A SIMPLE LAMBDA-CALCULUS MODEL OF PROGRAMMING LANGUAGES

Published @ 2017 Trieste Publishing Pty Ltd

ISBN 9781760576493

A simple lambda-calculus model of programming languages by S. Kamal Abdali

Edited by Trieste Publishing Pty Ltd.
Cover @ 2017

www.triestepublishing.com

S. KAMAL ABDALI

A SIMPLE LAMBDA-CALCULUS MODEL OF PROGRAMMING LANGUAGES

Trieste

COO-3077-28

Courant Institute of
Mathematical Sciences

AEC Computing and Applied Mathematics Center

A Simple Lambda-Calculus Model of Programming Languages

S. Kamal Abdali

AEC Research and Development Report

Mathematics and Computers
July 1973

New York University

UNCLASSIFIED

AEC Computing and Applied Mathematics Center
Courant Institute of Mathematical Sciences
New York University

Mathematics and Computers COO-3077-28

A SIMPLE LAMBDA-CALCULUS MODEL OF PROGRAMMING LANGUAGES

S. Kamal Abdali

July 1973

Contract No. AT(11-1)-3077

UNCLASSIFIED

ABSTRACT

We present a simple correspondence between a large subset of ALGOL 60 language and lambda-calculus. With the aid of this correspondence, a program can be translated into a single lambda-expression. In general, the representation of a program is specified by means of a system of simultaneous conversion relations among the representations of the program constituents. High-level programming language features are treated directly, not in terms of the representations of machine-level operations. The model includes input-output in such a way that when the representation of a (convergent) program is applied to the input item representations, the resulting combination reduces to a tuple of the representations of the output items. This model does not introduce any imperative notions into the calculus; the descriptive programming constructs, such as expressions, and the imperative ones, such as assignments and jumps, are translated alike into pure lambda-expressions. The applicability of the model to the problems of proving program equivalence and correctness is illustrated by means of simple examples.

CONTENTS

1. INTRODUCTION

If one wishes to study properties of programs, then one should either develop rules of deduction and inference that apply directly to programming language constructs (e.g., Hoare [1]), or represent programs by the objects of some mathematical system [e.g., 2,3,7-10] and work with these representations. As a step in the second direction, this paper describes a way of representing programs by lambda-expressions [4-6].

Since a number of lambda-calculus (or, related, combinatory logic) models of programming languages have already appeared in the literature [among them, 7-10], the proposal of yet another such model may require justification. So we will first indicate some distinguishing features of our model vis-à-vis others.

1. Our model does not introduce any imperative or otherwise foreign notions to lambda-calculus. This is in contrast to Landin [7], in which imperative features of programming languages are accounted for by ad hoc extensions of the calculus. We find that the calculus, in its purity, suffices as a model of programming languages. By not making any additions to the lambda-calculus, we have the guarantee that all its properties, in particular, the consistency and the Church-Rosser property [6], are valid in our model. For example, even when a program requires a fixed order of execution, the lambda-expression obtained by evaluating the program representation in any order, whatsoever, represents the program result correctly.

2. Programs are translated into lambda-expressions, not interpreted by a lambda-calculus interpreter (Reynolds [10]). Thus, programming semantics is completely reduced to lambda-calculus semantics, but without commitment to any particular view of the latter. Also, all lambda-expression transformations are applicable to programs.

3. Assignments are modelled by the substitution operation of lambda-calculus. Consequently, the notions of memory, addresses, and fetch and store operations do not enter our model in an explicit manner (Stratchey [8], Reynolds [10]).