Annales Univ. Sci. Budapest., Sect. Comp. **9** (1988) 105-114

A MATHEMATICAL APPROACH TO PROGRAMMING

ÁKOS FÓTHI

Abstract. In this paper we deal with a mathematical foundation of programming, providing a tool for teachers programming to deliver "scientific knowledge" rather than "the skills of the craft". This goal is hoped to be achieved by using mathematical tools, however, unlike other authors /e.g. Gries [3]/, we shall not follow the standard formalization of mathematical logics. We think that education of programming should be separated from the research of programming related to computer manipulations such as automatic program generation.

1. Introduction

We shall use a mathematical approach which is characteristic to the classical branches of mathematics such as mathematical analysis or probability calculus. First of all Mills' work [7] can be considered as a starting point to this approach but, similarly to [1,4,8,9] we use relations instead of functions, this way we can deal with nondeterministic and parallel programs. Our purpose is to set up a model that provides an unified mathematical treatment of the most important fundamental concepts and knowledge of programming. Talking of fundamental knowledge we have in mind first of all Dijkstra's [2] and Jackson's [6] works as well as the type, parallel programs and program transformation concepts.

This paper is devoted to some fundamental notions of the mathematical model of programming. The two most important

concepts are the problem and the program. Their abstract definitions are given in terms of state space. For the definition of solution /when a program is said to solve a problem/ establishing the main connection between the problem and program we introduce the notion of program function and study the equivalence of programs.

2. Preliminary definitions and notions

Let $\mathbf{N}$ denote the set of all natural numbers, $\mathbf{N}_0$ the set of all nonnegative integers, Z the set of all integers, $\mathbf{L}$ the set of logical values. 0 denotes the empty set.

$\alpha ::=< \alpha_1, \ldots >$, $\alpha_i \in A$ denotes a finite or infinite sequence of elements of A.

$\alpha ::=< \alpha_1, \ldots, \alpha_n >, \alpha_i \in A$ denotes a finite sequence of length n of elements of A. $\mid \alpha \mid$ stands for the length of such a finite sequence. We shall omit the symbol $\alpha_i \in A$ whenever it is clear from the context.

Denote by A^* the set of all finite and by A^∞ the set off all infinite sequences of elements of A. Put $A^{**} ::= A^\infty \cup A^*$.

The reduced sequence $\mathrm{red}(\alpha)$ corresponding to an $\alpha \in A^{**}$, is obtained by replacing each stationary subsequence by one of its single element.

Let $I ::= \{i_1, \ldots, i_m\} \subset \mathbf{N}$. The direct product of the sets $A_k, k \in I$ is defined as

$$X_{k \in I} A_k ::= \{(a_1, \ldots a_m) \mid \quad \forall j \in [1, m] : a_j \in A_{i_j}\}.$$

If $A = X_{k \in I} \quad A_k, B = X_{k \in J} A_k$ and $J \subset I \subset \mathbf{N}$, then B is called a subspace of A.

Let $A ::= X_{k \in I} H_k$ and $B ::= X_{k \in J} A_k$ where the sets $A_k (k \in I)$ are arbitrary, $I ::= \{i_1, \ldots, i_m\}, J ::= \{j_1, \ldots, j_n\}$ and $J \subset I \subset \mathbf{N}$. In this case $pr_B(a) ::= (a_{j_1}, \ldots, a_{j_n})$ denotes the projection of $a \in A$ into the subspace B.

If $C \subset A$ then

$$pr_B(C) ::= \{b \in B \mid \exists a \in C : b = pr_B(a)\},$$

if $\alpha ::= < \alpha_1, \ldots >, \alpha_1 \in A$ then
$pr_B(\alpha) ::= (pr_B(\alpha_1), \ldots)$,
if $H \subset A^{**}$ then

$$pr_B(H) ::= \{\beta \in B^{**} \mid \beta = pr_B(\alpha) \wedge \alpha \in H\}.$$

Any subset of any direct product is called a relation. Further we deal with relations which are subsets of the direct product of two sets /binary relations/.

Let $R \subset A \times B$ where A and B are arbitrary sets. The domain of the relation R is defined by

$$\mathcal{D}_R ::= \{a \in A \mid \exists b \in B : (a, b) \in R\}.$$

The range of R is

$$\mathcal{R}_R ::= \{b \in B \mid \exists a \in A : (a, b) \in R\}.$$

The range of R at $a \in A$ is

$$R(a) ::= \{b \in B \mid (a, b) \in R\}.$$

A relation R is called function, if $\forall a \in A : R(a)$ has at most one element. If $\mathcal{D}_R \neq A$ then R is a partial function.

3. The fundamental objects of the model

The concept of the state space has already been used in several senses. For Mills the state space is a model of a von Neumann type computer. Others, e.g. Dijkstra, associate this notion with the problem to be solved and, in this way, the elements of the state space are the possible states of the characteristics of the problem.

So the program is "outside" of the state space operating on it [2,9,10].

In this paper the notation of the state space is used in the second meaning.

DEFINITION D1. /State space/ Let $A_1, \ldots, A_n$ be an arbitrary finite or numerable sets. The set $A ::= A_1 \times A_2 \times \ldots \times A_n$ is called the state space. The components of the state space, i.e. the sets A_1 are called /type/ value sets. This name refers to the fact that each component of the state space is the range of a type /"representing function"/.

We usually have to handle the components of the state space separately. To this end we use variables.

DEFINITION D2. /Variable/ The projections $v_i : A \to A_i$ of the space $A = A_1 \times \ldots \times A_n$ are called variables.

The notion of the state space makes it possible to define the concept of a problem independently of any program.

DEFINITION D3. /Problem/ Any relation $F \subset A \times A$ is called a problem.

A program is defined in terms of sequences; the working of the program is characterized by a series of the elements of the state space [1].

DEFINITION D5. /Program/ A relation $S \subset A \times A^{**}$ is called a program in the state space if

$i/$ $\quad D_S = A,$

$ii/$ $\quad \alpha \in \mathcal{R}_S \Rightarrow \alpha = \mathrm{red}(\alpha),$

$iii/$ $\quad (a \in A \wedge \alpha \in S(a)) \Rightarrow \alpha_1 = a.$

Example

Let us consider the following program in the usual sense:
parbegin a := true; b := false; || a := false; parend.
Now, the state space has two components, each of them is the set of logical values. For example, the sequences, associated with the

"point" /true, true/ of the state space are

$$\alpha =< (true,true),(true,false),(false,false) >$$
$$\beta =< (true,true),(false,true),(true,true),(true,false).$$

The second part (ii) of the definition means that the program during its normal functioning always gets into a new state. No change of the state means an abnormal functioning of the program or, in other words, getting out of the system.

We introduce the notion of program function in order to express the result of the functioning.

Marks

Let $\tau : A^* \rightarrow A$ be a function which, with each sequence associates its last element. $\tau(\alpha) ::= \alpha_{|\alpha|}$.

DEFINITION D5. /Program function/ The program function of a program S is a relation $p(S) \subset A \times A$ defined as follows:

$$\mathcal{D}_P(S) ::= \{a \in A \mid S(a) \subset A^*\},$$
$$\forall a \in \mathcal{D}_P(S) :$$
$$p(S)(a) ::= \{b \in A \mid \exists \alpha \in S(a) : \tau(\alpha) = b\}.$$

We also notice that there are other options to define the domain of the program function taking a larger one

$$\{a \in A \mid S(a) \cap A^* \neq 0\}$$

or a smaller one

$$\{a \in A \mid S(a) \subset A^* \wedge \mid S(a) \mid < \infty\}$$

In the first case, the program function is called a liberal program function [2], in the second case a bounded one [4].

We use the notion of solution instead of that of program correctness because the problem is defined independently of the program. The concept of solution is similar to one used by Hoare [5] for representation of types.

DEFINITION D6. /Solution/ The program S is said to solve the problem F if

$$i/ \qquad \mathcal{D}_F \subset \mathcal{D}_P(S),$$
$$ii/ \qquad \forall a \in \mathcal{D}_F : p(S)(a) \subset F(a).$$

Thus it may seem reasonable to consider two programs equivalent, if their program functions are identical. The question of equivalence, however can be handled in a finer way using the above definition of the state space.

DEFINITION D7. /Extension of a problem/ Let the state space B be a subspace of the state space A. The relation $F' \subset A \times A$ is called the extension of the problem $F \subset B \times B$, if

$$F' ::= \{(x, y) \in A \times A \mid (pr_B(x), pr_B(y)) \in F\}.$$

In other words, the extension of a problem means that new variables are introduced without any restriction on them.

DEFINITION D8. /Extension of a program/ Let the state space B be a subspace of the state space A, namely $A ::= X_{i \in I} A_i$, $B ::= X_{i \in J} A_i$ and $J \subset I \subset \mathbf{N}$. Denote by K the set $I - J$, let $B' ::= X_{i \in K} A_i$ and let S be a program on the state space B. The program $S' \subset A \times A$ is called the extension of the program S onto the state space A if

$$\forall a \in A : S'(a) ::= \{\alpha \in A^{**} \mid pr_B(\alpha) \in S(pr_B(a)) \wedge \forall i \in \mathcal{D}_a : pr_{B'}(\alpha_i) = pr_{B'}(a)\}.$$

The extension of a program defined on a subspace gives rise to a program which operates on the subspace in the same way as the original program does and it does not change the rest of the components of the state space.

In definitions D7 and D8, we take one of the possible extensions, the trivial one.

DEFINITION D9. /Equivalence of programs/ Two programs /their state spaces may be different/ are called equivalent on a common subspace B, if the projections of their program functions onto B coincide.

The extensions of problems and programs make it possible for us to understand two practical things theoretically. One of them is, that during program writing, we often need to introduce new variables, that is extend the state space. In this case the program will not solve the original problem, but its extension. The other one is it in order, to solve a subproblem we use a procedure, a program defined on a state space has common components with the state space of the problem. In this case the program /possibly the problem too/ has to be extended.

The following theorems are related to these topics.

Theorem 1. *Let A be a state space, B a subspace of A. Let $F \subset B \times B$ be a problem, $S \subset B \times B^{**}$ a program, F' and S' the respective extensions of F and S onto the state space A. Then S solves F if only if S' solves F'.*

Proof.

1/ Suppose that S solves F that is, $\mathcal{D}_F \subset \mathcal{D}_P(S)$ and $\forall b \in \mathcal{D}_F$: $p(S)(b) \subset F(b)$. (D7) implies $\mathcal{D}_{F'} = pr_B^{-1}(\mathcal{D}_F)$. By (D8), $S'(a) \subset A^*$ if only if $S(pr_B(a)) \subset B^*$, that is, $\mathcal{D}_P(S') = pr_B^{-1}(\mathcal{D}_P(S))$. Consequently $\mathcal{D}_{F'} \subset \mathcal{D}_P(S')$. Also $\forall a \in \mathcal{D}_{F'} : F'(a) = pr_B^{-1}(F(pr_B(a)))$ and $p(S')(a) \subset pr_B^{-1}p(S)(pr_B(a))$. Since S solves F consequently $p(S)(pr_B(a)) \subset F(pr_B(a))$ and so $p(S')(a) \subset F'(a)$, that is, S' solves F'.

2/ Let S' solve F', that is, $\mathcal{D}_{F'} \subset \mathcal{D}_P(S')$ and $\forall a \in \mathcal{D}_F : p(S')(a) \subset F'(a)$ and by (D7) and (D8), we have $F = pr_B(F')$ and $S = pr_B(S')$. The projection do not change the length of a series, so $p(S) = pr_B(p(S'))$. Therefore $\mathcal{D}_F \subset \mathcal{D}_P(S)$ and $\forall b \in B : p(S)(b) \subset F(b)$, which means that S solves F. □

Theorem 2.*Let A be a state space and B be a subspace of*

A. Let $G \subset A \times A$ be a problem, $S \subset B \times B^{**}$ a program and S' the extension of S onto A. In this case, if S' solves G then S solves $H = pr_B(G)$.

Proof. Let S' solve G, that is, $\mathcal{D}_G \subset \mathcal{D}_P(S')and\forall a \in \mathcal{D}_G$: $p(S')(a) \subset G(a)$. Since $p(S) = pr_B(p(S'))$, $\mathcal{D}_H \subset \mathcal{D}_P(S)$ and$\forall b \in \mathcal{D}_H$: $p(S)(b) \subset H(b)$, we have that S solves H. □

The inverse of this theorem does not hold, although it would be convenient to apply it to a procedure. The next theorem gives the condition of invertibility of the previous theorem.

Theorem 3. *Suppose that $A = X_{i \in I} A_i, B = X_{j \in J} A_j, J \subset I \subset \mathbf{N}$ and $K = I - J$, $B' = X_{k \in K} A_k$. Let $G \subset A \times A$ be a problem, $S \subset B \times B^{**}$ a program and S' be the extension of S onto A. In this case, if S solves $H = pr_B(G)$,$\forall x, y \in \mathcal{D}_G$: if $pr_B(x) = pr_B(y)$ then $pr_B(G(x)) = pr_B(G(y))$ and $\forall (a,b) \in G$: $(a,c) \in G$, where $pr_B(c) = pr_B(b)$ and $pr_{B'}(c) = pr_{B'}(a)$ then S' solves G.*

Proof. From the proof of Theorem 2. it is clear that, if $\mathcal{D}_p(S) \supset \mathcal{D}pr_B(G)$, then $\mathcal{D}_P(S') \supset \mathcal{D}_G$. All we have to check is that $\forall a \in \mathcal{D}_G$: $p(S')(a) \subset G(a)$. Let $a \in \mathcal{D}_G$ and $b \in p(S')(a)$. Then the relation $pr_B(b) \in p(S)(pr_B(a))$ holds, and by condition, the latter is contained in $H(pr_B(a))$. Consequently the components of b in B' have to correspond to $G(a)$, but they are equal to $pr_{B'}(a)$ by the definition of the extension. □

References

[1] BEST, E., Relational Semantics of Concurrent Programs, In: *Formal Description of Programming Concepts II.* North-Holland, Amsterdam, 1978.

[2] DIJKSTRA, E.W., A Discipline of Programming Prentice-Hall Inc, Englewood Cliffs, New York, 1976.

[3] GRIES, D., The Science of Programming Springer Verlag, Berlin, 1981.

[4] GUERREIRO, P., Another characterization of weakest preconditions, In: *Internal Symposium on Programming Springer-Verlag*, Berlin, 1982.

[5] HOARE, C.A., Proof of correctness of data representations Acta informatica 1(1972), 271-281

[6] JACKSON, M.A., Principles of Programming Design Academic Press, New York, 1975.

[7] MILLS, H.D., Mathematical functions for structured programming, SC 72-6012, IBM, Gaithersburg, Md 1972.

[8] SANDERSON, J.G., A Relational Theory of Computing Springer-Verlag, Berlin, 1980.

[9] FÓTHI, Á., Unu matematika modelo por programado In: *Teoriaj kaj Praktikaj Problemaj de la Programado*, INTERCOMPUTO, Budapest, 1982.

[10] FÓTHI, Á. and VARGA, Z., Programming via recursive functions in Hungarian. In: *Alkalmazott Matematikai Lapok* 6 (1980) 331-336.

(Recived November 26, 1987)

ÁKOS FÓTHI
Dept. of General Computer Science
H-1088 Budapest, Múzeum krt. 6-8.
HUNGARY