Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 343 (2005) 207–236
Theoretical Computer Science
www.elsevier.com/locate/tcs

Games for complexity of second-order call-by-name programs

Andrzej S. Murawski

Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK

Abstract

We use game semantics to show that program equivalence and program approximation in a second-order fragment of Idealized Algol are PSPACE-complete. The result relies on a PSPACE construction of deterministic finite automata representing strategies defined by second-order programs and is an improvement over the at least exponential space bounds implied by the work of other authors in which extended regular expressions were used.

The approach makes it possible to study the contribution of various constructs of the language to the complexity of program equivalence and demonstrates a similarity between call-by-name game semantics and call-by-name interpreters.

Keywords: Game semantics; Computational complexity; Program analysis

1. Introduction

Game semantics views computation as an exchange of moves between two players, who represent respectively the program under evaluation and the environment in which the program is evaluated. Programs can then be interpreted as strategies for the first player. This approach has led to the construction of first *fully abstract* models for a variety of programming languages, i.e. models in which the interpretations of two programs coincide if and only if the programs are equivalent [3,13,4,5,12,16,2,7]. The game models provide a semantic characterization of program equivalence and make it possible to recast questions about equivalence of programs as semantic problems. However, reasoning about programs

E-mail address: Andrzej.Murawski@comlab.ox.ac.uk.

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.05.013

with game models is not so easy, especially if one has automation in mind. Firstly, to achieve full abstraction, equivalence classes of strategies need to be considered instead of strategies, and in general the relation involved (the so-called *intrinsic* preorder) is very intricate. Secondly, positions arising in game semantics are not merely sequences of moves. In addition, they are endowed with pointers that connect moves subject to a number of combinatorial constraints.

The case of Idealized Algol in which expressions may have side effects is much more satisfying. There, the above-mentioned quotient set admits a direct characterization based on *complete* plays—plays that correspond to terminating computations. Consequently, the first obstacle is removed: questions about program equivalence (respectively approximation) can be restated as equivalence (respectively containment) queries for the induced sets of complete positions. Moreover, when one restricts the language to second order, positions can be treated as strings of moves, because the pointer structure is uniquely reconstructible and hence redundant. Then it turns out that complete plays generated by second-order programs form regular languages [10], which immediately implies decidability of second-order program equivalence and approximation, because the problems of equivalence and containment of regular languages are decidable.

Two expositions of the regular game semantics exist [1,10], both employing a class of semi-extended regular expressions with intersections to describe the sets of complete plays generated by programs. Because the equivalence and containment problems for such expressions are known to be EXPSPACE-complete, one might suspect that the corresponding problems concerning programs will inherit this complexity (intersections are crucial for modelling state). In this paper we show that this is not the case: program approximation as well as program equivalence in the fragment of Idealized Algol considered in these papers are in fact both PSPACE-complete.

Our approach consists of a direct construction of deterministic automata which represent the game semantics of programs. In order to avoid the use of exponential space this process has two stages: first we construct the automaton corresponding to programs in which state changes are not observed; then we refine it so that state changes are respected. Because the construction is conducted in polynomial space, and both equivalence and containment of deterministic automata are NL-complete, one can obtain a PSPACE algorithm for program approximation and equivalence by combining the two in a careful way.

To our knowledge this is the first time a complexity result like this has been proved using a denotational model.

1.1. Idealized Algol

Idealized Algol (IA) is the canonical language combining functional and imperative programming. We shall concern ourselves with its fragment, called IA_2, in which free identifiers are of base type or (first-order) function type and arguments to procedures are of base type. IA_2 types (denoted by T) are generated by the following grammar:

$$B ::= \quad com \quad | \quad exp \quad | \quad var \qquad\qquad T ::= \quad B \quad | \quad B \to T.$$

Those generated from B are called base types. *com* is the type of commands, *exp* is the type of expressions. We assume that values of type *exp* are taken from a finite initial segment