A Datalog recognizer for almost affine lambda-CFGs

Pierre Bourreau, Sylvain Salvati

▶ **To cite this version:**

Pierre Bourreau, Sylvain Salvati. A Datalog recognizer for almost affine lambda-CFGs. Mathematics of Language, Sep 2011, Nara, Japan. pp.21-38, 10.1007/978-3-642-23211-4 . hal-00740701

HAL Id: hal-00740701
https://hal.science/hal-00740701v1

Submitted on 10 Oct 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Datalog recognizer for almost-affine λ-CFGs

Pierre Bourreau and Sylvain Salvati

LaBRI, Université Bordeaux 1
INRIA Sud-Ouest
{bourreau, salvati}@labri.fr

Abstract. The recent emergence of linguistic formalisms exclusively based on the simply-typed λ-calculus to represent both syntax and semantics led to the presentation of innovative techniques which apply to both the problems of parsing and generating natural languages. A common feature of these techniques consists in using strong relations between typing properties and syntactic structures of families of simply-typed λ-terms. Among significant results, an efficient algorithm based on Datalog programming is presented in [Kan07] for context-free grammar of *almost linear λ-terms*, which are linear λ-terms augmented with a restricted form of copy. We present an extension of this method to terms for which deletion is allowed.

Key words: Parsing, Context-Free λ-Grammars, Abstract categorial grammars, Datalog, Deleting grammars, Typing properties

1 Introduction

Abstract categorial grammars (ACGs) and λ-grammars, introduced independently in [dG01] and [Mus01], are formalisms designed for linguistics purposes and which take their origins in two main ideas: on the one hand Montague's formalization [Mon74] of compositional semantics for natural language based on the λ-calculus; and on the other hand Curry's idea to dissociate the structure of languages (tectogrammar) from their realizations (phenogrammar) [Cur61]. This view on grammatical formalization is further advocated in [Mus10].

In the framework of ACGs and λ-grammars, surface and semantic realizations are both represented and computed by means of simply-typed λ-terms. Devising parsing algorithms in this general context amounts to devise uniform solutions to the problems of parsing and generation for natural languages with the compositional hypothesis. In a similar context, Pogodalla ([Pog00]) gave a first algorithm for generating sentences from meaning representations, and the first one specifically dedicated to λ-grammars and ACGs has been proposed in [Sal05]. The latter has then been extended in [Kan07] and [Sal10]. While the proposal of [Sal10] gives a general algorithmic solution to the parsing/generation problems in the Montagovian framework, Kanazawa's result is mostly concerned with describing a tractable algorithm for some restricted case. In particular, it contains an efficient extension of the Datalog recognizer for context-free grammars of strings [Ull88] to context-free grammars of linear λ-terms (*i.e.* for which

there is no copy or deletion operations) and of almost linear λ-terms (*i.e.* a relaxed form of linear terms for which a restricted form of copy is allowed). The purpose of this paper is to extend Kanazawa's technique to context-free grammars of terms for which the operation of deletion is allowed. Such grammars are context-free grammars of *almost affine λ-terms* (*i.e.* almost linear λ-terms with deletion). Yoshinaka [Yos06] has proved that allowing deletion in λ-grammars does not essentially improve the expressive power of non-deleting λ-grammars. Nevertheless, his construction gives rise to non-deleting grammars the size of which may be exponential with respect to the size of the original deleting grammars.

The central theorem in Kanazawa's method is that almost linear terms are the unique inhabitants of their most general typings; it was recently proved that this result can actually be extended to almost affine terms [Kan10,BS11]. It is hence natural to see whether it is possible to build a Datalog recognizer based on Kanazawa's method for grammars of almost affine terms. From the perspective of grammar design, the addition of deletion allows enhancing the flexibility in which entries can be represented and in particular, it allows some contextual information to flow across a derivation. For example, it permits to handle agreement with techniques similar to the ones used in the Grammatical Framework [Ran09]; it also gives the possibility of implementing certain ideas from lexical semantics [BMR10] in order to disambiguate lexical entries in their semantic interpretation thanks to contextual information.

The paper is structured as follows: section 2 introduces the notion of context-free λ-grammars; in section 3, we present a restricted intersection type system in which we study the typing properties of almost affine λ-terms that are given in section 4. The final section is dedicated to presenting Datalog recognizers for context-free grammars of almost affine λ-terms, as an extension of the programs given in [Kan07].

2 Context-free λ-grammars

Given a set of atomic types A, the set of simple types $\mathcal{T}(A)$ on A is defined as the closure of A by the right-associative operator $\rightarrow$. To a type $\alpha \in \mathcal{T}(A)$, we associate its order defined as $\mathtt{ord}(\alpha) = 1$ if α belongs to A and $\mathtt{ord}(\alpha) = max(1 + \mathtt{ord}(\alpha_1), \mathtt{ord}(\alpha_2))$ if $\alpha = \alpha_1 \rightarrow \alpha_2$. We also inductively define the *set of positions in α*, $\mathcal{P}(\alpha)$, as the prefix-closed finite set of sequences of naturals numbers given by $\mathcal{P}(\alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow a) = \{\epsilon\} \cup \bigcup_{i \in \{1,\ldots,n\}} \{i \cdot s \mid s \in \mathcal{P}(\alpha_i)\}$ where ϵ is the empty sequence and $\_ \cdot \_$ is the operation of concatenation of sequences (we write $\mathbb{N}^*$ for the set of sequences of natural numbers). Given $\alpha = \alpha_1 \rightarrow \ldots \rightarrow \alpha_n \rightarrow a$, to each position s in $\mathcal{P}(\alpha)$ we associate an atomic type $\mathrm{at}(s, \alpha)$, such that $\mathrm{at}(\epsilon, \alpha) = a$ and $\mathrm{at}(i \cdot s, \alpha) = \mathrm{at}(s, \alpha_i)$ (i being in $\{1, \ldots, n\}$); when $\mathrm{at}(s, \alpha) = a$ we say that *a is the atomic type at position s in α*. The notion of position in types is inspired from games semantics for the simply typed λ-calculus (see [BS11] for more details).

Given two sets of atomic types A and B, a *type substitution* is a homomorphism σ from $\mathcal{T}(A)$ to $\mathcal{T}(B)$ (noted $\sigma : \mathcal{T}(A) \mapsto \mathcal{T}(B)$), *i.e.* a function such that $\sigma(\alpha \to \beta) = \sigma(\alpha) \to \sigma(\beta)$. A *type substitution* σ is a *relabeling* when for every atomic type in A, its image by σ is atomic, *i.e.* belongs to B. A type relabeling is a *type renaming* when it is bijective. Given two type substitutions σ_1 and σ_2 from $\mathcal{T}(A)$ to $\mathcal{T}(B)$, σ_1 is said *more general* than σ_2 when there is a type substitution $\sigma : \mathcal{T}(B) \mapsto \mathcal{T}(B)$ such that $\sigma_2 = \sigma \circ \sigma_1$. Given two types α_1 and α_2 in $\mathcal{T}(A)$, they are *unifiable* when there is a type substitution σ such that $\sigma(\alpha_1) = \sigma(\alpha_2)$. In such a case, σ is said to *unify* α_1 and α_2 and is called a *unifier of* α_1 *and* α_2. In case two types are unifiable, it is well known that the set of their unifiers contains most general ones, and that most general unifiers are all equivalent up to renaming. In general we will call a most general unifier *the most general unifier*, considering that we work with type substitution up to composition with renamings.

Given a set of constants $\mathcal{C}$ (where constants are written $\mathbf{c}, \mathbf{c_1}, \ldots$) and a set of variables $\mathcal{V}$ (where variables are written $x, y, x_1, \ldots$), we define the set of λ-terms $\Lambda ::= \mathcal{V} \mid \mathcal{C} \mid \lambda\mathcal{V}.\Lambda \mid (\Lambda\Lambda)$. The usual conventions that limit the number of parentheses and sequences of λ's in the spelling of λ-terms are adopted. We also take for granted the notions of set of free variables of a term M (noted $FV(M)$); the set of constants in M is noted $Cst(M)$. A term M is *closed* when $FV(M) = \emptyset$. We consider terms typed *à la Church* so that variables are explicitly carrying their types as exponents, but for the sake of readability, we will often omit this typing notation when it is unnecessary to the understanding.

A *higher-order signature* (HOS) $\Sigma = (A, \mathcal{C}, \tau)$ is a tuple made of a finite set of atomic types A, a finite set of constants $\mathcal{C}$ and a funtion τ which associates types in $\mathcal{T}(A)$ to constants in $\mathcal{C}$. The order of a higher-order signature $\Sigma = (A, \mathcal{C}, \tau)$ is defined as $max_{\mathbf{c}\in\mathcal{C}}(\mathtt{ord}(\tau(\mathbf{c})))$. We now define $(\Lambda_\Sigma^\alpha)_{\alpha\in\mathcal{T}(A)}$ as the family of the smallest sets verifying:

1. $x^\alpha \in \Lambda_\Sigma^\alpha$ and $\mathbf{c} \in \Lambda_\Sigma^{\mathcal{T}(c)}$
2. if $M \in \Lambda_\Sigma^\beta$, then $\lambda x^\alpha.M \in \Lambda_\Sigma^{\alpha\to\beta}$,
3. if $M_1 \in \Lambda_\Sigma^{\beta\to\alpha}$, $M_2 \in \Lambda_\Sigma^\beta$, then $M = M_1M_2$.

We now define *linear*, *syntactically almost linear* and *syntactically almost affine* λ-terms:

1. x^α and $\mathbf{c}$ are linear, syntactically almost linear and syntactically almost affine,
2. given $M_1 \in \Lambda_\Sigma^{\alpha\to\beta}$ and $M_2 \in \Lambda_\Sigma^\alpha$, (M_1M_2) is linear (*resp.* syntactically almost linear, *resp.* syntactically almost affine) when M_1 and M_2 are both linear (*resp.* syntactically almost linear, *resp.* syntactically almost affine) and $FV(M_1) \cap FV(M_2) = \emptyset$ (*resp.* if $x^\gamma \in FV(M_1) \cap FV(M_2)$ then γ is atomic),
3. given M in Λ_Σ^β, $\lambda x^\alpha.M$ of $\Lambda_\Sigma^{\alpha\to\beta}$ is (syntactically almost) linear if M is (syntactically almost) linear and $x^\alpha \in FV(M)$. $\lambda x^\alpha.M$ is syntactically almost affine when M is syntactically almost affine.

Remark that linear terms are syntactically almost linear; also, syntactically almost linear terms are syntactically almost affine. When M is linear and $M \twoheadrightarrow_\beta M'$

then M' is also linear. But, in case M is syntactically almost linear or syntactically almost affine, it may be the case that M' is not syntactically almost linear or syntactically almost affine: while $\lambda f^{(o\to o)\to o}.(\lambda y^o.x^{o\to o\to o}y^o y^o)(f(\lambda z^o.z^o))$ is both syntactically almost linear and syntactically almost affine, its normal form $\lambda f^{(o\to o)\to o}.x^{o\to o\to o}(f(\lambda z^o.z^o))(f(\lambda z^o.z^o))$ is neither syntactically almost linear nor syntactically almost affine. Thus, we say that M is *almost linear* (*resp. almost affine*) when there is a λ-term M' that is syntactically almost linear (*resp.* syntactically almost affine) such that $M'\twoheadrightarrow_\beta M$. Remark that every linear term is both almost linear and almost affine and that every almost linear term is also almost affine.

Given two HOS $\Sigma_1 = (A_1, \mathcal{C}_1, \tau_1)$ and $\Sigma_2 = (A_2, \mathcal{C}_2, \tau_2)$, a homomorphism $\mathcal{H}$ between Σ_1 and Σ_2 is a function that maps $\mathcal{T}(A_1)$ to $\mathcal{T}(A_2)$, $\Lambda^\alpha_{\Sigma_1}$ to $\Lambda^{\mathcal{H}(\alpha)}_{\Sigma_2}$ for every $\alpha \in \mathcal{T}(A_1)$ and verifies:

1. $\mathcal{H}(\alpha \to \beta) = \mathcal{H}(\alpha) \to \mathcal{H}(\beta)$,
2. $\mathcal{H}(\lambda x^\alpha.M) = \lambda\mathcal{H}(x^\alpha).\mathcal{H}(M)$, $\mathcal{H}(MN) = \mathcal{H}(M)\mathcal{H}(N)$ and $\mathcal{H}(x^\alpha) = x^{\mathcal{H}(\alpha)}$,
3. $\mathcal{H}(\mathbf{c})$ is a closed λ-term of $\Lambda^{\mathcal{H}(\tau_1(c))}_{\Sigma_2}$.

A *context-free λ-grammar* (λ-CFG) $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{H}, s)$ is a tuple where:

1. $\Sigma_1 = (A_1, \mathcal{C}_1, \tau_1)$ is a second-order signature and $\Sigma_2 = (A_2, \mathcal{C}_2, \tau_2)$ a HOS, respectively called the *abstract* and the *object* signatures of $\mathcal{G}$.
2. $\mathcal{H}$ is a homomorphism between Σ_1 and Σ_2, called *the lexicon*.
3. $s \in A_1$ is *the distinguished type*.

This particular class of ACG is called context-free because Σ_1 is bound to be a second-order signature. Indeed such ACGs have derivation structures that are the same as context-free languages.

Given a λ-CFG $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{H}, s)$, we define, its *abstract language* as $\mathcal{A}(\mathcal{G}) = \{M \in \Lambda^s_{\Sigma_1} \mid FV(M) = \emptyset\}$ and its *object language* as $\mathcal{O}(\mathcal{G}) = \{M \in \Lambda^{\mathcal{H}(s)}_{\Sigma_2} \mid \exists M' \in \mathcal{A}(\mathcal{G}),\ |\mathcal{H}(M')|_\beta = M\}$. A λ-CFG $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{H}, s)$ is said *linear* (*resp. almost linear*, *resp. almost affine*) when for each constant $\mathbf{c}$ in the abstract signature of $\mathcal{G}$, $\mathcal{H}(\mathbf{c})$ is a linear (*resp.* almost linear, *resp.* almost affine) term. While the original definition of λ-CFGs [dG01,Mus01] corresponds to linear λ-CFGs, extensions to affine (*i.e.* with deletion and no copy) and almost linear (*i.e.* with limited copy and no deletion) λ-CFGs were introduced in [Yos06] and [Kan07] respectively. An almost affine λ-CFG is a combination of the two previous extensions of the original definition.

3 Listed Types

We here present a type system that allows to assign a restricted form of intersection types to simply typed λ-terms in the spirit of [Sal10]. Mostly we use this second layer of typing for two purposes. First, it allows to define a notion of most general typing for terms typed *à la Church* that slightly differs from the most general typing one would get by dropping the type annotations. For example, we

get the type $(a \to b) \to c$ as the most general type for the term $\lambda x^{o \to o}.y^o$, while the most general typing of $\lambda x.y$ would be $a \to b$. Second, this restricted form of intersection types allows us to assign informative types to constants that are deleted during β-reduction.

Given two countable sets of atomic types A and B, we define $(\mathcal{U}_\alpha(B))_{\alpha \in \mathcal{T}(A)}$, *where* $\mathcal{U}_\alpha(B)$ *is the set of types built on* B *and uniform with* α, to be the least subsets of $\mathcal{T}(B)$ that verify the following identities:

1. $\mathcal{U}_\alpha(B) = B$ when α is atomic,
2. $\mathcal{U}_{\alpha \to \beta}(B) = \{\gamma \to \delta \mid \gamma \in \mathcal{U}_\alpha(B) \text{ and } \delta \in \mathcal{U}_\beta(B)\}$.

Lemma 1. *Two types* γ_1 *and* γ_2 *in* $\mathcal{U}_\alpha(B)$ *are always unifiable and their most general unifier is a relabeling.*

Proof. A simple induction on the structure of α.

We now define the family $(\mathcal{L}_\alpha(B))_{\alpha \in \mathcal{T}(A)}$, *where* $\mathcal{L}_\alpha(B)$ *is the set of listed types built on* B *and uniform with* α, as the smallest sets such that:

1. $\mathcal{U}_\alpha(B) \subseteq \mathcal{L}_\alpha(B)$,
2. if l_1 and l_2 are in $\mathcal{L}_\alpha(B)$ then $l_1 \cap l_2$ are in $\mathcal{L}_\alpha(B)$.

Intuitively, the elements of $\mathcal{L}_\alpha(B)$ are intersection types without the universal type [CDC80] where the use of the intersection is restricted to the most external level in order to combine types that are uniform with α. While atomic types will be noted by small roman letters $a, b, c, \ldots$, and simple types by small greek letters $\alpha, \beta, \gamma, \ldots$, listed types will be written as overlined small greek letters $\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \ldots$ As for intersection types, $\cap$ is associative, commutative and idempotent. We may therefore confuse the elements of $\mathcal{L}_\alpha(B)$ with non-empty finite subsets of $\mathcal{U}_\alpha(B)$ and we use the notation $\delta \in \overline{\gamma}, \overline{\gamma_1} \subseteq \overline{\gamma_2}$ and $\overline{\gamma_1} = \overline{\gamma_2}$ with their obvious meanings. In the sequel, we will need to represent listed types that are built as intersection of types that only differ on specific positions; as a shorthand, we shall write certain listed types as $\{a; b\} \to \{c; d\} \to e$ so as to denote the listed type $a \to c \to e \cap a \to d \to e \cap b \to c \to e \cap b \to d \to e$.

Given α such that $\overline{\gamma} = \gamma_1 \cap \ldots \cap \gamma_n$ belongs to $\mathcal{L}_\alpha(B)$, we write $\mathcal{P}(\overline{\gamma})$, *the set of positions in* $\overline{\gamma}$, for $\mathcal{P}(\alpha) \times \{1, \ldots, n\}$; and for (s, k) in $\mathcal{P}(\overline{\gamma})$, we write $\mathrm{at}((s, k), \overline{\gamma})$ for $\mathrm{at}(s, \gamma_k)$.

Let us fix a HOS $\Sigma = (A, C, \tau)$ and a countable set of atomic types B; a *typing environment* is a pair $\langle \Gamma; \Delta \rangle$, where:

1. Γ is a partial function (with a finite domain denoted by $\mathrm{Dom}(\Gamma)$) that maps constants to listed types so that $\Gamma(\mathbf{c})$ is in $\mathcal{L}_{\tau(\mathbf{c})}(B)$,
2. Δ is a partial function (with a finite domain denoted by $\mathrm{Dom}(\Delta)$) that maps variables to uniform types so that $\Delta(x^\alpha)$ is in $\mathcal{U}_\alpha(B)$.

As it is usual, we write both components of a typing environment as sequences; when writing $\mathbf{c_1} : \overline{\gamma_1}, \ldots, \mathbf{c_n} : \overline{\gamma_n}$ for Γ, we mean that the domain of Γ is the set $\{\mathbf{c_1}, \ldots, \mathbf{c_n}\}$ and that for all i in $\{1, \ldots, n\}$, $\Gamma(\mathbf{c_i}) = \overline{\gamma_i}$; the same convention

$$\frac{\alpha \in \Gamma(\mathbf{c})}{\langle \Gamma; \Delta \rangle \vdash \mathbf{c} : \alpha}\ \text{AxC} \qquad \frac{\Delta(x^\alpha) = \gamma}{\langle \Gamma; \Delta \rangle \vdash x^\alpha : \gamma}\ \text{AxV}$$

$$\frac{\langle \Gamma; \Delta, x^\alpha : \gamma \rangle \vdash M : \delta}{\langle \Gamma; \Delta \rangle \vdash \lambda x^\alpha . M : \gamma \to \delta}\ \text{Abs}$$

$$\frac{\langle \Gamma; \Delta \rangle \vdash M_1 : \gamma \to \delta \quad \langle \Gamma; \Delta \rangle \vdash M_2 : \gamma}{\langle \Gamma; \Delta \rangle \vdash M_1 M_2 : \delta}\ \text{App}$$

Fig. 1. Derivation system for listed types

is adopted for the type assignments $x_1^{\alpha_1} : \delta_1, \ldots x_m^{\alpha_m} : \delta_m$ in Δ. We write '_' for an empty typing environment. A *typing pair* is a pair noted $\langle \Gamma; \Delta \rangle \vdash \gamma$ where $\langle \Gamma; \Delta \rangle$ is a typing environment and where γ is an element of $\mathcal{T}(B)$. Given M in Λ_Σ^α, we say that M is an *inhabitant of the typing pair* $\langle \Gamma; \Delta \rangle \vdash \gamma$ or that $\langle \Gamma; \Delta \rangle \vdash \gamma$ *is a typing of* M, when $\langle \Gamma; \Delta \rangle \vdash M : \gamma$ is derivable with the rules of Figure 1. Remark that when $M \in \Lambda_\Sigma^\alpha$ is an inhabitant of $\langle \Gamma; \Delta \rangle \vdash \gamma$ then γ is an element of $\mathcal{U}_\alpha(B)$.

In order to have a homogeneous notation for positions in typing pairs, positions in a type $\gamma \in \mathcal{U}_\alpha(B)$ will be noted $\{(s,1) \mid s \in \mathcal{P}(\alpha)\}$. The set of positions of $\langle \Gamma; \Delta \rangle \vdash \gamma$ is defined as $\mathcal{P}(\langle \Gamma; \Delta \rangle \vdash \gamma) = \{(\mathbf{c}, p) \mid \mathbf{c} \in \mathrm{Dom}(\Gamma), p \in \mathcal{P}(\Gamma(\mathbf{c}))\} \cup \{(x^\alpha, p) \mid x^\alpha \in \mathrm{Dom}(\Delta), p \in \mathcal{P}(\Delta(x^\alpha))\} \cup \{(\top, p) \mid p \in \mathcal{P}(\gamma)\}$; every position in $\mathcal{P}(\langle \Gamma; \Delta \rangle \vdash \gamma)$ is therefore of the form (h, p) where h is either a constant, a variable or the special symbol $\top$. Finally, we define $\mathrm{at}((h,p), \langle \Gamma; \Delta \rangle \vdash \gamma)$ to be $\mathrm{at}(p, \gamma)$ if $h = \top$, $\mathrm{at}(p, \Delta(x^\alpha))$ if $h = x^\alpha$ and $\mathrm{at}(p, \Gamma(\mathbf{c}))$ if $h = \mathbf{c}$.

Note that, because variables and constants can only be assigned types of a specific shape, the set of typings of a term M with listed types is not closed under substitution as it is the case for the typing judgements that are derivable with simple types. Instead, the set of typings with listed types that can be derived for a term M are closed under *relabelings*. A typing $\langle \Gamma_1; \Delta_1 \rangle \vdash \gamma_1$ is said *more general* than a typing $\langle \Gamma_2; \Delta_2 \rangle \vdash \gamma_2$ when there is a relabeling σ such that:

1. $\gamma_2 = \gamma_1.\sigma$,
2. for every x^α in $\mathrm{Dom}(\Delta_1)$, x^α is in $\mathrm{Dom}(\Delta_2)$ and $\Delta_2(x^\alpha) = \Delta_1(x^\alpha).\sigma$,
3. for every $\mathbf{c}$ in $\mathrm{Dom}(\Gamma_1)$, $\mathbf{c}$ is in $\mathrm{Dom}(\Gamma_2)$ and $\Gamma_1(\mathbf{c}).\sigma \subseteq \Gamma_2(\mathbf{c})$.

Moreover, if $\langle \Gamma_2; \Delta_2 \rangle \vdash \gamma_2$ is not more general than $\langle \Gamma_1; \Delta_1 \rangle \vdash \gamma_1$, we say that $\langle \Gamma_1; \Delta_1 \rangle \vdash \gamma_1$ is *strictly more general* $\langle \Gamma_2; \Delta_2 \rangle \vdash \gamma_2$.

Obviously, if $\langle \Delta_1; \Gamma_1 \rangle \vdash \gamma_1$ is more general than $\langle \Delta_2; \Gamma_2 \rangle \vdash \gamma_2$, whenever M is an inhabitant of $\langle \Delta_1; \Gamma_1 \rangle \vdash \gamma_1$, it is also an inhabitant of $\langle \Delta_2; \Gamma_2 \rangle \vdash \gamma_2$. It can easily be proved that for every term M there is a most general typing in the set of typings of M. However, this most general typing is not unique (even up to renaming) simply because we may assign an arbitrarily large listed type to any constant (see example 1). Nevertheless, when introducing an order on the

most general typings of a term M with respect to the number of different atomic types that occur in them, it appears that the smallest ones are all equivalent up to renaming. Thus, working up to renaming of atomic types, we call such a typing *the most general listed typing of* M or the *mglt of* M. The mglt of M can be obtained by replacing each occurrence of a constant in M by a variable (the resulting term is noted *c-linear(M)*), computing the most general typing with simple types (taking into account the shape constraint imposed by listed types), and then type each constant with the intersection of the types assigned to the variables that are replacing its occurrences.

Example 1. Given, two constants $\mathbf{f}$ and $\mathbf{c}$ of respective type $o \to o$ and o, the mglt of $M = \lambda P^{o\to o}.\mathbf{f}((\lambda x^o.\mathbf{f}\mathbf{c})(P^{o\to o}\mathbf{c}))$ is $\langle \mathbf{c} : a_1 \cap a_2, \mathbf{f} : a_1 \to b_1 \cap b_1 \to b_2; \_\rangle \vdash (a_2 \to c) \to b_2$. Note that $\langle \mathbf{c} : a_1 \cap a_2 \cap d, \mathbf{f} : a_1 \to b_1 \cap b_1 \to b_2 \cap e_1 \to e_2; \_\rangle \vdash (a_2 \to c) \to b_2$ is also a most general typing of M.

Lemma 2. *If* $\langle \Gamma_1; \Delta_1\rangle \vdash \gamma_1$ *is a typing of term* M *and is more general than* $\langle \Gamma_2; \Delta_2\rangle \vdash \gamma_2$ *then* $\langle \Gamma_2; \Delta_2\rangle \vdash \gamma_2$ *is a typing of* M.

Proof. Simple induction on the structure of M.

The following properties are naturally inherited from derivations of the terms in the simply-typed λ-calculus.

Property 1. Given a term M, every typing of M is less general than its mglt.

Property 2. (Subject Reduction) Given terms M and M' such that $M' \twoheadrightarrow_\beta M$, every typing of M' is a typing of M.

4 Typing properties of almost affine terms

Throughout this section we assume that λ-terms are built on a HOS $\Sigma = (A, C, \tau)$ and that listed types are built on a countable set B of atomic types.

4.1 Negatively non-duplicating typings are not sufficient

In [Aot99], it was proved that all the terms N that are typable with a given negatively non-duplicating typing are $\beta\eta$-equivalent. This result is obtained on terms that do not contain constants. We here adapt this correspondence to the case of λ-terms that contain occurrences of constants. Let us first introduce *negatively non-duplicating listed typings* and the notion of *polarity*. Given a typing $\langle \Gamma; \Delta\rangle \vdash \gamma$, the set of *positive* (*resp. negative*) positions $\mathcal{P}^+(\langle \Gamma; \Delta\rangle \vdash \gamma)$ (*resp.* $\mathcal{P}^-(\langle \Gamma; \Delta\rangle \vdash \gamma)$) of $\langle \Gamma; \Delta\rangle \vdash \gamma$ is the set of positions $(h, (s, k))$ such that, h is either a constant or a variables and s has an odd (*resp.* even) length and of positions $(\top, (s, k))$ such that s has an even (*resp.* odd) length. We also write $\mathcal{P}^\theta_a\langle \Gamma; \Delta\rangle \vdash \gamma$ (with θ in $\{-; +\}$)for the set $\{p \mid \mathcal{P}^\theta\langle \Gamma; \Delta\rangle \vdash \gamma \wedge at(p, \langle \Gamma; \Delta\rangle \vdash \gamma) = a\}$. A *negatively non-duplicating* typing $\langle \Gamma; \Delta\rangle \vdash \gamma$ is a typing such that for every atomic type a, the number of elements of $\mathcal{P}^-_a\langle \Gamma; \Delta\rangle \vdash \gamma$ is less than 1.

From [Kan10] and [BS11] it can easily be derived that the mglt of a β-normal term M is negatively non-duplicating iff M is almost affine. This result can also be, rather easily, generalized to any almost affine term.

Theorem 1. *If a term is almost affine then its most general listed typing is negatively non-duplicating.*

We now briefly outline how Kanazawa [Kan07] exploited negatively non-duplicating typings so as to build a Datalog program for solving the recognition problem for almost linear λ-CFG and show on an example why a naive translation of this method cannot succeed for almost affine λ-CFG.

Given an almost linear term M, its mglt is a negatively non-duplicating typing, and it is also the case that the almost linear terms N that are $\beta\eta$-convertible to M also have a negatively non-duplicating mglt. So if $\langle \Gamma; \Delta \rangle \vdash \gamma$ is the mglt of M, and $\langle \Gamma'; \Delta' \rangle \vdash \gamma'$ is that of N, $\langle \Gamma'; \Delta' \rangle \vdash \gamma'$ is a typing of M by subject reduction, while $\langle \Gamma; \Delta \rangle \vdash \gamma$ might not be a typing of N. This problem can be overcome by pointing out that M has a negatively non-duplicating typing $\langle \Gamma_l; \Delta_l \rangle \vdash \gamma_l$ that is less general than any other of its negatively non-duplicating typings. Thus, because $\langle \Gamma'; \Delta' \rangle \vdash \gamma'$ is a negatively non-duplicating typing of M, $\langle \Gamma_l; \Delta_l \rangle \vdash \gamma_l$ is less general than $\langle \Gamma'; \Delta' \rangle \vdash \gamma'$ and is therefore a typing of N. As a conclusion, $\langle \Gamma_l; \Delta_l \rangle \vdash \gamma_l$ is a negatively non-duplicating typing of M that is also a typing of all the almost linear terms that are $\beta\eta$-convertible to M. Kanazawa's technique [Kan07] is precisely based on this property, and for an almost linear λ-CFG $\mathcal{G}$, he builds a datalog program that can compute the typings of every term in the language of $\mathcal{O}(G)$. Thus given an almost linear term M, verifying whether M is in $\mathcal{O}(G)$ amounts to find whether there is a term in $\mathcal{O}(G)$ that can be typed with the least general negatively non-duplicating typing of M. This is done by querying the Datalog program on an extensional database that represents the least general negatively non-duplicating typing of M.

When trying to extend this technique to almost affine λ-CFG, we face the problem that given an almost affine term M in β-normal form, it is not possible to find a negatively non-duplicating typing of M that is a typing of all the almost affine λ-terms that are $\beta\eta$-convertible to M.

Example 2. Let us consider the affine term $M = \lambda f^{o \to o} x^o y^o . \mathbf{c_1} x^o y^o$ (in the sequel, we will write it $\lambda f x y . \mathbf{c_1}\, x\, y$) built on the signature Σ that declares two constants $\mathbf{c_1}$ and $\mathbf{c_2}$, both of type $o \to o \to o$. The mglt of M is $\mathbf{c_1} : a \to b \to c \vdash (u \to v) \to a \to b \to c$. We are now going to consider two almost affine terms N_1 and N_2 that are $\beta\eta$-convertible to M, and have a look at their mglt:

1. $N_1 = \lambda f x y . (\lambda g . \mathbf{c_1}\, x\, y)(f\, x)$ with $\mathbf{c_1} : a \to b \to c \vdash (a \to v) \to a \to b \to c$ as mglt,
2. $N_2 = \lambda f x y . (\lambda g . \mathbf{c_1}\, x\, y)(f\, y)$ with $\mathbf{c_1} : a \to b \to c \vdash (b \to v) \to a \to b \to c$ as mglt.

If we take the least general negatively non-duplicating typing of M, *i.e.* $\mathbf{c_1} : a \to b \to c \vdash (v \to v) \to a \to b \to c$ it is neither a typing for N_1 nor for N_2. And,

$\mathbf{c_1} : d \to d \to c \vdash (d \to v) \to d \to d \to c$, the most general typing that is both a typing of N_1 and N_2, is not negatively non-duplicating. As a consequence, it is inhabited by terms different from M like $\lambda f x y.\mathbf{c_1}\, y\, x$.

Furthermore in the case of almost linear λ-terms, the least general negatively non-duplicating typing of an almost linear λ-term M is also the least general typing that has M as unique inhabitant. This is no longer the case with almost affine λ-terms: first, there may be several least general typing for which M is the unique inhabitant; second, such a typing may not be negatively non-duplicating.

Example 3. Considering the term M as in example 2, amongst the least general typings for which M is the unique inhabitant, there is: $\mathbf{c_1} : a \to b \to c \cap v \to a \to v \vdash (a \to v) \to a \to b \to c$ and $\mathbf{c_1} : a \to b \to c$, $\mathbf{c_2} : v \to a \to v \vdash (b \to v) \to a \to b \to c$ and none is negatively non-duplicating because they both contain two negative occurrences of v.

4.2 Potentially negatively non-duplicating typings

Given an almost affine term M, even if its mglt is negatively non-duplicating, there exist typings with weaker syntactic constraints that still fully characterize M. These syntactic constraints are close to the ones of negatively non-duplicating typings. We need new concepts to decipher and explain these constraints.

Given a typing $\langle \Gamma ; \Delta \rangle \vdash \gamma$, and p_1, p_2 in $\mathcal{P}(\langle \Gamma ; \Delta \rangle \vdash \gamma)$, p_1 *enables* p_2 if:

1. either, $p_1 = (h, (s_1, k))$ and $p_2 = (h, (s_1 \cdot i, k))$ for some $i \in \mathbb{N}$,
2. or, $p_1 = (\top, (\epsilon, 1))$ and, either $p_2 = (\boldsymbol{c}, (\epsilon, k))$ or $p_2 = (x^\alpha, (\epsilon, 1))$.

For p in $\mathcal{P}(\langle \Gamma ; \Delta \rangle \vdash \gamma)$, we write $\mathrm{en}(p)$ the set of positions that are enabled by p. Furthermore, given p in $\mathcal{P}(\langle \Gamma ; \Delta \rangle \vdash \gamma)$ and $\mathcal{Q} \subseteq \mathcal{P}(\langle \Gamma ; \Delta \rangle \vdash \gamma)$ we write $\mathrm{eq}(\mathcal{Q}, p)$ for the set $\{q \in \mathcal{Q} \mid \mathrm{at}(q, \langle \Gamma ; \Delta \rangle \vdash \gamma) = \mathrm{at}(p, \langle \Gamma ; \Delta \rangle \vdash \gamma)\}$. We now define, $\mathtt{Pot}(\langle \Gamma ; \Delta \rangle \vdash \gamma)$ the set of *potential positions of the pair* $\langle \Gamma ; \Delta \rangle \vdash \gamma$ by:

$$\mathrm{Pot}(\langle \Gamma ; \Delta \rangle \vdash \gamma) = \mathtt{Pot}(\mathrm{en}(\top, \epsilon), (\top, \epsilon))$$

where $\mathtt{Pot}(\mathcal{Q}, p)$ is defined as the smallest set such that:

$$\mathtt{Pot}(\mathcal{Q}, p) = \{p\} \cup \bigcup_{q \in \mathrm{eq}(\mathcal{Q}, p)} \left(\{q\} \cup \bigcup_{r \in \mathrm{en}(q)} \mathtt{Pot}(\mathcal{Q} \cup \mathrm{en}(r), r) \right)$$

$\mathtt{Pot}(\mathcal{Q}, p)$ approximates the derivation system in Figure 1 at the level of positions. Intuitively, it builds the set of positions that correspond to subformulae of the typing $\langle \Gamma ; \Delta \rangle \vdash \gamma$ that may be used in an axiom rule in a derivation. Thus, if we let $\mathrm{Irr}(\langle \Gamma ; \Delta \rangle \vdash \gamma)$ be $\mathcal{P}(\langle \Gamma ; \Delta \rangle \vdash \gamma) - \mathrm{Pot}(\langle \Gamma ; \Delta \rangle \vdash \gamma)$, it is a set of positions that cannot be used in any derivation of a judgement of the form $\langle \Gamma ; \Delta \rangle \vdash M : \overline{\gamma}$ when M is in normal form. For θ in $\{-; +\}$, we will use the following notations:

1. $\mathrm{Irr}^\theta(\langle \Gamma ; \Delta \rangle \vdash \gamma)$ for $\mathrm{Irr}(\langle \Gamma ; \Delta \rangle \vdash \gamma) \cap \mathcal{P}^\theta(\langle \Gamma ; \Delta \rangle \vdash \gamma)$, and $\mathrm{Irr}^\theta_a(\langle \Gamma ; \Delta \rangle \vdash \gamma)$ for $\mathrm{Irr}(\langle \Gamma ; \Delta \rangle \vdash \gamma) \cap \mathcal{P}^\theta_a(\langle \Gamma ; \Delta \rangle \vdash \gamma)$

2. $\mathrm{Pot}^{\theta}(\langle \Gamma; \Delta\rangle \vdash \gamma)$ for $\mathrm{Pot}(\langle \Gamma; \Delta\rangle \vdash \gamma) \cap \mathcal{P}^{\theta}(\langle \Gamma; \Delta\rangle \vdash \gamma)$, and $\mathrm{Pot}^{\theta}_{a}(\langle \Gamma; \Delta\rangle \vdash \gamma)$ for $\mathrm{Pot}(\langle \Gamma; \Delta\rangle \vdash \gamma) \cap \mathcal{P}^{\theta}_{a}(\langle \Gamma; \Delta\rangle \vdash \gamma)$

Property 3. For every irrelevant position p in the mglt $\langle \Gamma; \Delta\rangle \vdash \gamma$ of a β-normal term M, every position $p' \neq p \in \mathrm{Irr}(\langle \Gamma; \Delta\rangle \vdash \gamma)$ verifies $\mathrm{at}(p, \langle \Gamma; \Delta\rangle \vdash \gamma) \neq \mathrm{at}(p', \langle \Gamma; \Delta\rangle \vdash \gamma)$.

If $\Gamma(\boldsymbol{c}) = \gamma_1 \cap \ldots \cap \gamma_n$, we say that *the k^{th} simple type assigned to $\boldsymbol{c}$ is equivalent to its l^{th} simple type* when for every $p_1 = (\boldsymbol{c}, (s, k))$ and $p_2 = (\boldsymbol{c}, (s, l))$, the following properties hold:

1. $p_1 \in \mathrm{Pot}(\langle \Gamma; \Delta\rangle \vdash \gamma)$ iff $p_2 \in \mathrm{Pot}(\langle \Gamma; \Delta\rangle \vdash \gamma)$,
2. if $p_1 \in \mathrm{Pot}(\langle \Gamma; \Delta\rangle \vdash \gamma)$, then $\mathrm{at}(p_1, \langle \Gamma; \Delta\rangle \vdash \gamma) = \mathrm{at}(p_2, \langle \Gamma; \Delta\rangle \vdash \gamma)$

In case the k^{th} simple type assigned to $\boldsymbol{c}$ is equivalent to its l^{th} simple type and $p_1 = (\boldsymbol{c}, (s, k))$ and $p_2 = (\boldsymbol{c}, (s, l))$, we write $p_1 \equiv p_2$. The smallest equivalence relation on $\mathcal{P}(\langle \Gamma; \Delta\rangle \vdash \gamma)$ that contains the relation $\equiv$ is written $\approx$. Remark that whenever $p_1 \approx p_2$, p_1 and p_2 have the same polarity so that we can extend in the obvious way the notion of polarity to the equivalence classes of $\mathcal{P}(\langle \Gamma; \Delta\rangle \vdash \gamma)/\approx$. Furthermore, given $\mathbf{p}$ in $\mathcal{P}(\langle \Gamma; \Delta\rangle \vdash \gamma)/\approx$ either all the elements of $\mathbf{p}$ belong to $\mathrm{Pot}(\langle \Gamma; \Delta\rangle \vdash \gamma)$, in which case we say that $\mathbf{p}$ is *potential*, or they all belong to $\mathrm{Irr}(\langle \Gamma; \Delta\rangle \vdash \gamma)$, in which case we say that $\mathbf{p}$ is *irrelevant*. Note that, if $\mathbf{p}$ is potential, for every p_1, p_2 in $\mathbf{p}$ we have $\mathrm{at}(p_1, \langle \Gamma; \Delta\rangle \vdash \gamma) = \mathrm{at}(p_2, \langle \Gamma; \Delta\rangle \vdash \gamma)$ so that we may write $\mathrm{at}(\mathbf{p}, \langle \Gamma; \Delta\rangle \vdash \gamma)$ for the atomic type associated to the elements of $\mathbf{p}$; also, if $\mathbf{p}$ is irrelevant, we may have p_1 and p_2 in $\mathbf{p}$ such that $\mathrm{at}(p_1, \langle \Gamma; \Delta\rangle \vdash \gamma) \neq \mathrm{at}(p_2, \langle \Gamma; \Delta\rangle \vdash \gamma)$. Thus given an atomic type a, we call the *number of potential positive (resp. negative) occurrences of a in* $\langle \Gamma; \Delta\rangle \vdash \gamma$ the number of potential positive (*resp.* negative) equivalence classes $\mathbf{p}$ in $\mathcal{P}(\langle \Gamma; \Delta\rangle \vdash \gamma)/\approx$ such that $a = \mathrm{at}(\mathbf{p}, \langle \Gamma; \Delta\rangle \vdash \gamma)$.

Definition 1. *A typing $\langle \Gamma; \Delta\rangle \vdash \gamma$ is* potentially negatively non-duplicating *(PN-typing) if and only if*

- *every atomic type has at most one negative potential occurrence in it.*
- *for an atomic type a, if $\mathrm{Pot}^{-}_{a}(\Gamma \vdash \gamma) \neq \emptyset$ then $\mathrm{Irr}^{-}_{a}(\Gamma \vdash \gamma) = \emptyset$*

In example 3, the typings we exhibited as not being negatively non-duplicating typings are actually PN-typings.

As PN-typings are a natural extensions of negatively non-duplicating typings, it is easy to show that they also enjoy the property of being uniquely inhabited.

Theorem 2. ***(Coherence)*** *Given a β-normal term M such that $\langle \Gamma; \Delta\rangle \vdash M : \gamma$ is derivable, if $\langle \Gamma; \Delta\rangle \vdash \gamma$ is a PN-typing and $\langle \Gamma; \Delta\rangle \vdash N : \gamma$ is derivable, then $M =_{\beta\eta} N$.*

Proof. Suppose M is η-long form for $\langle \Gamma; \Delta\rangle \vdash \gamma$. By induction M, we prove that any term N in β-normal η-long form for $\langle \Gamma; \Delta\rangle \vdash \gamma$ verifies $M = N$.

While Kanazawa takes the least negatively non-duplicating typing of a β-normal almost linear term M as the typing which fully characterizes M and the almost linear terms that are $\beta\eta$-equivalent to M, we will next show that some PN-typings, we call *the least PN-typings*, of an almost affine term M fully characterizes M and the almost affine terms that are $\beta\eta$-equivalent to M.

5 Least PN-typings

In what follows, all the terms considered are closed (*i.e.* their set of free variables is empty); hence, instead of writting $\langle \Gamma; \_ \rangle \vdash \gamma$ for a typing, we will simply write $\Gamma \vdash \gamma$. We also fix a HOS Σ, an almost affine and β-normal term $M \in \Lambda_\Sigma$ and we let $\Gamma \vdash \gamma$ be the mglt of M.

Similarly to what is done in [Kan07], we aim at constructing a PN-typing of M that is less general than any PN-typing of M. In general, such a typing does not exist, instead, there are finitely many PN-typings (up to renaming) of M such that any typing of M which is strictly less general than one of them is not a PN-typing. We call these typings *the least PN-typings of* M.

We divide the construction of the least PN-typings of M into two parts. The first part of the construction is mostly dealing with the problems inherent to deletion, while the second can be considered as a mere rephrasing for PN-typings of what is done in [Kan07] for negatively non-duplicating typings. The constructions we describe in these two parts are based on relabelings that act on irrelevant positions in the first case, and on potential positions in the second one. We thus adopt the following notations:

- $\texttt{Irrat}(\Gamma \vdash \gamma) = \{\mathrm{at}(p, \Gamma \vdash \gamma) \mid p \in \mathrm{Irr}(\Gamma \vdash \gamma)\}$
- $\texttt{Potat}(\Gamma \vdash \gamma) = \{\mathrm{at}(p, \Gamma \vdash \gamma) \mid p \in \mathrm{Pot}(\Gamma \vdash \gamma)\}$

The set of atomic types $\texttt{Irrat}^+(\Gamma \vdash \gamma)$ (*resp.* $\texttt{Irrat}^-(\Gamma \vdash \gamma)$) is defined as $\{\mathrm{at}(p, \Gamma \vdash \gamma) \mid p \in \mathrm{Irr}^+(\Gamma \vdash \gamma)\}$ (*resp.* $\{\mathrm{at}(p, \Gamma \vdash \gamma) \mid p \in \mathrm{Irr}^-(\Gamma \vdash \gamma)\}$).

A relabeling σ is said *irrelevant* (*resp. potential*) with respect to $\Gamma \vdash \gamma$ when the set $\{a \mid a \cdot \sigma \neq a\}$ is included in $\texttt{Irrat}(\Gamma \vdash \gamma)$ (*resp.* $\texttt{Potat}(\Gamma \vdash \gamma)$). When $\Gamma \vdash \gamma$ is a PN-typing, σ is said PN-preserving if $\Gamma \cdot \sigma \vdash \gamma \cdot \sigma$ is also a PN-typing. A substitution is *PN-irrelevant* (*resp. PN-potential*) with respect to $\Gamma \vdash \gamma$ when it is both irrelevant (*resp.* potential) and PN-preserving with respect to $\Gamma \vdash \gamma$.

We are now going to study PN-preserving relabelings as the composition of PN-irrelevant and PN-potential relabelings.

5.1 PN-irrelevant relabelings

Lemma 3. *Given a PN-typing $\Gamma \vdash \gamma$, a relabeling σ is PN-irrelevant iff for every a in $\texttt{Irrat}^-(\Gamma \vdash \gamma)$, $a.\sigma$ is not in $\texttt{Potat}(\Gamma \vdash \gamma)$.*

According to Property 3, given the mglt $\Gamma \vdash \gamma$ of a β-reduced term M, for every $a \in \texttt{Irrat}(\Gamma \vdash \gamma)$ there is a unique position $p \in \mathcal{P}(\Gamma \vdash \gamma)$ such that $\mathrm{at}(p, \Gamma \vdash \gamma) = a$. Given a fresh atomic type $\omega \notin \texttt{Potat}(\Gamma \vdash \gamma)$, a PN-irrelevant relabeling σ on $\texttt{Irrat}(\Gamma \vdash \gamma)$, is said *maximal* when it satisfies the following properties:

1. $a \cdot \sigma = \omega$ when a is in $\texttt{Irrat}^-(\Gamma \vdash \gamma)$,
2. $a \cdot \sigma \in \texttt{Potat}(\Gamma \vdash \gamma) \cup \{\omega\}$ when a is in $\texttt{Irrat}^+(\Gamma \vdash \gamma)$.

Lemma 4. *Given a PN-tpying $\Gamma \vdash \gamma$, a maximal irrelevant relabeling σ on this typing, and a relabeling σ' such that, $\Gamma \cdot (\sigma' \circ \sigma) \vdash \gamma \cdot (\sigma' \circ \sigma)$ is a PN-typing iff $\sigma' = \sigma_1 \circ \sigma_2$ where σ_1 PN-potential substitution and σ_2 is a renaming.*

We now define the *Σ-mglt* $\Gamma_\Sigma \vdash \gamma$ of M such that for every $\mathbf{c} \in \mathcal{C}$, $\Gamma_\Sigma(\mathbf{c}) = \Gamma(\mathbf{c}) \cup \{\alpha\}$, where α is made of fresh atomic types for which there is a unique occurrence in $\Gamma_\Sigma \vdash \gamma$. Let us now define the set $\Omega(\Gamma_\Sigma \vdash \gamma)$ of maximal PN-irrelevant relabelings on $\Gamma_\Sigma \vdash \gamma$. Because $\texttt{Potat}(\Gamma_\Sigma \vdash \gamma)$ is finite, $\Omega(\Gamma_\Sigma \vdash \gamma)$ is also finite (up to renaming).

Definition 2. *Given a HOS Σ, a term $M \in \Lambda_\Sigma$, $\Gamma_\Sigma \vdash \gamma$ its Σ-mglt and $\Omega(\Gamma_\Sigma \vdash \gamma)$ the set of maximal PN-irrelevant substitutions on $\Gamma_\Sigma \vdash \gamma$, a* Σ-saturated typing *$\Gamma_{sat} \vdash \delta$ of M is a typing which verifies:*

- *there is $\sigma \in \Omega(\Gamma_\Sigma \vdash \gamma)$ such that $\delta = \gamma \cdot \sigma$*
- *for every constant $\mathbf{c} \in \mathrm{Dom}(\Gamma_\Sigma)$, every $\alpha \in \Gamma_\Sigma(\mathbf{c})$ and every $\sigma \in \Omega(\Gamma_\Sigma \vdash \gamma)$, we have $\alpha \cdot \sigma \in \Gamma_{sat}(\mathbf{c})$.*

Example 4. Let us consider the term M as in Example 2. Recall that we suppose that this term is built on a signature Σ that contains two constants $\mathbf{c_1}$ and $\mathbf{c_2}$ of type $o \to o \to o$ and that the mglt of M is: $\mathbf{c_1} : a \to b \to c \vdash (u \to v) \to a \to b \to c$. Let A be the set $\{a, b, c, \omega\}$, then the set of Σ-saturated typings of M is $\{\mathbf{c_1} : \overline{\gamma_1}, \mathbf{c_2} : \overline{\gamma_2} \vdash \gamma \mid \gamma \in (A \to \omega) \to a \to b \to c\}$ where $\overline{\gamma_1} = a \to b \to c \cap A \to A \to \omega$ and $\overline{\gamma_2} = A \to A \to \omega$

A Σ-saturated typing of an almost affine term M in β-normal form is a PN-typing; indeed, for every relabeling $\sigma \in \Omega(\Gamma_\Sigma \vdash \gamma)$, $\mathrm{Pot}(\Gamma_\Sigma \cdot \sigma \vdash \gamma \cdot \sigma)/\approx$ and $\mathrm{Pot}(\Gamma_\Sigma \vdash \gamma)/\approx$ are in a one to one correspondence that preserves the atomic type associated to classes of positions. This implies that $\mathrm{Pot}(\Gamma_{sat} \vdash \delta)/\approx$ and $\mathrm{Pot}(\Gamma_\Sigma \vdash \gamma)/\approx$ verify the same property. We can then deduce that $\Gamma_{sat} \vdash \delta$ is a PN-typing.

5.2 PN-potential relabelings

Example 5. Let us consider the term $M = \lambda f^{o \to o \to o} x^o y^o . f(\mathbf{c_1} x)(\mathbf{c_1} x)$ built on a signature declaring $\mathbf{c_1}$ with the type $o \to o$. The mglt of M is $\mathbf{c_1} : a \to b_1 \cap a \to b_2 \vdash (b_1 \to b_2 \to c) \to a \to v \to d$. Let A be the set $\{a, b_1, b_2, c, \omega\}$, then M has a unique Σ-saturated typing:

$$\mathbf{c_1} : a \to b_1 \cap a \to b_2 \cap A \to \omega \vdash (b_1 \to b_2 \to c) \to a \to \omega \to c$$

The PN-typing $\mathbf{c_1} : a \to b \cap A' \to \omega \vdash (b \to b \to c) \to a \to c$, where $A' = \{a, b, c, \omega\}$ is a typing of M', which is obtained by applying a relabeling σ such that $b_1.\sigma = b_2.\sigma = b$. The relabeling σ is potential since b_1 and b_2 are both potential atomic types.

In this example, it appears that the potential relabeling σ is a PN-potential relabeling with respect to the unique Σ-saturated typing of M; moreover, it assigns the same type to both occurrences of $\mathbf{c_1}$, which corresponds to the method used in [Kan07] at the level of the syntax of the term.

This example shows that even though maximal PN-irrelevant relabelings lead to PN-typings that are less general than the mglt of M, and which are the least general PN-typings of M generated by PN-irrelevant relabelings, there are still less general typings generated by applying the composition of maximal PN-irrelevant relabelings and PN-potential relabelings.

Theorem 3. *Given a PN-typing $\Gamma \vdash \gamma$ and two of its PN-potential relabelings σ_1 and σ_2, there is a PN-potential relabeling σ of $\Gamma \vdash \gamma$ such that $\Gamma.\sigma \vdash \gamma.\sigma$ is less general than both $\Gamma.\sigma_1 \vdash \gamma.\sigma_1$ and $\Gamma.\sigma_2 \vdash \gamma.\sigma_2$.*

This theorem leads to the existence of a relabeling σ_{max} (called the *maximal PN-potential relabeling on Γ_{sat}*) on $\mathrm{Pot}(\Gamma_{sat} \vdash \delta)$ such that for every relabeling σ which verifies $\mathrm{Dom}(\sigma) \subseteq \mathrm{Pot}(\Gamma_{sat} \vdash \delta)$, $\Gamma_{sat} \cdot (\sigma \circ \sigma_{max}) \vdash \delta \cdot (\sigma \circ \sigma_{max})$ is a PN-typing iff $\sigma = \sigma_1 \circ \sigma_2$ where σ_1 is PN-irrelevant and σ_2 is a renaming.

Definition 3. *A typing $\Gamma \vdash \gamma$ of M is called a* least PN-typing of M *if there is Σ-saturated typing $\Gamma_{sat} \vdash \delta$ of M such that $\Gamma \vdash \gamma = \Gamma_{sat} \cdot \sigma_{max} \vdash \delta \cdot \sigma_{max}$, where σ the maximal PN-potential relabeling on Γ_{sat}.*

We are now in position to state the main theorem that allows us to construct a Datalog recognizer.

Theorem 4. *Given a signature Σ, a closed almost affine term $M \in \Lambda_\Sigma$ in β-normal form, for every closed almost affine term $M' \in \Lambda_\Sigma$ such that $M' \twoheadrightarrow_\beta M$ there is a least PN-typing $\Gamma_{sat} \vdash \gamma$ of M that types M'.*

Proof. Let us consider the mglts $\Gamma \vdash \gamma$ and $\Gamma' \vdash \gamma'$ of M and M' respectively. Because M and M' are almost affine terms, these typings are negatively non-duplicating (Theorem 1), and by subject reduction $\Gamma' \vdash M : \gamma'$. Then, it can be easily remarked that there is a least PN-typing of M that is less general than $\Gamma' \vdash \gamma'$.

6 The Datalog recognizer

6.1 Description of the program

Let us consider an almost affine λ-CFG $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{H}, s)$ (where for $i \in \{1, 2\}$, $\Sigma_i = (A_i, \mathcal{C}_i, \tau_i)$), a closed almost affine term M in β-normal form and $\Gamma \vdash \gamma$ its mglt. We now give the details of the construction of a Datalog program that checks whether M belongs to $\mathcal{O}(\mathcal{G})$. This construction is done in the same fashion as in [Kan07].

In what follows we adopt the notation $\gamma[\overrightarrow{\alpha}]$ for a type $\alpha \in \mathcal{U}_\gamma(A)$, where $\overrightarrow{\alpha}$ is the sequence of atomic types appearing in α in a left-to-right order. Also, we write σ_ω for the substitution such that:

$$\sigma_\omega(a) = \begin{cases} \omega \text{ when } a \in \texttt{Irrat}^-(\Gamma \vdash \gamma) \\ a \text{ otherwise} \end{cases}$$

The term-related database Let's consider the term to parse M such that $Cst(M) = \{\mathbf{c}_1, \ldots, \mathbf{c}_l\} \subseteq \mathcal{C}_2$. Given a typing $\Gamma \vdash \gamma$ of M, we let $\Gamma_{\Sigma_2} \vdash \gamma$ be the Σ-mglt of M and $\Gamma_\omega \vdash \gamma_\omega = \Gamma_{\Sigma_2} \cdot \sigma_\omega \vdash \gamma \cdot \sigma_\omega$.

1. to every constant $\mathbf{c} \in \mathrm{Dom}(\Gamma_\omega)$ and every type $\alpha \in \Gamma_\omega(\mathbf{c})$ we associate the rule:
$$c(\overrightarrow{\alpha}) \text{ :- } \texttt{atom}(x_1), \ldots, \texttt{atom}(x_n).$$
where an atomic types a of α that belongs to either $\texttt{Potat}(\Gamma_\omega \vdash \gamma_\omega)$ or $\texttt{Irrat}^-(\Gamma_\omega \vdash \gamma_\omega)$ is considered as a Datalog constant a; and each atomic type of α that belongs to $\texttt{Irrat}^+(\Gamma_\omega \vdash \gamma_\omega)$ is considered as a Datalog variable x_i ($i \in \{1, \ldots, n\}$). Note that for $i \in \{1, \ldots, n\}$ there is one and only one occurrence of x_i in $\overrightarrow{\alpha}$ according to Property 3.
2. for every atomic type a in $\texttt{Potat}(\Gamma \vdash \gamma) \cup \{\omega\}$, add a fact:
$$\texttt{atom}(a).$$

The grammar-related database Given a constant $\mathbf{c} \in \mathcal{C}_1$ and the mglt $\Gamma \vdash \gamma$ of $\mathcal{H}(\mathbf{c})$, the following datalog rule $\rho_\mathbf{c}$:

$$p_0(\overrightarrow{x_0}) \text{ :- } p_1(\overrightarrow{x_1}), \ldots, p_n(\overrightarrow{x_n}), e_{f(1)}(\overrightarrow{y_1}), \ldots e_{f(m)}(\overrightarrow{y_m}), \texttt{atom}(z_1), \ldots, \texttt{atom}(z_l).$$

is associated to $\mathbf{c}$ and verifies.

- $\tau_1(\mathbf{c}) = p_1 \to \ldots \to p_n \to p_0$.
- $\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}, \overrightarrow{y_1}, \ldots, \overrightarrow{y_m}$ are exclusively made of Datalog variables
- given $\mathrm{Dom}(\Gamma) = \{\mathbf{e}_1, \ldots, \mathbf{e}_p\} = Cst(M)$:
  - there exists $j \in \{1, \ldots, m\}$, such that $\alpha = \tau_2(\mathbf{e}_{f(j)})[\overrightarrow{y_j}]$ iff $\alpha \in \Gamma(\mathbf{e}_{f(j)})$
  - $\gamma = \mathcal{H}(p_1)[\overrightarrow{x_1}] \to \ldots \to \mathcal{H}(p_n)[\overrightarrow{x_n}] \to \mathcal{H}(p_0)[\overrightarrow{x_0}]$
  - x_i belongs to $\{z_1, \ldots, z_l\}$ iff $x_i \in \overrightarrow{x_0}$ and x_i has a unique occurrence in $\overrightarrow{x_0}, \ldots, \overrightarrow{x_n}, \overrightarrow{y_1}, \ldots, \overrightarrow{y_m}$.

Note that the Datalog variables $\{z_1, \ldots, z_l\}$ are associated to irrelevant positions in $\Gamma_\omega \vdash \gamma_\omega$; this way, we ensure irrelevant positions are forced to belong to the desired set of atomic types (thanks to $\texttt{atom}(z_i)$), and we ensure the safety condition on the rules of our Datalog program.

For each constant $\mathbf{c} \in \mathcal{C}_1$, it is easy to see that the rule ρ_c is in fact associated to every typing $\Gamma' \cdot \sigma \vdash \gamma' \cdot \sigma$, where $\Gamma' \vdash \gamma'$ is the mglt of $\mathcal{H}(\mathbf{c})$ and the relabeling σ maps atomic types in $\texttt{Irrat}(\Gamma' \vdash \gamma')$ to atomic types in $\texttt{Potat}(\Gamma \vdash \gamma) \cup \{\omega\}$, such that $\Gamma \vdash \gamma$ is a typing of the term to parse M. Morever, if $\Gamma \vdash \gamma$ is taken as the mglt of M in the constrution of the term-related database, the rules

$c(\overrightarrow{\alpha})$:- $\texttt{atom}(x_1), \ldots, \texttt{atom}(x_n)$. derives any type $\alpha \in \Gamma_{sat}(\mathbf{c})$, for $\Gamma_{sat} \vdash \delta$ a least PN-typing of M.

We note a program built as above $\texttt{recog}(\mathcal{G}, \Gamma)$; moreover, given any atomic type p in the abstract signature Σ_1 of $\mathcal{G}$, we next prove that there is a term $N \in \Lambda^p_{\Sigma_1}$ such that $\mathcal{H}(N) \twoheadrightarrow_\beta M$ iff there is a derivation in answer to the Datalog query $?$:- $p(\overrightarrow{\gamma})$ (where every irrelevant occurrence in γ of an atomic type to $\texttt{Irrat}^+(\Gamma \vdash \gamma)$ is replaced by a Datalog variable in $\overrightarrow{\gamma}$). Note that, by construction, the query $?$:- $p(\overrightarrow{\gamma})$ is the same for any least PN-typing of M.

6.2 Completeness and Correctness of the method

Given a Datalog program $\texttt{recog}(\mathcal{G}, \Gamma)$, a request $?$:- $p(\overrightarrow{\gamma})$ built as previously detailed, and $\Gamma \vdash \gamma$ a typing of M, we note

$$\texttt{recog}(\mathcal{G}, M) \text{ :- } p(\overrightarrow{\gamma})$$

whenever there is a derivation of $p(\overrightarrow{\gamma})$ in $\texttt{recog}(\mathcal{G}, M)$; such a derivation must assign a type in $\texttt{Potat}(\Gamma \vdash \gamma) \cup \{\omega\}$ to each Datalog variable present in $\overrightarrow{\gamma}$.

Lemma 5. *Let's consider $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{H}, s)$ an almost affine λ-CFG. There exist a type assignment Γ and a type γ such that*

$$\texttt{recog}(\mathcal{G}, \Gamma) \vdash p(\gamma)$$

iff there is a closed term $N \in \Lambda^p_{\Sigma_1}$ such that $\Gamma \vdash \mathcal{H}(N) : \gamma$.

Proof. Proceed by induction on the Datalog derivation for the first implication, by a simple induction on N for the second one.

Theorem 5. *Let's consider an almost affine λ-CFG $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{H}, s)$. Then the two following propositions are equivalent:*

1. *there is a term $M \in \mathcal{O}(\mathcal{G})$*
2. *$\texttt{recog}(\mathcal{G}, \Gamma) \vdash s(\overrightarrow{\gamma})$, such that $\Gamma \vdash \gamma$ is a least PN-typing typing of M*

Proof. $(1 \Rightarrow 2)$ Let's first consider a term $M \in \mathcal{O}(\mathcal{G})$; by definition of an object language, there is a term $N \in \mathcal{A}(\mathcal{G})$, such that $|\mathcal{H}(N)|_\beta = M$. Hence, according to Lemma 5, for every typing $\Delta \vdash \delta$ of $\mathcal{H}(N)$, $\texttt{recog}(\mathcal{G}, \Delta) \vdash s(\overrightarrow{\delta})$. This is in particular true for some least PN-typing $\Gamma \vdash \gamma$ of M, according to Theorem 4. By construction, for any least PN-typing of M, $\texttt{recog}(\mathcal{G}, \Gamma)$ and $s(\overrightarrow{\gamma})$ are identical, so 2. is verified.

$(2 \Rightarrow 1)$ Let's now suppose $\texttt{recog}(\mathcal{G}, \Gamma) \vdash s(\overrightarrow{\gamma})$, such that every least-PN-typings of M are represented by $\Gamma \vdash \gamma$. According to Lemma 5, there is a term $N \in \mathcal{A}(\mathcal{G})$ such that $\mathcal{H}(N)$ inhabits $\Gamma \vdash \gamma$; but the least PN-typings of M are PN-typings and by Theorem 2, the terms M and $\mathcal{H}(N)$ must be $\beta\eta$-equivalent.

Abstract Grammar	Object Grammar
$John : NP$	$\lambda QP.P(Q$ **john_pers undefined undefined**$)$
$read : NP \to NP \to S$	$\lambda PQ.(P\pi_1)(\lambda x.(Q\pi_2)(\lambda y.$**read** $x\ y))$
$Hamlet : NP$	$\lambda QP.P(Q$ **ham_pers ham_bcont ham_bphys**$)$

where $V = NP \to NP \to S$ and $\pi_i = \lambda x_1 x_2 x_3 . x_i$

Fig. 2. Example of a semantically enriched λ-CFG

Extensional Rules of $\texttt{recog}(\mathcal{G}, \Gamma)$

$NP(x_1, x_2, x_3, x_4, x_4, x_5, x_5)$:- $\boldsymbol{john\_pers}(x_1), \boldsymbol{undefined}(x_2), \boldsymbol{undefined}(x_3)$.

$S(x_6)$:- $NP(x_1, x_2, x_3, x_1, x_4, x_5, x_6)$,
$NP(x'_1, x'_2, x'_3, x'_2, x_7, x_8, x_5)$,
$\boldsymbol{read}(x_4, x_7, x_8)$,

$NP(x_1, x_2, x_3, x_4, x_4, x_5, x_5)$:- $\boldsymbol{ham\_pers}(x_1), \boldsymbol{ham\_bcont}(x_2), \boldsymbol{ham\_bphys}(x_3)$.

$\boldsymbol{read}(x_1, x_2, \omega)$:- $\texttt{atom}(x_1), \texttt{atom}(x_2)$.

Intensional Rules of $\texttt{recog}(\mathcal{G}, \Gamma)$

$\boldsymbol{john\_pers}(\omega)$.	$\texttt{atom}(1)$.	$\boldsymbol{read}(2,3,1)$.
$\boldsymbol{undefined}(\omega)$.	$\texttt{atom}(2)$.	$\boldsymbol{john\_pers}(2)$.
$\boldsymbol{ham\_pers}(\omega)$.	$\texttt{atom}(3)$.	$\boldsymbol{ham\_bcont}(3)$.
$\boldsymbol{ham\_bcont}(\omega)$.		
$\boldsymbol{ham\_bphys}(\omega)$.		

Fig. 3. Example of Datalog program

6.3 Example

As an example, let us consider the almost affine λ-CFG in Figure 2, where semantic aspects are considered; for instance, **ham_bcont** and **ham_bphys** stand respectively for the content and the object denoted by the book "Hamlet". We consider the following semantic representation of the sentence *John read Hamlet*

read john_pers ham_bcont

and build the associated Datalog program as in Figure 3. There is a derivation in this program for the Datalog request :-$?S(1)$ which corresponds to the PN-typing:

$$\begin{cases} \textbf{read} : 2 \to 3 \to 1, \textbf{john\_pers} : 2, \textbf{ham\_bcont} : 3, \\ \textbf{ham\_bphys} : \omega, \textbf{ham\_bpers} : \omega, \textbf{undefined1} : \omega, \\ \textbf{undefined2} : \omega \end{cases} \vdash \mathcal{Lex}(read\ John\ Hamlet) : 1$$

Readers should note that in practice, only the last 6 rules depend on the term to parse M and on its least PN-typings. It is therefore possible to enhance

the construction of the grammar by constructing every rule of the grammar-related database plus, for every constant $\mathbf{c}$ in the object signature the rules $c(\overrightarrow{\alpha})$:- $\mathtt{atom}(x_1), \ldots, \mathtt{atom}(x_n)$ such that $\tau_2(\mathbf{c})[\overrightarrow{\alpha}]$ is of the form $\alpha_1 \to \ldots \to \alpha_m \to \omega$. Then, given a term M to parse, we only need to add the facts on the predicate $\mathtt{atom}$ and for every constant $\mathbf{c}$ in the object signature the rules $c(\overrightarrow{\alpha})$:- $\mathtt{atom}(x_1), \ldots, \mathtt{atom}(x_n)$ such that α is not of the previously mentionned form. Therefore, the complexity of the fixed-language recognition should be close to the one given in [Kan07] for almost linear λ-CFGs.

7 Conclusion

While Datalog programs for almost linear λ-CFGs led to efficient parsing algorithms, we give a Datalog recognizer construction which subsumes Kanazawa's, and which recognize λ-term in the language of an almost affine λ-CFG. Even though almost affine λ-CFGs are not more expressive than almost linear λ-CFGs, they facilitate the design of grammars which, for instance, take account of feature agreements or semantic aspects.

Technically, the solution proposed in this paper is exclusively based on typing properties of almost affine terms, in particular, we introduced PN-potential relabelings so as to identify occurrences of constants in a term M that originate from the same occurrence in the β-expanded term M. On the theoretical point of view, the introduction of deletion in the grammar implies the introduction of intersection types in the typing theory, just as it is done in [Sal07] for parsing generalized λ-grammar. Simple types seem therefore enough only to study linearity and almost linearity in λ-grammars. The use of intersection types gives also a great framework to handle occurrences of constants, and the listed types we introduced offer the possibility of considering parsing context-free grammars of non-simply-typed λ-terms, by assigning types of different shape to constants. Also, while [Kan07] results are based on the properties of negatively non-duplicating typings, we introduced the family of potentially negatively non-duplicating typings (to which negatively non-duplicating typings also belong).

While Kanazawa's method performs in LOGCFL, it remains to know what the complexity of the algorithm given in this article is exactly; though, we know the Datalog program ensures a polynomial time parsing algorithm. We also plan to compare the size of the final program with the size of a the Datalog recognizer linearized almost affine λ-CFG as given in [Yos06] so as to assess whether there is a significant gain in considering almost affine λ-CFG. In the future, we also plan to enhance this algorithm with an extraction of the associated derivations so as to develop a parser based on these results. Finally, it would also be interesting to see whether the operation of deletion can help modeling in a simple manner other linguistic phenomena such as ellipsis.

References

Aot99. T. Aoto. Uniqueness of normal proofs in implicational intuitionistic logic. *Journal of Logic, Language and Information*, 8:217–242, 1999.

BMR10. C. Bassac, B. Mery, and C. Retoré. Towards a type-theoretical account of lexical semantics. *Journal of Logic, Language and Information*, 19(2):229–245, 4 2010.

BS11. P. Bourreau and S. Salvati. The game of characterizing uniquely typed sequents in nj, 2011. Submitted to TLCA'11.

CDC80. M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, 1980.

Cur61. H.B. Curry. Some logical aspects of grammatical structure, 1961.

dG01. P. de Groote. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155, 2001.

Kan07. M. Kanazawa. Parsing and generation as Datalog queries. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 176–183, Prague, 2007. Association for Computational Linguistics.

Kan10. M. Kanazawa, 2010. Work presented at the logic seminar in NII.

Mon74. Richard Montague. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven, CT, 1974.

Mus01. R. Muskens. Lambda Grammars and the Syntax-Semantics Interface. In R. van Rooy and M. Stokhof, editors, *Proceedings of the Thirteenth Amsterdam Colloquium*, pages 150–155, Amsterdam, 2001.

Mus10. R. Muskens. New Directions in Type-Theoretic Grammars. *Journal of Logic, Language and Information*, 19(2):129–136, 4 2010.

Pog00. Sylvain Pogodalla. Generation, lambek calculus, montague's semantics and semantic proof nets. In *proceedings of the International Conference on Computational Linguistics*, August 2000.

Ran09. A. Ranta. The gf resource grammar library. *Linguistic Issues in Language Technology*, 2(2):1–63, 2009.

Sal05. S. Salvati. *Problèmes de filtrage et problèmes d'analyse pour les grammaires catégorielles abstraites*. PhD thesis, Institut National Polytechnique de Lorraine, 2005.

Sal07. S. Salvati. On the membership problem for non-linear abstract categorial grammars. In R. Muskens, editor, *Proceedings of the Workshop on New Directions in Type-theoretic Grammars*, pages 43–50, Dublin, Ireland, August 2007. Foundation of Logic, Language and Information (FoLLI).

Sal10. S. Salvati. On the Membership Problem for Non-Linear Abstract Categorial Grammars. *Journal of Logic, Language and Information*, 19(2):163–183, 2010.

Ull88. J. Ullman. *Principles of Database and Knowledge-Base Systems. Volume I*. W. H. Freeman & Co., New York, NY, USA, 1988.

Yos06. R. Yoshinaka. Linearization of affine abstract categorial grammars. In *Proceedings of the 11th Conference on Formal Grammar*, pages 185–199, Malaga, Spain, 2006.