Abstract

A nondeterministic Language for recursive definitions, L, is defined. It includes simple functions for manipulating data values, in this case the natural numbers, a choice operator OR and recursive procedures. Various kinds of parameter-passing mechanisms are allowed: the well-known call-by-value, run-time choice which models the Algol60 substitution rule and call-time choice in which no nondeterminism is allowed in the actual parameters once the procedure has been called.

An operational semantics is defined for this language in the form of a next state relation. Using this relation three different operational preorders are defined between programs. The difference in these orders reflect different views of divergent computations.

On the other hand we define three different mathematical models, in the sense of Scott-Theory and show that they are fully-abstract; that is programs are identified in the model if and only if they are identified by the corresponding operational preorder. The three different models are defined using three different powerdomains and these are shown to arise naturally by considering different properties of the semantic counter-part of the operator OR.

Finally we discuss the suitability of L for defining elements of the various domains. An element of a domain is computable if it is the least upper bound of a recursively enumerable set of finite elements. We show that for certain domains all computable elements are definable in L and point to difficulties in extending L so as to define all such elements in the remaining domains.

Introduction

Nondeterminism is implicit in many aspects of programming. A typical example is an operating system where the response to a user depends on such factors as workload, number of other users, etc. If all the parameters are known then the response is determined uniquely. But usually the relationship between the various parameters is so complicated that from the point of view of the user the response may be taken to be nondeterministic. Another example is a data base system where a query made at different points in time can have different responses. As a final example the behaviour of a distributed computing system in general depends on the relative speeds of the different processors. If these systems are examined at a level of
abstraction which ignores time then the behaviour will be nondeterministic. It follows that an adequate model of the behaviour of programs in general must be capable of representing nondeterminism.

Explicit nondeterminism has appeared in programming languages under varying disguises over the last ten to fifteen years. See for example [5],[9]. More recently Dijkstra has popularised nondeterministic features of programming languages in [4], where under-specification of the behaviour of a program is reflected in nondeterministic computations. Various semantic models of this language have appeared in the literature ([12]). In this paper we investigate another language with explicit nondeterminism. The language of recursive definitions (for example in [16]) is augmented by adding a choice operator OR. Then $T_1 \text{ OR } T_2$ can behave as $T_1$ or as $T_2$. The exact definition of the syntax of the language is taken from [6], where three different types of parameter-passing mechanisms are allowed: call-by-value, as in Algol60, run-time choice which models the substitution rule of Algol60 and finally call-time choice in which the actual parameters to a procedure must be deterministic. The operational semantics of call-time choice is somewhat different than that of [6], in that we use bindings [1] to associate the actual parameters to the formal parameters.

Three different operational preorders are defined for this language. Indeed the main point of this paper is that there is no one correct or incorrect model for nondeterministic programs. Such programs may be used in many different ways and different models reflect the different uses. The three different preorders arise from the different views one may have on divergent computations and we argue that these preorders arise quite naturally from consideration of desired input-output behaviour of programs.

On the other hand we consider mathematical models for this language, in the sense of Scott [19]. In this approach mathematical functions from domains are associated with every program. In [11],[14], the theory of domains is extended so as to model nondeterministic features of languages. The two different definitions of powerdomains are given which are intended to play the role of the "set of subsets of a set". Here we show that three different powerdomains can be defined in a natural way by considering the desirable properties of the semantic counterpart of the operator OR. Using these powerdomains three models are defined for the language and are shown to be fully abstract with respect to the operational preorders. This means that programs are identified in the model if and only if they are equivalent under the corresponding preorder. It should be emphasised that we only consider the case of unstructured data, in particular the set of integers, and the corresponding data domains are simply flat cpos. It remains to see how suitable powerdomains are for modelling languages with more complicated data structures. More generally it would be interesting to give a connection between the powerdomain approach and the theory of algebraic semantics [8]. A first step in this direction