1. INTRODUCTION

The article belongs to an area of program schemes theory. The theory studies semantic properties of programs on their models called program schemes. The concepts that give foundation to the theory of such models are stated along with a description of their implementation. The key point of the theory is the equivalence of program schemes that belong to a particular model. A class of special algebraic models with procedures, called gateway models, is studied. Conditions of the equivalence problem decidability in such models are analyzed.

About Algebraic Program Models with Procedures

R. I. Podlovchenko* and A. E. Molchanov**

Moscow State University, the CMC Dept.
e-mail: *podlovchenko.rimma@gmail.com, **gurux13@gmail.com
Received November 20, 2013

Abstract—Algebraic program models with procedures are designed to analyze program semantic properties on their models called program schemes. The concepts that give foundation to the theory of such models are stated along with a description of their implementation. The key point of the theory is the equivalence of program schemes that belong to a particular model. A class of special algebraic models with procedures, called gateway models, is studied. Conditions of the equivalence problem decidability in such models are analyzed.

DOI: 10.3103/S0146411613070250

The concepts are stated. Traditionally, an evolution of algebraic program models followed a path in which all basic operators’ compositions common for programming languages are used in programs formalization, except procedures. Models that can be constructed that way are called simple. Currently, there are many results known about equivalence and E.T. problems in simple program models. There are two main methods of solving an equivalence problem ([3] and [4]). A method of solving E.T. problem is described in [5].
It is natural to add procedures to the set of allowed compositions of operators and tests. Once the set is extended that way, algebraic program models with procedures emerge. Such models are the subject for consideration in this article.

Besides the introduction, the article contains Sections 2, 3, 4.

Section 2 is dedicated to the definition of an extended program formalization, that is, the structure, semantics and equivalence of formalized program are described. Each basis set of operators and tests defines a separate class of formalized programs.

Section 3 contains a description of algebraic program models with procedures. Given a basis of operators and tests, structure and semantics of a program scheme are defined. All program models are united into a set. Conditions on a model it must satisfy to be approximating are stated. A clean matrix scheme is defined, also it is proven that in any program model any scheme can be transformed into an equivalent clean matrix scheme.

Section 4 describes special algebraic program models with procedures called gateway program models. Such models are induced by simple program models and contain the latter as submodels. The research of gateway models originates from the idea of extending the facts known for simple models to some class of program models with procedures.

We present an algorithm that transforms an arbitrary scheme of a model in a particular class of gateway program models into an equivalent free scheme. A free scheme is a scheme that contains no elements that don’t operate during the scheme execution.

The obtained result is new. It precedes the solution to another problem: find conditions the model parameters must satisfy for the equivalence problem to be reducable to the equivalence problem in the inducing simple model.

This problem will be addressed in future work.

2. FORMALIZED PROGRAMS WITH PROCEDURES

A formalized program with procedures (further we will use the term ‘program’) is built over a finite basis that consists of elements of four nonempty and disjoint alphabets—Y, C, R, P. Elements of Y, C, R are called symbols, of P—logical variables. Each logical variable can have a value from {0, 1}. Symbols denote operators, while logical variables denote boolean expressions.

A program is a vertex-marked graph. Marks are chosen from the basis alphabets. The graph consists of subgraphs that have disjoint vertex sets. One of subgraphs is called main, and it has two selected vertices: the start vertex, that has no incoming edges and has exactly one outgoing edge; and the end vertex, that has no outgoing edges. Two vertices are selected in each non-main subgraph as well. One of them is called an initial vertex, and the other is called a final vertex. All other vertices are of one of four types: executor, test, call and return. A test has two outgoing edges that are marked with 0 and 1 correspondingly; an executor, call and return have one outgoing (unmarked) edge each. A test is marked with a variable from P, an executor, call and return are marked with symbols from Y, C, R, correspondingly. There is a one-to-one relation between calls and returns, and pairs that are related belong to the same subgraph. Each related call-return pair is assigned a unique number. The outgoing edge of a call is directed to initial vertex of some subgraph and there is an outgoing edge that goes from a final vertex of that subgraph to a related return. There is no other outgoing edges from final vertices of non-main subgraphs. All edges that begin in vertices other than call and final go to vertices of the same subgraph.

Figure shows a program over the following basis:

\[ Y = \{ y_1, y_2 \}, \quad C = \{ c_1, c_2, c_3 \}, \quad R = \{ r_1, r_2, r_3 \}, \quad P = \{ p \}. \]

It consists of a main program expressed by the main subgraph and of a procedure expressed by a non-main subgraph.

Functional meaning of a program is based upon a basis semantics. Basis semantics is an algebraic system \( \sigma \) with a freely chosen set \( \Xi_\sigma \) that for each basis item \( b \) defines a functor \( \sigma b \) with the following type:

\[
\sigma b : \begin{cases} 
\Xi_\sigma \to \Xi_\sigma, & \text{if } b \in Y \cup C, \\
\Xi_\sigma \times \Xi_\sigma \to \Xi_\sigma, & \text{if } b \in R, \\
\Xi_\sigma \to \{ 0, 1 \}, & \text{if } b \in P.
\end{cases}
\]