Journal of Applied Logic 16 (2016) 72–91

Contents lists available at ScienceDirect

Journal of Applied Logic

www.elsevier.com/locate/jal

Behavioral equivalence of hidden k-logics: An abstract algebraic approach ☆

Sergey Babenyshev [a], Manuel A. Martins [b,*]

[a] *Siberian Fire-Rescue Academy, Severnaya Str. 1, Zheleznogorsk, 662972, Russia*
[b] *CIDMA – Center for R&D in Mathematics and Applications, Dep. of Mathematics, U. Aveiro, Portugal*

ARTICLE INFO

Article history:
Received 17 February 2015
Received in revised form 27 October 2015
Accepted 1 February 2016
Available online 4 March 2016

Keywords:
Behavioral equivalence
Hidden logic
Leibniz congruence

ABSTRACT

This work advances a research agenda which has as its main aim the application of Abstract Algebraic Logic (AAL) methods and tools to the specification and verification of software systems. It uses a generalization of the notion of an abstract deductive system to handle multi-sorted deductive systems which differentiate visible and hidden sorts. Two main results of the paper are obtained by generalizing properties of the Leibniz congruence — the central notion in AAL.
In this paper we discuss a question we posed in [1] about the relationship between the behavioral equivalences of equivalent hidden logics. We also present a necessary and sufficient intrinsic condition for two hidden logics to be equivalent.

1. Introduction

Behavioral abstraction plays an important role in modern specification theory by providing a more satisfactory way to prove correctness of a program with respect to a given specification. Computational systems often have interfaces that encapsulate the local states of program objects and focus instead on the operations that modify the local states and some distinguished set of specific properties — *attributes* (in particular, these features are inherent to the *object oriented* (OO) *paradigm*). In order to implement this approach, many programming languages use techniques that hide internal data types, providing abstraction from unimportant details and protection for internal data.

Like a state of a transition system, a state of an OO program can be viewed as encapsulating all pertinent information about the abstract machine when it reaches a certain stage while executing a sequence of methods and procedures. The information about the local state conceptually is partitioned into a *visible*

☆ This work was supported in part by the Portuguese Foundation for Science and Technology FCT through CIDMA, within project UID/MAT/04106/2013. The second author also acknowledges the financial assistance by EU FP7 Marie Curie PIRSES-GA-2012-318986 project GeTFun and the project FFI2013-47126-P by the Spanish Ministry of Research.
* Corresponding author.
E-mail address: martins@ua.pt (M.A. Martins).

http://dx.doi.org/10.1016/j.jal.2016.03.002
1570-8683/

and a *hidden* part, with the former representing visible data, like attributes, and the latter representing the hidden states of objects in the OO paradigm. Methods can modify the hidden state of the object (giving, as a result, a new hidden state). Hidden states (data) can only be indirectly compared by considering the visible outputs of the same programs, applied to this hidden data. Using this approach, software can be designed according to a *behavioral* specification, the latter specifying certain *visible* behavior, instead of providing detailed requirements for all internal aspects of execution. In this respect, the above-mentioned approach turned out to be related to the *coalgebraic* approach (cf. [36]).

Behaviorally, two terms are said to be equivalent if and only if they cannot be distinguished in any visible context. This basic notion of behavioral equivalence is due to Reichel [47]. The idea of using the satisfaction relation on hidden terms for determining behavioral equivalence was also introduced by Reichel in the 80's [47] and it seems to be a useful way of defining equivalence between hidden terms. In fact, in applied settings, there are some well designed pieces of software that may fail to satisfy their requirements strictly, but do satisfy them behaviorally, i.e., under any program executed on the system (see [2,5,8]). More formally, in such approach, the standard equality predicate is augmented by *behavioral equivalence* (two data elements representing states are said to be *behaviorally equivalent* if every function returns the same visible value when executed on the same visible input). Behavioral equivalence has been adopted and generalized by many researches. The most significant contributions have been provided by Goguen, Bidoit, Bouhoula and their associates.

Hidden algebras were introduced by Goguen in [19] and further developed in [21,24], in order to generalize many-sorted algebras to give an algebraic semantics for the object oriented paradigm. In fact, the behavioral aspects of modern software make hidden algebras in practice more suitable than standard algebras as abstract machine implementations. Consequently, there has been an increasing interest in this field. Goguen and his collaborators have been improving their theory and applying it in more general settings. Now almost all of the results may be established for the general setting of *polyadic loose-data semantics*. Polyadic loose-data semantics allow any kind of operation symbols and, in order to have more freedom in choosing an adequate implementation, the visible part of the algebras is no longer fixed: it may be any sorted algebra in which the requirements (axioms) of the given specification are valid. However, some authors are interested in applying coalgebraic methods, and therefore they have to restrict their signatures to the monadic fixed-data semantics. Malcolm [29] had shown that behavioral equivalence may be formulated in the context of coalgebra (see also [28] and [48]).

Several generalizations of the notion of behavioral equivalence have been considered. Goguen et al. [21,49] consider Γ-*behavioral equivalence*, where Γ is a subset of the set of all operation symbols in the signature. Γ-behavioral equivalence is defined analogously to ordinary behavioral equivalence, but only makes use of the contexts built from the operation symbols in Γ. It can be proven that the Γ-behavioral equivalence is the largest Γ-congruence with the identity as the visible part. Based on this fact, the coinduction methods may still be formulated for this more general notion. Other interesting questions concerning Γ-behavioral equivalence also arise, such as the study of the compatibility of some operation symbols outside of Γ with respect to Γ-behavioral equivalence. This problem has been studied by Diaconescu et al. [16] and Bidoit et al. [3]. On the other hand, Bidoit and Hennicker [4] generalized the notion by endowing the hidden algebras with a binary relation, that may be partial. In particular, one can apply the algebraic approach to the behavioral setting by considering the algebras together with the Γ-behavioral equivalence.

Various notions of behavioral logics have been considered for reasoning about behavioral equivalence. The most relevant to the topic of this paper are the variations developed by Goguen et al. (*hidden logics* [20,21]) and by Bidoit and Hennicker (*observational logics* [2,27]). These approaches formalize behavioral validity (correctness) as follows: hidden logic is a variant of the equational logic in which some part of the specification is visible and another is hidden. The basic syntactic structures for hidden logics are equations and the behavioral satisfaction relation is defined by interpreting the equality symbol as the *behavioral equality*. Observational logics are different from hidden logics in that respect that instead of dealing with