Recognizability in the Simply Typed Lambda-Calculus

Sylvain Salvati

INRIA Bordeaux – Sud-Ouest,
351, cours de la Libration Bâtiment A29
33405 Talence cedex France
sylvain.salvati@labri.fr

Abstract. We define a notion of recognizable sets of simply typed λ-terms that extends the notion of recognizable sets of strings or trees. This definition is based on finite models. Using intersection types, we generalize the notions of automata for strings and trees so as to grasp recognizability for λ-terms. We then expose the closure properties of this notion and present some of its applications.

1 Introduction

Formal language theory is mainly concerned with the study of structures like strings, trees or even graphs. In this paper we try to add simply typed λ-terms to the scope of this theory. This article is a first step: the definition of recognizable sets which are a fundamental notion of formal language theory.

Languages of λ-terms appear in several research areas, but there has been really few research explicitly mentioning them and even fewer studying them. To our knowledge the first work explicitly defining a notion of language of λ-terms is that of de Groote [1]. In mathematical linguistics, the pioneering work Montague [2] shows how to connect syntax and semantics of natural language with the simply typed λ-calculus. Syntactic structures are interpreted via a homomorphism built with λ-terms. The normal forms obtained this way denote formulae of higher-order logic whose interpretation in a suitable model gives the semantics of the sentence. The set of formulae that this technique allows to generate can be seen as a language and one may wonder whether such a language can be *parsed* similarly to other languages like context-free languages. Parsing such languages results in generating sentences from their meaning representation. We have showed that this could effectively be done [3].

Still in mathematical linguistics, the type-logical tradition originating from Lambek's work [4], defines syntactic structures as proofs in some substructural logic. Several proposals have emerged in order to control the structure of those proofs such as in Moortgat's work [5] and his followers. These proofs may be represented as simply typed λ-terms and the set of syntactic structures defines a language of λ-terms.

Since simply typed λ-terms generalize both strings and trees, a notion of recognizable language of simply typed λ-terms should naturally extend those already defined for strings and trees. Furthermore, these languages should also be closed under the operational semantics of the λ-calculus, *i.e.* $\beta\eta$-convertibility. The easiest way to obtain such an extension is to use the algebraic characterization of recognizable sets of strings or trees which says that recognizable sets are precisely the sets that are the union of equivalence classes of a finite congruence. Generalizing this definition to sets of simply typed λ-terms consists in saying that such sets are recognizable if and only if they the set of terms that are interpreted as certain points in a finite model. But such a definition may not be useful in certain situations, this is the reason why we need a notion of automaton of λ-terms that coincides with that of recognizability. We define such automata using intersection types.

This work provides a natural framework in which several results that have appeared in the literature on simply typed λ-calculus can be related. In particular, our work shows that Urzyczyn's result on the undecidability of the emptiness problem for intersection types[6] can be seen as a corollary of Loader's result on the undecidability of λ-definability [7]. Moreover, we have showed in [3] that the singleton language can be defined with intersection types, the equivalence we establish here between recognizability in terms of finite models and in terms of automata gives an alternate proof of Statman's completeness theorem [8] (see also [9]). Furthermore, Statman [8] has showed that higher order matching is related to λ-definability. Since our notion of recognizability is related to λ-definability it gives tools with which we can study this problem.

The paper is organized as follows: section 2 gives the necessary definitions, section 3 gives the definition of recognizable sets of λ-terms. In section 4 we give an automaton-like characterization of recognizability. Section 5 gives its closure properties and section 6 shows some basic applications of the notion of recognizability for parsing and higher order matching. Finally we conclude in section 7.

2 Preliminaries

We here briefly review various notions concerning the simply typed λ-calculus and some related notions.

2.1 Simply typed λ-calculus

Higher Order Signatures (HOS) declare a finite number of constants by assigning them types. An HOS Σ is a triple $(\mathcal{A}, \mathcal{C}, \tau)$, where $\mathcal{A}$ is a finite set of *atomic types*, from which the set of *complex types*, $\mathcal{T}_\Sigma$, is built using the binary infix operator $\rightarrow$, $\mathcal{C}$ is a finite set of constants and τ is a function from $\mathcal{C}$ to $\mathcal{T}_\Sigma$. As usual, we will consider that $\rightarrow$ associates to the right and write $\alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \alpha$ for the type $(\alpha_1 \rightarrow (\cdots \rightarrow (\alpha_n \rightarrow \alpha) \cdots))$. The *order* of a type α, $ord(\alpha)$, is 1 when α is atomic and $\max(ord(\alpha_1) + 1, ord(\alpha_2))$ when $\alpha = \alpha_1 \rightarrow \alpha_2$. By extension,

the order of a HOS is the maximal order of type it associates to a constant. We suppose that we are given an infinite countable set of λ-variables V. We use types *à la Church* and variables explicitly carry their types. So we will write x^α, y^α or z^α (possibly with indices) the elements of $\mathcal{V} \times \{\alpha\}$, the variables of type α. A HOS Σ defines a set of simply typed λ-terms Λ_Σ. This set is the union of the sets of the family $(\Lambda^\alpha_\Sigma)_{\alpha \in \mathcal{T}_\Sigma}$ defined as the smallest sets verifying:

1. $x^\alpha \in \Lambda^\alpha_\Sigma$,
2. for $c \in \mathcal{C}$, $c \in \Lambda^{\tau(c)}_\Sigma$,
3. if $M_1 \in \Lambda^{\alpha\to\beta}_\Sigma$ and $M_2 \in \Lambda^\alpha_\Sigma$ then $(M_1 M_2) \in \Lambda^\beta_\Sigma$,
4. if $M \in \Lambda^\beta$ then $\lambda x^\alpha.M \in \Lambda^{\alpha\to\beta}_\Sigma$.

We take for granted the notions of free variables, closed terms, substitution, the various notions of conversions and reductions associated to the λ-calculus, the notions of normal form, of η-long form (or long form) and the notion of linearity. We write $\Lambda^\alpha_{\Sigma,W}$ to designates the set of terms of type α built on Σ whose set of free variables is included in W.

2.2 Trees and strings as λ-terms

A HOS is said to be a tree-HOS when it is a second order HOS and that it uses only one type namely o. We write $o^n \to o$ in the place of $\underbrace{o \to \cdots \to o}_{n} \to o$ and say that a constant of type $o^n \to o$ has *arity* n. It is easy to see that every freely and finitely generated sets of ranked trees can be seen as a the closed normal terms of type o built on a tree-HOS.

A HOS is said to be a string-HOS when it is a tree-HOS whose constants all have arity 1. Strings are represented as closed terms of type $o \to o$ and the string $c_1 \dots c_n$ is represented by the term $\lambda x^o.c_1(\dots(c_n x^o)\dots)$ denoted by $/c_1 \dots c_n/$. The empty string is $\lambda x^o.x^o$ and the concatenation operation can be represented by function composition $\lambda x^o.s_1(s_2\, x^o)$ (*c.f.* [1]).

2.3 Homomorphisms

A *homomorphism* between the signatures Σ_1 and Σ_2 is a pair (g, h) such that g maps $\mathcal{T}_{\Sigma_1}$ to $\mathcal{T}_{\Sigma_2}$, h maps Λ_{Σ_1} to Λ_{Σ_2} and verify the following properties:

1. $g(\alpha \to \beta) = g(\alpha) \to g(\beta)$,
2. $h(x^\alpha) = x^{g(\alpha)}$,
3. $h(c)$ is a closed term of $\Lambda^{g(\tau(c))}_{\Sigma_2}$,
4. $h(M_1 M_2) = h(M_1)h(M_2)$ and
5. $h(\lambda x^\beta.M) = \lambda x^{g(\beta)}.h(M)$.

A homomorphism is said to be *linear* whenever constants are mapped to linear terms. We write $\mathcal{H}(\alpha)$ and $\mathcal{H}(M)$ respectively instead of $g(\alpha)$ and of $h(M)$ for a given homomorphism $\mathcal{H} = (g, h)$. Note that if $\mathcal{H}$ is a homomorphism from Σ_1 to Σ_2 and $M \in \Lambda^\alpha_{\Sigma_1}$ then $\mathcal{H}(M) \in \Lambda^{\mathcal{H}(\alpha)}_{\Sigma_2}$.

The order of a homomorphism $\mathcal{H}$ is the maximal order of the type it associates to an atomic type. The usual notion of tree-homomorphism (*resp.* string homomorphism) is a first order homomorphism (in our sense) between tree-signatures (*resp.* string-signatures). A first order homomorphism between Σ_1 and Σ_2 that maps constants of Σ_1 to constants of Σ_2 is called a *relabeling*.

2.4 Models

Given a signature Σ, a *full model* of Σ is a pair $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ where:

1. $(\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}$ is a family of sets verifying:
   (a) for all $\alpha, \beta \in \mathcal{T}_\Sigma$, $\mathcal{M}^{\alpha\to\beta} = \mathcal{M}^{\alpha\mathcal{M}^\beta}$,
   (b) the sets $\mathcal{M}^\alpha$ such that α is atomic are pairwise disjoint.
2. ρ is a function from $\mathcal{C}$ to $\mathcal{M}^\alpha$ so that $\alpha = \rho(c)$.

A full model, $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$, of Σ is said *finite* when for all $\alpha \in \mathcal{T}_\Sigma$, $\mathcal{M}^\alpha$ is a finite set. Remark that $\mathbb{M}$ is finite if and only if for all atomic types α, $\mathcal{M}^\alpha$ is finite.

Given $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ a full model of Σ, the terms of Λ^α_Σ are interpreted as elements of $\mathcal{M}^\alpha$. This interpretation necessitates the definition of *variable assignments* which are partial functions that associate elements of $\mathcal{M}^\alpha$ to variables like x^α. A variable assignment is said *finite* when its domain is finite. Given a variable assignment ν, a variable x^α and $m \in \mathcal{M}^\alpha$ we define $\nu[x^\alpha \leftarrow m]$ to be the variable assignment verifying:

$$\nu[x^\alpha \leftarrow m](y^\beta) \begin{cases} m \text{ if } y^\beta = x^\alpha \\ \nu(y^\beta) \text{ otherwise} \end{cases}$$

Given a full model $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ a variable assignment ν, the interpretation of the elements of Λ_Σ (whose sets of free variables are included in the domain of definition of ν) in $\mathbb{M}$ is inductively defined as follows:

1. $[\![x^\alpha]\!]^{\mathbb{M}}_\nu = \nu(x^\alpha)$
2. $[\![c]\!]^{\mathbb{M}}_\nu = \rho(c)$
3. $[\![M_1 M_2]\!]^{\mathbb{M}}_\nu = [\![M_1]\!]^{\mathbb{M}}_\nu([\![M_2]\!]^{\mathbb{M}}_\nu)$
4. $[\![\lambda x^\alpha.M]\!]^{\mathbb{M}}_\nu$ is the function which maps $m \in \mathcal{M}^\alpha$ to $[\![M]\!]^{\mathbb{M}}_{\nu[x^\alpha\leftarrow m]}$.

It is well-known that the semantics of λ-terms is invariant modulo $\beta\eta$-reduction.

3 Recognizable sets of λ-terms

We wish to extend the notion of recognizability that already exists for strings and trees to λ-terms. An abstract way of defining recognizability for strings and trees is to use Myhill-Nerode theorem [10], [11], that describes it in terms of congruence of finite index over strings or trees which is equivalent to describing it in terms of finite algebra for trees or finite semigroups for strings. This approach has been successfully extended in the seminal paper [12] to any abstract algebra. We shall follow this line of work in order to define recognizability for the simply typed λ-calculus. Since the finite full models form the functional closure of finite algebra, we use them so as to extend recognizability to λ-terms.

Definition 1. *Given a HOS Σ and $\alpha \in \mathcal{T}_\Sigma$ a set $\mathcal{R}$ included in Λ^α_Σ is said to be recognizable iff there is a finite and full model $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ a finite variable assignment ν and a subset $\mathcal{P}$ of $\mathcal{M}^\alpha$ such that: $\mathcal{R} = \{M | [\![M]\!]^{\mathbb{M}}_\nu \in \mathcal{P}\}$.*

Note that in this definition when ν is chosen to be the empty assignment function then the set $\mathcal{R}$ only contains closed terms. In particular, when Σ is a tree (*resp.* string) signature, and that α is the atomic type o (*resp.* the type $o \to o$) then the set of closed λ-terms that are recognizable correspond exactly to set of recognizable trees (*resp.* strings).

The result by Loader [7] shows that in general it is undecidable to check whether a recognizable set is empty. But as soon as the finite and full model and the assignment function are given we can check whether a term is in the set. In what follows we give a mechanical way (corresponding to automata for trees or strings) to define recognizable sets and check whether a certain term belongs to that set.

A classical and simple example of recognizable set of trees being the set of true boolean formulae, we exemplify the notion of recognizability for λ-terms with the set of true Quantified Boolean Formulae (QBF). For this it suffices to use a HOS B whose constants are: $\wedge : b^2 \to b$, $\vee : b^2 \to b$, $\neg : b \to b$, $\mathbf{1} : b$, $\mathbf{0} : b$, $\forall : (b \to b) \to b$ and $\exists : (b \to b) \to b$. We use a finite model $\mathbb{B} = ((\mathcal{B}^\alpha)_{\alpha\in\mathcal{T}_B}, \rho)$ such that $\mathcal{B}^b = \{0; 1\}$ and ρ associates the obvious functions to the constants of B. Then the set of terms representing a true QBF is the set of closed λ-terms of B^b which are interpreted as 1 in $\mathbb{B}$ and therefore this set is recognizable.

4 Automata characterizing recognizable sets

We here generalize the notion of automata for trees and strings in order to obtain a mechanical definition of recognizability for λ-terms. Our notion of automaton is based on the notion of *Higher Order Intersection Signature (HOIS)* introduced in [3] which, in turn, is based on intersection types [13]. A HOIS is a tuple $\Pi = (\Sigma, I, \iota, \chi)$ where Σ is a HOS, I is a finite set of *atomic intersection types*, ι is a function from I to the atomic types of Σ, χ is a function that associates to every element of $\mathcal{C}$ a subset of $\cap^{\tau(c)}_\Pi$ where $(\cap^\alpha_\Pi)_{\alpha\in\mathcal{T}_\Sigma}$ is the smallest family verifying:

1. for α and atomic type $\cap^\alpha_\Pi = \iota^{-1}(\alpha)$,
2. $\cap^{\alpha\to\beta}_\Pi = 2^{\cap^\alpha_\Pi} \times \{\alpha\} \times \cap^\beta_\Pi$ (we write 2^P, the powerset of the set P)

A trivial induction on the structure of α shows that for each type α, the set $\cap^\alpha_\Pi$ is finite.

We now define the type system that allows to associate types to λ-terms. Given a HOIS $\Pi = (\Sigma, I, \iota, \chi)$, a *$\Pi$-typing environment* (or simply, *typing environment*, when there is no ambiguity) , is a partial mapping from typed variables to $2^{\cap_\Pi}$ whose domain is finite and such that $\Gamma(x^\alpha)$, when it is defined, is included in $\cap^\alpha_\Pi$. We write $\overline{\Gamma}$ to denote the domain of Γ. *Judgements over Π* are objects of the form $\Gamma \vdash_\Pi M : p$ where Γ is a typing environment, M is an element of

Λ_Σ and p is an element of $\cap_\Pi$. Judgements are derived by using the following inference system:

$$\frac{p \in \Gamma(x^\alpha)}{\Gamma \vdash_\Pi x^\alpha : p}\ \textsc{Axiom} \qquad \frac{p \in \chi(c)}{\Gamma \vdash_\Pi c : p}\ \textsc{Const} \qquad \frac{\Gamma \vdash_\Pi M : p \quad p \sqsubseteq^\alpha_\Pi q}{\Gamma \vdash_\Pi M : q}\ \textsc{Sub}$$

$$\frac{\Gamma, x^\alpha : S \vdash_\Pi M : p}{\Gamma \vdash_\Pi \lambda x^\alpha.M : (S, \alpha, p)}\ \textsc{Abst}$$

$$\frac{\Gamma \vdash_\Pi M : (S, \alpha, p) \quad N \in \Lambda^\alpha_{\Sigma,\overline{\Gamma}} \quad \forall q \in S.\ \Gamma \vdash_\Pi N : q}{\Gamma \vdash (MN) : p}\ \textsc{App}$$

Where the relation $\sqsubseteq^\alpha_\Pi$ is defined as follows:

$$\frac{i \in \iota(\alpha)}{i \sqsubseteq^\iota_\Pi i} \qquad \frac{T \subseteq \cap^\alpha_\Pi \quad \forall p \in S. \exists q \in T. q \sqsubseteq^\alpha_\Pi p}{T \trianglelefteq^\alpha_\Pi S} \qquad \frac{S \trianglelefteq^\alpha_\Pi T \quad q \sqsubseteq^\beta_\Pi p}{(T, \alpha, q) \sqsubseteq^{\alpha\to\beta}_\Pi (S, \alpha, p)}$$

Notice that the rule App, has two premises, concerning N. The reason of being of the premise $N \in \Lambda^\alpha_{\Sigma,\overline{\Gamma}}$ is that when M has an intersection type of the form $(\emptyset, \alpha, p)$, the premise $\forall q \in S.\ \Gamma \vdash_\Pi N : q$ is trivially true and without such a premise we would derive judgments on terms which would not be simply typed.

We will use the notation $\Gamma \vdash_\Pi M : S$ where S is a subset of $\cap^\alpha_\Pi$ to denote the all the judgements of the form $\Gamma \vdash_\Pi M : p$ where p in S. In the same spirit, given S and T that are respectively subsets of $\cap^\alpha_\Pi$ and of $\cap^\beta_\Pi$, we write (S, α, T) to denote the subset of $\cap^{\alpha\to\beta}_\Pi$, $\{(S, \alpha, p) | p \in T\}$.

We now give the principal properties of that system. The proofs of those properties can be found in [3].

Theorem 1 (subject reduction). *If $\Gamma \vdash_\Pi M : p$ is derivable and $M \xrightarrow{*}_{\beta\eta} N$ then $\Gamma \vdash_\Pi N : p$ is derivable.*

Notice that this Theorem would only hold for β-reduction without the use of the rule Sub.

Theorem 2 (subject expansion). *If $M \in \Lambda^\alpha_\Sigma$, $M \xrightarrow{*}_{\beta\eta} N$ and $\Gamma \vdash_\Pi N : p$ is derivable then $\Gamma \vdash_\Pi M : p$ is derivable.*

Theorem 3 (Singleton). *Given $M \in \Lambda^\alpha_\Sigma$, there is Π, Γ and $S \subseteq \cap^\alpha_\Pi$ such that given $N \in \Lambda^\alpha_\Sigma$, $\Gamma \vdash_\Pi N : S$ is derivable if and only if $M =_{\beta\eta} N$.*

This Singleton Theorem, requires some comments. We proved it referring to coherence theorems such as the one proved in [14]. It is also related to a Theorem proved by Statman [8], [9], since we will see that HOIS and finite full models can be represented one in the other.

Since, with intersection type we can represent graphs of functions, the set of λ-terms that are interpreted as a certain element in a finite full model are exactly the λ-terms that verify a certain judgement.

Theorem 4. *Given a HOS Σ, a finite model of Σ, $\mathbb{M} = (\mathcal{M}^\alpha, \rho)$, a finite variable assignment ν over $\mathbb{M}$ and an element e of $\mathcal{M}^\alpha$ then there is a HOIS over Σ, Π, a typing environment Γ and a subset S of $\cap^\alpha_\Pi$ such that for every λ-term M the two following properties are equivalent:*

1. $[\![M]\!]^{\mathbb{M}}_\nu = e$
2. $\Gamma \vdash_\Pi M : S$

A consequence of the previous theorem is that we can obtain the undecidability result by [6] about the emptiness of intersection type as a corollary of the undecidability of λ-decidability [7].

We now prove the converse of the previous theorem, that is, typability properties in HOIS can be seen as properties of interpretations in finite full models. The principle of this proof is to interpret intersection types as functions operating over intersection types.

We define the operator **app** which maps, for all α and β, $2^{\cap^{\alpha\to\beta}_\Pi} \times 2^{\cap^\alpha_\Pi}$ to $2^{\cap^\beta_\Pi}$. It is defined by: $\mathbf{app}(S,T) = \{p | \exists (Q,\alpha,p) \in S. T \trianglelefteq Q\}$

The finite model in which we will interpret intersection types built over Π is $\mathbb{M}_\Pi = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{I}_\mathcal{A}}, \rho)$ where for α atomic we let $\mathcal{M}^\alpha$ be $2^{\iota^{-1}(\alpha)}$. The definition of ρ requires an auxiliary function $\mathcal{F}^\alpha$ that sends subsets of $\cap^\alpha_\Pi$ to subsets of $\mathcal{M}^\alpha$ and that verifies:

1. for α atomic and S included in $\cap^\alpha_\Pi$ we let $\mathcal{F}^\alpha(S) = \{T \subseteq \cap^\alpha_\Pi | S \subseteq T\}$,
2. for S included in $\cap^{\alpha\to\beta}_\Pi$ we let

$$\mathcal{F}^{\alpha\to\beta}(S) = \{h \in \mathcal{M}^{\alpha\to\beta} | \forall T \subseteq \cap^\alpha_\Pi . \forall g \in \mathcal{F}^\alpha(T). h(g) \in \mathcal{F}^\beta(\mathbf{app}(S,T))\}$$

It is easy to verify that for every S included in $\cap^\alpha_\Pi$, the set $\mathcal{F}^\alpha(S)$ is not empty. We choose $\rho(c)$ as an element of $\mathcal{F}^{\tau(c)}(\chi(c))$. We then have the following theorem.

Theorem 5. *Given a HOS Σ, a HOIS Π over Σ, Γ and S a subset of $\cap^\alpha_\Pi$, we set $\nu(x^\alpha)$ to be an element of $\mathcal{F}_\alpha(\Gamma(x^\alpha))$, then the two following properties are equivalent:*

1. $\Gamma \vdash_\Pi M : S$
2. $[\![M]\!]^{\mathbb{M}_\Pi}_\nu$ *belongs to* $\mathcal{F}_\alpha(S)$

The Theorems 4 and 5 relate finite models and typability in HOIS. This leads us to the definition of a generalized notion of automaton, *typing-automata*.

Definition 2. *A* typing-automaton*, $\mathcal{A}$, over a HOS Σ is a tuple $(\alpha, \Pi, \Gamma, \{S_1; \dots S_n\})$ where: $\alpha \in \mathcal{T}_\Sigma$, Π is a HOIS over Σ, Γ is a Π-typing environment, for all i in $\{1; \dots; n\}$, S_i is a subset of $\cap^\alpha_\Pi$. The language defined by $\mathcal{A}$ is*

$$L(\mathcal{A}) = \{M | \exists i \in \mathbb{N}.\ \Gamma \vdash_\Pi M : S_i\}$$

Using Theorems 4 and 5 we get:

Theorem 6. *A language of λ-terms L is recognizable if and only if there is a typing-automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.*

5 Closure properties

5.1 Boolean closure

In this section we shall quickly outline how to construct of typing-automata for the boolean closure properties of recognizable sets of λ-terms. Interestingly these constructions can be seen as generalizations of the usual constructions that are used for tree/string-automata. For example, concerning the intersection of two recognizable languages, we can construct the product of two typing-automata. We first start by defining the product of two HOIS.

Definition 3. *Given $\Pi_1 = (\Sigma, I_1, \iota_1, \chi_1)$ and $\Pi_2 = (\Sigma, I_2, \iota_2, \chi_2)$ we define the HOIS $\Pi_1 \otimes \Pi_2$ to be (Σ, I, ι, χ) where:*

1. *I is a subset of $I_1 \times I_2$ which is equal to $\{(p_1, p_2) | \iota_1(p_1) = \iota_2(p_2)\}$,*
2. *$\iota((p_1, p_2)) = \iota_1(p_1)$, note that by definition of I, $\iota((p_1, p_2)) = \iota_2(p_2)$,*
3. *$\chi(c) = \{p_1 \otimes p_2 | p_1 \in \chi_1(c)$ and $p_2 \in \chi_2(c)\}$.*

where given p_1 in $\cap^{\alpha}_{\Pi_1}$ and p_2 in $\cap^{\alpha}_{\Pi_2}$ we define $p_1 \otimes p_2$ by:

1. *if α is atomic then $p_1 \otimes p_2 = (p_1, p_2)$*
2. *if $\alpha = \alpha_1 \to \alpha_2$ then $p_1 = (S_1, \alpha_1, q_1)$ and $p_2 = (S_2, \alpha_2, q_2)$ and $p_1 \otimes p_2 = (S_1 \otimes S_2, \alpha_1, q_1 \otimes q_2)$ where $S_1 \otimes S_2 = \{r_1 \otimes r_2 | r_1 \in S_1$ and $r_2 \in S_2\}$*

If we define the product of two typing environment Γ and Δ to be $\Gamma \otimes \Delta$ such that $\Gamma \otimes \Delta(x) = \Gamma(x) \otimes \Delta(x)$, we can prove the following property:

Theorem 7. *The judgements $\Gamma \vdash_{\Pi_1} M : P$ and $\Delta \vdash_{\Pi_2} M : Q$ are derivable if and only if the judgement $\Gamma \otimes \Delta \vdash_{\Pi_1 \otimes \Pi_2} M : P \otimes Q$ is derivable.*

This allows us to define the product $\mathcal{A} \otimes \mathcal{B}$ of two typing-automata $\mathcal{A}$ and $\mathcal{B}$.

Definition 4. *Given two typing-automata over some HOS Σ, $\mathcal{A} = (\alpha, \Pi_1, \Gamma, T_1)$ and $\mathcal{B} = (\alpha, \Pi_2, \Delta, T_2)$, we let $\mathcal{A} \otimes \mathcal{B}$ be $(\alpha, \Pi_1 \otimes \Pi_2, \Gamma \otimes \Delta, T_1 \otimes T_2)$ where $T_1 \otimes T_2$ is the set $\{S_1 \otimes S_2 | S_1 \in T_1$ and $S_2 \in T_2\}$.*

Theorem 8. *Given two typing automata of the same type over some HOS Σ, $\mathcal{A}$ and $\mathcal{B}$ we have $L(\mathcal{A} \otimes \mathcal{B}) = L(\mathcal{A}) \cap L(\mathcal{B})$.*

The closure under complement of the class of recognizable sets of λ-terms, is a direct consequence of its definition in terms of finite models. Interestingly, if one wishes to construct the typing-automaton recognizing the complementary language of a given typing-automaton, then one would use the construction that serves in Theorem 5 which on a tree or string automaton would corresponds to *determinization*. This induces a notion of deterministic typing-automata that grasps the notion of recognizability, and corresponding to the fact that intersection types correspond to partial function over the finite model generated by the atomic intersection types.

5.2 Homomorphisms

It is well-known that recognizable sets of strings are closed under arbitrary homomorphisms while recognizable sets of trees are closed under linear homomorphisms. We will see that recognizable sets of λ-terms are not even closed under relabeling. This has the consequence, that Monadic Second Order Logic (MSO) over the structure of normal λ-terms is not grasped by our notion of recognizability, since relabelings allow to represent set quantification. On the other hand, alike strings and trees, recognizable sets of λ-terms are closed under arbitrary inverse homomorphisms.

We now turn to show that recognizable sets of λ-terms are not closed under relabeling. In order to show this we use the following signature $\Sigma = \{\forall : (b \to b) \to b, \wedge : b \to b \to b, \vee : b \to b \to b, \neg : b \to b, \lhd : b \to b \to b, \rhd : b \to b \to b\}$. Since terms built on Σ are usual boolean expressions, we shall use the standard notation for those expressions instead of the λ-term notation. Thus we shall write $\forall x.t$, $t_1 \wedge t_2$ and $t_1 \vee t_2$ instead of $\forall(\lambda x.t)$, $\wedge t_1\, t_2$ and $\vee t_1\, t_2$. The terms built on Σ are interpreted in a finite model $\mathbb{B} = ((\mathcal{B}^\alpha)_{\alpha \in \mathcal{T}_\Sigma}, \rho)$ where $\mathcal{B}^b = \{0; 1\}$ and ρ interprets the usual boolean connectives and quantifiers ($\wedge$, $\vee$, $\neg$ and $\forall$) with their usual truth tables and ρ interprets the connectives $\lhd$ and $\rhd$ as the functions such that $\rho(\lhd)xy = x$ and $\rho(\rhd)xy = y$. By definition the set $\mathcal{T}$ of closed terms whose semantic interpretation in $\mathbb{B}$ is 1 is recognizable.

We use a relabeling $\mathcal{H}$ which maps the terms built on Σ to terms built on $\Sigma' = \{\forall : (b \to b) \to b, \wedge : b \to b \to b, \vee : b \to b \to b, \neg : b \to b, \bowtie : b \to b \to b\}$ so the constants $\forall$, $\wedge$, $\vee$ and $\neg$ are mapped to themselves by $\mathcal{H}$ and $\lhd$ and $\rhd$ are both mapped to $\bowtie$.

We let $\Leftrightarrow$ be the λ-term $\lambda xy.(x \wedge y) \vee (\neg x \wedge \neg y)$; as for the other connective, we adopt an infix notation, *i.e.* we shall write $t_1 \Leftrightarrow t_2$ instead of $\Leftrightarrow t_1\, t_2$.

As the connective $\lhd$ (*resp.* $\rhd$) takes the value of its left (*resp.* right) argument, if f and g are terms whose free variables are $x_1, \ldots, x_n$, then we have the following identities $[\![\forall x_1 \ldots \forall x_n.(\lhd f\ g) \Leftrightarrow f]\!]^{\mathbb{B}} = 1$ and $[\![\forall x_1 \ldots \forall x_n.(\rhd f\ g) \Leftrightarrow g]\!]^{\mathbb{B}} = 1$.

The closed term $\lambda x_1^b \ldots x_n^b.t$ built on Σ can be interpreted as a function from $\{0; 1\}^n$ to $\{0; 1\}$ (modulo curryfication) in $\mathbb{B}$, *i.e.* an n-ary boolean function. For a given n there are 2^{n+1} such functions and we know that for each such function f we can build, using only $\wedge$, $\vee$ and $\neg$, a term $\widetilde{f}$ such that $[\![\lambda x_1 \ldots x_n.\widetilde{f}]\!]^{\mathbb{B}} = f$. Given $\mathcal{F} = \{f_1; \ldots; f_p\}$ a set of such functions, we write $[\mathcal{F}]$ the term $\bowtie \widetilde{f_1}(\bowtie \widetilde{f_2}(\ldots(\bowtie \widetilde{f_{p-1}}\widetilde{f_p})\ldots))$. Remark that for all i in $\{1; \ldots; p\}$, there is H_i such that $\mathcal{H}(H_i) = [\mathcal{F}]$ and $[\![\forall(x_1 \ldots \forall(x_n.H_i \Leftrightarrow \widetilde{f_i})\ldots)]\!]^{\mathbb{B}} = 1$ and thus $\forall(x_1 \ldots \forall(x_n.[\mathcal{F}] \Leftrightarrow \widetilde{f_i})$ is in $\mathcal{H}(\mathcal{T})$. Furthermore for every H such that $\mathcal{H}(H) = \mathcal{F}$ there is i in $\{1; \ldots; p\}$ such that $[\![\forall(x_1 \ldots \forall(x_n.H \Leftrightarrow \widetilde{f_i})\ldots)]\!]^{\mathbb{B}} = 1$. If we suppose that $\mathcal{H}(\mathcal{T})$ is recognizable, then there is a finite model $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha \in \mathcal{T}_\Sigma}, \rho)$ and a subset $\mathcal{N}$ of $\mathcal{M}^b$ such that the closed terms M are in $\mathcal{H}(\mathcal{T})$ if and only if $[\![M]\!]^{\mathbb{M}} \in \mathcal{N}$; we assume that $\mathcal{M}^b$ contains q elements. Each closed term $\lambda x_1^b \ldots x_n^b.M$ built on Σ' is interpreted in $\mathbb{M}$ as a function from $\{1; \ldots; q\}^n$ to $\{1; \ldots; q\}$ (modulo curryfication). We are going to show that for every sets of n-ary boolean functions $\mathcal{F}$ and $\mathcal{G}$, it is necessary that $[\![\lambda x_1^b \ldots x_n^b.[\mathcal{F}]]\!]^{\mathbb{M}}$ and

$[\![\lambda x_1^b \dots x_n^b.[\mathcal{G}]]\!]^{\mathbb{M}}$ are different when $\mathcal{F}$ and $\mathcal{G}$ are different. Indeed, if $\mathcal{F}$ and $\mathcal{G}$ are different, we can assume without loss of generality that $\mathcal{F}$ is not empty, and then there is a boolean function f which is in $\mathcal{F}$ and which is not in $\mathcal{G}$. Since there is H such that $\mathcal{H}(H) = [\mathcal{F}]$ and $[\![\forall x_1 \dots \forall x_n.H \Leftrightarrow \widetilde{f}]\!]^{\mathbb{B}} = 1$, then $\forall x_1 \dots \forall x_n.[\mathcal{F}] \Leftrightarrow \widetilde{f}$ is in $\mathcal{H}(\mathcal{T})$. But for an n-ary boolean g, there is an H' such that $\mathcal{H}(H') = [\mathcal{G}]$ and $[\![\forall x_1 \dots \forall x_n.H' \Leftrightarrow \widetilde{g}]\!]^{\mathbb{B}} = 1$ iff g is in $\mathcal{G}$. Thus the term $\forall x_1 \dots \forall x_n.[G] \Leftrightarrow \widetilde{f}$ is not in $\mathcal{H}(\mathcal{T})$ and $[\![\lambda x_1^b \dots x_n^b.[\mathcal{F}]]\!]^{\mathbb{M}}$ is different from $[\![\lambda x_1^b \dots x_n^b.[\mathcal{G}]]\!]^{\mathbb{M}}$. But there are $2^{2^{n+1}}$ sets of n-ary boolean functions while there are q^{n+1} functions from $\{1; \dots; q\}^n$ to $\{1; \dots; q\}$ and thus for n sufficiently big, it is not possible to verify that $[\![\lambda x_1^b \dots x_n^b.[\mathcal{F}]]\!]^{\mathbb{M}}$ and $[\![\lambda x_1^b \dots x_n^b.[\mathcal{G}]]\!]^{\mathbb{M}}$ are different when $\mathcal{F}$ and $\mathcal{G}$ are different. Therefore, $\mathcal{H}(\mathcal{T})$ is not a recognizable set. This implies that the class of recognizable sets of λ-terms is not closed under relabeling.

While there seems to be no interesting class of homomorphisms under which our notion of recognizability is closed, we can show that recognizable sets of λ-terms are closed under inverse homomorphism.

Theorem 9. *Given Σ_1, Σ_2 two HOS and $\mathcal{H}$ a homomorphism between Σ_1 and Σ_2, if R is a recognizable set of Σ_2 then $\mathcal{H}^{-1}(R) \cap \Lambda^{\alpha}_{\Sigma,V}$ is also recognizable.*

Recognizable sets contain only λ-terms of a given type and there is no reason why $\mathcal{H}^{-1}(R)$ is a set containing terms having all the same type. So intersecting $\mathcal{H}^{-1}(R)$ with set set of the form $\Lambda^{\alpha}_{\Sigma,V}$ is necessary.

6 Some applications of recognizability

We here quickly review some direct applications of the notion of recognizability in the simply typed λ-calculus.

6.1 Parsing

Theorem 9 gives a very simple definition of parsing for many formalisms. Indeed in formalisms, such as Context Free Grammars, Tree Adjoining Grammars, Multiple Context Free Grammars, Parallel Multiple Context Free Grammars *etc...* can be seen as the interpretation of trees via homomorphism (see [15]). Thus these grammars can be seen a 4-tuple $(\Sigma_1, \Sigma_2, \mathcal{H}, S)$ where Σ_1 is a multi-sorted tree signature, Σ_2 is a string signature, $\mathcal{H}$ is a homomorphism from Σ_1 to Σ_2 and S is the type of the trees that are considered as analyses. Thus if we want to parse a word w we try to find the set $\{M \in \Lambda^S_{\Sigma_1} | M \text{ is closed and } \mathcal{H}(M) =_{\beta\eta} w\}$ which is actually $\mathcal{H}^{-1}(\{w\})$. But we know from Theorem 3 that $\{w\}$ is a recognizable set and thus parsing amounts to compute the inverse homomorphic image of a recognizable set. This gives a new proof of the theorem of [16] which proves that the set of parse trees of a sentence in a context free grammars is a recognizable set, and it furthermore generalizes the result to a wide family of formalisms. Moreover, this view on parsing also applies to grammars generating tree or λ-terms, it also shows that parsing a structure is similar to parsing

recognizable sets. Parsing recognizable sets instead of singleton structures has the advantage that it allows to parse ambiguous inputs, such as noisy phonetic transcriptions, or ambiguous tagging of sentences...

6.2 Higher order matching

The γ-higher-order matching problem (γ-HOM), with $\gamma \in \{\beta; \beta\eta\}$, consists in solving an equation of the form $M \stackrel{?}{=}_\gamma N$ where N is a closed term. A solution of such an equation is a substitution σ such that $M.\sigma =_\gamma N$. Using the extraction Lemma of [3], and Theorem 3, it is easy to see that the solutions of $\beta\eta$-HOM form finite unions of cartesian products of recognizable sets. Observing this, allows us to obtain in an alternative way the relation between λ-definability and $\beta\eta$-HOM showed in [8]. Furthermore, we can easily obtain the result that $\beta\eta$-HOM is decidable (see [17]) when the terms in a solution are *arity bounded*, *i.e.* under the constraint that the number of variables that can be free in a subterm is bounded by some number k. Indeed, because of the subformula property and the bound on the number of free variables, arity-bounded terms of a given type can all be represented with finitely many combinators; this means that we can represent those terms in a tree-HOS Σ and recover them with a homomorphism $\mathcal{H}$. Thus, the set S of terms that are solution of arity bounded $\beta\eta$-HOM can be effectively represented as a recognizable set of trees, namely $\mathcal{H}^{-1}(S)$, the emptiness of recognizable sets of trees being decidable this gives the decidability of arity bounded $\beta\eta$-HOM. In particular, this leads to the decidability of arity bounded $\beta\eta$-HOM. Since arity-bounded $\beta\eta$-HOM is more general than 3^{rd} and 4^{th} order $\beta\eta$-HOM [17], this technique sheds some light on the results obtained by [18] that relate the solutions of these special cases to tree automata. Contrary to most approach to HOM, the one we use is completely direct, we do not need to transform the problem within a set of interpolation equations.

β-HOM [19] is undecidable while $\beta\eta$-HOM seems to be decidable [20]. But there is no satisfying explanation on the difference between β-HOM and $\beta\eta$-HOM so as to account satisfactorily of that difference. But as we have seen, intersection types make a discrimination between β-reduction and $\beta\eta$-reduction with the rule Sub, without which the subject reduction Theorem does not hold for $\beta\eta$-reduction. Thus intersection types seem to be a good tool to investigate this problem.

7 Conclusion and future work

We have defined a notion of recognizability for the λ-calculus that naturally extends recognizability for trees or strings. We have exhibited the closure properties of this notion and showed how it could be exploited to understand parsing of the higher order matching problem. Contrary to strings and trees where recognizability comes with three kinds of characterization, a mechanical one (automata), an algebraic one and a logical one (Monadic Second Order Logic, MSOL), here our notion only comes with a mechanical and an algebraic characterization. It

seems difficult to come up with a logical characterization since this notion is not closed under relabeling. And closure under relabeling is central to represent quantification in MSOL. As we wish to use this notion of recognizability so as to describe particular sets of λ-terms, it would be nice to obtain a connection with some logic. First-order logic would be a first step. A more general question would be whether there is a logic that exactly corresponds to this notion of recognizability.

Another question is to characterize the restrictions under which the emptiness of recognizable sets is decidable. Theorem 9 gives a positive answer when the terms are bound to be generated with a finite set of combinators since it reduces this emptiness problem to the emptiness problem of some recognizable set of trees. But we do not know whether there are weaker constraints for which this holds. When we look at the situation for graphs, there is no class of graphs [21] which can be generated only with infinitely many combinators (this means that the class of graphs has an infinite treewidth) for which this emptiness problem is decidable. Thus, this question can be related to the definition of a suitable notion for normal λ-terms that would be similar to treewidth for graphs.

Finally we hope that the notion of recognizability for λ-terms can be of interest in the study of trees generated by higher-order programming schemes. It has been showed that those trees had a decidable MSO theory[22]. It is likely that intersection types should be more adapted to conduct this proof, and yield to new techniques.

References

1. de Groote, P.: Towards abstract categorial grammars. In for Computational Linguistic, A., ed.: Proceedings 39th Annual Meeting and 10th Conference of the European Chapter, Morgan Kaufmann Publishers (2001) 148–155
2. Montague, R.: Formal Philosophy: Selected Papers of Richard Montague. Yale University Press, New Haven, CT (1974)
3. Salvati, S.: On the membership problem for Non-linear Abstract Categorial Grammars. In Muskens, R., ed.: Proceedings of the Workshop on New Directions in Type-theoretic Grammars (NDTTG 2007), Dublin, Ireland, Foundation of Logic, Language and Information (FoLLI) (August 2007) 43–50
4. Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly **65** (1958) 154–170
5. Moortgat, M.: Categorial Investigations: Logical & Linguistic Aspects of the Lambek Calculus. Foris Pubns USA (1988)
6. Urzyczyn, P.: The emptiness problem for intersection types. J. Symb. Log. **64**(3) (1999) 1195–1215
7. Loader, R.: The undecidability of λ-definability. In Anderson, C.A., Zeleny, M., eds.: Logic, Meaning and Computation: Essays in memory of Alonzo Church. Kluwer (2001) 331–342
8. Statman, R.: Completeness, invariance and λ-definability. Journal of Symbolic Logic **47**(1) (1982) 17–26
9. Statman, R., Dowek, G.: On statman's finite completeness theorem. Technical Report CMU-CS-92-152, University of Carnegie Mellon (1992)

10. Myhill, J.: Finite automata and the representation of events. Technical Report WADC TR-57-624, Wright Patterson Air Force Base, Ohio, USA (1957)
11. Nerode, A.: Linear automaton transformations. In: Proceedings of the American Mathematical Society. Volume 9., American Mathematical Society (1958) 541–544
12. Mezei, J., Wright, J.: Algebraic automata and context-free sets. Informcation and Control **11** (1967) 3–29
13. Dezani-Ciancaglini, M., Giovannetti, E., de' Liguoro, U.: Intersection Types, Lambda-models and Böhm Trees. In: MSJ-Memoir Vol. 2 "Theories of Types and Proofs". Volume 2. Mathematical Society of Japan (1998) 45–97
14. Babaev, A.A., Soloviev, S.V.: Coherence theorem for canonical maps in cartesian closed categories. Journal of Soviet Mathematics **20** (1982)
15. de Groote, P., Pogodalla, S.: On the expressive power of abstract categorial grammars: Representing context-free formalisms. Journal of Logic, Language and Information **13**(4) (2005) 421–438
16. Thatcher, J.W.: Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. Journal of Computer and System Sciences **1**(4) (December 1967) 317–322
17. Schmidt-Schauß, M.: Decidability of arity-bounded higher-order matching. In: CADE-19. LNCS 2741, Springer (2003) 488–502
18. Comon, H., Jurski, Y.: Higher-order matching and tree automata. In: CSL. (1997) 157–176
19. Loader, R.: Higher order β matching is undecidable. Logic Journal of the IGPL **11**(1) (2003) 51–68
20. Stirling, C.: A game-theoretic approach to deciding higher-order matching. In Bugliesi, M., Preneel, B., Sassone, V., Wegener, I., eds.: ICALP (2). Volume 4052 of Lecture Notes in Computer Science., Springer (2006) 348–359
21. Robertson, N., Seymour, P.D.: Graph minors. v. excluding a planar graph. J. Comb. Theory, Ser. B **41**(1) (1986) 92–114
22. Ong, C.H.L.: On model-checking trees generated by higher-order recursion schemes. In: LICS, IEEE Computer Society (2006) 81–90