Information and Computation 243 (2015) 205–221

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

The IO and OI hierarchies revisited [☆]

Gregory M. Kobele [a,1], Sylvain Salvati [b,*]

[a] University of Chicago, United States
[b] Bordeaux University/INRIA, France

ARTICLE INFO

Article history:
Received 1 September 2013
Available online 12 December 2014

Keywords:
Higher-order grammars
Simply typed lambda-calculus
IO
OI
Formal language theory
Montague semantics
Membership problem

ABSTRACT

We study languages of λ-terms generated by IO and OI unsafe grammars. These languages can be used to model meaning representations in the formal semantics of natural languages following the tradition of Montague. Using techniques pertaining to the denotational semantics of the simply typed λ-calculus, we show that the emptiness and membership problems for both types of grammars are decidable. In the course of the proof of the decidability results for OI, we identify a decidable variant of the λ-definability problem, and prove a stronger form of Statman's finite completeness Theorem.

1. Introduction

At the end of the sixties, similar but independent lines of research were being pursued in formal language theory and in the formal semantics of natural language. Formal language theory was refining the Chomsky hierarchy so as to find an adequate syntactic model of programming languages lying in between the context-free and context-sensitive languages. Among others, this period resulted in the definition of *IO and OI macro languages* by Fischer [13] and the notion of *indexed languages* (which coincide with OI macro languages) by Aho [2]. At the same time, Richard Montague [25] was proposing a systematic way of mapping natural language sentences to logical formulae representing their meanings, providing thereby a solid foundation for the field of formal semantics. The main idea behind these two lines of research can be summed up in the phrase '*going higher-order*.' For macro and indexed grammars, this consisted in parameterizing non-terminals with strings and indices (stacks) respectively, and in Montague's work it consisted in using the simply typed λ-calculus to map syntactic structures to their meanings. In this respect, Montague was ahead of the formal language theory community, which took another decade to go higher-order with the work of Damm [7]. However, the way Damm defined higher-order grammars used (implicitly) a restricted version of the λ-calculus that is now known as the *safe λ-calculus*. This restriction was made explicit by Knapik *et al.* [19] and further studied by Blum and Ong [4]. For formal grammars this restriction was first lifted by de Groote [8] and Muskens [27] in the context of computational linguistics as a way of applying Montague's techniques to syntactic modeling.

In the context of higher-order recursive schemes, Ong [28] showed that safety was not a necessary condition for the decidability of the MSO model checking problem. The safety restriction has been shown to be a real restriction by Parys [29].

[☆] Supported by ANR project POLYMNIE: ANR-12-CORD-0004 POLYMNIE.
[*] Corresponding author.
E-mail addresses: kobele@uchicago.edu (G.M. Kobele), salvati@labri.fr (S. Salvati).
[1] The first author was funded by LaBRI while working on this research.

http://dx.doi.org/10.1016/j.ic.2014.12.015
0890-5401/© 2014 Published by Elsevier Inc.

Nevertheless, concerning the IO and OI hierarchies, the question as to whether safety is a genuine restriction in terms of the definable languages is still an open problem. Aehlig *et al.* [1] showed that, for second order OI grammars, safety was in fact *not* a restriction. It is nevertheless generally conjectured that safety is a restriction for higher-order grammars.

As we wish to extend Montague's technique with the OI hierarchy so as to enrich it with fixed-point computation as proposed by Moschovakis [26], or as in proposals to handle presuppositions in natural languages by Lebedeva and de Groote [10,9,22], we work with languages of λ-terms rather than with just languages of strings or trees. In the context of languages of λ-terms, safety clearly appears to be a restriction since, as shown by Blum and Ong [4], not every λ-term is safe. Moreover the terms generated by Montague's technique appear to be unsafe in general.

This paper is thus studying the formal properties of the unsafe IO and OI languages of λ-terms. A first property that the use of unsafe grammars brings into the picture is that the class of unsafe IO languages hierarchy is strictly included within the class of unsafe OI languages. The inclusion can be easily shown using a standard continuation passing style (CPS) transform on the grammars, and its strictness is implied by decidability results. Nevertheless, it is worth noting that such a transform does not result in safe grammars, and so it is unclear whether safe IO languages are safe OI languages. This paper focuses primarily on the emptiness and the membership problems for unsafe IO and OI languages, using simple techniques related to the denotational semantics of the λ-calculus. For the IO case, we recast some known results from Salvati [31,30] so as to emphasize that they derive from the fact that, given an IO language and a finite model of the λ-calculus, one can effectively compute the elements of the model which are the interpretations of terms in the language. This allows us to show that the emptiness problem is decidable, and also, using Statman's finite completeness theorem [35], to show that the membership problem is decidable. In contrast to the case for IO languages, we show that this proof method does not work for OI languages. Indeed, we prove that the set of closed λ-terms of a given type is an OI language, and thus, since λ-definability is undecidable [23], the set of elements in a finite model that are the interpretation of terms in an OI language cannot be effectively computed. To show the decidability of the emptiness and membership problems for OI, we prove a theorem that we call the *Observability Theorem*; it characterizes some semantic properties of the elements of an OI language in monotonic models, and leads directly to the decidability of the emptiness problem. For the membership problem we prove a generalization of Statman's finite completeness theorem which, in combination with the Observability Theorem, entails the decidability of the membership problem of OI languages.

The work we present here is closely related to the research that is being carried out on higher-order recursive schemes. It differs from it in one important respect: the main objects of study in the research on higher-order recursive schemes are the infinite trees generated by schemes, while our work is related to the study of the Böhm trees of λY-terms, which may contain λ-binders. Such Böhm trees are closer to the configuration graphs of Higher-order Collapsible Pushdown Automata, whose first-order theory has been shown undecidable [6]. If we were only interested in grammars generating trees or strings, the decidability of MSO for higher-order recursion schemes [28] would yield the decidability of both the emptiness and the membership problems of unsafe OI grammars, but this is no longer the case when we turn to languages of λ-terms. It is also important to notice that for the cases of string languages and tree languages, our results give more specific algorithms than those induced by the theorem of Ong [28] and are proved using very different techniques.

Organization of the paper We start by giving the definitions related to the λ-calculus, its finitary semantics, and how to define higher-order grammars in Section 2. We then present the decidability results concerning higher-order IO languages and explain why the techniques used there cannot be extended to OI languages in Section 3. Section 4 contains the main contributions of the paper: the notion of hereditary prime elements of monotone models together with the Observability Theorem, and a strong form of Statman's finite completeness Theorem. Finally we present conclusions and a broader perspective on our results in Section 5.

2. Preliminaries

In this section, we introduce the various calculi we are going to use in the course of the article. Then we show how these calculi may be used to define IO and OI grammars. We give two presentations of these grammars, one using traditional rewriting systems incorporating non-terminals, and the other as terms in one of the calculi; these two perspectives are equivalent. In the remainder of the paper we will switch between these two formats as is most convenient. Finally we introduce the usual notions of full and monotone models for the calculi we work with.

2.1. λ-Calculi

We introduce here various extensions of the simply typed λ-calculus. Given an atomic type 0 (our results extend without difficulty to any finite number of atomic types), the set *type* of types is built inductively using the binary right-associative infix operator $\rightarrow$. We write $\alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \alpha_0$ for $(\alpha_1 \rightarrow (\cdots(\alpha_n \rightarrow \alpha_0)))$. As in [16], the order of a type is: $\mathrm{order}(0) = 1$, $\mathrm{order}(\alpha \rightarrow \beta) = \max(\mathrm{order}(\alpha) + 1, \mathrm{order}(\beta))$. Constants are declared in higher-order signatures Σ which are finite sets of typed constants $\{A_1^{\alpha_1}, \ldots, A_n^{\alpha_n}\}$. We use constants to represent non-terminal symbols.

We assume that we are given, for each type, a countably infinite set of typed λ-variables $(x^\alpha, y^\beta, \ldots)$. The families of typed $\lambda Y{+}\Omega$-terms $(\Lambda^\alpha)_{\alpha \in type}$ built on a signature Σ are inductively constructed according to the following rules, where $c^\alpha \in \Sigma$: x^α, c^α and Ω^α are in Λ^α; Y^α is in $\Lambda^{(\alpha\rightarrow\alpha)\rightarrow\alpha}$; if M is in $\Lambda^{\alpha\rightarrow\beta}$ and N is in Λ^α, then (MN) is in Λ^β; if M is in