The IO and OI hierarchies revisited[☆]

Gregory M. Kobele[,☆☆]

University of Chicago

Sylvain Salvati

Bordeaux University/INRIA

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 [25]. 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 [35].

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

[☆]Supported by ANR project POLYMNIE: ANR-12-CORD-0004 POLYMNIE

[☆☆]The first author was funded by LaBRI while working on this research.

Email addresses: kobele@uchicago.edu (Gregory M. Kobele[,☆☆]), salvati@labri.fr (Sylvain Salvati)

The IO and OI hierarchies revisited[☆]

Gregory M. Kobele[☆☆]

University of Chicago

Sylvain Salvati

Bordeaux University/INRIA

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

[☆]Supported by ANR project POLYMNIE: ANR-12-CORD-0004 POLYMNIE

[☆☆]The first author was funded by LaBRI while working on this research.

Email addresses: kobele@uchicago.edu (Gregory M. Kobele[☆☆]), salvati@labri.fr (Sylvain Salvati)

problem. The safety restriction has been shown to be a real restriction by Parys [29]. 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 $\to$. We write $\alpha_1 \to \cdots \to \alpha_n \to \alpha_0$ for $(\alpha_1 \to (\cdots(\alpha_n \to \alpha_0)))$. As in [16], the order of a type is: $\mathrm{order}(0) = 1$, $\mathrm{order}(\alpha \to \beta) = \max(\mathrm{order}(\alpha) + 1, \mathrm{order}(\beta))$. 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^α, y^β,...). 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 $\Lambda^{(\alpha\to\alpha)\to\alpha}$; if M is in $\Lambda^{\alpha\to\beta}$ and N

is in Λ^α, then (MN) is in Λ^β; if M is in Λ^β then $(\lambda x^\alpha.M)$ is in $\Lambda^{\alpha\to\beta}$; if M and N are in Λ^0 then $M+N$ is in Λ^0. When M is in Λ^α we say that it has type α, we write M^α to indicate that M has type α; the order of a term M, is the order of its type. As it is customary, we omit type annotations when they can be easily inferred or when they are irrelevant. We adopt the usual conventions about dropping parentheses in the syntax of terms. We write $M^{\alpha\to\beta}+N^{\alpha\to\beta}$ as an abbreviation for $\lambda x^\alpha.Mx+Nx$. The set of free variables of the term M is written $FV(M)$; a term M is closed when $FV(M)=\emptyset$. Finally we write $M[x_1\leftarrow N_1,\ldots,x_n\leftarrow N_n]$ for the simultaneous capture-avoiding substitutions of the terms N_1, ..., N_n for the free occurrences of the variables x_1, ..., x_n in M.

The set of λ-terms is the set of terms that do not contain occurrences of Y, $+$ or Ω, and for any $S\subseteq\{Y,+,\Omega\}$, the λS-terms are the $\lambda Y{+}\Omega$-terms that may contain only those constructors in S. For example, $\lambda{+}\Omega$-terms are the terms that do not contain occurrences of Y.

We assume the reader is familiar with the notions of β- and η-contraction and η-long forms (see [16]). The constant Ω^α stands for the undefined term of type α, Y^α is the fixpoint combinator of type $(\alpha\to\alpha)\to\alpha$, and $+$ is the non-deterministic choice operator. The families of terms that may contain occurrences of Ω are naturally ordered with the least compatible relation $\sqsubseteq$ such that $\Omega^\alpha\sqsubseteq M$ for every term M of type α; δ-contraction provides the operational semantics of the fixpoint combinator: $YM\to_\delta M(YM)$, and $+$-contraction gives the operational semantics of the non-deterministic choice operator: $M+N\to_+M$ and $M+N\to_+N$. Given a set $\mathcal{R}$ of symbols denoting compatible relations, for $S\subseteq\mathcal{R}$, S-contraction is the union of the contraction relations denoted by the symbols in S; it will generally be written as $\to_S$. For example, $\to_{\beta\eta+}$ denotes $\beta\eta+$-contraction. S-reduction, written $\overset{*}{\to}_S$, is the reflexive transitive closure of S-contraction, and S-conversion, written $=_S$, is the smallest equivalence relation containing $\to_S$. A term is in S-normal form when it cannot be further reduced by any of the contraction relations denoted by symbols in S; we simply say it is in *normal form* when S is clear from the context. We recall (see [16]) that when $M\overset{*}{\to}_{\beta\eta\delta+}N$, and M' and N' are the η-long forms of M and N respectively, then $M'\overset{*}{\to}_{\beta\delta+}N'$. This means that η-long forms provide a means of forgetting about η-reduction; in addition, they strongly relate the syntax of terms to the structure of their types. Therefore, in the remainder of the paper, we assume that we are working with terms in η-long form.

2.2. IO and OI grammars

The notions of IO and OI macro grammars were introduced by Fischer [13] so as to extend the expressive power of context free grammars. As for context free grammars, these grammars are defined by means of non-terminals that can be rewritten with production rules. Fischer's extension consists in using non-terminals that are parametrized by strings. These non-terminals can be understood as non-deterministic functions from strings to strings. Going higher-order consists in parametrizing non-terminals not only with first order data such

as strings, but also with functions over those data and functionals over those functions and so on. A simple way to represent such grammars is to assign non-terminals a type, and to rewrite non-terminals to λ-terms of the same type. We define a higher-order macro grammar $\mathcal{G}$ as a triple (Σ, R, S) where Σ is a higher-order signature of non-terminals, R is a finite set of rules $A \to M$, for A a non-terminal of Σ, and M a λ-term built on Σ that has the same type as A, and where S is a distinguished non-terminal of Σ, the *start symbol*. We do not require M to be a closed term; free variables in the right hand side of grammatical rules play the role of terminal symbols (as there are finitely many rules, there can be only finitely many such variables). We also do not require S to be of a particular type. This permits (higher-order) macro grammars to define languages of λ-terms of arbitrary types. As noted in [8], languages of strings and trees are special cases of languages of λ-terms. A grammar has order n when its non-terminals have at most order n. Our higher-order macro grammars generalize those of Damm [7] in two ways. First, they do not necessarily fulfill the safety condition that hold of Damm's grammars. According to [4], a term M is safe when no subterm N of M contains free λ-variables (excluding the free variables of M which play the role of constants) of order lower than that of N, unless N occurs as part of a subterm NP or $\lambda x.N$. A grammar is safe when the right hand sides of its rules are all safe terms. Safe terms can be *safely* reduced using substitution in place of capture-avoiding substitutions. Second, instead of only defining languages of strings or trees, our higher-order macro grammars can define languages of λ-terms of higher type, following Montague's tradition in the formal semantics of natural languages.

The rules of a grammar $\mathcal{G} = (\Sigma, R, S)$ define a natural relation $\to_{\mathcal{G}}$ on terms built on Σ. We write $M \to_{\mathcal{G}} N$ when N is obtained from M by replacing (without capturing free variables) an occurrence of a non-terminal A in M by a term P, such that $A \to P$ is a rule of $\mathcal{G}$. A term is in $\mathcal{G}$-normal form when it does not contain any occurrence of a non-terminal and in $\beta\mathcal{G}$-normal form when it is both $\mathcal{G}$ normal and β-normal. The grammar $\mathcal{G}$ defines two languages: $\mathcal{L}_{OI}(\mathcal{G}) = \{M \text{ in } \beta\mathcal{G}\text{-normal form} \mid S \overset{*}{\to}_{\beta\mathcal{G}} M\}$ and $\mathcal{L}_{IO}(\mathcal{G}) = \{M \text{ in } \beta\text{-normal form} \mid \exists P \text{ in } \mathcal{G}\text{-normal form}. S \overset{*}{\to}_{\mathcal{G}} P \wedge P \overset{*}{\to}_{\beta} M\}$. These two languages can be defined in a different manner, in particular M is in $\mathcal{L}_{OI}(\mathcal{G})$ iff S can be reduced to M with the head reduction strategy that consists in always contracting top-most redices of the relation $\to_{\beta\mathcal{G}}$. For a given grammar $\mathcal{G}$, we always have that $\mathcal{L}_{IO}(\mathcal{G}) \subseteq \mathcal{L}_{OI}(\mathcal{G})$, but, in general, $\mathcal{L}_{IO}(\mathcal{G}) \neq \mathcal{L}_{OI}(\mathcal{G})$. For a grammar $\mathcal{G}$, we say that the terms in $\mathcal{L}_{OI}(\mathcal{G})$ are obtained in the OI ('outside-in') mode of derivation while those in $\mathcal{L}_{IO}(\mathcal{G})$ are obtained in the IO ('inside-out') mode of derivation. For a grammar that is used in the IO or in the OI mode of derivation we may speak respectively of an IO or of an OI grammar. The class of languages defined by IO or OI grammars respectively define IO and OI languages and when restricted to order n they respectively define order n IO and OI languages which are denoted IO_n and OI_n. The class IO_n is included in the class IO_{n+1} and the class OI_n is included in the class OI_{n+1}. The same argument as for safe grammars should show that those inclusions are strict [12]. The IO and

OI hierarchies are the families of increasing classes of languages $(\mathrm{IO}_n)_{n\in\mathbb{N}}$ and $(\mathrm{OI}_n)_{n\mathbb{N}}$.

Here follows an example of a second order macro grammar $\mathcal{G}$ whose free variables (i.e. terminals) are $\textsc{ex}^{(0\to 0)\to 0}$, $\textsc{and}^{0\to 0\to 0}$, $\textsc{not}^{0\to 0}$ and $\mathrm{P}^{0\to 0}$, and whose non-terminals are S^0 (the start symbol), $S'^{0\to 0}$ and $\mathbf{cons}^{0\to 0\to 0}$ (extending BNF notation to macro grammars).

$$\begin{array}{rcl} S & \to & \textsc{ex}(\lambda x.(S' x)) \mid \textsc{and}\, S\, S \mid \textsc{not} S \\ S' & \to & \lambda y.\textsc{ex}(\lambda x.S'(\mathbf{cons}\, x\, y)) \mid \lambda y.\textsc{and}(S' y)(S' y) \mid \lambda y.\textsc{not}(S' y) \mid \lambda y.\mathrm{P} y \\ \mathbf{cons} & \to & \lambda xy.x \mid \lambda xy.y \end{array}$$

The language $\mathcal{L}_{OI}(\mathcal{G})$ represents the set of formulae of first-order logic built with one predicate P (we use a similar construction later on to prove Theorem 10). For example the formula $\textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}x)(\mathrm{P}y)))$, written in the more familiar syntax of first-order logic as $\exists x.\exists y.P(x) \wedge P(y)$, can be derived as shown on Figure 1. In derivation representations, we underline the redex that the next step is reducing.

$$\begin{array}{rcl} \underline{S} & \to_{\mathcal{G}} & \textsc{ex}(\lambda x.(\underline{S' x})) \\ & \to_{\mathcal{G}} & \textsc{ex}(\lambda x.(\underline{(\lambda z.\textsc{ex}(\lambda y.S'(\mathbf{cons}\, y\, z)))x})) \\ & \to_{\beta} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\underline{S'}(\mathbf{cons}\, y\, x))) \\ & \to_{\mathcal{G}} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\underline{(\lambda z.\textsc{and}(S' z)(S' z))(\mathbf{cons}\, y\, x)})) \\ & \to_{\beta} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(S'(\underline{\mathbf{cons}}\, y\, x))(S'(\mathbf{cons}\, y\, x)))) \\ & \to_{\mathcal{G}} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(S'(\underline{(\lambda uv.v)\, y}\, x))(S'(\mathbf{cons}\, y\, x)))) \\ & \to_{\beta} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(S'(\underline{(\lambda v.v)\, x}))(S'(\mathbf{cons}\, y\, x)))) \\ & \to_{\beta} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(S' x)(S'(\underline{\mathbf{cons}}\, y\, x)))) \\ & \to_{\mathcal{G}} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(S' x)(S'(\underline{(\lambda uv.u)\, y}\, x)))) \\ & \to_{\beta} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(S' x)(S'(\underline{(\lambda v.y)\, x})))) \\ & \to_{\beta} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\underline{S' x})(S' y))) \\ & \to_{\mathcal{G}} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\underline{(\lambda z.\mathrm{P}z)x})(S' y))) \\ & \to_{\beta} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}x)(\underline{S' y}))) \\ & \to_{\mathcal{G}} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}x)(\underline{(\lambda z.\mathrm{P}z)y}))) \\ & \to_{\beta} & \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}x)(\mathrm{P}y))) \end{array}$$

Figure 1: OI derivation of $\textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}x)(\mathrm{P}y)))$ with the grammar $\mathcal{G}$

The language $\mathcal{L}_{IO}(\mathcal{G})$ represents the formulae of first-order logic that can be built with only one variable (that is each subformula of a formula represented in $\mathcal{L}_{IO}(\mathcal{G})$ contains at most one free variable). We give an example of the

derivation of the formula $\textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}y)(\mathrm{P}y)))$, representing the (one-variable) FOL sentence $\exists x.\exists y.P(y) \wedge P(y)$, in Figure 2.

$$
\begin{aligned}
\underline{S} &\to_{\mathcal{G}} \textsc{ex}(\lambda x.(\underline{S'}x)) \\
&\to_{\mathcal{G}} \textsc{ex}(\lambda x.((\lambda z.\textsc{ex}(\lambda y.\underline{S'}(\mathbf{cons}\, y\, z)))x)) \\
&\to_{\mathcal{G}} \textsc{ex}(\lambda x.((\lambda z.\textsc{ex}(\lambda y.(\lambda u.\textsc{and}(\underline{S'}u)(S'u))(\mathbf{cons}\, y\, z)))x)) \\
&\to_{\mathcal{G}} \textsc{ex}(\lambda x.((\lambda z.\textsc{ex}(\lambda y.(\lambda u.\textsc{and}((\lambda v.\mathrm{P}v)u)(\underline{S'}u))(\mathbf{cons}\, y\, z)))x)) \\
&\to_{\mathcal{G}} \textsc{ex}(\lambda x.((\lambda z.\textsc{ex}(\lambda y.(\lambda u.\textsc{and}((\lambda v.\mathrm{P}v)u)((\lambda v.\mathrm{P}v)u))(\underline{\mathbf{cons}}\, y\, z)))x)) \\
&\to_{\mathcal{G}} \textsc{ex}(\lambda x.((\lambda z.\textsc{ex}(\lambda y.(\lambda u.\textsc{and}((\lambda v.\mathrm{P}v)u)((\lambda v.\mathrm{P}v)u))(\underline{(\lambda uv.u)\, y}\, z)))x)) \\
&\to_{\beta} \textsc{ex}(\lambda x.((\lambda z.\textsc{ex}(\lambda y.(\lambda u.\textsc{and}((\lambda v.\mathrm{P}v)u)((\lambda v.\mathrm{P}v)u))(\underline{(\lambda v.y)\, z})))x)) \\
&\to_{\beta} \textsc{ex}(\lambda x.((\lambda z.\textsc{ex}(\lambda y.\underline{(\lambda u.\textsc{and}((\lambda v.\mathrm{P}v)u)((\lambda v.\mathrm{P}v)u))y})x)) \\
&\to_{\beta} \textsc{ex}(\lambda x.((\lambda z.\textsc{ex}(\lambda y.\textsc{and}(\underline{(\lambda v.\mathrm{P}v)y})((\lambda v.\mathrm{P}v)y)))x)) \\
&\to_{\beta} \textsc{ex}(\lambda x.((\lambda z.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}y)(\underline{(\lambda v.\mathrm{P}v)y})))x)) \\
&\to_{\beta} \textsc{ex}(\lambda x.(\underline{(\lambda z.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}y)(\mathrm{P}y)))x})) \\
&\to_{\beta} \textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}y)(\mathrm{P}y)))
\end{aligned}
$$

Figure 2: IO derivation of $\textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}y)(\mathrm{P}y)))$ with the grammar $\mathcal{G}$

As the description of $\mathcal{L}_{IO}(\mathcal{G})$ given above suggests, the term

$$\textsc{ex}(\lambda x.\textsc{ex}(\lambda y.\textsc{and}(\mathrm{P}x)(\mathrm{P}y)))$$

is not in $\mathcal{L}_{IO}(\mathcal{G})$. The reason why this is so illustrates well the difference between IO and OI. In the IO mode of derivation, terms must be reduced using the rules of the grammar until all non-terminals have been eliminated, before β-reduction is performed. In the OI mode of derivation, on the other hand, a β-contraction step may be performed at any time during the derivation. This feature, particular to OI, allows (in the derivation of Figure 1) the term $\mathbf{cons}\, y\, x$ to be copied, resulting in two occurrences which are reduced independently. This is not possible in the IO mode of derivation since the term $\mathbf{cons}\, y\, x$ contains the non-terminal **cons**, and thus must be further reduced using the rules of the grammar before it may be copied in a β-reduction step.

Given the definition of safety in [4], it is easily verified that the terms of $\mathcal{L}_{IO}(\mathcal{G})$ and $\mathcal{L}_{OI}(\mathcal{G})$ of the example grammar $\mathcal{G}$ are not safe; this illustrates that unsafe IO and OI languages of λ-terms are more general than their safe counterparts. Moreover, when seen as graphs, the terms of $\mathcal{L}_{OI}(\mathcal{G})$ form a class of graphs which has an unbounded treewidth; the MSO theory of these terms is undecidable. This explains why the decidability results we obtain later on cannot be seen as immediate corollaries of Ong's Theorem [28].

2.3. IO and OI languages in the $\lambda Y{+}\Omega$-calculus

We extend the notions of IO and OI languages to $\lambda Y{+}\Omega$-terms. Given a $\lambda Y{+}\Omega$-term M, its IO language $\mathcal{L}_{IO}(M)$ is the set of λ-terms N in normal form such that there is P such that $M \xrightarrow{*}_{\delta+} P \xrightarrow{*}_{\beta} N$. Its OI language $\mathcal{L}_{OI}(M)$ is the set of λ-terms N in normal form such that $M \xrightarrow{*}_{\beta\delta+} N$ (we can also restrict our attention to head-reduction). An alternative characterization of $\mathcal{L}_{OI}(M)$, which we make use of later on, is the following. Given a term M we write $\omega(M)$ for the *immediate approximation of* M, that is the term obtained from M as follows: $\omega(\lambda x^{\alpha}.M) = \Omega^{\alpha\to\beta}$ if $\omega(M) = \Omega^{\beta}$, and $\lambda x^{\alpha}.\omega(M)$ otherwise; $\omega(MN) = \Omega^{\beta}$ if $\omega(M) = \Omega^{\alpha\to\beta}$ or $M = \lambda x.P$, and $\omega(MN) = \omega(M)\omega(N)$ otherwise; $\omega(Y^{\alpha}) = \Omega^{(\alpha\to\alpha)\to\alpha}$, $\omega(x^{\alpha}) = x^{\alpha}$, $\omega(\Omega^{\alpha}) = \Omega^{\alpha}$, and $\omega(N_1 + N_2) = \omega(N_1) + \omega(N_2)$. Note that $\omega(M)$ is a $\lambda{+}\Omega$-term that contains no β-redices. A $\lambda{+}\Omega$-term Q is a *finite approximation of* M if there is a P such that $M \xrightarrow{*}_{\beta\delta} P$ and $Q = \omega(P)$. The language $\mathcal{L}_{OI}(M)$ is the union of the languages $\mathcal{L}_{OI}(Q)$ so that Q is a finite approximation of M.

In both IO and OI modes of evaluation, $\lambda{+}\Omega$-terms define finite languages, and the $\lambda Y{+}\Omega$-calculus defines exactly the same classes of languages as higher-order macro grammars.

Theorem 1 *Given a higher-order macro grammar $\mathcal{G}$, there is a $\lambda Y{+}\Omega$-term M such that $\mathcal{L}_{OI}(\mathcal{G}) = \mathcal{L}_{OI}(M)$ and $\mathcal{L}_{IO}(\mathcal{G}) = \mathcal{L}_{IO}(M)$.*

Given a $\lambda Y{+}\Omega$-term M there is a higher-order macro grammar $\mathcal{G}$ such that $\mathcal{L}_{OI}(\mathcal{G}) = \mathcal{L}_{OI}(M)$ and $\mathcal{L}_{IO}(\mathcal{G}) = \mathcal{L}_{IO}(M)$.

The proof of this theorem is based on the correspondence between higher-order schemes and the λY-calculus given in [33]. Going from a $\lambda Y{+}\Omega$-term to a grammar is simply a direct transposition of the procedure described in [33] with the obvious treatment for $+$. For the other direction, it suffices to see the grammar as a non-deterministic scheme, which is done by viewing all the rules $A \to M_1, \ldots, A \to M_n$, of a non-terminal A as a unique rule of a scheme $A \to M_1 + \cdots + M_n$; and then to transform the scheme into a $\lambda Y{+}$-term using the transformation given in [33]. There is a minor technical difficulty concerning IO languages; one needs to start with a grammar where every non-terminal can be rewritten into a $\mathcal{G}$-normal form using $\xrightarrow{*}_{\mathcal{G}}$ only. This is related to the slight differences in the treatment of recursion in grammars and in terms, which could be suppressed by following the slightly modified syntax of [33] where Y is a variable binding operator rather than a combinator.

The grammar $\mathcal{G}$ that we took as an example is represented by the term:

$$
\begin{array}{l}
Y(\lambda S.\ \textsc{ex}(Y(\lambda S' y.\ \textsc{ex}(\lambda x.S'(x+y)) \\
\qquad\qquad\qquad\qquad + \textsc{and}(S' y)(S' y) \\
\qquad\qquad\qquad\qquad + \textsc{not}(S' y) \\
\qquad\qquad\qquad\qquad + P y)) \\
\qquad + \textsc{and}(S)(S) \\
\qquad + \textsc{not}(S))
\end{array}
$$

2.4. Models of the λ-calculi

Full models of the λ-calculus. We start by giving the simplest notion of model for the λ-calculus, that of full models. A full model $\mathcal{F}$ is a collection of type-indexed sets $(\mathcal{F}_\alpha)_{\alpha \in type}$, where $\mathcal{F}_{\alpha\to\beta}$ is $\mathcal{F}_\beta^{\mathcal{F}_\alpha}$, the set of functions from $\mathcal{F}_\alpha$ to $\mathcal{F}_\beta$. Note that $\mathcal{F}$ is completely determined by $\mathcal{F}_0$. A full model is said to be finite when $\mathcal{F}_0$ is a finite set; in that case $\mathcal{F}_\alpha$ is finite for every $\alpha \in type$. A valuation ν is a function that maps variables to elements of $\mathcal{F}$ respecting typing, which means that, for every x^α, $\nu(x^\alpha)$ is in $\mathcal{F}_\alpha$. Given a valuation ν and a in $\mathcal{F}_\alpha$, we write $\nu[x^\alpha \leftarrow a]$ for the valuation which maps the variable x^α to a but is otherwise equal to ν. We can now interpret λ-terms in $\mathcal{F}$, using the following interpretation scheme: $[\![x^\alpha]\!]^\nu_{\mathcal{F}} = \nu(x^\alpha)$, $[\![MN]\!]^\nu_{\mathcal{F}} = [\![M]\!]^\nu_{\mathcal{F}}([\![N]\!]^\nu_{\mathcal{F}})$ and $[\![\lambda x^\alpha.M]\!]^\nu_{\mathcal{F}}$ is the function such that for a in $\mathcal{F}_\alpha$, $[\![\lambda x^\alpha.M]\!]^\nu_{\mathcal{F}}(a) = [\![M]\!]^{\nu[x^\alpha\leftarrow a]}_{\mathcal{F}}$. For a closed term M, $[\![M]\!]^\nu_{\mathcal{F}}$ does not depend on ν, and thus we simply write $[\![M]\!]_{\mathcal{F}}$. The following facts are known about full models:

Theorem 2 (Soundness (see [15])) *If $M =_{\beta\eta} N$ then for every full model $\mathcal{F}$ and valuation ν, $[\![M]\!]^\nu_{\mathcal{F}} = [\![N]\!]^\nu_{\mathcal{F}}$.*

Theorem 3 (Finite Completeness [35]) *Given a λ-term M, there is a finite full model $\mathcal{F}_M$ and a valuation ν, such that, for every N, $[\![N]\!]^\nu_{\mathcal{F}_M} = [\![M]\!]^\nu_{\mathcal{F}_M}$ iff $N =_{\beta\eta} M$.*

Theorem 3 implies in particular that it suffices for two λ-terms to have the same interpretation in every finite full model so as to be $\beta\eta$-convertible. In this theorem, the constructions of $\mathcal{F}_M$ and ν are effective.

For a full model $\mathcal{F}$, an element f of $\mathcal{F}_\alpha$ is said to be *λ-definable* when there is a closed λ-term M such that $[\![M]\!]_{\mathcal{F}} = f$. The problem of *λ-definability* is the problem whose input is a finite full model $\mathcal{F}$ together with an element f of $\mathcal{F}_\alpha$, and whose answer is whether f is λ-definable.

Theorem 4 (Loader [23]) *The λ-definability problem is undecidable.*

Given a language of λ-terms L, a full model $\mathcal{F}$, and a valuation ν, we write $[\![L]\!]^\nu_{\mathcal{F}}$ for the set $\{[\![N]\!]^\nu_{\mathcal{F}} \mid N \in L\}$. So in particular, for a $\lambda Y{+}\Omega$-term M, we may write $[\![\mathcal{L}_{IO}(M)]\!]^\nu_{\mathcal{F}}$ or $[\![\mathcal{L}_{OI}(M)]\!]^\nu_{\mathcal{F}}$.

Monotone models of λY+Ω-calculus. Given two complete lattices L_1 and L_2, we write $\mathbf{Mon}[L_1 \to L_2]$ for the lattice of *monotonic functions* from L_1 to L_2 ordered pointwise; f is *monotonic* if $a \leq_1 b$ implies $f(a) \leq_2 f(b)$, and given f and g in L, $f \leq g$ whenever for every a in L_1, $f(a) \leq_2 g(a)$. It is worth noticing that for f, g in $L = \mathbf{Mon}[L_1 \to L_2]$ we have $(f \vee g)(a) = f(a) \vee_2 g(a)$ and $(f \wedge g)(a) = f(a) \wedge_2 g(a)$. Among the functions in $\mathbf{Mon}[L_1 \to L_2]$, of special interest are the step functions. A step function $a \mapsto b$ is determined from elements a in L_1 and b in L_2, and is defined such that $(a \mapsto b)(c)$ is equal to b when $a \leq_1 c$ and to $\bot_2$ otherwise. A monotone model $\mathcal{M}$ is a collection of finite lattices indexed by types, $(\mathcal{M}_\alpha)_{\alpha\in type}$ where $\mathcal{M}_{\alpha\to\beta} = \mathbf{Mon}[\mathcal{M}_\alpha \to \mathcal{M}_\beta]$ (we

write $\bot_\alpha$ and $\top_\alpha$ respectively for the least and greatest elements of $\mathcal{M}_\alpha$). The notion of valuation on monotone models is similar to the one on full models and we use the same notation. Terms are interpreted in monotone models according to the following scheme: $[\![x^\alpha]\!]^\nu_\mathcal{M} = \nu(x^\alpha)$, $[\![MN]\!]^\nu_\mathcal{M} = [\![M]\!]^\nu_\mathcal{M}([\![N]\!]^\nu_\mathcal{M})$ and for a in $\mathcal{M}_\alpha$, $[\![\lambda x^\alpha.M]\!]^\nu_\mathcal{M}(a) = [\![M]\!]^{\nu[x^\alpha \leftarrow a]}_\mathcal{M}$, $[\![\Omega^\alpha]\!]^\nu_\mathcal{M} = \bot_\alpha$, $[\![M+N]\!]^\nu_\mathcal{M} = [\![M]\!]^\nu_\mathcal{M} \vee [\![N]\!]^\nu_\mathcal{M}$ and for every a in $\mathcal{M}_{\alpha\to\alpha}$, $[\![Y]\!]^\nu_\mathcal{M}(a) = \bigvee\{a^n(\bot_\alpha) \mid n \in \mathbb{N}\}$.

The following Theorem gives well known results on monotone models (see [3]):

Theorem 5 *Given two $\lambda Y{+}\Omega$terms of type α, M and N, the following obtain for every monotone model $\mathcal{M}$ and valuation ν:*

1. *if $M =_{\beta\delta} N$ then $[\![M]\!]^\nu_\mathcal{M} = [\![N]\!]^\nu_\mathcal{M}$,*
2. *$[\![M]\!]^\nu_\mathcal{M} = \bigvee\{[\![Q]\!]^\nu_\mathcal{M} \mid Q$ is a finite approximation of $M\}$,*
3. *if $M \xrightarrow{*}_{\beta\delta+} N$ then $[\![N]\!]^\nu_\mathcal{M} \leq [\![M]\!]^\nu_\mathcal{M}$,*
4. *if $N \sqsubseteq M$ then $[\![N]\!]^\nu_\mathcal{M} \leq [\![M]\!]^\nu_\mathcal{M}$.*

3. Relations between IO, OI and full models

In this section, we investigate some basic properties of IO and OI languages. We will see that the class of higher-order OI languages strictly subsumes the class of higher-order IO languages. We will then see that the emptiness and membership problems for higher-order IO languages are decidable, by showing that for a higher-order grammar $\mathcal{G}$, a finite full model $\mathcal{F}$, and a valuation ν, the set $[\![\mathcal{L}_{IO}(\mathcal{G})]\!]^\nu_\mathcal{F}$ is effectively computable. On the other hand, we show that $[\![L]\!]^\nu_\mathcal{F}$ is *not* in general effectively computable when L is an OI language. This is done by a reduction to the λ-definability problem, which proceeds by showing that the set of closed λ-terms of type α is actually an OI language.

3.1. OI subsumes IO

We first start by showing that OI subsumes IO; for this given a higher-order grammar $\mathcal{G}$, we construct a grammar $\mathcal{G}'$ so that $\mathcal{L}_{IO}(\mathcal{G}) = \mathcal{L}_{OI}(\mathcal{G}')$. The construction of $\mathcal{G}'$ can be seen as a continuation passing style (CPS) transformation of $\mathcal{G}$. In the community on higher-order schemes, Haddad [14] showed that the IO and OI modes of evaluation on schemes define the same classes of infinite trees. Nevertheless, in the context of schemes, IO is understood in terms of a call-by-value strategy that is quite close to that used in programming languages: Haddad considers that a value is either a possibly infinite tree or a non-terminal that is not applied to all its possible arguments. If we were to present IO macro grammars in terms of call-by-value strategy, then the notion of a value would be that of a term that does not contain non-terminals, that is, a term whose evaluation is completely deterministic (non-determinism emerging from non-terminal rewriting when working with grammars). This difference

of interpretation of what IO means in the context of schemes and in that of higher-order grammars explains why our results, even though they might superficially look similar, are quite different. In particular, in contrast to the case of higher-order schemes where IO and OI coincide, we show that, in the context of higher-order grammars, OI strictly subsumes IO.

Theorem 6 (OI subsumes IO) *Given a higher-order grammar $\mathcal{G}$, one can construct a higher-order grammar $\mathcal{G}'$ such that $\mathcal{L}_{IO}(\mathcal{G}) = \mathcal{L}_{OI}(\mathcal{G}')$.*

Proof
Let us assume that $\mathcal{G} = (\Sigma, R, S)$. Before we begin the proof, we adopt the convention that when, for $A_1, \ldots, A_n$ a sequence of non-terminals, we write $M = P[x_1 \leftarrow A_1, \ldots, x_n \leftarrow A_n]$ (where $P[x_1 \leftarrow A_1, \ldots, x_n \leftarrow A_n]$ is the result of simultaneously substituting $A_1, \ldots, A_n$ for the free occurrences of $x_1, \ldots, x_n$ in P) for a λ-term M built on Σ, we implicitly assume that P is a λ-term that does not contain any non-terminal and also that the variables $x_1, \ldots, x_n$ have exactly one free occurrence in P.

We now define the higher-order macro grammar $\mathcal{G}' = (\Sigma', R', S)$ as follows: for every A^α in Σ, we let $A'^{\alpha'}$ be in Σ' such that $\alpha' = (\alpha \to 0) \to 0$, we also let S be in Σ'. Now for every rule $A \to P[x_1 \leftarrow A_1, \ldots, x_n \leftarrow A_n]$ in R, we let $A' \to \lambda k.A'_1(\lambda x_1 \ldots A'_n(\lambda x_n.kP)\ldots)$ be in R'; if S has type $\alpha_1 \to \cdots \to \alpha_n \to 0$, we also add to R' the rule $S \to \lambda x_1 \ldots x_n.S'(\lambda Q.Qx_1 \ldots x_n)$. It now remains to show that $\mathcal{L}_{IO}(\mathcal{G}) = \mathcal{L}_{OI}(\mathcal{G}')$.

We start by showing that $\mathcal{L}_{IO}(\mathcal{G}) \subseteq \mathcal{L}_{OI}(\mathcal{G}')$. For this, we show that, when $A \xrightarrow{*}_{\mathcal{G}} M$ with $M = P[x_1 \leftarrow A_1, \ldots, x_n \leftarrow A_n]$, then

$$A' \xrightarrow{*}_{\beta\mathcal{G}'} \lambda k.A'_{\sigma(1)}(\lambda x_{\sigma(1)} \ldots A'_{\sigma(n)}(\lambda x_{\sigma(n)}.kP)\ldots)$$

for some permutation σ of $[1, n]$. This is proved by induction on the reduction $A \xrightarrow{*}_{\mathcal{G}} M$. In case this reduction has length one the conclusion follows from the definition of $\mathcal{G}'$. Let us now assume the reduction has length $m = o + 1$ for $o > 0$. This implies that A can be rewritten in o steps into $N = Q[y_1 \leftarrow B_1, \ldots, y_r \leftarrow B_r]$, and that for some i in $[1, r]$, $B_i \to O$ is a rule of $\mathcal{G}$ such that M is obtained from N by rewriting B_i into O. Without loss of generality we assume that $i = 1$ and thus that $M = Q[y_1 \leftarrow O, y_2 \leftarrow B_2 \ldots, y_r \leftarrow B_r]$ and $O = O'[z_1 \leftarrow C_1, \ldots, z_q \leftarrow C_q]$. Therefore we have:

$$M = (Q[y_1 \leftarrow O'])[z_1 \leftarrow C_1, \ldots, z_q \leftarrow C_q, y_2 \leftarrow B_2 \ldots, y_r \leftarrow B_r]$$

By the induction hypothesis, A' can be rewritten in the OI mode of evaluation with the grammar $\mathcal{G}'$ to a term $Q' = \lambda k.B'_{\tau(1)}(\lambda y_{\tau(1)} \ldots B'_{\tau(r)}(\lambda y_{\tau(r)}.kQ))$. Without loss of generality we assume that $\tau(1) = 1$. By the definition of $\mathcal{G}'$, $B'_1 \to \lambda k'.C'_1(\lambda z_1 \ldots C'_q(\lambda z_q.k'O')\ldots)$ is in R'. Therefore Q' can be rewritten to

$$\lambda k.(\lambda k'.C'_1(\lambda z_1 \ldots C'_q(\lambda z_q.k'O')\ldots))(\lambda y_1.B'_{\tau(2)}(\lambda y_{\tau(2)} \ldots B'_{\tau(r)}(\lambda y_{\tau(r)}.kQ)\ldots))$$

which itself reduces to

$$\lambda k.C'_1(\lambda z_1.\dots C'_q(\lambda z_q.B'_{\tau(2)}(\lambda y_{\tau(2)}.\dots B'_{\tau(r)}(\lambda y_{\tau(r)}.kQ[y_1 \leftarrow O']))\dots))$$

This finally proves the claim.

Now we can use the claim so as to prove the inclusion. If N is in $\mathcal{L}_{IO}(\mathcal{G})$, it means that there is some P which does not contain any non-terminals (P is in G-normal form), such that $S \xrightarrow{*}_{\mathcal{G}} P$ and $P \xrightarrow{*}_{\beta} N$. Thus, using the previous claim, we have that $S' \xrightarrow{*}_{\beta\mathcal{G}'} \lambda k.kN$. But then

$$\begin{array}{lll} S & \xrightarrow{*}_{\mathcal{G}'} & \lambda x_1 \dots x_n.S'(\lambda Q.Qx_1 \dots x_n) \\ & \xrightarrow{*}_{\beta\mathcal{G}'} & \lambda x_1 \dots x_n.(\lambda k.kN)(\lambda Q.Qx_1 \dots x_n) \\ & \xrightarrow{*}_{\beta} & N \end{array}$$

so that N is in $\mathcal{L}_{OI}(\mathcal{G}')$.

Let us now prove the converse inclusion: $\mathcal{L}_{OI}(\mathcal{G}') \subseteq \mathcal{L}_{IO}(\mathcal{G})$. Here we use head-reduction as the evaluation strategy for OI. We prove that, under this reduction strategy, when a non-terminal A' can be rewritten in $3m+1$ steps into a term M that still contains a non-terminal, then M is of the form $\lambda k.A'_1(\lambda x_1.\dots A'_n(\lambda x_n.kN)\dots)$, and $A \xrightarrow{*}_{\mathcal{G}} N[x_1 \leftarrow A_1, \dots, x_n \leftarrow A_n]$. This claim can be proved by induction on m. The case where $m = 0$ is clear from the definition of $\mathcal{G}'$. If $m = o + 1$, then by the induction hypothesis A' can be rewritten in $3o + 1$ steps into $\lambda k.A'_1(\lambda x_1.\dots A'_n(\lambda x_n.kN)\dots)$ and $A \xrightarrow{*}_{\mathcal{G}} N[x_1 \leftarrow A_1, \dots, x_n \leftarrow A_n]$. Now there must be a rule $A'_1 \rightarrow \lambda k'.B'_1(\lambda z_1.\dots B'_p(\lambda z_p.k'P)\dots)$ in R' so that:

$$\begin{array}{l} \lambda k.A'_1(\lambda x_1.A'_2(\lambda x_2.\dots A'_n(\lambda x_n.kN)\dots)) \\ \quad \rightarrow_{\mathcal{G}'} \lambda k.(\lambda k'.B'_1(\lambda z_1.\dots B'_p(\lambda z_p.k'P)\dots))(\lambda x_1.A'_2(\lambda x_2.\dots A'_n(\lambda x_n.kN)\dots)) \\ \quad \rightarrow_{\beta} \lambda k.B'_1(\lambda z_1.\dots B'_p(\lambda z_p.(\lambda x_1.A'_2(\lambda x_2.\dots A'_n(\lambda x_n.kN)\dots))P)\dots) \\ \quad \rightarrow_{\beta} \lambda k.B'_1(\lambda z_1.\dots B'_p(\lambda z_p.A'_2(\lambda x_2.\dots A'_n(\lambda x_n.kN[x_1 \leftarrow P])\dots))\dots) \end{array}$$

But since $A'_1 \rightarrow \lambda k'.B'_1(\lambda z_1.\dots B'_p(\lambda z_p.k'P))$ is in R' we must have $A_1 \rightarrow P[z_1 \leftarrow B_1, \dots, z_p \leftarrow B_p]$ in R and thus $N[x_1 \leftarrow A_1, \dots, x_n \leftarrow A_n]$ can be rewritten into $N[x_1 \leftarrow P[z_1 \leftarrow B_1, \dots, z_p \leftarrow B_p], x_2 \leftarrow A_2, \dots, x_n \leftarrow A_n]$ that is into $(N[x_1 \leftarrow P])[z_1 \leftarrow B_1, \dots, z_p \leftarrow B_p, x_2 \leftarrow A_2, \dots, x_n \leftarrow A_n]$ and thus the claim is proved by induction.

With the claim we easily obtain that when m is the smallest number such that A' is rewritten into a term M which does not contain any non-terminal in $3m+1$ steps, then $M = \lambda k.kP$ and $A \xrightarrow{*}_{\mathcal{G}} P$. This implies that, if $\mathcal{L}_{IO}(A)$ is the language of terms that A defines in $\mathcal{G}$ in the IO mode of derivation and $\mathcal{L}_{OI}(A')$ is the language of terms that A' defines in $\mathcal{G}'$ in the OI mode of derivation, $\lambda k.kN$ is in $\mathcal{L}_{OI}(A')$ implies that N is in $\mathcal{L}_{IO}(A)$. The fact that $\mathcal{L}_{OI}(\mathcal{G}')$ is included into $\mathcal{L}_{IO}(\mathcal{G})$ is an immediate consequence. □

Regarding the proof of the previous theorem, several remarks need to be made. First, the CPS transform naturally makes the order of $\mathcal{G}'$ be two more

than the order of $\mathcal{G}$. Second, even if $\mathcal{G}$ is safe, the grammar $\mathcal{G}'$ constructed in the proof is not necessarily so. Indeed, consider the safe rule $A \to aAA$ where A is a non-terminal of type 0. The new rule $A' \to \lambda k.A'(\lambda x.A'(\lambda y.kaxy))$ is not safe since the term of type $0 \to 0$ of order 2, $(\lambda y.kaxy)$, contains a free variable x of type 0 that is of order 1 and is in argument position. Thus, Theorem 6 only applies to unsafe grammars and it is unclear even whether the safe OI languages contain the safe IO languages. Finally Fischer [13] has shown that the classes of languages defined by macro grammars in the OI and IO modes of derivation are incomparable. We conjecture that for a fixed order n, the classes of languages IO_n and OI_n are incomparable.

3.2. Decidability results for IO

We now show that for a full model $\mathcal{F}$, a valuation ν and a given grammar $\mathcal{G}$, the set $[\![\mathcal{L}_{IO}(\mathcal{G})]\!]^{\nu}_{\mathcal{F}}$ can be effectively computed. A natural consequence of this is that the emptiness and the membership problems for higher-order IO languages are decidable. These results are known in the literature [30, 31, 32], nevertheless, we include them here so as to emphasize that they are related to the effectivity of the set $[\![\mathcal{L}_{IO}(\mathcal{G})]\!]^{\nu}_{\mathcal{F}}$, a property that, as we will see later, does not hold in the case of OI languages. Also, as one can note, the decidability results for emptiness and membership for IO can be derived from the one we shall prove later on for OI by using Theorem 6. However, we think that the decidability of the set $[\![\mathcal{L}_{IO}(\mathcal{G})]\!]^{\nu}_{\mathcal{F}}$ and the undecidability of the set $[\![\mathcal{L}_{OI}(\mathcal{G})]\!]^{\nu}_{\mathcal{F}}$ underline a key difference between IO and OI, and that the decidability of $[\![\mathcal{L}_{IO}(\mathcal{G})]\!]^{\nu}_{\mathcal{F}}$ gives a much simpler route to prove decidability results for IO.

Theorem 7 (Effective finite interpretation of IO) *Given a higher-order macro grammar $\mathcal{G}$, a full model $\mathcal{F}$ and a valuation ν, one can effectively construct the set $[\![\mathcal{L}_{IO}(\mathcal{G})]\!]^{\nu}_{\mathcal{F}}$.*

Proof

Since this theorem is some reformulation of results that already appeared in the literature, we simply sketch its proof.

The idea behind this theorem is reminiscent of the usual techniques used when facing some problem related to context free grammars. Basically the proof of this theorem consists in computing a least fixpoint.

For this we assume that $\mathcal{G} = (\Sigma, R, S)$ and that $\mathcal{F} = (\mathcal{F}_\alpha)_{\alpha \in type}$. We define the family $\mathcal{P} = (\mathcal{P}_\alpha)_{\alpha \in type}$ so that $\mathcal{P}_\alpha = 2^{\mathcal{F}_\alpha}$. So as to give an interpretation to terms that contain non-terminals, we adopt the same convention as in the proof of Theorem 6: we write $M = P[x_1 \leftarrow A_1, \ldots, x_n \leftarrow A_n]$ to mean that M contains n occurrences of non-terminals each of them materialized in P by the variables $x_1, \ldots, x_n$ that each occur exactly once. Now if we let ξ be a type consistent mapping of non-terminals to elements of $\mathcal{P}$, we say that a valuation ρ is compatible with ν and ξ relative to P written $\rho \in_P (\nu, \xi)$, when, for every y, $\rho(y) = \nu(y)$ and for every i in $[1, n]$, $\rho(x_i) \in \xi(A_i)$; we then let $[\![M]\!]^{\nu,\xi}_{\mathcal{P}}$ be the set $\{[\![P]\!]^{\rho}_{\mathcal{F}} \mid \rho \in_P (\nu, \xi)\}$.

We then use this interpretation to compute the least valuation ξ so that for every non-terminal A, if the rules with left-hand side A are $A \to M_1$, ..., $A \to M_n$, then $\xi(A) = \bigcup_{i\in[1,n]} [\![M_i]\!]^{\nu,\xi}_{\mathcal{P}}$. For this, it suffices to let ξ_0 be the valuation of non-terminals so that for every non-terminal A, $\xi_0(A) = \emptyset$, and then let $\xi_{k+1}(A) = \bigcup_{i\in[1,n]} [\![M_i]\!]^{\nu,\xi_k}_{\mathcal{P}}$. It is then easy to see that for every k, and every A, $\xi_k(A) \subseteq \xi_{k+1}(A)$, which guarantees the existence of the least valuation ξ. Then the usual techniques show that an element f of $\mathcal{M}$ is in $\xi(A)$ iff there are P and M respectively in $\mathcal{G}$-normal form and in $\beta\mathcal{G}$-normal form so that $A \xrightarrow{*}_{\mathcal{G}} P \xrightarrow{*}_{\beta} M$ and $[\![M]\!]^{\nu}_{\mathcal{F}} = f$. □

Corollary 8 Given a higher-order macro grammar $\mathcal{G}$, the problem of deciding whether $\mathcal{L}_{IO}(\mathcal{G}) = \emptyset$ is P-complete.

Proof
The P-hardness of the problem comes from the fact that it subsumes the problem of the emptiness of a context-free language which is P-complete. The fact that it is in P comes from that if we take the full model $\mathcal{S} = (\mathcal{S}_\alpha)_{\alpha\in type}$ so that $\mathcal{S}_0$ is a singleton set, then for every $\alpha \in type$, $\mathcal{S}_\alpha$ is a singleton set. Then $\mathcal{T}_\alpha = 2^{\mathcal{S}_\alpha}$ is a two elements set and computing the interpretation of a term as explained in the proof of Theorem 7 is then linear in the size of the term, so that computing a fixpoint of a grammar in $(\mathcal{T}_\alpha)_{\alpha\in type}$ is linear in the number of non-terminals and in the size of the terms involved in the rules of the grammar. □

Corollary 9 Given a higher-order macro grammar $\mathcal{G}$ and a term M, it is decidable whether $M \in \mathcal{L}_{IO}(\mathcal{G})$.

Proof
From the finite completeness Theorem, we know that there is a full model $\mathcal{F}_M$ and a valuation ν so that for every term N, $[\![N]\!]^{\nu}_{\mathcal{F}_M} = [\![M]\!]^{\nu}_{\mathcal{F}_M}$ iff $N =_{\beta\eta} M$. It then follows that M is in $\mathcal{L}_{IO}(\mathcal{G})$ iff $[\![M]\!]^{\nu}_{\mathcal{F}_M}$ is in $[\![\mathcal{L}_{IO}(\mathcal{G})]\!]^{\nu}_{\mathcal{F}_M}$. Since from Theorem 7 the set $[\![\mathcal{L}_{IO}(\mathcal{G})]\!]^{\nu}_{\mathcal{F}_M}$ can effectively be computed, and since the constructions of $\mathcal{F}_M$ and ν are also effective, it follows that we can decide whether $[\![M]\!]^{\nu}_{\mathcal{F}_M}$ is in $[\![\mathcal{L}_{IO}(\mathcal{G})]\!]^{\nu}_{\mathcal{F}_M}$. □

3.3. OI generates all closed terms of a given type

We will now see that the set of all closed λ-terms of a given type α is an OI language. Combined with Theorem 4, it follows that the set $[\![\mathcal{L}_{OI}(\mathcal{G})]\!]_{\mathcal{F}}$ cannot be effectively computed. Moreover, Theorems 6 and 7 imply that the class of IO languages is strictly included in that of OI languages.

Theorem 10 *For every type α, there is a higher-order macro grammar $\mathcal{G}_\alpha$ such that $\mathcal{L}_{OI}(\mathcal{G}_\alpha)$ is the set of all closed λ-terms of type α in β-normal form.*

Proof
Fix α. We are going to build a grammar $\mathcal{G}_\alpha$ so that $\mathcal{L}_{OI}(\mathcal{G}_\alpha)$ is the set of closed λ-terms of type α. For this we let $\mathcal{T}_\alpha$ be the *finite* set of types which are subformulae of α, *i.e.* the types that have a syntactic occurrence within α. We assume a total order on $\mathcal{T}_\alpha$ so that when we deal with a subset $S = \{\alpha_1, \ldots, \alpha_n\}$ of $\mathcal{T}_\alpha$, we take the ordering of the elements to be $\alpha_1, \ldots, \alpha_n$. We define the non-terminals of $\mathcal{G}_\alpha$ to be either pairs $\langle \gamma, S\rangle$ where γ is in $\mathcal{T}_\alpha$ and $S \subseteq \mathcal{T}_\alpha$, or to be of the form $\mathbf{cons}_\gamma$. The finiteness of $\mathcal{T}_\alpha$ guaranties that the set of non-terminals of $\mathcal{G}_\alpha$ is finite. The type of the non-terminals $\langle \gamma, S\rangle$ is $\alpha_1 \to \cdots \to \alpha_n \to \gamma$ when $S = \{\alpha_1, \ldots, \alpha_n\}$ (it is γ when $S = \emptyset$; and otherwise the order of the α_i is fixed according to the order of $\mathcal{T}_\alpha$), and the type of $\mathbf{cons}_\gamma$ is $\gamma \to \gamma \to \gamma$. The non-terminals $\mathbf{cons}_\gamma$ will serve to construct non-empty finite sets of variables of type γ; they allow the addition of a variable to such a set. For example the set of variables $\{x^\gamma, y^\gamma, z^\gamma\}$ will be represented as $\mathbf{cons}_\gamma x^\gamma(\mathbf{cons}_\gamma y^\gamma z^\gamma)$. The grammar will reduce such a term non-deterministically to one of the variables in the set it represents. The non-terminals of the form $\langle \gamma, S\rangle$, where $S = \{\alpha_1, \ldots, \alpha_n\}$, will always be applied to terms representing non-empty sets of variables of type $\alpha_1, \ldots, \alpha_n$. If $T_{V_1}, \ldots, T_{V_n}$ are terms representing respectively the finite set V_1 of variables of type $\alpha_1, \ldots$, and the finite set V_n of variables of type α_n, the grammar will rewrite the term $\langle \gamma, S\rangle T_{V_1} \ldots T_{V_n}$ into any term of type γ whose free variables are in the set $V_1 \cup \cdots \cup V_n$. In particular, closed terms of type γ will be generated by the non-terminal $\langle \gamma, \emptyset\rangle$.

We now describe the rules of $\mathcal{G}_\alpha$. For the non-terminals $\langle \gamma, S\rangle$, we assume that $S = \{\alpha_1, \ldots, \alpha_n\}$. The rules of $\mathcal{G}_\alpha$ are given by:

1. $\mathbf{cons}_\gamma \to \lambda xy.x$ and $\mathbf{cons}_\gamma \to \lambda xy.y$ are rules of $\mathcal{G}_\alpha$. These two rules implement the non-deterministic choice of the grammar concerning a set of variables,

2. in case $\gamma = \gamma_1 \to \gamma_2$ and $\gamma_1 = \alpha_i$ for some $i \leq n$, then

$$\langle \gamma, S\rangle \to \lambda y_1^{\alpha_1} \ldots y_n^{\alpha_n} x^{\alpha_i}.\langle \gamma_2, S\rangle y_1 \ldots y_{i-1}(\mathbf{cons}_{\alpha_i}\, x\, y_i) y_{i+1} \ldots y_n$$

   is a rule of $\mathcal{G}_\alpha$. This rule constructs a λ-abstraction that introduces a fresh variable x of type α_i; the variable x is added to the set of variables of type α_i; while the non-terminal $\langle \gamma_2, S\rangle$ is used to construct a term of type γ_2 with the updated sets of variables.

3. in case $\gamma = \gamma_1 \to \gamma_2$, $\gamma_1 \notin S$ and γ_1 appears in between α_i and α_{i+1} in the order of $\mathcal{T}_\alpha$, then

$$\langle \gamma, S\rangle \to \lambda y_1^{\alpha_1} \ldots y_n^{\alpha_n} x^{\gamma_1}.\langle \gamma_2, S \cup \{\gamma_1\}\rangle y_1 \ldots y_i x y_{i+1} \ldots y_n$$

   is a rule of $\mathcal{G}_\alpha$. This rule is similar to the previous one, except that γ_1 is not in S so that we initiate a new set of variables of type γ_1.

4. in case $\gamma = 0$ and $S \neq \emptyset$, if $\alpha_i = \beta_1 \to \cdots \to \beta_p \to 0$ then

$$\langle \gamma, S\rangle \to \lambda y_1^{\alpha_1} \ldots y_n^{\alpha_n}.y_i(\langle \beta_1, S\rangle y_1 \ldots y_n) \ldots (\langle \beta_p, S\rangle y_1 \ldots y_n)$$

is a rule of $\mathcal{G}_\alpha$. This rule chooses one of the variables in the set of variables of type α_i represented by the variable y_i, and then inductively constructs the arguments of the right types for that variable.

It is rather easy to show that the non-terminals behave as we explained above. As a consequence the non-terminal $\langle \alpha, \emptyset \rangle$ generates all the closed terms of type α. □

Theorem 11 (Undecidable finite interpretation of OI) *Given a higher-order macro grammar $\mathcal{G}$, a finite full model $\mathcal{F}$, and f an element of $\mathcal{F}$, it is undecidable whether $f \in [\![\mathcal{L}_{OI}(\mathcal{G})]\!]$.*

Proof
This is a direct consequence of Theorems 10 and 4. □

Theorem 12 *The class of higher-order IO language is strictly included in the class of higher-order OI languages.*

Proof
If there were an IO grammar that could define the set of closed terms of type α, Theorem 7 would contradict Theorem 4. □

4. Emptiness and membership for the OI hierarchy

In this section we prove the decidability of the emptiness and membership problems for higher-order OI languages. For this we use monotone models as approximations of sets of elements of full models.

4.1. Hereditary primality and the Observability Theorem

Theorem 11 implies that the decision techniques we used for the emptiness and the membership problems for IO do not extend to OI. So as to show that those problems are nevertheless decidable, we are going to prove a theorem that we call the *Observability Theorem*, which allows us to *observe* certain semantic properties of λ-terms in the OI language of a $\lambda Y{+}\Omega$-term M by means of the semantic values of M in monotone models. For this we introduce the notion of *hereditary prime elements* of a monotone model.

Definition 13 In a lattice $\mathcal{L}$, an element f is *prime* (or $\vee$*-prime*) when for every g_1 and g_2 in $\mathcal{L}$, $f \leq g_1 \vee g_2$ implies that $f \leq g_1$ or $f \leq g_2$.

Given a monotone model $\mathcal{M} = (\mathcal{M}_\alpha)_{\alpha \in type}$, for every type α we define the sets $\mathcal{M}^+_\alpha$ and $\mathcal{M}^-_\alpha$ by:

1. $\mathcal{M}^+_0$ and $\mathcal{M}^-_0$ contain the prime elements of $\mathcal{M}_0$ that are different from $\bot_0$,
2. $\mathcal{M}^+_{\alpha\to\beta} = \{(\bigvee F) \mapsto g \mid F \subseteq \mathcal{M}^-_\alpha \wedge g \in \mathcal{M}^+_\beta\}$,

3. $\mathcal{M}^-_{\alpha\to\beta} = \{f \mapsto g \mid f \in \mathcal{M}^+_\alpha \wedge g \in \mathcal{M}^-_\beta\}$.

A valuation ν on $\mathcal{M}$ is said *hereditary prime* when, for every variable x^α, $\nu(x^\alpha) = \bigvee F$ for some $F \subseteq \mathcal{M}^-_\alpha$. The elements of $\mathcal{M}^+_\alpha$ are called the *hereditary prime elements* of $\mathcal{M}_\alpha$.[1]

The main interest of primality lies in that, if f is prime and $f \leq [\![M+N]\!]^\nu_{\mathcal{M}}$, then either $f \leq [\![M]\!]^\nu_{\mathcal{M}}$ or $f \leq [\![N]\!]^\nu_{\mathcal{M}}$. This can be interpreted as the fact that either there is an element in $\mathcal{L}_{OI}(M)$ whose semantics is greater than f or there is one in $\mathcal{L}_{OI}(N)$ whose semantics is greater than f. The difficulty that is solved by hereditary prime elements is how to transfer this property to the whole hierarchy of types while remaining compatible with all the constructs of $\lambda Y{+}\Omega$-terms. Hereditary prime elements are reminiscent of finite state automata for λ-terms, while other elements in $\mathcal{M}^\alpha$ are more similar to alternating finite state automata. The elements in $\mathcal{M}^-_\alpha$ can be thought of as representing transitions; to a variable x of type α, we associate a set of transitions F, in a manner similar to finite state automata which may have several transitions associated to a given symbol. The elements of $\mathcal{M}^+_\alpha$ can then be seen as the states of the automaton. Then, when we want to check whether $\lambda x.M$ has a semantics greater than $\bigvee F \mapsto g$ in $\mathcal{M}^+_{\alpha\to\beta}$, this amounts to checking whether M has a semantics greater than g while associating the transitions in F to x. The proof of the following technical Lemma, from which we derive the Observability Theorem, can be seen as constructing a run of this kind of automaton on one of the λ-terms in the language defined by a $\lambda{+}\Omega$-term. The role that is fulfilled by primality is to ensure that there is a run in at least one of the arguments of the $+$.

Lemma 14 Given a $\lambda{+}\Omega$-term M^α, a monotone model $\mathcal{M} = (\mathcal{M}_\gamma)_{\gamma\in type}$, a hereditary prime valuation ν and a hereditary prime element f of $\mathcal{M}_\alpha$, we have the equivalence:

$$f \leq [\![M^\alpha]\!]^\nu_{\mathcal{M}} \Leftrightarrow \exists N \in \mathcal{L}_{OI}(M).f \leq [\![N]\!]^\nu_{\mathcal{M}}$$

Proof
The direction from right to left is a simple consequence of Theorem 5.3.

Let us now prove the other direction. We assume that M^α is in β-normal and η-long form. We then proceed by induction on the structure of M.

In case $M = hM^{\gamma_1}_1 \dots M^{\gamma_n}_n$, we have $\alpha = 0$, and since f is strictly greater than $\perp$, and $f \leq [\![M]\!]^\nu_{\mathcal{M}}$ we cannot have $h = \Omega^\gamma$. Thus h must be a variable x^γ with $\gamma = \gamma_1 \to \dots \to \gamma_n \to 0$. Since ν is hereditary prime there is $G \subseteq \mathcal{M}^-_\gamma$ such that $\nu(x) = \bigvee G$. Since $f \leq [\![M]\!]^\nu_{\mathcal{M}}$ and f is prime, there must be g in G such that $f \leq g[\![M_1]\!]^\nu_{\mathcal{M}} \dots [\![M_n]\!]^\nu_{\mathcal{M}} = g'$. But as g is in $\mathcal{M}^-_\gamma$, $g = g_1 \mapsto \dots \mapsto g_n \mapsto g'$ with $g_1, \dots, g_n$ respectively in $\mathcal{M}^+_{\gamma_1}, \dots, \mathcal{M}^+_{\gamma_n}$. Thus, by induction, for every

[1] As is usual, we assume that, when $F \subseteq \mathcal{M}^+_\alpha$ is such that $F = \emptyset$, then $\bigvee F = \perp_\alpha$.

i in $[1,n]$, there is N_i in $\mathcal{L}_{OI}(M_i)$ such that $g_i \leq [\![N_i]\!]^{\nu}_{\mathcal{M}}$. We then have that $N = xN_1 \dots N_n$ is in $\mathcal{L}_{OI}(M)$ moreover $g' \leq [\![N]\!]^{\nu}_{\mathcal{M}}$ such that $f \leq [\![N]\!]^{\nu}_{\mathcal{M}}$.

In case $M = \lambda x^{\alpha_1}.P^{\alpha_2}$, we have $f = f_1 \mapsto f_2$, where $f_1 = \bigvee F$ and $F \subseteq \mathcal{M}^-_{\alpha_1}$ and f_2 is in $\mathcal{M}^+_{\alpha_2}$. As $f \leq [\![M]\!]^{\nu}_{\mathcal{M}}$, we have that $f_2 \leq [\![P]\!]^{\nu[x^{\alpha_1} \leftarrow f_1]}_{\mathcal{M}}$. By induction hypothesis, there is Q in $\mathcal{L}_{OI}(P)$ such that $f_2 \leq [\![Q]\!]^{\nu[x^{\alpha_1} \leftarrow f_1]}_{\mathcal{M}}$. Thus, letting $N = \lambda x^{\alpha_1}.Q$, we have that $f \leq [\![N]\!]^{\nu}_{\mathcal{M}}$ and N in $\mathcal{L}_{OI}(M)$.

In case $M = M_1 + M_2$, we have $[\![M]\!]^{\nu}_{\mathcal{M}} = [\![M_1]\!]^{\nu}_{\mathcal{M}} \vee [\![M_2]\!]^{\nu}_{\mathcal{M}}$. Since f is in $\mathcal{M}^+_0$, f is prime and thus either $f \leq [\![M_1]\!]^{\nu}_{\mathcal{M}}$ or $f \leq [\![M_2]\!]^{\nu}_{\mathcal{M}}$. Let us assume, without loss of generality, that $f \leq [\![M_1]\!]^{\nu}_{\mathcal{M}}$. By induction hypothesis, there is N in $\mathcal{L}_{OI}(M_1)$ such that $f \leq [\![N]\!]^{\nu}_{\mathcal{M}}$. The conclusion follows from the fact that N is an element of $\mathcal{L}_{OI}(M)$. □

Theorem 5 allows to extend Lemma 14 to $\lambda Y{+}\Omega$-terms.

Theorem 15 (Observability) *Given a $\lambda Y{+}\Omega$-term M, a monotone model $\mathcal{M} = (\mathcal{M}_\alpha)_{\alpha \in type}$, a hereditary prime valuation ν and a hereditary prime element f of $\mathcal{M}_\alpha$, we have the equivalence:*

$$f \leq [\![M^\alpha]\!]^{\nu}_{\mathcal{M}} \Leftrightarrow \exists N \in \mathcal{L}_{OI}(M).f \leq [\![N]\!]^{\nu}_{\mathcal{M}}$$

Proof
Since for every α, $\mathcal{M}_\alpha$ is finite, according to Theorem 5.2 (and the fact that the set of finite approximations of M is directed for the partial order $\sqsubseteq$), there is a finite approximation Q of M such that $f \leq [\![Q^\alpha]\!]^{\nu}_{\mathcal{M}}$. But then Q is a $\lambda{+}\Omega$-term and by the previous Lemma this is equivalent to there being some N in $\mathcal{L}_{OI}(Q)$ such that $f \leq [\![N]\!]^{\nu}_{\mathcal{M}}$. The conclusion follows from the fact that obviously $\mathcal{L}_{OI}(Q) \subseteq \mathcal{L}_{OI}(M)$. The other direction follows from Theorem 5.3. □

4.2. Decidability results

We are now going to use the Observability Theorem so as to prove the decidability of both the emptiness and the membership problems for OI languages.

Decidability of emptiness. We consider the monotone model $\mathcal{E} = (\mathcal{E}_\alpha)_{\alpha \in type}$ such that $\mathcal{E}_0$ is the lattice with two elements $\{\top, \bot\}$ such that $\bot \leq \top$. We then define for every α, the element $\mathbf{e}_\alpha$ of $\mathcal{E}^+_\alpha \cap \mathcal{E}^-_\alpha$ such that: $\mathbf{e}_0 = \top$, and $\mathbf{e}_{\alpha\to\beta} = \mathbf{e}_\alpha \mapsto \mathbf{e}_\beta$. We let ξ be the valuation such that for each variable x^α, $\xi(x^\alpha) = \mathbf{e}_\alpha$. We first prove that every λ-term of type α has an interpretation that is greater than $\mathbf{e}_\alpha$ in $\mathcal{E}$ with the valuation ξ.

Lemma 16 For every λ-term M of type γ, $\mathbf{e}_\gamma \leq [\![M]\!]^{\xi}_{\mathcal{E}}$.

Proof
This is a simple induction on the structure of M. □

We are now in position to use the Observability Theorem to prove a characterization of terms that define a non-empty OI language.

Proposition 1 *Given a $\lambda Y{+}\Omega$-term M of type α, we have that*

$$\mathcal{L}_{OI}(M) \neq \emptyset \Leftrightarrow \mathbf{e}_\alpha \leq [\![M]\!]^\xi_{\mathcal{E}}$$

Proof
If $\mathcal{L}_{OI}(M) \neq \emptyset$, then there is N in normal form such that $M \xrightarrow{*}_{\beta\delta+} N$. Lemma 16 implies that $\mathbf{e}_\alpha \leq [\![N]\!]^\xi_{\mathcal{E}}$ and thus, using Theorem 5, $\mathbf{e}_\alpha \leq [\![M]\!]^\xi_{\mathcal{E}}$. If $\mathbf{e}_\alpha \leq [\![M]\!]^\xi_{\mathcal{E}}$, since $\mathbf{e}_\alpha$ is in $\mathcal{E}^+_\alpha$, from Theorem 15, there is N in $\mathcal{L}_{OI}(M)$ such that $\mathbf{e}_\alpha \leq [\![N]\!]^\xi_{\mathcal{E}}$; so in particular that $\mathcal{L}_{OI}(M) \neq \emptyset$. □

The decidability of the emptiness problem for OI languages follows from the fact that we can effectively compute interpretations of terms in monotone models and in particular we can test whether the interpretation of a term is greater than a given element of the model.

Theorem 17 *The emptiness problem for OI languages is decidable.*

Decidability of membership. For the decidability of the membership problem, we are going to prove a stronger version of Theorem 3 (Statman's finite completeness Theorem). The proof technique we use together with bringing a generalization of Statman's Theorem, is also simplifying its argument. In particular, logical relations (see [3]) play a significant role in the simplification. Most of our effort deals with properties of models by means of logical relations. If we take the view of recognizability adopted in [30] with respect to finite models of the λ-calculus, Statman's finite completeness Theorem shows that singleton sets of λ-terms are recognizable. From that point of view, our proof can be best understood as a generalization from tree to λ-terms that singleton sets are recognizable. Statman's Theorem has already received a lot of attention in the literature and has been given various proofs [35, 36, 34] and it has been proven using intersection types in [30]. We here give a proof of a stronger statement that allows us to decide the membership problem for higher-order OI languages.

Given a finite set A, we write $\mathcal{M}(A) = (\mathcal{M}_\alpha(A))_{\alpha \in type}$ for the monotone model such that $\mathcal{M}_0(A)$ is the lattice of subsets of A ordered by inclusion. We let $\bot_{A,\alpha}$ be the least element of $\mathcal{M}_\alpha(A)$.

Definition 18 Given a λ-term M of type α, a triple $T = (A, \nu, f)$, where A is a finite set, ν is a valuation on $\mathcal{M}(A)$ and f is an element of $\mathcal{M}_\alpha(A)$, is *characteristic* of M when:

1. for every λ-term N of type α, $M =_\beta N$ iff $f \leq [\![N]\!]^\nu_{\mathcal{M}(A)}$,
2. f is a hereditary prime element of $\mathcal{M}_\alpha(A)$ and ν is a hereditary prime valuation.

The stronger form of Statman's finite completeness theorem is formulated as:

Theorem 19 (Monotone finite completeness) *For every type α and λ-term M of type α, one can effectively construct a triple T that is characteristic of M.*

We here start proving Theorem 19. For this we first need to introduce the notion of logical relation.

Given two monotone models $\mathcal{M} = (\mathcal{M}_\alpha)_{\alpha \in type}$ and $\mathcal{N} = (\mathcal{N}_\alpha)_{\alpha \in type}$, a logical relation $\mathcal{R}$ between $\mathcal{M}$ and $\mathcal{N}$ is a family of relations $(\mathcal{R}_\alpha)_{\alpha \in type}$, such that $\mathcal{R}_\alpha \subseteq \mathcal{M}_\alpha \times \mathcal{N}_\alpha$ and $\mathcal{R}_{\alpha \to \beta} = \{(f, g) \in \mathcal{M}_{\alpha \to \beta} \times \mathcal{N}_{\alpha \to \beta} \mid \forall (f', g') \in \mathcal{R}_\alpha . (f(f'), g(g')) \in \mathcal{R}_\beta\}$. Note that a logical relation $\mathcal{R}$ is completely determined by $\mathcal{R}_0$. In general, when the type is irrelevant we write $f \ R \ g$ to mean that $(f, g) \in \mathcal{R}_\alpha$ for some appropriate α. Given two valuations ν and μ, we write $\nu \ \mathcal{R} \ \mu$ when for all variables x, $\nu(x) \ \mathcal{R} \ \mu(x)$. Logical relations satisfy the following property:

Lemma 20 (Fundamental Lemma) Given two monotone models $\mathcal{M}$ and $\mathcal{N}$, a logical relation $\mathcal{R}$ between $\mathcal{M}$ and $\mathcal{N}$, and two valuations ν and μ over $\mathcal{M}$ and $\mathcal{N}$ respectively, such that $\nu \ \mathcal{R} \ \mu$, then for every $\lambda Y{+}\Omega$-term M we have $[\![M]\!]^\nu_{\mathcal{M}} \ \mathcal{R} \ [\![M]\!]^\mu_{\mathcal{N}}$.

We now turn to the proof of Theorem 19 itself. The idea is to prove it by induction on the structure of M, using a proof that is very similar to the proof one would use to prove that there is a finite state tree automaton which recognizes a unique given tree. Let us quickly have a look at this proof: if we need to prove that there is an automaton recognizing the tree $f(t_1, \ldots, t_n)$ we first construct, by induction, n automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ that respectively recognize the trees $t_1, \ldots, t_n$ with states $q_1, \ldots, q_n$. Assuming that $\mathcal{A}_1, \ldots, \mathcal{A}_n$ have disjoint sets of states we construct an automaton $\mathcal{A}$ that is the union of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ with an additional state q and an additional transition $f q_1 \ldots q_n \to q$. It is then easy to prove that $\mathcal{A}$ recognizes exactly the tree $f(t_1, \ldots, t_n)$. Crucially, in proving this, one relies on the fact that any term that is recognized by $\mathcal{A}$ in state q_i must be equal to t_i. This property, which follows obviously from the construction of $\mathcal{A}$, is precisely the one that is the source of difficulty for proving Theorem 19. Indeed, in the induction the only difficult case is the case where $M = x M_1 \ldots M_n$, we then construct by induction the characteristic triples $(A_1, \nu_1, f_1), \ldots, (A_n, \nu_n, f_n)$ of $M_1, \ldots, M_n$. Then, assuming that $A_1, \ldots, A_n$ are pairwise disjoint and taking an fresh element b, we try to construct a triple for the set $A = A_1 \cup \cdots \cup A_n \cup \{b\}$. This requires being able to retrieve the properties we had in each of the monotone models $\mathcal{M}(A_i)$ inside the monotone model $\mathcal{M}(A)$ and, in particular, to be able to find elements in $\mathcal{M}(A)$ that characterize the terms $M_1, \ldots, M_n$. This is where the logical relations play a central role.

As a preliminary for the proof of Theorem 19, we study how to embed the model $\mathcal{M}(A)$ into $\mathcal{M}(B)$ when $A \subseteq B$. So, given A and B two finite sets such that $A \subseteq B$, we define $\mathcal{I}_{A,B} = (\mathcal{I}_{A,B,\alpha})_{\alpha \in type}$ to be the logical relation between $\mathcal{M}(A)$ and $\mathcal{M}(B)$ such that $\mathcal{I}_{A,B,0} = \{(C \cap A, C) \mid C \subseteq B\}$. We now define a function $E_{A,B}$ that maps every element f of $\mathcal{M}_\alpha(A)$ to an element $E_{A,B}(f)$ of $\mathcal{M}_\alpha(B)$, where:

- $E_{A,B}(f) = f$ when $\alpha = 0$,
- $E_{A,B}(f) = \bigvee\{E_{A,B}(g) \mapsto E_{A,B}(f(g)) \mid g \in \mathcal{M}_\beta(A)\}$ when $\alpha = \beta \to \gamma$.

The idea is that when (f, g) is in $I_{A,B,\alpha}$, then f represents *the restriction* of g to $\mathcal{M}(A)$. For a fixed f in $\mathcal{M}(A)$, the set $\{g \in \mathcal{M}(B) \mid (f, g) \in I_{A,B,\alpha}\}$ is the set of *extensions* of f to $\mathcal{M}(B)$. Among those extensions $E_{A,B}(f)$ is the least one. All these facts come from the following lemma:

Lemma 21 Given a type α we have the following properties:

1. for every f in $\mathcal{M}_\alpha(A)$, $(f, E_{A,B}(f))$ is in $\mathcal{I}_{A,B,\alpha}$,
2. for every (f, g) in $\mathcal{I}_{A,B,\alpha}$, for every h in $\mathcal{M}_\alpha(A)$, we have $E_{A,B}(h) \leq g$ iff $h \leq f$.

Indeed this lemma allows us to see:

1. if (f, g) is in $I_{A,B,\alpha}$ then $E_{A,B}(f) \leq g$. We indeed have from the lemma that $(f, E_{A,B}(f))$ is in $I_{A,B,\alpha}$ and $f \leq f$ so that we obtain $E_{A,B}(f) \leq g$ (using the lemma again). This shows that $E_{A,B}(f)$ is the least extension of f in $\mathcal{M}(B)$.
2. If (f_1, g) and (f_2, g) are in $I_{A,B,\alpha}$ then $f_1 = f_2$. Indeed, from the previous fact, we have that $E_{A,B}(f_1) \leq g$ and thus, using the lemma, $f_1 \leq f_2$. Symmetrically we obtain that $f_2 \leq f_1$ and thus $f_1 = f_2$, so that the elements of $\mathcal{M}(B)$ have at most one restriction to $\mathcal{M}(A)$ (notice that there are some elements of $\mathcal{M}(B)$ that do not have a restriction to $\mathcal{M}(A)$).

Technically, Lemma 21 proves that, for each α, $E_{A,B}$ is a Galois connection between $\mathcal{M}_\alpha(A)$ and $\mathcal{M}_\alpha(B)$. We now focus on proving some technical properties about $I_{A,B,\alpha}$ and $E_{A,B}$, which will be useful either in proving Lemma 21 or Theorem 19.

Lemma 22 If (f_1, g_1) and (f_2, g_2) are in $\mathcal{I}_{A,B,\alpha}$, then $(f_1 \vee f_2, g_1 \vee g_2)$ is in $\mathcal{I}_{A,B,\alpha}$.

Proof
The proof is done by induction on the structure of α. In case $\alpha = 0$, we have that $f_1 = g_1 \cap A$ and $f_2 = g_2 \cap A$, and so $f_1 \vee f_2 = f_1 \cup f_2 = (g_1 \cap A) \cup (g_2 \cap A) = (g_1 \cup g_2) \cap A = (g_1 \vee g_2) \cap A$ whence $(f_1 \vee f_2, g_1 \vee g_2)$ is in $\mathcal{I}_{A,B,\alpha}$. In case $\alpha = \beta \to \gamma$, then given (h, l) in $\mathcal{I}_{A,B,\beta}$ we have that $(f_1(h), g_1(l))$ and $(f_2(h), g_2(l))$ are in $\mathcal{I}_{A,B,\gamma}$, so that by induction $(f_1(h) \vee f_2(h), g_1(l) \vee g_2(l)) = (f_1 \vee f_2(h), g_1 \vee g_2(l))$ is in $\mathcal{I}_{A,B,\gamma}$, showing finally that $(f_1 \vee f_2, g_1 \vee g_2)$ is in $\mathcal{I}_{A,B,\alpha}$. □

Lemma 23 For every type α, $(\bot_{A,\alpha}, \bot_{B,\alpha})$ is in $\mathcal{I}_{A,B,\alpha}$.

Proof
A simple induction on α. □

Lemma 24 If $(\bot_{A,\alpha}, f)$ is in $\mathcal{I}_{A,B,\alpha}$ and $(\bot_{A,\beta}, g)$ is in $\mathcal{I}_{A,B,\beta}$, then $(\bot_{A,\alpha\to\beta}, f \mapsto g)$ is in $\mathcal{I}_{A,B,\alpha\to\beta}$.

Proof
Given (h, l) is in $\mathcal{I}_{A,B,\alpha}$, we have $f \mapsto g(l) = g$ or $f \mapsto g(l) = \bot_{B,\beta}$, and in each case $\bot_{A,\alpha\to\beta}(h) = \bot_{A,\beta}$. As we have, by assumption $(\bot_{A,\beta}, g)$ in $\mathcal{I}_{A,B,\beta}$ and, by Lemma 23, $(\bot_{A,\beta}, \bot_{B,\beta})$ also in $\mathcal{I}_{A,B,\beta}$, we obtain that $(\bot_{A,\alpha\to\beta}, f \mapsto g)$ is in $\mathcal{I}_{A,B,\alpha\to\beta}$. □

The next lemma is central in the proof of Theorem 19, it allows us to use disjoint parts of a model without one interfering with the other.

Lemma 25 If A_1 and A_2 are disjoint subsets of B, and f is in $\mathcal{M}_\alpha(A_1)$ then we have:

1. $(\bot_{A_2,\alpha}, E_{A_1,B}(f))$ is in $\mathcal{I}_{A_2,B,\alpha}$
2. if $\alpha = \beta \to \gamma$ and g is in $\mathcal{M}_\beta(B)$ then $(\bot_{A_2,\gamma}, E_{A_1,B}(f)(g))$ is in $\mathcal{I}_{A_2,B,\gamma}$.

Proof

1. We proceed by induction on α. In case $\alpha = 0$, $\bot_{A_2,\alpha} = \emptyset$, and since $f \subseteq A_1$ and A_1 is disjoint from A_2, we have that $f \cap A_2 = \emptyset$. It thus follows that $(\emptyset, f)$ is in $\mathcal{I}_{A_2,B,\alpha}$. In case $\alpha = \beta \to \gamma$, let (h, l) be in $\mathcal{I}_{A_2,B,\beta}$. We need to show that $(\bot_{A_2,\gamma}, E_{A_1,B}(f)(l))$ is in $\mathcal{I}_{A_2,B,\gamma}$. We have that $E_{A_1,B}(f)(l) = \bigvee\{E_{A_1,B}(f(k)) \mid E_{A_1,B}(k) \leq l, k \in \mathcal{M}_\beta(A_1)\}$. As, by induction, we have that $(\bot_{A_2,\gamma}, E_{A_1,B}(f(k)))$ is in $\mathcal{I}_{A_2,B,\gamma}$ and that, by the previous Lemma $(\bot_{A_2,\gamma}, \bot_{B,\gamma})$ is in $\mathcal{I}_{A_2,B,\gamma}$, we obtain, using iteratively Lemma 22, that $(\bot_{A_2,\gamma}, E_{A_1,B}(f)(l))$ is in $\mathcal{I}_{A_2,B,\gamma}$.
2. when $\alpha = \beta \to \gamma$, by definition of $E_{A_1,B}(f)$, we have that for a given g in $\mathcal{M}_\beta(B)$, $E_{A_1,B}(f)(g) = \bigvee\{E_{A_1,B}(f(h)) \mid E_{A_1,B}(h) \leq g,\ h \in \mathcal{M}_\beta(A_1)\}$. From the previous statement of the Lemma, we have $(\bot_{A_2,\gamma}, E_{A_1,B}(f(h)))$ in $\mathcal{I}_{A_2,B,\gamma}$; Lemma 22 gives that $(\bot_{A_2,\gamma}, \bigvee\{E_{A_1,B}(f(h)) \mid E_{A_1,B}(h) \leq g,\ h \in \mathcal{M}_\beta(A_1)\})$ is in $\mathcal{I}_{A_2,B,\gamma}$ which gives the result.

□

We are now in a position to prove Lemma 21 (we reproduce its statement for ease of reading).
Lemma 21. *Given a type α we have the following properties:*

1. *for every f in $\mathcal{M}_\alpha(A)$, $(f, E_{A,B}(f))$ is in $\mathcal{I}_{A,B,\alpha}$,*
2. *for every (f, g) in $\mathcal{I}_{A,B,\alpha}$, for every h in $\mathcal{M}_\alpha(A)$, we have $E_{A,B}(h) \leq g$ iff $h \leq f$.*

Proof
We proceed by induction on the structure of α. The case where $\alpha = 0$ is clear.
In case $\alpha = \beta \to \gamma$, we have:

1. Given (g_1, g_2) in $\mathcal{I}_{A,B,\beta}$, we want to show that $(f(g_1), E_{A,B}(f)(g_2))$ is in $\mathcal{I}_{A,B,\gamma}$. By definition, $E_{A,B}(f)(g_2) = \bigvee\{E_{A,B}(f(l)) \mid E_{A,B}(l) \leq g_2\}$. But by the induction hypothesis, given l in $\mathcal{M}_\beta(A)$, $E_{A,B}(l) \leq g_2$ iff $l \leq g_1$. Therefore $E_{A,B}(f)(g_2) = \bigvee\{E_{A,B}(f(l)) \mid l \leq g_1\}$. But the induction hypothesis also implies that $(f(g_1), E_{A,B}(f(g_1)))$ is in $\mathcal{I}_{A,B,\gamma}$, and by monotonicity of f, if $l \leq g_1$, then $f(l) \leq f(g_1)$, which by induction is equivalent to $E_{A,B}(f(l)) \leq E_{A,B}(f(g_1))$. Therefore $\bigvee\{E_{A,B}(f(l)) \mid l \leq g_1\} = E_{A,B}(f(g_1))$, $E_{A,B}(f)(g_2) = E_{A,B}(f(g_1))$ and $(f(g_1), E_{A,B}(f)(g_2))$ is in $\mathcal{I}_{A,B,\gamma}$.

2. We first prove that if $h \leq f$ then $E_{A,B}(h) \leq g$. As $h \leq f$, we have that for l in $\mathcal{M}_\beta(A)$, $h(l) \leq f(l)$ and by induction we have $(f(l), E_{A,B}(f(l)))$ in $\mathcal{I}_{A,B,\gamma}$ so that, by induction again, $E_{A,B}(h(l)) \leq E_{A,B}(f(l))$. But, given k in $\mathcal{M}_\beta(B)$, we have

$$\begin{aligned} E_{A,B}(h)(k) &= \bigvee\{E_{A,B}(h(l)) \mid E_{A,B}(l) \leq k\} \\ E_{A,B}(f)(k) &= \bigvee\{E_{A,B}(f(l)) \mid E_{A,B}(l) \leq k\} \end{aligned}$$

   and since we have seen that for every l in $\mathcal{M}_\beta(A)$, $E_{A,B}(h(l)) \leq E_{A,B}(f(l))$, we necessarily have $E_{A,B}(h)(k) \leq E_{A,B}(f)(k)$ and therefore $E_{A,B}(h) \leq E_{A,B}(f)$. But by induction we have that for every l in $\mathcal{M}_\beta(A)$, $(l, E_{A,B}(l))$ is in $\mathcal{I}_{A,B,\beta}$ thus $(f(l), g(E_{A,B}(l)))$ is in $\mathcal{I}_{A,B,\gamma}$. Using the induction hypothesis again, we obtain $E_{A,B}(f(l)) \leq g(E_{A,B}(l))$. Now given k in $\mathcal{M}_\beta(B)$, we have that $E_{A,B}(f)(k) = \bigvee\{E_{A,B}(f(l)) \mid E_{A,B}(l) \leq k\}$, but as when $E_{A,B}(l) \leq k$, $g(E_{A,B}(l)) \leq g(k)$, we have $E_{A,B}(f(l)) \leq g(k)$ and thus $E_{A,B}(f)(k) \leq g(k)$ proving $E_{A,B}(f) \leq g$ and since we already showed $E_{A,B}(h) \leq E_{A,B}(f)$ we have $E_{A,B}(h) \leq g$.

   Now suppose that $E_{A,B}(h) \leq g$, given l in $\mathcal{M}_\beta(A)$, by induction $(l, E_{A,B}(l))$ is in $\mathcal{M}_\beta(A)$. Therefore, $(f(l), g(E_{A,B}(l)))$ is in $\mathcal{I}_{A,B,\gamma}$, but also, $E_{A,B}(h)(E_{A,B}(l)) \leq g(E_{A,B}(l))$. As, by definition,

$$E_{A,B}(h)(E_{A,B}(l)) = \bigvee\{E_{A,B}(h(l')) \mid E_{A,B}(l') \leq E_{A,B}(l)\}$$

   and as, by induction, $E_{A,B}(l') \leq E_{A,B}(l)$ iff $l' \leq l$, we obtain

$$E_{A,B}(h)(E_{A,B}(l)) = E_{A,B}(h(l)).$$

   Thus, from $E_{A,B}(h)(E_{A,B}(l)) \leq g(E_{A,B}(l))$ we obtain that $E_{A,B}(h(l)) \leq g(E_{A,B}(l))$, then the induction hypothesis gives that $h(l) \leq f(l)$ and finally that $h \leq f$.

□

Lemma 26 Given (f, g) in $\mathcal{I}_{A,B,\beta}$ and h in $\mathcal{M}_\alpha(A)$, we have

$$(h \mapsto f, E_{A,B}(h) \mapsto g) \text{ is in } \mathcal{I}_{A,B,\alpha\to\beta}.$$

Proof
Given (k,l) in $\mathcal{I}_{A,B,\alpha}$, from Lemma 21 we have that $h \leq k$ iff $E_{A,B}(h) \leq l$. Therefore we have that either $(h \mapsto f(k), E_{A,B}(h) \mapsto g(l)) = (f,g)$ or $(h \mapsto f(k), E_{A,B}(h) \mapsto g(l)) = (\bot_{A,\beta}, \bot_{B,\beta})$ and in both cases we have that $(h \mapsto f(k), E_{A,B}(h) \mapsto g(l))$ is in $\mathcal{I}_{A,B,\beta}$. □

We are now able to prove Theorem 19.

Theorem 19. *For every type α and λ-term M of type α, one can effectively construct a triple T that is characteristic of M.*

Proof
We assume that M is in η-long form and we proceed by induction on M. There are two cases.

Case $M = x^\alpha M_1 \dots M_n$. Let us assume that $\alpha = \alpha_1 \to \dots \to \alpha_n \to 0$; by the induction hypothesis, for every i in $[1,n]$ there is $T_i = (A_i, \nu_i, f_i)$ that is characteristic of M_i. Let $B = \{b\} \cup \bigcup_{i=1}^n A_i$, where b is not in any of the A_i, let $A_{n+1} = \{b\}$, and let ν be the valuation such that, for every variable y^β different from x^α:

$$\nu(y^\beta) = \bigvee_{i=1}^n E_{A_i,B}(\nu_i(y^\beta))$$

and

$$\begin{aligned} \nu(x^\alpha) &= g \vee \bigvee_{i=1}^n E_{A_i,B}(\nu_i(x^\alpha)) \text{ where} \\ g &= E_{A_1,B}(f_1) \mapsto \dots \mapsto E_{A_n,B}(f_n) \mapsto \{b\} \end{aligned}$$

We will see that $T = (B, \nu, \{b\})$ is a characteristic triple for M. It is easy to see that $\{b\}$ is hereditary prime and that ν is a hereditary prime valuation. Thus to prove that T is a characteristic triple for M, it just remains to prove that for every λ-term N, $M =_{\beta\eta} N$ iff $\{b\} \leq [\![N]\!]^\nu_{\mathcal{M}(B)}$.

Lemma 25 gives that, for every variable y^β, every (i,j) in $[1,n+1] \times [1,n]$, if $i \neq j$ then $(\bot_{A_i,\beta}, E_{A_j,B}(\nu_j(y^\beta)))$ is in $\mathcal{I}_{A_i,B,\beta}$. An iterative use of Lemma 22 then shows that

$$(\bot_{A_i,\beta}, \bigvee_{j\in\{1,\dots,n\}-\{i\}} E_{A_j,B}(\nu_j(y^\beta)) \in \mathcal{I}_{A_i,B,\beta}$$

Notice that instantiating i with $n+1$ in the above gives

$$(\bot_{A_{n+1},\beta}, \bigvee_{j\in\{1,\dots,n\}} E_{A_j,B}(\nu_j(y^\beta)) \in \mathcal{I}_{A_i,B,\beta}$$

Moreover Lemma 21 implies that $(\nu_i(y^\beta), E_{A_i,B}(\nu_i(y^\beta)))$ is in $\mathcal{I}_{A_i,B,\beta}$. Another use of Lemma 22 then gives that for every i in $[1,n]$,

$$(\nu_i(y^\beta), \bigvee_{j\in[1,n]} E_{A_j,B}(\nu_j(y^\beta))) \in \mathcal{I}_{A_i,B,\beta}.$$

So that when $y^\beta \neq x^\alpha$, $(\nu_i(y^\beta), \nu(y^\beta))$ is in $\mathcal{I}_{A_i,B,\beta}$ for each i in $[1,n]$ and $(\bot_{A_{n+1},\beta}, \nu(y^\beta))$ is in $\mathcal{I}_{A_i,B,\beta}$.

We are now going to see that we also have that $(\nu_i(x^\alpha), \nu(x^\alpha))$ is in $\mathcal{I}_{A_i,B,\alpha}$ for each i in $[1,n]$. As an intermediate result we need to prove that for every i in $[1,n]$, $(\bot_{A_i,\alpha}, g)$ is in $\mathcal{I}_{A_i,B,\alpha}$. Let $g_0 = \{b\}$, $\delta_0 = 0$ and $g_{j+1} = E_{A_{n-j},B}(f_{n-j}) \mapsto g_j$ and $\delta_{j+1} = \alpha_{n-j} \to \delta_j$. Note that $g_{n-1} = g$ and $\delta_{n-1} = \alpha$. We prove by induction on j that for every j in $[0, n-1]$ and every i in $[1,n]$, $(\bot_{A_i,\delta_j}, g_j)$ is in $\mathcal{I}_{A_i,B,\delta_j}$. The case where $j = 0$ is clear. We now show that if $(\bot_{A_i,\delta_j}, g_j)$ is in $\mathcal{I}_{A_i,B,\delta_j}$ then $(\bot_{A_i,\delta_{j+1}}, g_{j+1})$ is in $\mathcal{I}_{A_i,B,\delta_{j+1}}$. There are two cases depending on whether $i = n-j$ or not. If $i = n-j$, then, Lemma 26, gives that $(f_i \mapsto \bot_{A_i,\delta_j}, E_{A_i,B}(f_i) \mapsto g_j) = (\bot_{A_i,\delta_{j+1}}, g_{j+1})$ is in $\mathcal{I}_{A_i,B,\delta_{i+1}}$. If $i \neq n-j$, then Lemma 25 gives that $(\bot_{A_i,\alpha_{n-j}}, E_{A_i,B}(f_{n-j}))$ is in $\mathcal{I}_{A_i,B,\alpha_{n-j}}$ and Lemma 24 thus gives that $(\bot_{A_i,\delta_{j+1}}, g_{j+1})$ is in $\mathcal{I}_{A_i,B,\delta_{j+1}}$. Thus finally we have that, for every i, $(\bot_{A_i,\alpha}, g)$ is in $\mathcal{I}_{A_i,B,\alpha}$ and thus $(\nu_i(x^\alpha), \nu(x^\alpha))$ is in $\mathcal{I}_{A_i,B,\alpha}$.

Using the fundamental Lemma of logical relations, given a λ-term N_i of type α_i, we obtain that

$$(\llbracket N_i \rrbracket^{\nu_i}_{\mathcal{M}(A_i)}, \llbracket N_i \rrbracket^{\nu}_{\mathcal{M}(B)}) \in \mathcal{I}_{A_i,B,\alpha_i}$$

moreover, from Lemma 21, we have that $E_{A_i,B}(f_i) \leq \llbracket N_i \rrbracket^{\nu}_{\mathcal{M}(B)}$ iff $f_i \leq \llbracket N_i \rrbracket^{\nu_i}_{\mathcal{M}(A_i)}$, but, since T_i is a characteristic triple for M_i, we obtain that $E_{A_i,B}(f_i) \leq \llbracket N_i \rrbracket^{\nu}_{\mathcal{M}(B)}$ iff $N_i =_\beta M_i$.

Let us now suppose that we are given a λ-term N of type 0 such that $\{b\} \leq \llbracket N \rrbracket^{\nu}_{\mathcal{M}(B)}$. Without loss of generality, we assume that N is in β-normal, η-long form. We then prove that $N =_\beta M$. We must have $N = y^\beta N_1 \dots N_p$. If y^β is different from x^α, then, we have by Lemma 25 that $(\bot_{A_{n+1},B,\beta}, E_{A_i,B,\beta}(\nu_i(y^\beta)))$ is in $\mathcal{I}_{A_{n+1},B,\beta}$. Therefore, again by Lemma 25, we obtain that

$$(\bot_{A_{n+1},0}, E_{A_i,B,\beta}(\nu_i(y_\beta))\llbracket N_1 \rrbracket^{\nu}_{\mathcal{M}(B)} \dots \llbracket N_n \rrbracket^{\nu}_{\mathcal{M}(B)}) \in \mathcal{I}_{A_{n+1},B,0}$$

so that by Lemma 22 $(\bot_{A_{n+1},0}, \llbracket N \rrbracket^{\nu}_{\mathcal{M}})$ is also in $\mathcal{I}_{A_{n+1},B,0}$, which is possible only if $b \notin \llbracket N \rrbracket^{v}_{\mathcal{M}(B)}$ or, equivalently, only if we do not have $\{b\} \leq \llbracket N \rrbracket^{\nu}_{\mathcal{M}(B)}$. Therefore if $\{b\} \leq \llbracket N \rrbracket^{v}_{\mathcal{M}(B)}$ we must have $y^\beta = x^\alpha$ and $N = x^\alpha N_1 \dots N_n$. We are now going to show that we must also have $E_{A_i,B}(f_i) \leq \llbracket N_i \rrbracket^{v}_{\mathcal{M}(B)}$, which, as we have seen above, is equivalent to $N_i =_\beta M_i$ and thus to $N =_\beta M$. Let us suppose that for some i we do not have $E_{A_i,B}(f_i) \leq \llbracket N_i \rrbracket^{v}_{\mathcal{M}(B)}$. This means that $g(\llbracket N_1 \rrbracket^{v}_{\mathcal{M}(B)}) \dots (\llbracket N_n \rrbracket^{v}_{\mathcal{M}(B)}) = \bot_{B,0}$ and that

$$\nu(x^\alpha)(\llbracket N_1 \rrbracket^{\nu}_{\mathcal{M}(B)}) \dots (\llbracket N_n \rrbracket^{\nu}_{\mathcal{M}(B)}) = \bigvee_{i \in [1,n]} E_{A_i,B}(\nu_i(x^\alpha))(\llbracket N_1 \rrbracket^{\nu}_{\mathcal{M}(B)}) \dots (\llbracket N_n \rrbracket^{\nu}_{\mathcal{M}(B)})$$

but then, we have that $(\bot_{A_{n+1},B,\alpha}, E_{A_i,B}(\nu_i(x^\alpha)))$ is in $\mathcal{I}_{A_{n+1},B,\alpha}$. This implies that for all i $(\bot_{A_{n+1},B,0}, E_{A_i,B}(\nu_i(x^\alpha))(\llbracket N_1 \rrbracket^{\nu}_{\mathcal{M}(B)}) \dots (\llbracket N_n \rrbracket^{\nu}_{\mathcal{M}(B)}))$ is in $\mathcal{I}_{A_{n+1},B,0}$ and, again, Lemma 25 gives that it cannot be the case that $\{b\} \leq \nu(x^\alpha)(\llbracket N_1 \rrbracket^{v}_{\mathcal{M}(B)}) \dots (\llbracket N_n \rrbracket^{v}_{\mathcal{M}(B)})$. Therefore, for every i in $[1,n]$ we must have $E_{A_i,B}(f_i) \leq \llbracket N_i \rrbracket^{v}_{\mathcal{M}(B)}$.

Case $M = \lambda x^{\beta}.P$. We assume that P is of type γ and $\alpha = \beta \to \gamma$. By induction there is a triple $U = (A, \nu, f)$ that is characteristic of P; we let $T = (A, \nu', g)$ with $\nu' = \nu[x^{\alpha} \leftarrow \bot_{A,\alpha}]$ and $g = \nu(x^{\alpha}) \mapsto f$. Clearly, g is hereditary prime and ν' is a hereditary prime valuation.

To prove that T is a characteristic triple for M, it just remains to show that for every λ-term N, $g \leq [\![N]\!]^{\nu'}_{\mathcal{M}(A)}$ iff $N =_{\beta} M$. So, given N such that $g \leq [\![N]\!]^{\nu'}_{\mathcal{M}(A)}$; we have that $g(\nu(x^{\alpha})) \leq [\![N]\!]^{\nu'}_{\mathcal{M}(A)}(\nu(x^{\alpha}))$ so that $f \leq [\![Nx^{\alpha}]\!]^{\nu}_{\mathcal{M}(A)}$, therefore $Nx^{\alpha} =_{\beta} P$ and thus $\lambda x^{\alpha}.Nx^{\alpha} =_{\beta} \lambda x^{\alpha}.P$ finally giving $N =_{\beta} M$. □

Now, using the Observability Theorem as in the proof of Proposition 1 we obtain:

Theorem 27 *Given a λ-term M of type α and a $\lambda Y{+}\Omega$-term N of type α, if $T = (A, \nu, f)$ is a characteristic triple for M then $f \leq [\![N]\!]^{\nu}_{\mathcal{M}(A)}$ iff $M \in \mathcal{L}_{OI}(N)$.*

Proof
By the Observability Theorem, since f is hereditary prime and ν is hereditary prime, $f \leq [\![N]\!]^{\nu}_{\mathcal{M}(A)}$ iff there is a λ-term P in $\mathcal{L}_{OI}(N)$ so that $f \leq [\![P]\!]^{\nu}_{\mathcal{M}(A)}$, but, according to Theorem 19, this is equivalent to that $P =_{\beta} M$, which is equivalent to M being in $\mathcal{L}_{OI}(N)$. □

And finally we obtain the decidability of the membership problem for OI grammars.

Theorem 28 *Given M a λ-term of type α and N a $\lambda Y{+}\Omega$-term of type α, it is decidable whether $M \in \mathcal{L}_{OI}(N)$.*

Proof
Theorem 19 gives a constructive method so as to construct a characteristic triple (A, ν, f) for M. Then $[\![N]\!]^{\nu}_{\mathcal{M}(A)}$ can effectively be compared to f. □

5. Conclusion

We have seen how to use models of λ-calculus so as to solve algorithmic questions, in particular the emptiness and membership problems, related to the classes of higher-order IO and OI languages of λ-terms. In so doing, we have revisited various questions related to finite models of the λ-calculus. In particular, we have seen that hereditary prime elements, via the Observability Theorem, play a key role in finding effective solutions for higher-order OI languages. In combination with Theorem 10, we obtain that it is decidable whether there is a term M whose interpretation in a monotone model is greater than a given hereditary prime element of that model, which gives a decidability result for a restricted notion of λ-definability. This raises at least two questions: (i) what properties of λ-terms can be captured with hereditary prime elements, and (ii) is

there a natural extension of this notion that still defines some decidable variant of λ-definability.

We have also proved that the class of OI languages strictly subsumes the class of IO languages. Nevertheless, the strictness of the inclusion is based on an argument involving λ-definability and thus we do not know whether, when restricting OI and IO languages to strings and trees, this inclusion is still strict. We think that this is likely to be the case, in particular, because it seems that IO languages are not closed under finite language substitutions while OI languages are. Unfortunately, this conjecture does not appear to be trivial to establish. Another problem that this result raises is whether safe IO languages are all safe OI languages. Indeed the proof technique we use to prove that IO languages are all OI languages does not preserve safety in general. It remains as well to see how the respective levels of the IO and OI hierarchies compare to one another. As far as we know, this problem is also open for the safe IO and OI hierarchies. Fischer's [13] result that IO and OI languages are incomparable suggests that for every n, IO_n and OI_n are also incomparable and that this should also hold when attention is restricted to safe languages.

For proving the strictness of the inclusion of the IO languages into the OI ones we have shown that the set of all closed λ-terms of a given type is an OI language. As those λ-terms represent proofs in minimal logic via the Curry-Howard isomorphism, this result gives a language theoretic representation of these proofs. This new connection to proof representations within standard classes of languages allows one to use the concepts and techniques of formal language theory for studying those representations and may be of use in designing new proof search or transformation algorithms. This also raises the question of whether one can see proofs in other logics as OI languages (or at least of some interesting classes thereof).

On the complexity side, we expect that, using techniques similar to those in [37], it might be possible to prove that verifying whether the value of a $\lambda Y{+}\Omega$-term is greater than a hereditary prime element of a monotone model is of the same complexity as the emptiness and membership problems for the safe OI hierarchy, which is $(n-2)$-EXPTIME-complete for order n-grammars (see [12]; with Huet's convention, the order of a grammar is one more than the order of its corresponding higher-order pushdown automaton).

While most string sets corresponding to natural languages seem to be describable using simple (first order) macro grammars [17] (although some data suggests that this is not enough [24, 20]), it is common to find proposals about syntactic mechanisms which ultimately involve higher order types [21], which are needed to associate the desired semantic representation with the derived string. In the domain of semantics, where higher types are commonplace, least fixed point computations have been proposed by [26] (and are a natural way to understand certain prominent proposals about ellipsis resolution [38]) and nondeterminism has been (implicitly) used to model pronoun resolution [11]. The $\lambda Y{+}\Omega$-calculus studied herein provides a way of representing a wide range of linguistic proposals. That being said, the high complexity of the algorithmic problems this paper studies underscores the need to identify linguistically

motivated restrictions which point to tractable subclasses of OI grammars [39]. Some restricted classes of IO grammars are already known to have low complexity [18, 5]. A natural next step is to see whether in the OI mode of derivation those grammars still have tractable emptiness and membership problems.

References

[1] K. Aehlig, J.G. de Miranda, and C.-H.L. Ong. Safety is not a restriction at level 2 for string languages. In *FOSSACS*, volume 3441 of *LNCS*, pages 490–504, 2005.

[2] A. V. Aho. Indexed grammars - an extension of context-free grammars. *J. ACM*, 15(4):647–671, 1968.

[3] R. M. Amadio and P-L. Curien. *Domains and Lambda-Calculi*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.

[4] W. Blum and C.-H.L. Ong. The safe lambda calculus. *Logical Methods in Computer Science*, 5(1:3):1–38, 2009.

[5] Pierre Bourreau and Sylvain Salvati. A datalog recognizer for almost affine λ-CFGs. In *MOL 12*, volume 6878 of *LNCS*, pages 21–38. Springer, 2011.

[6] Christopher H. Broadbent. The limits of decidability for first order logic on CPDA graphs. In *STACS*, pages 589–600, 2012.

[7] W. Damm. The IO- and OI-hierarchies. *Theor Comput Sci*, 20:95–207, 1982.

[8] P. de Groote. Towards abstract categorial grammars. In ACL, editor, *Proceedings 39th Annual Meeting of ACL*, pages 148–155, 2001.

[9] P. de Groote and E. Lebedeva. On the dynamics of proper names. Technical report, INRIA, 2010.

[10] P. de Groote and E. Lebedeva. Presupposition accommodation as exception handling. In *SIGDIAL*, pages 71–74. ACL, 2010.

[11] Philippe de Groote. Towards a montagovian account of dynamics. In Masayuki Gibson and Jonathan Howell, editors, *Proceedings of SALT 16*, pages 1–16, 2006.

[12] Joost Engelfriet. Iterated stack automata and complexity classes. *Inf. Comput.*, 95(1):21–75, 1991.

[13] M. J. Fischer. *Grammars with macro-like productions*. PhD thesis, Harvard University, 1968.

[14] A. Haddad. IO vs OI in higher-order recursion schemes. In *FICS*, volume 77 of *EPTCS*, pages 23–30, 2012.

[15] J. Roger Hindley and Jonathan P. Seldin. *Lambda-calculus and Combinators, an Introduction*. London, New York: Cambridge University Press, 2008.

[16] G. Huet. *Résolution d'équations dans des langages d'ordre 1,2,...,ω*. Thèse de doctorat en sciences mathématiques, Université Paris VII, 1976.

[17] Aravind K. Joshi. How much context-sensitivity is necessary for characterizing structural descriptions. In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Processing: Theoretical, Computational and Psychological Perspectives*, pages 206–250. Cambridge University Press, NY, 1985.

[18] M. Kanazawa. Parsing and generation as datalog queries. In *Proceedings of the 45th Annual Meeting of ACL*, pages 176–183. ACL, 2007.

[19] T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-order pushdown trees are easy. In *FoSSaCS*, volume 2303 of *LNCS*, pages 205–222, 2002.

[20] Gregory M. Kobele. *Generating Copies: An investigation into structural identity in language and grammar*. PhD thesis, UCLA, 2006.

[21] Gregory M. Kobele and Jens Michaelis. CoTAGs and ACGs. In Denis Béchet and Alexandre Dikovsky, editors, *Logical Aspects of Computational Linguistics*, volume 7351 of *Lecture Notes in Computer Science*, pages 119–134, Berlin, 2012. Springer.

[22] E. Lebedeva. *Expressing Discourse Dynamics Through Continuations*. PhD thesis, Université de Lorraine, 2012.

[23] R. Loader. The undecidability of λ-definability. In *Logic, Meaning and Computation: Essays in memory of Alonzo Church*, pages 331–342. Kluwer, 2001.

[24] Jens Michaelis and Marcus Kracht. Semilinearity as a syntactic invariant. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*, pages 37–40, NY, 1997. Springer-Verlag (Lecture Notes in Computer Science 1328).

[25] R. Montague. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven, CT, 1974.

[26] Y. Moschovakis. Sense and denotation as algorithm and value. In *Logic Colloquium'90: ASL Summer Meeting in Helsinki*, volume 2, pages 210–249. Springer, 1993.

[27] Reinhard Muskens. Lambda Grammars and the Syntax-Semantics Interface. In *Proceedings of the Thirteenth Amsterdam Colloquium*, pages 150–155, 2001.

[28] C.-H.L. Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS*, pages 81–90, 2006.

[29] Pawel Parys. On the significance of the collapse operation. In *LICS*, pages 521–530, 2012.

[30] S. Salvati. Recognizability in the Simply Typed Lambda-Calculus. In *16th WOLLIC*, volume 5514 of *LNCS*, pages 48–60. Springer, 2009.

[31] S. Salvati. On the membership problem for non-linear ACGs. *Journal of Logic Language and Information*, 19(2):163–183, 2010.

[32] S. Salvati, G. Manzonetto, M. Gehrke, and H. Barendregt. Loader and Urzyczyn are logically related. In *ICALP (2)*, LNCS, pages 364–376, 2012.

[33] S. Salvati and I. Walukiewicz. Recursive schemes, Krivine machines, and collapsible pushdown automata. In *RP*, volume 7550 of *LNCS*, pages 6–20, 2012.

[34] B. Srivathsan and I. Walukiewicz. An alternate proof of statman's finite completeness theorem. *Inf. Process. Lett.*, 112(14-15):612–616, 2012.

[35] R. Statman. Completeness, invariance and λ-definability. *Journal of Symbolic Logic*, 47(1):17–26, 1982.

[36] R. Statman and G. Dowek. On statman's finite completeness theorem. Technical Report CMU-CS-92-152, University of Carnegie Mellon, 1992.

[37] K. Terui. Semantic evaluation, intersection types and complexity of simply typed lambda calculus. In *RTA*, pages 323–338, 2012.

[38] Satoshi Tomioka. A step-by-step guide to ellipsis resolution. In Kyle Johnson, editor, *Topics in Ellipsis*, chapter 9, pages 210–228. Cambridge University Press, 2008.

[39] Iris van Rooij. The tractable cognition thesis. *Cognitive Science*, 32:939–984, 2008.