Notre Dame Journal of Formal Logic
Volume XVI, Number 4, October 1975
NDJFAM

NON-DEFINABILITY OF CERTAIN SEMANTIC PROPERTIES OF PROGRAMS

RICHARD A. DeMILLO

1 *Introduction* This paper has a two-fold purpose. First, we address ourselves to the problem of describing programs with languages which are, in some sense, "related" to the programming language in question. In particular, we examine generalizations of the halting and equivalence problems and show that no uniform first order methods of description can exist. Second, we illustrate the very natural sort of correspondence which may be established between results in mathematical logic and a class of foundational problems in computer science. What we will refer to as *programs*, are variously called *program schemata* and *abstract programs* in the literature. The question of description or *definability* has been examined by a number of authors. Positive results relating to definability of termination appear in Engeler [2] and Manna [4]. Equivalence and decision problems are given thorough treatment in Luckham, Park, and Paterson [3]. The import of definability results is indicated in Manna and Waldinger [5]. Details of the logical results are available in Bell and Slomson [1] and vanFraassen [6].

2 *The Languages* $\mathcal{L}$ *and* $\mathcal{P}(\mathcal{L})$ By the language $\mathcal{L}$ we mean a first order predicate calculus with identity and the following primitive alphabet:

1. denumerably many individual variables: $x_0, x_1, \ldots$ (in some cases a variable may be denoted by y_j);
2. function symbols of specified addicity and unspecified number: $f_0, f_1, \ldots$;
3. predicate symbols of specified addicity and unspecified number: $q_0, q_1, \ldots$.

Function symbols taking 0—arguments are also called individual constants (denoted $a_0, a_1, \ldots$). When f is either a function or a function symbol, we let $f^n(z)$ be z when $n = 0$ and $f(f^{n-1}(z))$ when $n > 0$. $\mathcal{L}$ is completely specified when we set down its rules of formation, logical axioms, and rules of inference, all in the usual fashion. $\mathcal{P}(\mathcal{L})$ is a programming language derived from $\mathcal{L}$. About $\mathcal{P}(\mathcal{L})$'s exact structure we can be quite informal. Programs in $\mathcal{P}(\mathcal{L})$ are taken to be constructed from the following components:

Received May 7, 1973