Towards a uniform topological treatment of streams and functions on streams

J.W. de Bakker*
J.N. Kok**

Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB AMSTERDAM, The Netherlands

ABSTRACT

We study the semantics of functional languages on streams such as Turner’s SASL or KRC. The basis of these languages is recursive equations for (functions on) finite or infinite sequences. The paper presents a start towards a mathematical (denotational) description of such languages using tools from metric topology. The description is based on the Banach fixed point theorem and a restricted version of a typed lambda calculus. To a system of recursive stream (function) declarations a system of functions is associated in an appropriate topological domain. These functions have to be contracting in certain arguments and non distance increasing in others; a syntax is designed which ensures the right interplay between these conditions. Nondeterminism is handled by considering compact sets of streams, and preservation of compactness is another important technical issue. Not all concepts in a language such as KRC are covered, and some indications on possible extensions of the framework are provided.

1. Introduction

We present a semantic study of languages with streams and functions on streams as exemplified by Turner’s languages SASL and KRC. The tools we use are from metric topology; ultimately, our model relies on Banach’s fixed point theorem for contracting functions on a complete metric space.

A stream is a finite or infinite sequence of values from a set V. (The examples below always take for V the set of integers.) A program is a set of recursive declarations of streams and stream functions, together with an expression to be evaluated with respect to the declarations.

Example: \( v \leftarrow \cdot v \cdot (\text{head}(z)+1) \cdot v \cdot (\text{tail}(z)) \mid \cdot v \cdot \), we see two declarations, viz. that of the stream \( v \) and of the stream function \( v_i \). The expression to be evaluated is \( v \) itself. In \( v_i \), the formal \( z \) is used which can have a stream (or, in general, a set of streams) as actual. `-` denotes concatenation, and `\mid` separates the declarations from the main program expression. The functions head and tail are as usual. The intended meaning of \( v \) is the infinite sequence 1-2-3-….

Our main task is the development of a semantic framework to assign meaning to declarations of streams and stream functions. First, we define various metrics. The distance \( d \) between two streams is smaller if the elements where they exhibit their first difference occurs further to the right in the streams. For example, \( d(23,24) = \frac{1}{2} \), \( d(123,124) = \frac{1}{4} \). By standard topological methods, we extend this metric to sets of streams and to stream functions. Section 2, on topological preliminaries, collects these definitions. Also, the important notions of contracting, non distance increasing, and continuous functions are introduced, and various properties of compactness needed below are described. In fact, compactness, as a limit case of finiteness, is the topological counterpart of the familiar notion of bounded nondeterminism present in various order theoretic approaches.

Section 3 presents the definitions of the syntax and the semantics of our language. The syntax is designed in such a way that the associated semantic functions have the right contracting c.q. non distance increasing properties, as developed in section 4. Two main themes arise here: in a declaration such as \( (v \leftarrow \cdots v \cdots) \), recursive occurrences of \( v \) have to be guarded by some expression (e.g. \( v \leftarrow \cdots z v \cdots \)) in order to ensure contractivity. Moreover, in order to guarantee that such contractivity is preserved throughout, stream functions of the type \( (v_i \leftarrow \var z : \cdots z \cdots) \) have to be non distance increasing in \( z \). Appropriate syntactic categories are introduced in order to enforce the right combination of these properties. The format of the syntax follows the usual pattern of a typed lambda calculus, restricted, however, to ground types and first order functional types. We envisage no problems in generalizing this aspect of the syntax.

* Supported in part by ESPRIT project 415: Parallel Architectures and Languages
** Supported by the Netherlands Organisation for the Advancement of Pure Research (Z.W.O.), grant 125-20-04.
The main theorem of the paper is in section 4 where the basic contractivity result, for functions associated with a set of declarations, is established. The intended meaning of the declared streams or stream functions as unique fixed points by Banach’s theorem is then immediate. A minor issue to be faced is the possibility that a “guarding” term \( \mathfrak{s} \) in \( s.v \) has the empty word in the set \( \{ x \} \) denoting its meaning.

Section 5 treats a program as a pair consisting of a set of declarations and an expression. In the latter, we can be more liberal as to the allowed functions occurring in it, since it has only calls (and no declarations) of recursive objects.

Section 6 discusses some limitations and possible extensions. First we discuss functions which, instead of being contracting or non distance increasing, allow a bounded increase in distance. The problems which arise in this case are related to those studied by Wadge [22]. Secondly, with a more refined syntax we can also cater for the use of external (i.e. not programmer declared) functions which are not required to be contracting or non distance increasing. However, we then must restrict the way in which such functions occur in our expressions. Thirdly, we mention an important case of a function declaration which is allowed in KRC but does not fit into the present framework (a “permutation” function). We expect that the use of Painlevé limits (rather than of Cauchy sequences of compact sets with respect to the Hausdorff distance) will be useful here, but we have not worked out this idea.

Functional programming in general, and programming on streams in particular, have received wide attention in recent years. For the general background we refer, e.g., to [9] and references contained therein. It will be clear from the above that the languages SASL and KRC [20,21] have been a source of inspiration to us. Further references concerned with programming on streams are [8,10,12,15,23].

Our use of topological techniques goes back to the work of Nivat and his coworkers (e.g. [1,16]). Many of the technical results we use below are also described or applied in [4,5].

An order-theoretic approach to stream semantics is also possible; a basic reference is Broy [6], see also [7] for a more introductory presentation. Advantages of the metric approach are that certain distinctions can be made, in particular between contracting, non distance increasing and (only) continuous functions, which have no direct order-theoretic counterpart. Also, contractivity leads to the attractive situation that uniqueness of fixed points is ensured. However, further work is needed to cope with the problem of unguarded recursion in a topological setting. For connections between metric, order-theoretic, algebraic approaches in general see [17,18,19].

2. Topological preliminaries

Let \( M, M_1, M_2 \) be metric spaces with (bounded) distances \( d, d_1, d_2 \). A function \( \phi : M_1 \rightarrow M_2 \) is called continuous whenever for each sequence \( \{ x_i \} \) in \( M_1 \) and \( x \in M_1 \) such that \( x = \lim_{i \rightarrow \infty} x_i \), we have \( \phi(x) = \lim_{i \rightarrow \infty} \phi(x_i) \). \( \phi \) is called contractive whenever, for each \( x, y \in M_1 \), \( d(\phi(x), \phi(y)) < c \cdot d(x, y) \) for some constant \( c \) with \( 0 < c < 1 \), and \( \phi \) is called non distance increasing whenever for each \( x, y \in M_1 \), \( d(\phi(x), \phi(y)) \leq d(x, y) \).

Given a metric space \((M, d)\), \( d \) is said to be an ultrametric on \( M \) if it satisfies the ‘strong triangle inequality’: for all \( x, y, z \in M \), \( d(x, z) \leq \max(d(x, y), d(y, z)) \).

Let \((M, d)\) be a complete metric space. For \( X, Y \subseteq M \) we define the so called Hausdorff distance
\[
H(X, Y) = \max(\sup_{x \in X} d'(x, Y), \sup_{y \in Y} d'(y, X))
\]
with \( d'(x, Y) = \inf_{y \in Y} d(x, y) \). By convention \( \inf \emptyset = 1 \) and \( \sup \emptyset = 0 \). We now define some spaces obtained from other spaces.

Let \( P_{\text{comp}}(M) \) denote the non empty compact subsets of \( M \), let \([M_1 \rightarrow M_2]\) denote the non distance increasing functions from \( M_1 \rightarrow M_2 \), and let \( M_1 \times \cdots \times M_n \) be the Cartesian product of \( M_1, \ldots, M_n \). We give these spaces the following metrics:

- \((P_{\text{comp}}(M), d)\) : Hausdorff metric induced by the metric on \( M \),
- \([M_1 \rightarrow M_2] \): \( \alpha(\phi_1, \phi_2) = \sup_{x \in M_1} d_2(\phi_1(x), \phi_2(x)) \),
- \((M_1 \times \cdots \times M_n, d)\) : \( d(\langle x_1, \ldots, x_n \rangle, \langle y_1, \ldots, y_n \rangle) = \max_{i \in \{1, \ldots, n\}} d_i(x_i, y_i) \).

2.1. Theorem. If \( M, M_1, \ldots, M_n \) are complete metric spaces then the following spaces with the above defined metrics are also complete: \( P_{\text{comp}}(M) \), \([M_1 \rightarrow M_2]\), \( M_1 \times \cdots \times M_n \).

(i) If \( \{ X_i \} \) is a Cauchy sequence of compact sets in \((P_{\text{comp}}(M), d)\) then there exists a limit and this limit is compact. For details see [4].
(ii) Let \( \{ \phi_i \} \) be a Cauchy sequence in \([M_1 \rightarrow M_2]\). Define \( \phi' : M_1 \rightarrow M_2 \) by \( \phi'(X) = \lim \phi_i(X) \). Then we have \( \lim \phi_i = \phi' \) and \( \phi' \in [M_1 \rightarrow M_2] \).
(iii) omitted \( \square \).