For metric spaces M_1 and M_2, we use $M_1 \xrightarrow{1} M_2$ to denote the closed subspace of $M_1 \to M_2$ consisting of all nonexpansive mappings from M_1 to M_2.

The branching model developed in Section 4 will be based on a system of domain equations. The general form for a domain equation is

$$\mathbb{P} \simeq \mathcal{F}(\mathbb{P})$$

where '$\simeq$' denotes isometry and $\mathcal{F}$ is a construction which yields a complete metric space $\mathcal{F}(\mathbb{P})$ from a given complete metric space $\mathbb{P}$. Several topological and category-theoretical techniques have been developed in the literature for obtaining solutions for such an equation for various types of $\mathcal{F}$. For our application we distinguish the nonempty-compact hyperspace construction $\mathcal{P}_{nco}(\cdot)$, disjoint sum $\cdot + \cdot$ and the prefixing construction $\mathcal{A} \times \cdot$. The hyperspace and disjoint sum operations are the familiar ones. The prefixing construction is nonstandard: For given metric space M the space $\mathcal{A} \times M$ has as its distance

$$d_{\mathcal{A}\times M}(\langle a, x\rangle, \langle a', x'\rangle) \quad = \quad \begin{cases} 1 & \text{if } a \neq a' \\ \frac{1}{2} d_M(x, x') & \text{otherwise} \end{cases} .$$

Under certain conditions (explained elsewhere, see Section 5) systems of equations involving constructions such as the ones mentioned here have solutions which are unique up to isometry. However, for the use of the solution as semantical domain only the mere existence of a solution is of interest since its metric is governed by the properties for distances enforced by the various constructions (cf. equations (4.8) and (4.9) in Section 4).

3 A linear denotational model

In this section we introduce the abstract programming language *Prog*. This simple language will serve as a vehicle for illustrating our method of developing and relating denotational models. The meaning function $\mathcal{D}_1$, also discussed in this section, is a linear interpretation of *Prog*. Its construction provides a first example of the metric approach to programming language semantics.

Definition 3.1 Fix two sets *Act* and *PVar* of atomic objects, which are called *actions* and *procedure variables,* respectively. The set *Prog* of syntactic objects called *programs* is built up from *Act* and *PVar* and the symbols $(,), ;, +$ such that

$$\begin{array}{l} \mathit{Act}, \mathit{PVar} \subseteq \mathit{Prog}, \text{ and} \\ \pi_1, \pi_2 \in \mathit{Prog} \text{ implies } (\pi_1 ; \pi_2), (\pi_1 + \pi_2) \in \mathit{Prog}. \end{array}$$

The sets *Act* and *PVar* are supposed to be taken nonempty, disjoint and may be finite or infinite. They are considered primitive for *Prog*. Typically, meta-variables a, x, π are use to range over *Act*, *PVar*, *Prog*, respectively. Let $\tau \in \mathit{Act}$ be a distinguished element in *Act* (that will be used below in the modeling of recursion).

Our actions a from Act can be thought of as abstract representations of constructs like assignments, tests, etc. that occur in concrete programming languages as `Pascal` or `C++`. Here, however, we concentrate on the flow of control, i.e. the consecutive actions that are executed and not on their effect. Therefore it suffices here to leave the precise form of the actions in Act unspecified.

A similar remark applies to $PVar$. For procedure calls x we omit details concerning procedure arguments or parameters. We fix for each procedure variable x a program $D(x)$, called its procedure body. Thus we consider to be given a map $D\colon PVar \rightarrow Prog$, allowing us to look up the program associated with a procedure variable. Execution of x amounts to execution of its associated body $D(x)$.

For notational convenience, usually outermost parentheses are dropped from programs. Also we consider the symbol ';' to bind tighter than '+'. Hence we may write, e.g., $a\,;x + b$ instead of $(a\,;x) + c$ or $((a\,;x) + b)$ where $a, b \in Act$ and $x \in PVar$.

Examples $a\,;(b + c)$, $a\,;b + a\,;c$, x, $x + a$.

The first model for $Prog$ that we present is a linear one. A subset of sequences in Act^{∞} will be assigned to each program in $Prog$. More precisely, we will define a denotational semantics $\mathcal{D}_1\colon Prog \rightarrow \mathbb{P}_1$ where the codomain $\mathbb{P}_1$ is the hyperspace of nonempty compacta of Act^{∞}. So, with each program we associate a (nonempty and compact) set of sequences.

Recall $Act^{\infty} = Act^{*} \cup Act^{\omega}$ (cf. Definition 2.2). A finite sequence from Act^{+} models a successfully terminating computation. An infinite sequence in Act^{ω} corresponds to a divergent or nonterminating computation. Divergence is inherent for a language featuring recursion (such as ours). Generally for $Prog$, due to the presence of the nondeterministic choice '+', a program π may have more than one behaviour, i.e. may perform several sequences of actions. The set $\mathcal{D}_1(\pi)$ consists of all these sequences.

The interpretation for $\mathcal{D}_1$ of the '+'-operation —which is the syntactical ingredient giving rise to the use of sets— will be based on set union for sets of sequences. So, $\mathcal{D}_1(\pi_1 + \pi_2) = \mathcal{D}_1(\pi_1) \cup \mathcal{D}_1(\pi_2)$. Typically for a linear semantics is the validity of the left-distributive law for ';' over '+'. Indeed for $\mathcal{D}_1$ we will have

$$\mathcal{D}_1(\pi_1\,;(\pi_2 + \pi_3)) = \mathcal{D}_1((\pi_1\,;\pi_2) + (\pi_1\,;\pi_3)). \tag{3.5}$$

This explains our calling $\mathcal{D}_1$ a linear semantics. In contrast, the model $\mathcal{D}_2$ that is discussed in the next section is called branching, and does not satisfy this law. (We will give a proof of equation (3.5) at the end of this section.)

We start with the definition of the semantical domain, i.e. the mathematical structure into which our programs will be mapped.

Definition 3.2 The metric spaces $\mathbb{P}_1$ and $\mathbb{Q}_1$ are given by

$$\left\{ \begin{array}{lcl} \mathbb{P}_1 & = & \mathcal{P}_{nco}(\mathbb{Q}_1) \\ \mathbb{Q}_1 & = & Act^\infty \end{array} \right.$$

The distances d_1 of $\mathbb{P}_1$ and d'_1 of $\mathbb{Q}_1$ are implicitly given. $\mathbb{Q}_1$ is assumed to be endowed with the Baire-distance (cf. Definition 2.2). Thus, by Theorem 2.3, $(\mathbb{Q}_1, d'_1)$ is a complete metric space. Also $(\mathbb{P}_1, d_1)$ is a complete metric space being the hyperspace of nonempty compacta of $\mathbb{Q}_1$ supplied with the Hausdorff-distance d_1 induced by the distance d'_1 on $\mathbb{Q}_1$ (see Theorem 2.5). Consequently we have

$$d_1(a \cdot p_1, a \cdot p_2) = \tfrac{1}{2} d_1(p_1, p_2) \tag{3.6}$$

for all $a \in Act$, $p_1, p_2 \in \mathbb{P}_1$. This is the basic fact exploited below in obtaining contractivity of various mappings. Note that equation (3.6) does not hold for the hyperspace $\mathcal{P}_{co}(\mathbb{Q}_1)$, i.e., for $\mathcal{P}_{nco}(\mathbb{Q}_1)$ with $\emptyset$ added as an isolated element. E.g., $d_1(a \cdot \emptyset, a \cdot \{b\}) = d_1(\emptyset, \{ab\}) = 1 \neq \frac{1}{2} = \frac{1}{2} \cdot 1 = \frac{1}{2} d_1(\emptyset, \{b\})$.

The denotational semantics $\mathcal{D}_1$ is a function mapping a program to a set of possible executions. For an action $a \in Act$ the behaviour is clear. The program 'a' has as its only possible execution performing 'a' and then terminating. In $\mathbb{P}_1$ this is expressed by the singleton set $\{a\}$. Hence

$$\mathcal{D}_1(a) \quad = \quad \{a\}.$$

The meaning of a procedure variable x depends on the associated procedure body. This is captured by the declaration D, i.e. the mapping $D \colon PVar \to Prog$ above. We will put

$$\mathcal{D}_1(x) \quad = \quad \tau \cdot \mathcal{D}_1(D(x)). \tag{3.7}$$

Here the symbol τ denotes a so-called silent step. In our set-up it models the call of the procedure.[§] So the possible behaviours of x, i.e. the elements of the set $\mathcal{D}_1(x)$, are the same as those for its procedure definition $D(x)$ but now all preceded by the silent step τ. Note that in equation (3.7) the

[§] Often in denotational models one prefers to avoid prefixing with τ and one has $\mathcal{D}_1(x) = \mathcal{D}_1(D(x))$. This is at the expense of restriction to so-called *guarded* recursion. For clarity of presentation this issue is completely ignored here.

argument $D(x)$ of $\mathcal{D}_1$ on the right-hand side is not syntactically simpler than the argument x of $\mathcal{D}_1$ on the left-hand side.

Finally, to the denotational semantics of the constructions $\pi_1 ; \pi_2$ and $\pi_1 + \pi_2$ the principle of compositionality applies. Thus, $\mathcal{D}_1(\pi_1 ; \pi_2) = \mathcal{D}_1(\pi_1) ; \mathcal{D}_1(\pi_2)$ and $\mathcal{D}_1(\pi_1 + \pi_2) = \mathcal{D}_1(\pi_1) + \mathcal{D}_1(\pi_2)$ where ';' and '+' occurring on the right-hand sides are functions on $\mathbb{P}_1$ still to be defined. These functions are the counterparts of the syntactical constructions ';' and '+', respectively in the semantical domain $\mathbb{P}_1$ and are therefore referred to as semantical operators. Taking all together we have the following definition of the denotational semantics $\mathcal{D}_1$.

Definition 3.3 The mapping $\mathcal{D}_1 \colon \mathit{Prog} \to \mathbb{P}_1$ is given by

$$\begin{array}{rcl} \mathcal{D}_1(a) & = & \{a\} \\ \mathcal{D}_1(x) & = & \tau \cdot \mathcal{D}_1(D(x)) \\ \mathcal{D}_1(\pi_1 ; \pi_2) & = & \mathcal{D}_1(\pi_1) ; \mathcal{D}_1(\pi_2) \\ \mathcal{D}_1(\pi_1 + \pi_2) & = & \mathcal{D}_1(\pi_1) + \mathcal{D}_1(\pi_2). \end{array}$$

The semantical operators ';' and '+' on the the right-hand side still have to be defined. This we will do next. However it should be noted that, since in the case of $\mathcal{D}_1(x)$ the syntactical complexity is not decreasing on the right-hand side the *existence* of a mapping which satisfies the above demands should be addressed (cf. Lemma 3.7).

Definition 3.4

(a) The mappings $; \colon \mathbb{P}_1 \times \mathbb{P}_1 \to \mathbb{P}_1$ and $;' \colon \mathbb{Q}_1 \times \mathbb{Q}_1 \to \mathbb{Q}_1$ are given by

$$\begin{array}{rcl} p_1 ; p_2 & = & \{ \, q_1 ;' q_2 \mid q_1 \in p_1, q_2 \in p_2 \, \} \\ a ;' q & = & a \cdot q \\ (a \cdot q') ;' q & = & a \cdot (q' ;' q). \end{array}$$

(b) The mapping $+ \colon \mathbb{P}_1 \times \mathbb{P}_1 \to \mathbb{P}_1$ is given by

$$p_1 + p_2 \;=\; p_1 \cup p_2.$$

The definition of ';' needs further comment. In the third defining clause the map ';′' occurs at both sides of the equations, but since we are also dealing with infinite sequences the argument q' at the right-hand side is not necessarily a shorter sequence than the argument $a \cdot q'$ at the left-hand side. Does there exists a pair of mappings $\langle ;, ;' \rangle$ satisfying Definition 3.4

In Lemma 3.5 we introduce a so-called higher-order transformation which will be a contraction on a complete metric space. By Banach's Fixed Point Theorem this contraction has a unique fixed point which is the only pair of mappings meeting the listed requirements.

Examples

- $ab \,;'\, cd = a \cdot (b \,;'\, cd) = a \cdot (b \cdot cd) = a \cdot (bcd) = abcd$.
- We claim $a^\omega \,;'\, cd = a^\omega$. This can be checked as follows: We have $a^\omega = a \cdot a^\omega$. Hence, $d_1'(a^\omega \,;'\, cd, a^\omega) = d_1'((a \cdot a^\omega) \,;'\, cd, a \cdot a^\omega) = d_1'(a \cdot (a^\omega \,;'\, cd), a \cdot a^\omega) = \frac{1}{2} d_1'(a^\omega \,;'\, cd, a^\omega)$. Therefore $d_1'(a^\omega \,;'\, cd, a^\omega) = 0$ and $a^\omega \,;'\, cd = a^\omega$ follows.
- $\{\, a, ab, a^\omega \,\} \,;\, \{c\} = \{\, ac, abc, a^\omega \,\}$.

Next we justify Definition 3.4 exploiting Theorem 2.1. Note the similarity of the definition of $\Omega_;(\phi,\psi)$ and $\Omega_{;'}(\phi,\psi)$ and the definition of ';' and ';'' above.

Lemma 3.5 *Put* $Op = \mathbb{P}_1 \times \mathbb{P}_1 \xrightarrow{1} \mathbb{P}_1$ *and* $Op' = \mathbb{Q}_1 \times \mathbb{Q}_1 \xrightarrow{1} \mathbb{Q}_1$. *Define* $\langle \Omega_;, \Omega_{;'} \rangle \colon Op \times Op' \to Op \times Op'$ *by* $\langle \Omega_;, \Omega_{;'} \rangle(\phi,\psi) = \langle \Omega_;(\phi,\psi), \Omega_{;'}(\phi,\psi) \rangle$ *for all* $\phi \in Op$, $\psi \in Op'$, *where*

$$\begin{array}{rcl} \Omega_;(\phi,\psi)(p_1,p_2) &=& \{\, \psi(q_1,q_2) \mid q_1 \in p_1, q_2 \in p_2 \,\} \\ \Omega_{;'}(\phi,\psi)(a,q) &=& a \cdot q \\ \Omega_{;'}(\phi,\psi)(a \cdot q', q) &=& a \cdot \psi(q',q). \end{array}$$

Then $\langle \Omega_;, \Omega_{;'} \rangle$ *is well-defined and* $\langle \Omega_;, \Omega_{;'} \rangle^2$ *is a* $\frac{1}{2}$*-contraction.*

Proof Well-definedness of $\Omega_{;'}$ is clear. For $\Omega_;$ we note that each $\psi \in Op'$ is nonexpansive, hence continuous. So $\Omega_;(\phi,\psi)(p_1,p_2)$ is the continuous image of a compactum, hence compact itself.

By a general argument is suffices to show that $\Omega_;$ is nonexpansive and $\Omega_{;'}$ is $\frac{1}{2}$-contractive to conclude that $\langle \Omega_;, \Omega_{;'} \rangle^2$ is a contraction. We have

$$\begin{array}{l} d_1(\Omega_;(\phi_1,\psi_1)(p_1,p_2), \Omega_;(\phi_2,\psi_2)(p_1,p_2)) \\ \quad = \; d_1(\{\, \psi_1(q_1,q_2) \mid q_i \in p_i \,\}, \{\, \psi_2(q_1,q_2) \mid q_i \in p_i \,\}) \\ \quad \leq \; [\text{prop. } d_1] \; \sup\{\, d_1'(\psi_1(q_1,q_2), \psi_2(q_1,q_2)) \mid q_i \in p_i \,\} \\ \quad \leq \; [\text{def. } d_F] \; d_F(\psi_1,\psi_2) \end{array}$$

Hence, we have $d_F(\Omega_;(\phi_1,\psi_1),\Omega_;(\phi_2,\psi_2)) \leq d_F(\psi_1,\psi_2) \leq d_P((\phi_1,\psi_1),(\phi_2,\psi_2))$. A case analysis for q_1 shows, along the same lines, $d_1'(\Omega_{;'}(\phi_1,\psi_1)(q_1,q_2),\Omega_{;'}(\phi_2,\psi_2)(q_1,q_2)) \leq \frac{1}{2}d_F(\psi_1,\psi_2) \leq d_P((\phi_1,\psi_1),(\phi_2,\psi_2))$. $\square$

Some variation in the set-up for the transformation $\langle \Omega_;, \Omega_{;'} \rangle$ is possible. E.g., we could have chosen to put

$$\Omega_;(\phi,\psi)(p_1,p_2) \quad = \quad \{\ \Omega_{;'}(\phi,\psi)(q_1,q_2) \mid q_1 \in p_1, q_2 \in p_2\ \}$$

and would than prove contractivity of $\langle \Omega_;, \Omega_{;'} \rangle$ itself. Although probably being a matter of taste, we feel that the presentation of Lemma 3.5 matches the most with the definitions of ';' and ';′' above.

In the justification of Definition 3.4 we have employed a higher-order transformation $\langle \Omega_;, \Omega_{;'} \rangle$. In fact we have $\langle ;, ;' \rangle = \mathit{fix}(\langle \Omega_;, \Omega_{;'} \rangle)$. The adjective higher-order refers to the fact that functions, viz. elements of $\mathit{Op} \times \mathit{Op}'$, are taken as arguments. For the justification of the definition of the denotational semantics $\mathcal{D}_1$ we use the same technique. We introduce a higher-order transformation $\Psi_1 \colon \mathit{Sem} \to \mathit{Sem}$ on a complete metric space Sem, show that it is a contraction and check $\mathcal{D}_1 = \mathit{fix}(\Psi_1)$. However, the contractivity of Ψ_1 relies on the nonexpansiveness of the semantical operators ';' and '+' as captured by the following lemma.

Lemma 3.6 *The mappings ';' and '+' on $\mathbb{P}_1$ are nonexpansive.*

Proof For '+' this is just a basic property of the Hausdorff distance. For ';' it follows from Lemma 3.5, since ';' is the first component of $\mathit{fix}(\langle \Omega_;, \Omega_{;'} \rangle)$. Hence, this function is nonexpansive by the definition of Op. $\square$

With Lemma 3.6 available we are in a position to justify the definition of the denotational semantics $\mathcal{D}_1$. The fixed point of a higher-order transformation Ψ_1 —to be given in a moment— will be the unique function from Prog to $\mathbb{P}_1$ satisfying the clauses of Definition 3.3.

Lemma 3.7 *Put $\mathit{Sem} = \mathit{Prog} \to \mathbb{P}_1$ and let $\Psi_1 \colon \mathit{Sem} \to \mathit{Sem}$ be given by*

$$\begin{array}{rcl}
\Psi_1(S)(a) & = & \{a\} \\
\Psi_1(S)(x) & = & \tau \cdot S(D(x)) \\
\Psi_1(S)(\pi_1 \,;\, \pi_2) & = & S(\pi_1) \,;\, S(\pi_2) \\
\Psi_1(S)(\pi_1 + \pi_2) & = & S(\pi_1) + S(\pi_2)
\end{array}$$

for all $S \in \mathit{Sem}$. Then Ψ_1 is well-defined and $\frac{1}{2}$-contractive.

Proof The definition of Ψ_1 is by structural induction. (Note that Ψ_1 does not occur on the right-hand side, in particular not in the case of x.) So suppose $S_1, S_2 \in Sem$. By the definition of d_F it suffices to verify

$$d_1(\Psi_1(S_1)(\pi), \Psi_2(S_1)(\pi)) \quad \leq \quad \tfrac{1}{2} d_F(S_1, S_2)$$

by structural induction on π. We exhibit the cases for x and $\pi_1 ; \pi_2$.

$$\begin{array}{ll} [x] & d_1(\Psi_1(S_1)(x), \Psi_1(S_2)(x)) \\ & \quad = \; d_1(\tau \cdot S_1(D(x)), \tau \cdot S_2(D(x))) \\ & \quad = \; [\text{prop. } (3.6)] \; \tfrac{1}{2} d_1(S_1(D(x)), S_2(D(x))) \\ & \quad \leq \; [\text{def. } d_F] \; \tfrac{1}{2} d_F(S_1, S_2) \end{array}$$

$$\begin{array}{ll} [\pi_1 ; \pi_2] & d_1(\Psi_1(S_1)(\pi_1 ; \pi_2), \Psi_1(S_2)(\pi_1 ; \pi_2)) \\ & \quad = \; d_1(S_1(\pi_1) ; S_1(\pi_2), S_2(\pi_1) ; S_2(\pi_2)) \\ & \quad \leq \; [\text{`;' nonexp.}] \; \max\{ \, d_1(S_1(\pi_1), S_2(\pi_1)), d_1(S_1(\pi_2), S_2(\pi_2)) \, \} \\ & \quad \leq \; [\text{ind. hyp. twice}] \; \tfrac{1}{2} d_F(S_1, S_2). \end{array}$$

$\square$

Note (';' and '+' having been fixed by Definition 3.6) that there is a 1–1 correspondence between mappings in $Prog \to \mathbb{P}_1$ satisfying Definition 3.3 on the one hand and fixed points of Ψ_1 on the other. Thus, since $\mathit{fix}(\Psi_1)$ is the unique fixed point, there is precisely one element of Sem, viz. $\mathit{fix}(\Psi_1)$ meeting the requirements.

Examples

- $\mathcal{D}_1(a ; (b + c)) = \{ab, ac\}$ and also $\mathcal{D}_1((a ; b) + (a ; c)) = \{ab, ac\}$.
- Suppose $D(x) = x + a$. We claim $\mathcal{D}_1(x) = \tau^+ a \cup \{\tau^\omega\}$, i.e. $\mathcal{D}_1(x) = \{\, \tau^n a \mid n \geq 1 \,\} \cup \{\tau^\omega\}$. This can be shown as follows: On the one hand, by the definition of $\mathcal{D}_1$,

  $$\begin{aligned} \mathcal{D}_1(x) &= \tau \cdot \mathcal{D}_1(x + a) = \tau \cdot (\mathcal{D}_1(x) \cup \mathcal{D}_1(a)) \\ &= \tau \cdot (\mathcal{D}_1(x) \cup \{a\}) = \tau \cdot D_1(x) \cup \{\tau a\}. \end{aligned}$$

  On the other hand,

  $$\begin{aligned} \tau^+ a \cup \{\tau^\omega\} &= \{\tau a\} \cup \{\, \tau^n a \mid n \geq 2 \,\} \cup \{\tau^\omega\} \\ &= \{\tau a\} \cup \tau \cdot (\tau^+ a \cup \{\tau^\omega\}). \end{aligned}$$

  Thus,

$$
\begin{aligned}
& d_1(\mathcal{D}_1(x), \tau^+ a \cup \{\tau^\omega\}) \\
& \quad = \; d_1(\tau \cdot \mathcal{D}_1(x) \cup \{\tau a\}, \{\tau a\} \cup \tau \cdot (\tau^+ a \cup \{\tau^\omega\})) \\
& \quad = \; [\text{prop. } d_1] \; d_1(\tau \cdot \mathcal{D}_1(x), \tau \cdot (\tau^+ a \cup \{\tau^\omega\})) \\
& \quad = \; [\text{prop. } d_1] \; \tfrac{1}{2} d_1(\mathcal{D}_1(x), \tau^+ a \cup \{\tau^\omega\}).
\end{aligned}
$$

Therefore, $d_1(\mathcal{D}_1(x), \tau^+ a \cup \{\tau^\omega\}) = 0$ and $\mathcal{D}_1(x) = \tau^+ a \cup \{\tau^\omega\}$ follows.

We still owe the reader the proof of equation (3.5) stating

$$\mathcal{D}_1(\pi_1 \,;(\pi_2 + \pi_3)) = \mathcal{D}_1((\pi_1 \,; \pi_2) + (\pi_1 \,; \pi_3)).$$

By the compositionality of $\mathcal{D}_1$ it suffices to check

$$p_1 \,; (p_2 + p_3) = (p_1 \,; p_2) + (p_1 \,; p_3)$$

for all $p_1, p_2, p_3 \in \mathbb{P}_1$. This is straightforward from the definitions:

$$
\begin{aligned}
& p_1 \,; (p_2 + p_3) \\
& \quad = \; \{\, q_1 \,; q \mid q_1 \in p_1, q \in p_2 \cup p_3 \,\} \\
& \quad = \; \{\, q_1 \,; q_2 \mid q_1 \in p_1, q_2 \in p_2 \,\} \cup \{\, q_1 \,; q_3 \mid q_1 \in p_1, q_3 \in p_3 \,\} \\
& \quad = \; (p_1 \,; p_2) + (p_1 \,; p_3).
\end{aligned}
$$

A particular instance of (3.5) gives that the programs $a\,;(b+c)$ and $(a\,;b) + (a\,;c)$ are identified by the semantics $\mathcal{D}_1$ (as claimed by the example above). In the next section we present a 'branching' model $\mathcal{D}_2$ for which we will see that the meanings of these two programs are different, i.e. $\mathcal{D}_2(a\,;(b+c)) \neq \mathcal{D}_2((a\,;b) + (a\,;c))$.

4 A branching denotational model

In this section we develop a so-called branching time model $\mathcal{D}_2$ for *Prog*. A complete metric space $\mathbb{P}_2$ of tree-like structures is used as semantical domain for $\mathcal{D}_2$. A system of 'domain equations' determines this space. Again the technique of higher-order transformations is used to define operators and the semantical mapping. Moreover, also for the comparison of the models $\mathcal{D}_1$ and $\mathcal{D}_2$ Banach's Fixed Point Theorem can be applied fruitfully.

Definition 4.1 The metric spaces $\mathbb{P}_2$ and $\mathbb{Q}_2$ are given by

$$
\begin{cases}
\mathbb{P}_2 = \mathcal{P}_{nco}(\mathbb{Q}_2) \\
\mathbb{Q}_2 = Act + Act \times (\mathbb{P}_2/2).
\end{cases}
$$

The metrics d_2 and d_2' of $\mathbb{P}_2$ and $\mathbb{Q}_2$, respectively, are implicitly given by (the functors constituting) the equations. In Section 5 references to the literature are given about solving such domain equations. As characteristic for the the distances on $\mathbb{P}_2$ and $\mathbb{Q}_2$ we have

$$d_2(p, p') = d_H(p, p') \tag{4.8}$$

(with d_H the Hausdorff-distance induced by d_2') and

$$d_2'(\langle a, p\rangle, \langle a, p'\rangle) = \tfrac{1}{2} d_2(p, p'). \tag{4.9}$$

(The factor $\frac{1}{2}$ occurring in (4.9)is due to the construct $Act \times \cdot$ in the equation for $\mathbb{Q}_2$.) The general theory of contracting functors on the category of complete metric spaces assures that a unique tuple of complete metric spaces solving the equations exists.

Example There is precisely one element p in $\mathbb{P}_2$ such that $p = \{\langle a, p\rangle\}$. Existence of p can be seen as follows: Put $p_0 = \{a\}$, $p_{n+1} = \{\langle a, p_n\rangle\}$ (The choice for p_0 is in fact immaterial). Then the sequence $(p_n)_n$ is Cauchy and $p = \lim_n p_n$ has the desired property. If both p', p'' are solutions then $d(p', p'') = d(\{\langle a, p'\rangle\}, \{\langle a, p''\rangle\} = \frac{1}{2} d(p', p'')$. So $d(p', p'') = 0$ and $p' = p''$.

The definition of the denotational semantics $\mathcal{D}_2$ follows the same pattern as for $\mathcal{D}_1$. However, new definitions for the semantical operators ';' and '+' on the new domain have to be given.

Definition 4.2 The mapping $\mathcal{D}_2$: $Prog \to \mathbb{P}_2$ is given by

$$\begin{array}{rcl} \mathcal{D}_2(a) & = & \{a\} \\ \mathcal{D}_2(x) & = & \{\langle \tau, \mathcal{D}_2(D(x))\rangle\ \} \\ \mathcal{D}_2(\pi_1 \,;\, \pi_2) & = & \mathcal{D}_2(\pi_1) \,;\, \mathcal{D}_2(\pi_2) \\ \mathcal{D}_2(\pi_1 + \pi_2) & = & \mathcal{D}_2(\pi_1) + \mathcal{D}_2(\pi_2) \end{array}$$

As before, the construction of the sequential composition ';' will need a further argument whereas the alternative composition '+' is modeled by set union.

Definition 4.3

(a) The mappings $;\colon \mathbb{P}_2 \times \mathbb{P}_2 \to \mathbb{P}_2$ and $;'\colon \mathbb{Q}_2 \times \mathbb{P}_2 \to \mathbb{Q}_2$ are given by

$$\begin{array}{rcl} p_1 \,;\, p_2 & = & \{\ q_1 \,;'\, p_2 \mid q_1 \in p_1\ \} \\ a \,;'\, p & = & \langle a, p\rangle \\ \langle a, p'\rangle \,;'\, p & = & \langle a, p' \,;\, p\rangle. \end{array}$$

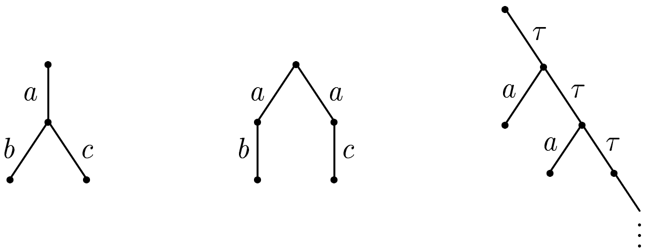
(b) The mapping $+\colon \mathbb{P}_2 \times \mathbb{P}_2 \to \mathbb{P}_2$ is given by $p_1 + p_2 = p_1 \cup p_2$.

Examples

- $\{a\} \,;\, p = \{\langle a, p\rangle\}$. If $p \in \mathbb{P}_2$ satisfies $p = \{\langle a, p\rangle\}$ then $p \,;\, p' = p$ for all $p' \in \mathbb{P}_2$.

- $\mathcal{D}_2(a \,;\, (b + c)) = \mathcal{D}_2(a) \,;\, \mathcal{D}_2(b + c) = \{a\} \,;\, (\mathcal{D}_2(b) + \mathcal{D}_2(c)) = \{a\} \,;\, (\{b\} \cup \{c\}) = \{a\} \,;\, \{b, c\} = \{\langle a, \{b, c\}\rangle\}$.
  $\mathcal{D}_2((a \,;\, b) + (a \,;\, c)) = \mathcal{D}_2(a \,;\, b) + \mathcal{D}_2(a \,;\, c) = (\mathcal{D}_2(a) \,;\, \mathcal{D}_2(b)) \cup (\mathcal{D}_2(a) \,;\, \mathcal{D}_2(c)) = (\{a\} \,;\, \{b\}) \cup (\{a\} \,;\, \{c\}) = \{\langle a, \{b\}\rangle\} \cup \{\langle a, \{c\}\rangle\} = \{\langle a, \{b\}\rangle, \langle a, \{c\}\rangle\}$.
  Note that the example shows $\mathcal{D}_2(a \,;\, (b + c)) \neq \mathcal{D}_2((a \,;\, b) + (a \,;\, c))$.

- Suppose $D(x) = x + a$, then $\mathcal{D}_2(x) = p$ where p is the (unique) element of $\mathbb{P}_2$ satisfying $p = \{\langle \tau, p \cup \{a\}\rangle\}$.



$a \,;\, (b + c)$ $\qquad$ $(a \,;\, b) + (a \,;\, c)$ $\qquad$ x where $D(x) = x + a$

As for ';' of Section 3 the definition of the semantical operator ';' here needs further justification. The argument is a little more involved now, due to the interdependence of the metric spaces involved. However, the technique of higher-order transformations can be applied fruitfully again.

Lemma 4.4 *Let* $Op = \mathbb{P}_2 \times \mathbb{P}_2 \xrightarrow{1} \mathbb{P}_2$ *and* $Op' = \mathbb{Q}_2 \times \mathbb{P}_2 \xrightarrow{1} \mathbb{Q}_2$*. Define the mapping* $\langle \Omega_{;}, \Omega_{;'} \rangle \colon Op \times Op' \to Op \times Op'$ *by* $\langle \Omega_{;}, \Omega_{;'} \rangle(\phi, \psi) = \langle \Omega_{;}(\phi, \psi), \Omega_{;'}(\phi, \psi)\rangle$ *for* $\phi \in Op$, $\psi \in Op'$, *where*

$$\begin{array}{rcl} \Omega_{;}(\phi, \psi)(p_1, p_2) & = & \{\, \psi(q_1, p_2) \mid q_1 \in p_1 \,\} \\ \Omega_{;'}(\phi, \psi)(a, p) & = & \langle a, p\rangle \\ \Omega_{;'}(\phi, \psi)(\langle a, p'\rangle, p) & = & \langle a, \phi(p, p')\rangle. \end{array}$$

Then $\langle \Omega_{;}, \Omega_{;'} \rangle$ *is well-defined and* $\langle \Omega_{;}, \Omega_{;'} \rangle^2$ *is contractive.*

Proof Similar to the proof of Lemma 3.5 using properties (4.8) and (4.9).
□

From Lemma 4.4 we obtain, as in Section 3, $\langle ;, ;' \rangle = \mathit{fix}(\Omega_{;}, \Omega_{;'})$). As a direct consequence of this, and of the definition of '+' on $\mathbb{P}_2$ as set union we have the following properties.

Lemma 4.5 *The mappings ';' and '+' on $\mathbb{P}_2$ are nonexpansive.*

Next we justify the definition of the denotational semantics $\mathcal{D}_2$ in Definition 4.2 by means of a higher-order transformation Ψ_2. A 1–1 correspondence between fixed points of Ψ_2 and mappings satisfying Definition 4.2 establishes the existence and uniqueness of the model $\mathcal{D}_2$.

Lemma 4.6 *Put $\mathit{Sem} = \mathit{Prog} \to \mathbb{P}_2$ and let $\Psi_2 \colon \mathit{Sem} \to \mathit{Sem}$ be given by*

$$\begin{array}{rcl} \Psi_2(S)(a) & = & \{a\} \\ \Psi_2(S)(x) & = & \{\langle \tau, S(D(x)) \rangle\} \\ \Psi_2(S)(\pi_1 \,;\, \pi_2) & = & S(\pi_1) \,;\, S(\pi_2) \\ \Psi_2(S)(\pi_1 + \pi_2) & = & S(\pi_1) + S(\pi_2) \end{array}$$

for all $S \in \mathit{Sem}$. Then Ψ_2 is well-defined and $\frac{1}{2}$-contractive. Moreover, $\mathit{fix}(\Psi_2)$ is the unique element of Sem satisfying Definition 4.2.

Proof Similar to the proof of Lemma 3.7 exploiting Lemma 4.5. $\square$

The natural question about the relationship between the models $\mathcal{D}_1$ of Section 3 and $\mathcal{D}_2$ of this section will be addressed next. Once again uniqueness of the fixed point of a contraction is exploited for the comparison of $\mathcal{D}_1$ and $\mathit{abs} \circ \mathcal{D}_2$ with abs a suitable abstraction function mediating between the branching domain $\mathbb{P}_2$ and the linear domain $\mathbb{P}_1$. First we will give a definition of this function.

Definition 4.7 The mappings $\mathit{abs} \colon \mathbb{P}_2 \to \mathbb{P}_1$ and $\mathit{abs}' \colon \mathbb{Q}_2 \to \mathbb{P}_1$ are given by

$$\begin{array}{rcl} \mathit{abs}(p) & = & \bigcup \{ \mathit{abs}'(q) \mid q \in p \} \\ \mathit{abs}'(a) & = & \{a\} \\ \mathit{abs}'(\langle a, p \rangle) & = & a \cdot \mathit{abs}(p). \end{array}$$

Note that, for $a \in \mathit{Act}$, $p \in \mathbb{P}_2$, we have $\mathit{abs}(\{\langle a, p \rangle\}) = a \cdot \mathit{abs}(p)$. A similar construction as in Lemma 3.7 of a higher-order transformation $\langle \Omega_{\mathit{abs}}, \Omega_{\mathit{abs}'} \rangle$ for $\mathit{Op} \times \mathit{Op}'$ to $\mathit{Op} \times \mathit{Op}'$ where $\mathit{Op} = \mathbb{P}_2 \to \mathbb{P}_1$ and $\mathit{Op}' = \mathbb{Q}_2 \to \mathbb{P}_1$ may be used to justify the above definition of abs and abs'. Again, the unique fixed point of the contraction $\langle \Omega_{\mathit{abs}}, \Omega_{\mathit{abs}'} \rangle$ will be the only pair of functions satisfying Definition 4.7.

Next we claim that $\mathcal{D}_1 = abs \circ \mathcal{D}_2$. For the verification of this claim it suffices to show that $abs \circ \mathcal{D}_2$ is a fixed point of the contraction Ψ_1 of Lemma 3.7. By Banach's Fixed Point Theorem the result then follows. In the proof of Theorem 4.9 we need the following result.

Lemma 4.8 *For all $p_1, p_2 \in \mathbb{P}_2$ it holds that:*

(a) $abs(p_1 ;_2 p_2) = abs(p_1) ;_1 abs(p_2)$ where '$;_2$' is '$;$' given by Definition 4.3 and '$;_1$' is '$;$' given by Definition 3.4, respectively.

(b) $abs(p_1 +_2 p_2) = abs(p_1) +_1 abs(p_2)$ where '$+_2$' is '$+$' given by Definition 4.3 and '$+_1$' is '$+$' given by Definition 3.4, respectively.

Proof Put $\varepsilon = \sup\{ d_1(abs'(q ;_2' p'), \bigcup\{ abs'(q) ;_1 abs'(q') \mid q' \in p' \}) \}$. By case analysis for q one verifies

$$d_1(abs'(q ;_2' p'), \bigcup\{ abs'(q) ;_1 abs'(q') \mid q' \in p' \}) \quad \leq \quad \tfrac{1}{2}\varepsilon.$$

So, $\varepsilon \leq \frac{1}{2}\varepsilon$, $\varepsilon = 0$ and $abs'(q ;_2' p') = \bigcup\{ abs'(q) ;_1 abs'(q') \mid q' \in p' \}$. Using this, part (a) follows directly. Part (b) is straightforward from the definitions. □

Theorem 4.9 $\mathcal{D}_1 = abs \circ \mathcal{D}_2$.

Proof It suffices to show, by structural induction,

$$\Psi_1(abs \circ \mathcal{D}_2)(\pi) \quad = \quad (abs \circ \mathcal{D}_2)(\pi)$$

for $\pi \in Prog$. We only treat the two cases x and $\pi_1 ; \pi_2$.

$[x]$
$$\begin{aligned} & \Psi_1(abs \circ \mathcal{D}_2)(x) \\ = \ & \tau \cdot (abs(\mathcal{D}_2(D(x)))) \\ = \ & abs(\{\langle \tau, \mathcal{D}_2(D(x)) \rangle\}) \\ = \ & abs(\mathcal{D}_2(x)) \\ = \ & (abs \circ \mathcal{D}_2)(x). \end{aligned}$$

$[\pi_1 ; \pi_2]$
$$\begin{aligned} & \Psi_1(abs \circ \mathcal{D}_2)(\pi_1 ; \pi_2) \\ = \ & (abs \circ \mathcal{D}_2)(\pi_1) ;_1 (abs \circ \mathcal{D}_2)(\pi_2) \\ = \ & abs(\mathcal{D}_2(\pi_1)) ;_1 abs(\mathcal{D}_2(\pi_2)) \\ = \ & [\text{lemma 4.8}]\ abs(\mathcal{D}_2(\pi_1) ;_2 \mathcal{D}_2(\pi_2)) \\ = \ & abs(\mathcal{D}_2(\pi_1 ; \pi_2)) \\ = \ & (abs \circ \mathcal{D})(\pi_1 ; \pi_2). \end{aligned}$$
□

5 Concluding remarks

Above we have illustrated the use of Banach's Fixed Point Theorem in the metric approach to programming languages semantics. Two denotational models for an abstract programming language with nondeterministic choice and recursion have been developed and related exploiting this classical result. Also for the definition of semantical domains and semantical operators fixed point constructions play a dominant role.

In semantics, the exploitation of unique fixed points of higher-order contractions on complete metric spaces has been introduced by Kok and Rutten in the important paper [KR90]. Prior, in [HP79], Hennessy and Plotkin presented an order-theoretic variant of the proof principle based on the Knaster-Tarski theorem ([Kna28, Tar55]). (The latter is the order-theoretic analogue of Banach's theorem, stating that each monotonic endofunction on a complete lattice has a *least* fixed point.)

The principles of denotational semantics were developed by Scott and Strachey from 1969 onwards. In order to deal with recursive constructs, one has to use domains which allow the definition of fixed points. Various mathematical structures have been used for this purpose. During the first decade, ordered sets were used primarily. In the late seventies, the use of *metric spaces* in semantics was proposed by Arnold and Nivat (for an introduction, see [ANN85]). The solving of domain equations over metric spaces was first studied by De Bakker and Zucker ([BZ82]). A generalization by America and Rutten in a categorical setting is published in [AR89b].

In a series of subsequent papers, De Bakker and his coworkers have designed metric semantic models for a large number of programming notions. In several theses, the methodology has been applied to a variety of language paradigms and associated foundational problems (America and Rutten [AR89a], Kok [Kok89], Eliëns [Eli92], Horita [Hor93], and Van Breugel [Bre94]). Metric denotational semantics has also been studied, e.g., by the Programming Research Group of Oxford University (see, for example, [RR88]) and by Majster-Cederbaum et al. (e.g., [MZ91]).

The monograph [BV96] presents a comprehensive account of the techniques for developing the semantics of programming notions using metric topology as covered here. The book [BR92] contains a selection of the papers from a decade of work on these topics. [Bre94] provides a concise treatment of the use of topological (in particular metric) models in semantics, including several more advanced investigations not dealt with in [BV96]. The introductory paper [BV97] discusses the metric treatment of so-called operational semantics and their comparison with denotational ones.

References

[ANN85] A. Arnold, P. Naudin, and M. Nivat. On semantics of nondeterministic recursive program schemes. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 1–33. Cambridge University Press, 1985.

[AR89a] P. America and J.J.M.M. Rutten. *A Parallel Object-Oriented Language: design and semantic foundations.* PhD theses, Vrije Universiteit Amsterdam, 1989.

[AR89b] P. America and J.J.M.M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *Journal of Computer and System Sciences*, 39:343–375, 1989.

[Bai09] R. Baire. Sur la représentation des fonctions discontinues. *Acta Mathematica*, 32:97–176, 1909.

[Ban22] S. Banach. Sur les opérations dans les ensembles abstraits et leurs applications aux equations intégrales. *Fundamenta Mathematicae*, 3:133–181, 1922.

[BR92] J.W. de Bakker and J.J.M.M. Rutten, editors. *Ten Years of Concurrency Semantics, selected papers of the Amsterdam Concurrency Group.* World Scientific, 1992.

[Bre94] F.C. van Breugel. *Topological Models in Comparative Semantics.* PhD thesis, Vrije Universiteit Amsterdam, 1994.

[BV96] J.W. de Bakker and E.P. de Vink. *Control Flow Semantics.* The MIT Press, 1996.

[BV97] J.W. de Bakker and E.P. de Vink. A metric approach to control flow semantics. In *Proc. 11th Summer Conference on General Topology and Applications.* Annals of the New-York Academy of Sciences, 1997. To appear.

[BZ82] J.W. de Bakker and J.I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54:70–120, 1982.

[Eli92] A. Eliëns. *DLP: A Language for Distributed Logic Programming: Design, Semantics and Implementation.* Wiley Series in Parallel Computing, 1992.

[Hau14] F. Hausdorff. *Grundzüge der Mengenlehre.* Leipzig, 1914.

[Hor93] E. Horita. *Fully Abstract Models for Concurrent Languages.* PhD thesis, Vrije Universiteit Amsterdam, 1993.

[HP79] M. Hennessy and G.D. Plotkin. Full abstraction for a simple parallel programming language. In J. Bečvář, editor, *Proc. of 8th Symposium on Mathematical Foundations of Computer Science*, pages 108–120. LNCS 74, Springer, 1979.

[Kna28] B. Knaster. Un théorème sur les fonctions d'ensembles. *Annales Societatis Mathematicae Polonae*, 6:133–134, 1928.

[Kok89] J.N. Kok. *Semantic Models for Parallel Computation in Data Flow, Logic- and Object-Oriented Programming.* PhD thesis, Vrije Universiteit Amsterdam, 1989.

[KR90] J.N. Kok and J.J.M.M. Rutten. Contractions in comparing concurrency semantics. *Theoretical Computer Science*, 76:179–222, 1990.

[Kur56] K. Kuratowski. Sur une méthode de métrisation complète des certains espaces d'ensembles compacts. *Fundamenta Mathematicae*, 43:114–138, 1956.

[MZ91] M.E. Majster-Cederbaum and F. Zetzsche. Towards a foundation for semantics in complete metric spaces. *Information and Computation*, 90:217–243, 1991.

[RR88] G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.