Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 337 (2005) 1–50
Theoretical Computer Science
www.elsevier.com/locate/tcs

Fundamental Study

Transforming semantics by abstract interpretation

Roberto Giacobazzi*, Isabella Mastroeni

Dipartimento di Informatica, Università di Verona, Strada Le Grazie 15, 37134 Verona, Italy

Received 3 February 2004; received in revised form 7 December 2004; accepted 13 December 2004

Communicated by G. Levi

Abstract

In 1997, Cousot introduced a hierarchy where semantics are related with each other by abstract interpretation. In this field we consider the standard abstract domain transformers, devoted to refine abstract domains in order to include attribute independent and relational information, respectively the reduced product and power of abstract domains, as domain operations to systematically design and compare semantics of programming languages by abstract interpretation. We first prove that natural semantics can be decomposed in terms of complementary attribute independent observables, leading to an algebraic characterization of the symmetric structure of the hierarchy. Moreover, we characterize some structural property of semantics, such as their compositionality, in terms of simple abstract domain equations. This provides an equational presentation of most well known semantics, which is parametric on the observable and structural property of the semantics, making it possible to systematically derive abstract semantics, e.g. for program analysis, as solutions of abstract domain equations.

Keywords: Abstract interpretation; Comparative semantics; Domain theory; Compositionality; Constraint programming

1. Introduction

Since its origin in 1977, abstract interpretation [11] has been widely used, implicitly or explicitly, to describe and formalize approximate computations in many different areas of computer science, from its very beginning use in formalizing (compile-time) program

* Corresponding author. Tel.: +39 45 802 7995; fax: +39 45 802 7982.
E-mail addresses: roberto.giacobazzi@univr.it (R. Giacobazzi), mastroeni@sci.univr.it (I. Mastroeni).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.12.021

analysis frameworks to more recent applications in model checking, program verification, data security, type inference, automated deduction, and comparative semantics. This justifies a now well established definition of abstract interpretation as *a general theory to approximate the semantics of discrete dynamic systems* [8]. This is particularly striking in comparative semantics, where semantics at different levels of abstraction can be compared with each other by abstract interpretation [10]. In this paper, we analyze the most well-known structural properties of semantics, such as their precision, compositionality, and relation between complementary observables, by using standard abstract interpretation techniques. We prove that most of these properties be characterized in terms of properties of the corresponding abstractions. This is achieved by isolating a suitable set of abstract domain transformers which allows us to design abstractions accordingly, providing a characterization of semantics of programming languages as solutions of simple abstract domain equations, involving both some basic observable property which has to be observed by the semantics and the abstract domain transformers necessary in order to achieve a suitable structural property.

1.1. The scenario

Semantics is central in the construction of any abstract interpretation. The so-called *concrete semantics* specifies the observable property of program behavior and any more abstract semantics, e.g. decidable semantics for program analysis, can be derived by abstraction. As a consequence, a semantics, at any level of abstraction, can be fully specified as an abstract interpretation of a more concrete semantics. This key idea is the basis of Cousot's design of a complete hierarchy of semantics of programming languages [9,15]. A number of semantics including big-step, termination and non-termination, Plotkin's natural, Smyth's demonic, Hoare's angelic relational and corresponding denotational, Dijkstra's predicate transformer weakest-precondition and weakest-liberal precondition and Hoare's partial and total axiomatic semantics, have all been derived by successive abstractions from an (operational) maximal trace semantics of a transition system. The resulting hierarchy (here called Cousot's hierarchy) provides a complete account on the structure and the relative precision of most well known semantics of programming languages. One of the major challenge in Cousot's construction is that *semantics are abstract domains*. Therefore they can be transformed, refined, decomposed, and composed similarly to what is usually done with abstract domains in static program analysis. This view of semantics as domains provides both a better insight on the structure and relative precision of traditional well known semantics of programming languages and the possibility to systematically specify new semantics by composition, decomposition, refinement and simplification of existing ones, by manipulating the corresponding domains.

1.2. The main results

In this paper, we treat the Cousot's hierarchy of semantics as an *algebra of semantics*, namely we apply algebraic operations to semantics, here seen as abstract domains. Our aim is to relate the properties of semantics with the properties of the abstract domain transformations used in their design. This is achieved by considering the main operations