On IO-Copying and Mildly-Context Sensitive Formalisms

Pierre Bourreau, Laura Kallmeyer, Sylvain Salvati

On IO-copying and mildly-context sensitive formalisms

Pierre Bourreau[1], Laura Kallmeyer[2], and Sylvain Salvati[1]

[1] Université Bordeaux 1
351, Cours de la Libération
33405 Talence Cedex, France
{bourreau,salvati}@labri.fr
[2] Heine-Heinrich Universitat Dusseldorf
Universitatstr. 1
40225 Dusseldorf, Germany
kallmeyer@phil.uni-duesseldorf.de

Abstract. The class of mildly context-sensitive languages is commonly accepted as fulfilling the requirements to describe natural language. Many formalisms are known to generate languages which belong to this class, such as tree-adjoining grammars, multiple context-free grammars or abstract categorial grammars. All these formalisms share the property of being describe thanks to linear transformations, *i.e.* which avoid erasing or copying of material along derivations. We show that restricted copying operations allow defining mildly context-sensitive languages, thanks to the introduction of a new operation: IO-substition of languages.

Keywords: mildly context-sensitive languages, abstract families of languages, IO substitution, PMCFG.

1 Introduction

The question of the amount of expressive power needed in order to deal with natural languages is still an open question. In this context, the notion of mild-context-sensitivity has been proposed. A grammar formalism is mildly context-sensitive if a) it is more powerful that CFG, b) it generates only languages of constant growth, c) it can generate languages that describe a limited amount of cross-serial dependencies and d) it is polynomially parsable. A well-known class of formalisms that is still mildly context-sensitive is the class containing Linear Context-Free Rewriting Systems (LCFRS) and equivalent formalisms such as Multiple Context-Free Grammars (MCFG), Minimalist Grammar (MG) and set-local multicomponent Tree Adjoining Grammars (MCTAG).

This paper explores ways to characterize formally and linguistically interesting extensions of the class of LCFRL/MFCL that are still mildly context-sensitive. It follows ideas from [Kal10] where a mildly context-sensitive LCFRS extension is defined that allows for a limited amount of copying during derivations. The definition of this formalisms called Literal Movement Grammars of constant non-linearity (CNL-LMG) is, however, based on properties of possible derivations in a grammar. Furthermore, it lacks an independent characterization of the resulting class of string languages.

In this paper, we define first an operation on string languages that amounts to copying a substring into different places, called *IO-substitution*. The idea is roughly that we generate some string w_2 and a string w_1 that contains several occurrences of a variable x. This variable marks all the positions where the string w_2 gets copied to. An important point is that the string that gets copied does not increase while being copied. Once we have such a copying operation, we define the closure of MCFL/LCFRL under this operation as a new class IO-MCFL. We show that the languages in this class still have the constant-growth property.

As an example, consider the languages $(xd)^*x$ and a^*. IO-substitution of x in the first language by all possible words from the second language means fixing a word a^n from the second and replacing all occurrences of x in words from the first by a^n. As a result, we obtain $\{(a^nd)^ma^n \mid n, m \geq 0\}$ which is a generalization of the counting language that is not a MCFL [Kal10].

In parallel to the definition of IO-substitution on strings, we then characterize possible rules in Literal Movement Grammars (LGM), an MCFG-extension, that describe exactly the IO-subsitition operation. Based on this, a new grammar formalism is defined that turns out to be mildly context-sensitive while being a proper extension of MCFG.

The structure of the paper is as follows: ... to be written ...

2 Preliminaries

2.1 Mildly context-sensitive languages

Let us consider a countable set Σ, called an alphabet. We write $\Sigma^* = (\Sigma, \cdot, \epsilon)$ the (free) monoid on Σ, where $\cdot$ is the operation of concatenation, and ϵ the identity element. Any set $L \subseteq \Sigma^*$ is called a language on Σ.

Given a language $L \in \Sigma^*$, and w one of its elements (called words), we define the usual notions of length and number of occurrences of a letter $a \in \Sigma$ in w by induction on w: $|\epsilon| = 0$, $|w| = 1$ if $w \in \Sigma$ and $|w_1 \cdot w_2| = |w_1| + |w_2|$, for the length of w and $|\epsilon|_a = 0$, $|w|_a = 1$ if $w = a$, $|w|_a = 0$ if $w \in \Sigma - \{a\}$ and $|w_1 \cdot w_2|_a = |w_1|_a + |w_2|_a$ for the number of occurrences of a in w.

The class **MCSL** of *mildly context-sensitive languages* [Jos85,Wei88] is defined as the smallest set such that:

- **MCSL** contains all context-free languages.
- some restricted crossing-dependencies are taken in account by the languages $L \in$ **MCSL**
- every language $L \in$ **MCSL** verifies the constant-growth property.
- every language $L \in$ **MCSL** is recognizable in polynomial-time.

The first property specifies that **MCSL** falls between the classes of context-free languages and context-sensitive languages in Chomsky's hierarchy: **CFL** $\subseteq$ **MCSL** $\subseteq$ **CSL**. While vague, the second property ensures this class is bigger than the class of context-free languages. Hence, these two properties are easily ensured when defining some new formalisms.

This work mainly focuses on the last two properties. For the constant-growth property, we consider the following definition [Kal10], which is stronger than the original definition in [Wei88], and is based on the commonly-known notion of the Parikh image:

Notation 21 *Let us consider an alphabet Σ and the vector space $\mathbb{N}^{|\Sigma|}$. Given a letter $a \in \Sigma$ and a vector $\overrightarrow{v}$ on $\mathbb{N}^{|\Sigma|}$, we note by $\overrightarrow{v}[a]$ the scalar of $\overrightarrow{v}$ on the coordinate associated to a.*

Definition 1. *Let us consider a word w in a language $L \subseteq \Sigma^*$. The* Parikh image of *w, written $\overrightarrow{p}(w)$ is such that, for every $a \in \Sigma$, $\overrightarrow{p}(w)[a] = |w|_a$. The Parikh image of L is defined as $\overrightarrow{p}(L) = \{\overrightarrow{p}(w) \mid w \in L\}$.*

Definition 2. *A language $L \in \Sigma^*$ is said to verify the constant-growth property if there exists a constant $c \in \mathbb{N} - \{0\}$ such that, for every word $w \in L$ verifying $|w| > c$, there exists $\overrightarrow{x}, \overrightarrow{y} \in \mathbb{N}^{|\Sigma|}$ for which:*

1. *$\overrightarrow{p}(w) = \overrightarrow{x} + \overrightarrow{y}$ and*
2. *for every $i \geq 1$, $\overrightarrow{x} + i\overrightarrow{y} \in \overrightarrow{p}(L)$.*

The constant-growth property is a property weaker than the condition of semilinearity:

Definition 3. *A set L of vectors on a vector space $\mathbb{N}^k$ is called a* linear set *if $L = \{\overrightarrow{x_0} + k_1\overrightarrow{x_1} + \ldots + k_n\overrightarrow{x_n} \mid n \in \mathbb{N}, k_1, \ldots, k_n \in \mathbb{N} - \{0\}\}$.*

A semilinear set is a finite union of linear sets.

Definition 4. *A language L is said* semilinear *if $\overrightarrow{p}(L)$ is a semilinear set.*

Many formalisms were created to generate languages that belong to **MCSL**, among which are tree-adjoining grammars (TAGs) [JLT75], multiple context-free grammars (MCFGs) [SMMK91], which will be introduced in the next section, or abstract categorial grammars (ACGs) [dG01,Mus01]. It is known that the class of multiple context-free languages (*i.e.* languages generated by MCFGs), which we will write **MCFL**, strictly entails the class of tree-adjoining languages (*i.e.* languages generated by TAGs). Hence, we consider **MCFL** as the best approximation of **MCSL**.

An important property of languages which belong to the class **MCFL** is that they are semilinear. In this paper, we introduce the operation of IO-substitution on languages in the class **MCFL** of languages; we will see that this operation allow the definition of a new class of languages which are not semilinear, but verify the constant-growth property.

2.2 Multiple context-free tree languages

In what follows, we define a multiple context-free language as the yield of a tree language derived by a linear multiple context-free tree language (linear MCFTG). Thanks to this definition, we will then prove that some specific MCFTGs are polynomial. The tree languages derived by Linear MCFTGs are exactly the tree languages derived by abstract categorial grammars[dG01,Mus01] of trees, and we write this class of languages **MCFTL**.

In order to define the formalism of MCFTG, let us first define trees and tree contexts. These objects will be defined as particular λ-terms types with simple types built on the single type o: given an atomic type o, we define the set of types on o as the smallest set $\mathscr{T}_o$ such that $o \in \mathscr{T}_o$ and $(\alpha \to \beta) \in \mathscr{T}_o$ if $\alpha,\beta \in \mathscr{T}_o$. Tthe usual convention of parenthesis are taken: a type $\alpha_1 \to (\alpha_2 \to \alpha_3)$ will be written $\alpha_1 \to \alpha_2 \to \alpha_3$. Given a type $\alpha \in \mathscr{T}_o$, we define the order $\texttt{ord}(\alpha)$ by induction on α as $\texttt{ord}(o) = 1$, and $\texttt{ord}(\alpha_1 \to \alpha_2) = max(1 + \texttt{ord}(\alpha_1), \texttt{ord}(\alpha_2))$.

Definition 5. *An alphabet Σ is said* typed *if to every element a in Σ is associated a type of $\mathscr{T}_o$, written $\tau(a)$. A typed alphabet is called an n-*order alphabet *(where $n \geq 0$) if $max_{a\in\Sigma}(\texttt{ord}(\tau(a))) \leq n$.*

In what follows, a type $\underbrace{\alpha \to \ldots \to \alpha}_{n} \to \gamma \in \mathscr{T}_o$ will be abreviated into $\alpha^n \to \gamma$.

Given a typed alphabet Σ and $n > 0$, we note by Σ^n the smaller subset of Σ which contains all the elements of Σ which associated type is of order less or equal to n.

Definition 6. *Let us consider an enumerable typed alphabet Σ and a type $\alpha \in \mathscr{T}_o$. We define $\mathbb{T}^\alpha(\Sigma)$ the trees of type α on Σ as the smallest set such that:*

- *$c \in \mathbb{T}^\alpha(\Sigma)$ if $c \in \Sigma$ and $\tau(c) = \alpha$*
- *$(t_1 t_2) \in \mathbb{T}^\alpha(\Sigma)$ if $t_1 \in \mathbb{T}^{\gamma\to\alpha}(\Sigma)$ and $t_2 \in \mathbb{T}^\gamma(\Sigma)$.*

The set $\mathbb{T}(\Sigma)$ of trees on Σ is defined as $\mathbb{T}(\Sigma) = \mathbb{T}^o(\Sigma)$. Moreover, if Σ is 2-order, we say that $\mathbb{T}(\Sigma)$ is a pure set of trees.

We take the usual conventions for parenthesis so that a tree $(t_1 t_2)t_3$ is written as $t_1 t_2 t_3$. The general form of a tree in $\mathbb{T}(\Sigma)$ is therefore $ct_1 \ldots t_n$, where $c \in \Sigma$, $\tau(c) = \alpha_1 \to \ldots \to \alpha_n \to o$ and for every $1 \leq i \leq n, t_i \in \mathbb{T}^{\alpha_i}(\Sigma)$.

Given a type $\alpha \in \mathscr{T}_o$ and a tree $t \in \mathbb{T}^\alpha(\Sigma)$ and $a \in \Sigma$, we define $|t|_a$ as the number of occurrences of a in t by induction on t: $|a|_a = 1$, $|c|_a = 0$ if $c \in \Sigma - \{a\}$, and $|t_1 t_2|_a = |t_1|_a |t_2|_a$.

The yield function is a function which takes a tree t as its inputs and returns the sequence of leafs of t from left to right. Formally, given a typed alphabet Σ, we define $\texttt{yield} : \mathbb{T}(\Sigma) \mapsto \Sigma^*$, as $\texttt{yield}(c) = c$ if $c \in \Sigma$, $\texttt{yield}(t_1 t_2) = \texttt{yield}(t_2)$ if $t_1 = c \in \Sigma$ and $\texttt{yield}(t_1 t_2) = \texttt{yield}(t_1) \cdot \texttt{yield}(t_2)$ otherwise. We extend this notion to sets of trees: given a set of trees L, $yL = \{\texttt{yield}(t) \mid t \in L\}$.

Definition 7. *Let us consider a finite second-order alphabet Σ and a type $\alpha \in \mathscr{T}_o$. We define the set $\mathbb{T}^\alpha_{[]}(\Sigma)$ of* tree contexts *of type α for $\mathbb{T}(\Sigma)$ as the smallest set of terms such that $\lambda x_1 \ldots x_n.t$ is in $\mathbb{T}^\alpha_{[]}(\Sigma)$ if $\mathbf{X} = \{x_1, \ldots, x_n\}$ is a typed alphabet, where for every $1 \leq i \leq n, \tau(x_i) = \alpha_i$, $\alpha = \alpha_1 \to \ldots, \to \alpha_n \to o$, and $t \in \mathbb{T}(\Sigma \cup \mathbf{X})$.*

The set of tree contexts on (Σ) is defined by $(\mathbb{T}^\alpha_{[]}(\Sigma))_{\alpha\in\mathscr{T}_o}$. Finally, such a tree context $\lambda x_1 \ldots x_n.t$ is said:

- linear *if $|t|_{x_i} = 1$ for every $1 \leq i \leq n$*
- almost affine *if for every $1 \leq i \leq n$, $|t|_{x_i} > 1$ iff $x_i \in \Sigma^1 \cup \mathbf{X}^1$*

Remark 1. In what follows, we will note trees by $t, t', t_1, \ldots$, contexts by $C, C', C_1, \ldots$. Moreover, finite sequences of variables $x_1, \ldots, x_n$ will often be written as $\overline{x}$, and identically, finite sequences of contexts of the form $C_1, \ldots, C_m$ will be noted by $\overline{C}$.

It is important to remark that $\mathbb{T}(\Sigma) = \mathbb{T}^o_{[]}(\Sigma)$. Moreover, tree contexts are defined as particular simply-typed λ-terms. In the rest of the document, we will take as granted the usual notions of substitutions, α-conversion, β-reduction and η-conversions defined in the simply-typed λ-calculus (see [Bar84,SU06] for details). These notions will be used to define a *tree context substition*: given a typed alphabet $\mathbf{X}$, $\sigma = [x_1 := C_1, \ldots, x_n := C_n]$ (where $x_i \in \mathbf{X}$ and $C_i \in \mathbb{T}^{\tau(x_i)}_{[]}(\Sigma)$ for every $i \in \{1, \ldots, n\}$) is a term substitution from $\mathbf{X}$ to $\mathbb{T}_{[]}(\Sigma)$; the application of σ to a tree context C is defined as $|C \cdot \sigma|_\beta$, *i.e.* the normal form of the application of the λ-term substitution σ to the λ-term C. Finally, we define the application of tree contexts $C_1, \ldots, C_n$ to a tree context $\lambda x_1 \ldots x_n.t$ by $\mathsf{app}(C, C_1, \ldots, C_n) = |C \cdot [x_1 := C_1, \ldots, x_n := C_n]|_\beta$.

Definition 8. *An alphabet Σ is said* multi-typed *if to every element a in Σ is associated a natural number $n \in \mathbb{N}$, called the* rank *of a (written $r(a)$), and to every pair (a, i), where $a \in \Sigma$ and $1 \leq i \leq r(a)$ is associated a type in $\mathscr{T}_o$ (written $\tau(a, i)$)*

We are in position of defining MCFTGs. These grammars can be seen as MCFGs on tree contexts.

Definition 9. *A* multiple context-free tree grammar *(MCFTG for short) is a tuple $G = (N, \Sigma, \mathbf{Y}, P, S)$ where:*

- *N is a finite multi-typed alphabet of elements called* non-terminals*.*
- *Σ is a finite second-order alphabet of elements called* terminals*.*
- *$\mathbf{Y}$ is an enumerable typed alphabet of variables disjoint from Σ.*
- *$S \in N$ is called the* starting non-terminal *of G and verifies $r(S) = 1$.*
- *P is a finite set of* production rules *of the form:*

$$A_0(C_1, \ldots, C_{r_0}) \to A_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, A_n(x_{n,1}, \ldots, x_{n,r_n})$$

where

- *for very $0 \leq i \leq n$, $r(A_i) = r_i$.*
- *for every $1 \leq k \leq r$, $C_k \in \mathbb{T}^{\alpha_i}_{[]}(\Sigma \cup \mathbf{Y})$, where $\alpha_i = t(A_0, k)$.*
- *for every $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, r_i\}$, $x_{i,j} \in \mathbf{Y}$ is of type $t(A_i, j)$ and verifies $\Sigma_{1 \leq k \leq r_0} |C_k|_{x_i,j} = 1$. Moreover, given $i_1, i_2 \in \{1, \ldots, n\}$ and $j_1 \in \{1, \ldots, r_{i_1}\}$, $j_2 \in \{1, \ldots, r_{i_2}\}$, we have $x_{i_1,j_1} = x_{i_2,j_2}$ iff $i_1 = i_2$ and $j_1 = j_2$*

Such a grammar is said linear *(resp.* almost affine*) if for every rule $\pi \in P$ of the form $A_0(C_1, \ldots, C_{r_0}) \to A_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, A_n(x_{n,1}, \ldots, x_{n,r_n})$, the contexts $C_1, \ldots, C_{r_0}$ are linear (resp. almost affine).*

Example 1. Let us consider the MCFTG $G = (N, \Sigma, \mathbf{Y}, P, S)$ where $N = \{S, A_1, A_2\}$, $\Sigma = \{a, b, c, d, \epsilon, e, f\}$, and P is made of the following production rules:

$$\begin{array}{ll} S(ex_1x_2x_3x_4) & \to A_1(x_1, x_3), A_2(x_2, x_4) \\ A_1(\epsilon, \epsilon) & \\ A_1(fax_1, fcx_3) & \to A_1(x_1, x_3) \\ A_1(\epsilon, \epsilon) & \\ A_1(fbx_2, fdx_4) & \to A_1(x_2, x_4) \end{array}$$

This MCFTG is a linear MCFTG. Moreover, $r(S) = 1$, $r(A_1) = r(A_2) = 2$ and $\tau(S, 1) = \tau(A_1, 1) = \tau(A_1, 2) = \tau(A_2, 1) = \tau(A_2, 2) = o$.

Definition 10. *Given a MCFTG $G = (N, \Sigma, \mathbf{Y}, P, S)$, and a non-terminal $A \in N$ of arity r, we define the relation of* derivation $\rightarrow_G^*$ *by:*

1. *if $A(C_1, \ldots, C_r)$ is a production rule in P, then $\rightarrow_G^* A(C_1, \ldots, C_n)$*
2. *$\rightarrow_G^* A(C_1 \cdot \sigma, \ldots, C_r \cdot \sigma)$ if:*
   - *σ is a substitution from $\mathbf{Y}$ to $\mathbb{T}_{[]}(\Sigma)$*
   - *there exists a rule $A(C_1, \ldots, C_r) \rightarrow A_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, A_n(x_{n,1}, \ldots, x_{n,r_n})$ in P*
   - *for every $1 \leq i \leq n$, the relation $\rightarrow_G^* A_i(x_{1,1} \cdot \sigma, \ldots, x_{i,r_i} \cdot \sigma)$ is verified.*

*The language $L(G)$ generated by a MCFTG $G = (N, \Sigma, \mathbf{Y}, P, S)$ is defined as the set $\{t \in \mathbb{T}_{[]}(\Sigma) \mid \rightarrow_G^* S(t)\}$. The class of tree languages **MCFTL** is defines as the smallest set which contains every language derived by some MCFTG.*

In the previous example, the language $L(G)$ derived by the MCFTG G is

$$\{c(\underbrace{fa(fa\ldots fa}_{m}\epsilon)...)(\underbrace{fb(fb\ldots fb}_{n}\epsilon)...)(\underbrace{fc(fc\ldots fc}_{m}\epsilon)...)(\underbrace{fd(fd\ldots fd}_{m}\epsilon)...) \mid n, m \in \mathbb{N}\}$$

Moreover, we can remark that $yL(G) = \{a^m b^n c^m d^n \ l \ n, m \in \mathbb{N}\}$ which is known to be a MCFL. This result is generalized in the following theorem:

Theorem 1 ([dGP04,Kan06]). *A language L belongs to **MCFL** iff there is a linear MCTFG G such that $yL(G) = L$.*

Finally, we give the following theorem related to the complexity of the recognition problem of tree languages derived by a MCFTG.

Theorem 2 ([Kan07,Yos06], [BS11]). *The recognition problem of an almost affine MCFTG is **LOGCFL**.*

Because the class of problems **LOGCFL** is a sub-class of the problem solvable in polynomial-time, this theorem will be of particular importance so as to prove that the new class of language we define in this article is recgnizable in polynomial time, and is hence included in **MCSL**.

3 IO-substitutions, semilinearity and constant-growth

3.1 IO-substitution and copies

In this section, we introduce a new operation on languages, which we will use to define constant-growth languages. This operation, which we call *IO-substitution* enables a specific kind of copying operation. Thanks to this operation, we want to capture some languages which do not belong to **MCFL**, but still verify the constant-growth property, as shown in [Kal10]:

Example 2. The language $L_{a,b}\{(a^n b^n)^m \mid n, m \in \mathbb{N} - \{0\}\}$ is not a **MCFL**. Nevertheless, it is a semilinear language, as its Parikh's image is $\{(1,1) + k(1,1) \mid k \in \mathbb{N}\}$.

The language $L_{count} = \{(a^n d)^m a^n \mid n, m \in \mathbb{N}\}$ is not a semilinear language, but verifies the constant-growth property.

The language $L_{exp} = \{a^{2^n} \mid n \in \mathbb{N}\}$ is neither a semilinear, nor a language which verifies the constant-growth property.

Interestingly, those languages are also known to be generated by IO-macro grammars [Fis68b,Fis68a], in which copying operations are allowed in a specific way, which we will not discuss in detail in this article. In brief words, Fischer makes a clean distinction between IO derivations, in which every copies of the same occurrence of a non-terminal are replaced using the same rewriting rule, and an OI derivation, in which such a constraint does not exist. For instance, let us consider a word $w = \alpha_1 A \alpha_2 A \alpha_3$, where A does not appear in $\alpha_1\alpha_2\alpha_3$, and the two occurrences of A are supposed to result from a copying operation. We consider the rewritings of A as w_1 or w_2. An IO derivation forces the two occurrences of A to be rewritten using the same rewriting rule; it hence defines the language $\{\alpha_1 w \alpha_2 w \alpha_3 \mid w \in \{w_1, w_2\}\}$. An OI derivation would lead to the derivation of the language $\{\alpha_1 w'_1 \alpha_2 w'_2 \alpha_3 \mid w'_1, w'_2 \in \{w_1, w_2\}\}$. We use the same ideas to define the IO-substitution on languages, as contrasted with the classic susbstitution, which can be seen as an OI-substitution.

Definition 11. *Let us consider an alphabet Σ and a variable x, and two languages $L_1 \in (\Sigma \cup \{x\})^*$ and $L_2 \in \Sigma^*$. Given words $w_1 \in L_1$, $w_2 \in L_2$ and $x \in \chi$, we define the word $w_1[x := w_2]_{IO} \in \Sigma$ by induction:*

- $x[x := w_2]_{IO} = w_2$
- $a[x := w_2]_{IO} = a$ *if* $a \neq x$
- $ww'[x := w_2]_{IO} = w[x := w_2]_{IO} w'[x := w_2]_{IO}$

The language $L_1[x := L_2]_{IO}$ is then defined as $\{w_1[x := w_2]_{IO} \in \Sigma \mid w_1 \in L_1, w_2 \in L_2\}$.

Based on this operation, we can build the languages $L_{a,b}$ and L_{count} easily from context-free languages. Indeed, given $L_1 = \{x^n \mid n \in \mathbb{N} - \{0\}\}$ and $L_2 = \{a^m b^m \mid m \in \mathbb{N} - \{0\}\}$, the language $L_{a,b}$ is the language $L_1[x := L_2]_{IO}$. In the same way, $L_{count} = L'_1[x := L'_2]_{IO}$, where $L'_1 = \{(xd)^m x \mid m \in \mathbb{N}\}$ and $L'_2 = \{a^n \mid n \in \mathbb{N}\}$. On the other hand, there is no trivial way to use the IO-substitution operation on context-free languages or mildly context-sensitive languages to generate L_{exp}. One can also remark that L_1, L_2, L'_1 and L'_2 are context-free languages, and are therefore semilinear languages (hence, verifiy the constant-growth property). It is therefore natural to investigate the conditions under which semilinearity and the constant-growth propery are preserved by the IO-substitution operation.

3.2 Preserving the semilinearity and constant-growth properties

In the previous examples, we have seen that the newly introduced operation of IO-substitution allows, in some cases, building semilinear languages or constant-growth languages from semilinear languages. We now investigate the conditions which allow preserving these properties.

Definition 12. *Let us consider a semilinear language $L \subseteq \Sigma^*$ and its Parikh image $p(L) = \bigcup_{i \in I} S_i$ ($I \subset \mathbb{N}$ finite), where for every $i \in I$,*

$$S_i = \{\overrightarrow{v_{i,0}} + n_1 \overrightarrow{v_{i,1}} + \ldots + n_{r_i} \overrightarrow{v_{i,r_i}} \mid n_1, \ldots, n_{r_i} \in \mathbb{N}\}$$

Given $a \in \Sigma$, L is said a-isolating *if, for every $i \in I$, there exists a unique $l_i \in \{0, \ldots, r_i\}$ such that:*

– *for every* $j \in \{0, \dots r_l\}$, $\overrightarrow{v_{i,j}}(k_a) \neq 0$ *iff* $j = l_i$.
– *for every* $c \in \Sigma - \{a\}$, $\overrightarrow{v_{i,l_i}}(k_c) = 0$

Example 3. The language $\{x^n \mid n \in \mathbb{N}\}$ is obviously x-isolating. The language $abx(a)^*x^*$ is not x-isolating as its Parikh image is $\{(1,1,1)+n(1,0,0)+m(0,0,1) \mid n,m \in \mathbb{N}\}$. Finally, the language $(xa)^*x$ is not x-isolating, its Parikh image being $\{(0,1)+n(1,1) \mid n \in \mathbb{N}\}$.

Remark 2. Given a language $L \in \Sigma^*$ and $a \in \Sigma$ such that L is a-isolating, each linear set $S \in \{S_1, \dots, S_l\}$, such that $\bigcup_{i=1}^{l} S_i = p(L)$ can be written as:

$$\Sigma_{i \in I} n_i \overrightarrow{x_i} + n_t \overrightarrow{x_t}$$

where $n_0 = 1$, $t \notin I$, for every $i \in I \cup \{t\}$ $\overrightarrow{x_i}(k_a) \neq 0$ iff $i = t$, and $\overrightarrow{x_t}(k_c) = 0$ for every $c \in \Sigma - \{a\}$.

We now show that the IO-substitution of a letter a by a semilinear language, on a a-isolating language generates a semilinear language.

Theorem 3. *Given a semilinear language $L_1 \in (\Sigma \cup \{x\})^*$, x-isolating, and a semilinear language L_2, the language $L_1[x := L_2]_{IO}$ is semilinear.*

Proof. Let us consider the language $L = L_1[x := L_2]_{IO}$, and a word $w \in L$. By definition, there exist $w_1 \in L_1$ and $w_2 \in L_2$ such that $w = w_1[x := w_2]_{IO}$. Because L_1 and L_2 are semilinear, for $p(L_1)$ and $p(L_2)$ the Parikh images of L_1 and L_2 respectively, there exist linear sets $S_1 \subset p(L_1)$ and $S_2 \subset p(L_2)$ such that $p(w_1) \in S_1$ and $p(w_2) \in S_2$; hence:

– $p(w_1) \in \{\overrightarrow{x_0} + n_1\overrightarrow{x_1} + \dots + n_p\overrightarrow{x_p} \mid n_1, \dots, n_p \in \mathbb{N}\}$
– $p(w_2) \in \{\overrightarrow{y_0} + m_1\overrightarrow{y_1} + \dots + m_r\overrightarrow{y_r} \mid m_1, \dots, m_r \in \mathbb{N}\}$

Moreover, because L_1 is x-isolating, we can write

$$p(w_1) \in \{\Sigma_{i \in I} n_i \overrightarrow{x_i} + n_t \overrightarrow{x_t} \mid \text{for all } i \in I \cup \{t\}, n_i \in \mathbb{N} \text{ and } n_0 = 1\}$$

where $t \notin I$, $I \cup \{t\} = \{0, \dots, p\}$, $\overrightarrow{x_i}(k_x) = 0$ for every $i \in I$, and $\overrightarrow{x_t}(k_c) = 0$ for every $c \in \Sigma$.

Then, by construction, the Parikh image of w can be written as

$$p(w) = \overrightarrow{z_0} + n_1\overrightarrow{z_1} + \dots + n_p\overrightarrow{z_p}$$

where, for every $i \in \{0, \dots, p\}$, $\overrightarrow{z_i} = \overrightarrow{x_i} + \overrightarrow{x_i}(k_x)\overrightarrow{p}(w_2)$. Let us write $\overrightarrow{x_i}(k_x) = K_i$, for every $i \in \{0, \dots, p\}$

The Parikh image of w can be rewritten according to the following results:

1. $\overrightarrow{z_i} = \overrightarrow{x_i}$, for every $i \in I$, because $K_i = 0$
2. $\overrightarrow{z_t} = K_t\overrightarrow{p}(w_2)$, because $\overrightarrow{x_t}(k_a) = 0$ for every $a \neq x$. Hence,

$$\begin{aligned} \overrightarrow{z_t} &= K_t(\overrightarrow{y_0} + m_1\overrightarrow{y_1} + \dots + m_r\overrightarrow{y_r}) \\ &= K_t\overrightarrow{y_0} + m_1K_t\overrightarrow{y_1} + \dots + m_rK_t\overrightarrow{y_r} \end{aligned}$$

Finally,

$$\begin{aligned} p(w) &\in \{\Sigma_{i\in I} n_i \overrightarrow{x_i} + n_t \Sigma_{j\in\{1,\dots,r\}} m_j K_t \overrightarrow{y_j} \mid \forall i \in \{1,\dots,r\}, j \in I \cup \{t\}, m_i, n_j \in \mathbb{N}\} \\ &\in \{\Sigma_{i\in I} n_i \overrightarrow{x_i} + n_t \Sigma_{j\in\{1,\dots,r\}} m_j \overrightarrow{y_{jt}} \mid \forall i \in \{1,\dots,r\}, j \in I \cup \{t\}, m_i, n_j \in \mathbb{N}\} \\ &\in \{\Sigma_{i\in I} n_i \overrightarrow{x_i} + \Sigma_{j\in\{1,r\}} m_j \overrightarrow{y'_{jt}} \mid \forall i \in \{1,\dots,r\}, j \in I \cup \{t\}, m_i, n_j \in \mathbb{N}\} \end{aligned}$$

belongs to a linear set.

Moreover, for $w' = w'_1[x := w'_2]_{IO}$, where $\overrightarrow{p}(w'_1)$ and $\overrightarrow{p}(w'_2)$ belong to the same linear sets as $\overrightarrow{p}(w_1)$ and $\overrightarrow{p}(w_2)$, it is easy to verify that $\overrightarrow{p}(w')$ belongs to the same linear set as $\overrightarrow{p}(w)$. The set of linear sets built in the proof is trivially finite because $\overrightarrow{p}(L_1)$ and $\overrightarrow{p}(L_2)$ are finite. Hence L is a semilinear language.

We now show that the preservation of the constant-growth property is more direct, as substituting a letter by a constant-growth language, in a constant-growth language suffices.

Theorem 4. *Given a constant-growth language $L_1 \subseteq (\Sigma \cup \{x\})^*$, and a constant-growth language $L_2 \subseteq \Sigma^*$. The language $L_1[x := L_2]_{IO}$ is constant-growth.*

Proof. Let us a consider a word w_1 in L_1, and a word w_2 in L_2. We note by k the number of occurrences of x in w_1 (*i.e.* $\overrightarrow{p}(w_1)(k_x) = k$).

By definition of a constant-growth language, for every $i \in \{1,2\}$, there exists a constant $c_i \in \mathbb{N}$ such that, for every word $w'_i \in L_i$ verifying $|w'_i| \geq c_i$, there exist $\overrightarrow{x}, \overrightarrow{y} \in \mathbb{N}^{|\Sigma|}$ such that $\overrightarrow{p}(w'_i) = \overrightarrow{x} + \overrightarrow{y}$ and, for every $k \geq 1$, $\overrightarrow{x} + k\overrightarrow{y} \in \overrightarrow{p}(L_i)$.

Let us first suppose $|w_2| \geq c_2$. Then $\overrightarrow{p}(w_1[x := w_2]_{IO}) = \overrightarrow{z} + k\overrightarrow{p'}(w_2) = \overrightarrow{z} + k\overrightarrow{x'_2} + k\overrightarrow{y'_2}$, where $\overrightarrow{z}$ is the Parikh image of w_1 on Σ, and $\overrightarrow{p'}(w_2)$ is the Parikh image of w_2 on the same alphabet. By hypothesis, we can consider an integer $i \geq 1$ and a word $w'_2 \in L_2$ such that $\overrightarrow{p}(w'_2) = \overrightarrow{x_2} + i\overrightarrow{y_2}$. Then, $\overrightarrow{p}(w_1[x := w'_2]_{IO}) = \overrightarrow{z} + k\overrightarrow{p'}(w'_2) = \overrightarrow{z} + k\overrightarrow{x'} + ki\overrightarrow{y'}$.

Otherwise, suppose $|w_2| < c_2$ and $|w_1| \geq c_1$. Then $\overrightarrow{p}(w_1) = x_1 + y_1$, and given $i > 1$, there exist a word $w'_1 \in L_1$ such that $\overrightarrow{p}(w'_1) = \overrightarrow{x_1} + i\overrightarrow{y_1}$. We give $k_1 = \overrightarrow{x_1}(x)$ and $k_2 = \overrightarrow{y_1}(x)$. Then

- $\overrightarrow{p}(w_1[x := w_2]_{IO}) = \overrightarrow{x'_1} + k_1\overrightarrow{p'}(w_2) + \overrightarrow{y'_1} + k_2\overrightarrow{p'}(w_2)$
- $\overrightarrow{p}(w'_1[x := w_2]_{IO}) = \overrightarrow{x'_1} + k_1\overrightarrow{p'}(w_2) + i\overrightarrow{y'_1} + ik_2\overrightarrow{p'}(w_2)$

where given $\overrightarrow{x}$ a Parikh vector on $\Sigma \cup \{x\}$, $\overrightarrow{x'}$ is the same vector on Σ; and $\overrightarrow{p'}(w_2)$ is the Parikh image of w_2 on the same alphabet.

As a conclusion, $L_1[w := L_2]_{IO}$ is a constant-growth language. We can consider the constant associated to the growth of this language as c_1c_2.

Thanks to the two previous theorems, we proved that the constant-growth and semilinearity properties of languages can be preserved by the IO-substitution operation. In the following section, we seek a class of languages bigger than **MCFL** which is included in **MCSL**, by using this operation.

4 IO-Multiple Context-Free Languages

4.1 Definition

In the previous section, we proved some properties on the IO-substitution which allows preserving the semilinearity or constant-growth properties. As mentionned in the first section, we can consider the class of multiple context-free languages as the biggest class approximating mildly context-sensitive languages. We now extend the class **MCFL** by using the IO-substition operation.

Definition 13. *We consider the family of* IO-mildly context sensitive languages *(written **IO-MCFL**) as the smallest family such that $L \in$ **IO-MCFL** if:*

- $L \in$ ***MCFL*** *or*
- *if $L = L_1[x := L_2]_{IO}$ where $L_1, L_2 \in$ **IO-MCFL**.*

Thanks to this definition, languages $L_{a,b}$ and L_{count} can be considered to belong to **IO-MCFL**. The following theorem is a direct corrolary of Theorem 4.

Theorem 5. *Every language L in **IO-MCFL** verifies the constant-growth property.*

Proof. We proceed by induction on L. If $L \in MCFL$, the result is given by the fact that L is semilinear. Otherwise, there exist $L_1 \in$ **IO-MCFL** and $L_2 \in$ **IO-MCFL** such that $L = L_1[x := L_2]_{IO}$. By induction hypothesis, L_1 and L_2 verify the constant-growth property. According to Theorem 4, L is constant-growth.

In order to see if **IO-MCFL** is a better approximation of the class **MCSL**, we then need to know if such languages are tractable in polynomial time. In order to do so, we give a formalism which exactly captures the **IO-MCFL** class of languages, thanks to the multiple context-free tree languages given in section 2.2.

4.2 IO-MCFGs as quasi-affine MCFTGs

As mentionned in section 2.2, the class **MCFTL** is connected to **MCFL** through theorem 1. In what follows, we seek the characterization of some MCFTGs which yield are exactly IO-MCFLs.

Intuitively, given a language $L_1 \in \Sigma \cup \{x\}$, where L_1 and L_2 are MCFLs, we can remark that the letter x may have many occurrences in $w_1 \in L_1$. Given a MCFTG G_1 such that $yL(G_1)$, the idea is too consider x not as a leaf in the trees derived by G_1, but by a variable which can be substituted by a word $w_2 \in L_2$. Therefore, we need to build a MCFTG G'_1 such that $t \in L(G_1)$ iff $\lambda x.t \in L(G'_1)$, and then apply a simple $S(x_1x_2) \to S'_1(x_1), S(x_2)$ where S'_1 and S_2 are the starting non-terminals of G'_1 and G_2 respectively.

Remark that, because x will now be considered as a variable, the contexts appearing in the left-hand side of a production rule in G'_1 will not be linear anymore. But x being a leaf, its type is $\tau(x) = o$, and the contexts will therefore be almost affine, which still ensures the recognizability problem belongs to **LOGCFL**, hence to the class of problems solvable in polynomial-time.

Definition 14. *Given a MCFTG* $G = (N, \Sigma \cup \{x\}, \mathbf{Y}, P, S)$ *where* $\tau(x) = \alpha$*, we define the MCFTG*

$$\mathsf{abs}(G,x) = (\mathsf{abs}(N,x), \Sigma, \mathsf{abs}(\mathbf{Y},x), \mathsf{abs}(P,x), \mathsf{abs}(S,x))$$

as follows:

- *given a type* $\alpha = \alpha_1 \to \ldots \to \alpha_n \to \alpha_0$ *in* $\mathscr{T}_o$*, we define* $\mathsf{abs}(\alpha,x) = \tau(x) \to \alpha_1 \to \ldots \to \alpha_n \to \alpha_0$.
- *for every* $A \in N$, $\mathsf{abs}(A,x) = A$ *such that* $r(A') = r(A)$ *and for every* $1 \leq i \leq r(A)$, $t(A',i) = \mathsf{abs}(t(A,i),x)$*. Finally,* $\mathsf{abs}(N,x) = \{\mathsf{abs}(A,x) \mid A \in N\}$.
- $y \in \mathbf{Y}$ *and* $\tau(y) = \gamma$ *in* $\mathbf{Y}$ *iff* $y \in \mathsf{abs}(\mathbf{Y},x)$ *and* $\tau(y) = \mathsf{abs}(y,x)$ *in* $\mathsf{abs}(\mathbf{Y},x)$.
- *given a tree context* C *in* $\mathbb{T}_{[]}(\Sigma \cup \mathbf{Y})$*, we define* $\mathsf{abs}(C,x) \in \mathbb{T}_{[]}(\Sigma \cup \mathsf{abs}(\mathbf{Y},x))$*, as* $\lambda x.\mathsf{abs}(C,x)$*, where:*
  - $\mathsf{abs}(\lambda y.C',x) = \lambda y.\mathsf{abs}(C',x)$*, for* $C' \in \mathbb{T}_{[]}(\Sigma \cup \mathbf{Y}\})$
  - $\mathsf{abs}(c,x) = c$ *for* $c \notin \mathbf{Y}$
  - $\mathsf{abs}(y,x) = (yx)$*, for* $y \in \mathbf{Y}$.
  - $\mathsf{abs}(t_1 t_2,x) = \mathsf{abs}(t_1,x)\mathsf{abs}(t_2,x)$.

  Finally a rule

  $$A_0(C_1,\ldots,C_r) \to A_1(x_{1,1},\ldots,x_{1,r_1}),\ldots,A_n(x_{n,1},\ldots,x_{n,r_n})$$

  belongs to P *iff*

  $$A'_0(C'_1,\ldots,C'_r) \to A'_1(x'_{1,1},\ldots,x'_{1,r_1}),\ldots,A'_n(x'_{n,1},\ldots,x'_{n,r_n})$$

  belongs to $\mathsf{abs}(P,x)$*, where* $A'_i = \mathsf{abs}(A_i,x)$ *and* $x'_{i,j} = \mathsf{abs}(x_{i,j},x)$ *for every* $0 \leq i|eqn$ *and* $1 \leq j \leq r_i$*, and* $C'_k = \mathsf{abs}(C_k,x)$ *for every*$1 \leq k \leq r$.

The transformation given by abs intuitively result in considering x as a variable, on which tree context substitutions can be applied. This is made explicit in the following theorem.

Lemma 1. *Given a MCFTG* $G = (N, \Sigma \cup \{x\}, \mathbf{Y}, P, S)$*, the language* $L(\mathsf{abs}(G,x))$ *is equal to* $\{\lambda x.C \in \mathbb{T}_{[]}(\Sigma) \mid C \in L(G)\}$.

Proof. We consider the general form of a MCFTG so that the language derived is a tuple of tree contexts, and We proceed by induction on the derivation of such a tuple $\overline{C} \in L$. Let us consider the last rule π used to derive $\overline{C}$ in G, which general form is:

$$A_0(C_1,\ldots,C_r) \to A_1(\overline{x_1}),\ldots,A_n(\overline{x_n})$$

If $n = 0$, then $\overline{C} = (C_1,\ldots,C_r)$ and $\mathsf{abs}(\pi,x) = \mathsf{abs}(A_0,x)(C'_1,\ldots,C'_r)$, where for every $1 \leq i \leq r$, $C'_i = \mathsf{abs}(C_i,x)$. Moreover, $C'_i \in \mathbb{T}_{[]}(\Sigma)$ and $C'_i = \lambda x.C_i$ by construction. The tuple $(\lambda x.C_1,\ldots,\lambda x.C_r)$ is therefore recognized by $\mathsf{abs}(G,x)$.

If $n \neq 0$; by induction hypothesis, for every $1 \leq k \leq n$, the following statement stands: $\to^*_G A_k(C_{k,1},\ldots,C_{k,r_k})$ iff $\to^*_{\mathsf{abs}(G,x)} \mathsf{abs}(A_k,x)(\lambda x.C_{k,1},\ldots,\lambda x.C_{k,r_k})$. Therefore, given a tree context substitution $\sigma = [x_{1,1} := C_{1,1},\ldots,x_{1,r_1} := C_{1,r_1},\ldots,C_{n,1},\ldots,x_{n,r_n} := C_{n,r_n}]$, such that $\overline{C} = (C_1 \cdot \sigma,\ldots,C_r \cdot \sigma)$, we build $\sigma' = [x'_{1,1} := \lambda x.C'_{1,1},\ldots,x'_{n,r_n} := \lambda x.C'_{n,r_n}]$ where $x'_{i,j} := \mathsf{abs}(x_{i,j},x)$ and $C'_{i,j} := \mathsf{abs}(C_{i,j},x)$, for every $1 \leq i \leq n$ and every $1 \leq j \leq r_i$. By construction, we obtain $\overline{C} = (\lambda x.C_1,\ldots,\lambda x.C_r)$.

We now need to build the MCFTG which produces the substitution of x in a tree context C_1 derived in $\mathtt{abs}(G_1, x)$ by a tree context C_2 derived by G_2. In particular, if $\not\in tau(x) = o$, x may appear in the yield of any tree derived by G_1, and replacing every occurrence of x in $t_1 = C_1$ by the tree $t_2 = C_2$, resumes to $yt_1[x := yt_2]_{IO}$.

First, we define disjunction of MCFTGs as follows: given two MCFTGs $G_1 = (N_1, \Sigma_1, \mathbf{Y_1}, P_1, S_1)$ and $G_2 = (N_2, \Sigma_2, \mathbf{Y_2}, P_2, S_2)$, G_1 and G_2 are said disjoint if $N_1 \cap N_2 = \emptyset$. Remark that such a relation implies $P_1 \cap P_2 = \emptyset$ and $S_1 \neq S_2$.

Definition 15. *A MCFTG G is called an IO-MCFTG if*

1. *G is a linear MCFTG or*
2. *there exist $G_1 = (N_1, \Sigma \cup \{x\}, \mathbf{Y_1}, P_1, S_1)$ and $G_2 = (N_2, \Sigma, \mathbf{Y_2}, P_2, S_2)$, two disjoint IO-MCFTGs, such that given $G'_1 = (N'_1, \Sigma, \mathbf{Y'_1}, P'_1, S'_1) = \mathtt{abs}(G_1, x)$, $G = (N'_1 \cup N_2 \cup \{S\}, \Sigma, \mathbf{Y'_1} \cup \mathbf{Y_2}, P'_1 \cup P_2 \cup \{\pi\}, S)$ where:*
   (a) $S \notin N'_1 \cup N'_2$ and $r(S) = 1$, $t(S, 1) = o$.
   (b) $\pi \notin P'_1 \cup P_2$ and $\pi = S(x_1 x_2) \to S'_1(x_1), S_2(x_2)$.

We now prove that this formalism exactly captures tree languages which yield form IO-MCFLs:

Theorem 6. *A language L is a $IO - MCFL$ iff there exists a $IO - MCFTG$ such that $L = yL(G)$.*

Proof. We proceed by induction on L. If L is a MCFL, there exists a linear MCFTG G such that $yL(G) = L$, and G si a IO-MCFTG by definition. Conversely, for G a linear MCFTG, $yL(G)$ is a IO-MCFL. If there exist $L_1 \subseteq (\Sigma \cup \{x\})^*$ and $L_2 \subseteq \Sigma^*$ two MCFLs, such that $L = L_1[x := L_2]_{IO}$, by induction hypothesis, there exist two MCFTGs $G_1 = (N_1, \Sigma \cup \{x\}, \mathbf{Y}_1, P_1, S_1)$ and $G_2 = (N_2, \Sigma, \mathbf{Y}_2, P_2, S_2)$, which can be considered disjoint, and such that $L_1 = yL(G_1)$ and $L_2 = yL(G_2)$. According to Lemma 1, $L(\mathtt{abs}(G_1, x)) = \{\lambda x.t \in \mathbb{T}_{[]}^{o \to o}(\Sigma) \mid t \in L(G_1)\}$. We consider the grammar

$$G = (\mathtt{abs}(N_1, x) \cup N_2 \cup \{S\}, \Sigma, \mathtt{abs}(\mathbf{Y_1}, x) \cup \mathbf{Y_2}, \mathtt{abs}(P_1, x) \cup P_2\{\pi\}, S)$$

such that $S \notin \mathtt{abs}(N_1, x) \cup N_2$ and $\pi = S(x_1 x_2) \cup \mathtt{abs}(S_1, x)(x_1), S_2(x_2)$. A t tree is recognized by this grammar iff there exist $t_1 \in L(G_1)$ and $t_2 \in L(G_2)$ such that $t = \mathtt{app}(t_1, t_2)$ which is the result of substituting every occurrence of x in t_1 by t_2. Finally, $yL(G) = \{w_1[x := w_2]_{IO} \in \Sigma \mid w_1 \in L_1, w_2 \in L_2\}$. Conversely, the yield of the language of an IO-MCFTG is an IO-MCFL.

It now suffices to show that the recognizability problem in this formalism can be solved in polynomial-time, which is a direct consequence of the following lemma.

Lemma 2. *If G is an almost affine MCFTG, then $\mathtt{abs}(G, x)$ is an almost affine MCFTG.*

Proof. In order to prove this property, it suffices to prove that, given an almost affine tree context C in $\mathbb{T}_{[]}(\Sigma \cup \{x\})$, $\lambda x.\mathtt{abs}(C, x)$ is a tree context in $\mathbb{T}_{[]}(\Sigma)$. Therefore, given the general form $\lambda x_1^{\alpha_1} \dots x_n^{\alpha_n}.t$ of C, if a variable $y^\alpha \notin \Sigma \cup \{x\})$ verifies $|C|_y \neq 1$ then there exists $1 \leq j \leq n$ such that $y = x_j$ and $\alpha_j = o$. Moreover, x might have a number of occurrences in C different from 0, which implies the same property is verified on $\lambda x.C$, which therefore remains almost affine.

Corollary 1. *The recognition problem of an IO-MCFL belongs to **LOCGCFL**.*

As a conclusion, we proved that **MCFL** $\subsetneq$ **IO-MCFL** $\subseteq$ **MCSL**. This new class of languages can therefore be taken as the best approximation of the class of mildly context-sensitive languages.

While we proved that the recognition problem can be solved in polynomial-time, we next prove some usual properties taken to test the robustness of parsing in a given formalism, so as to explore additional proeprties of **IO-MCFL**.

4.3 Closure properties

In what follows, we study other properties enjoyed by IO-MCFLs. In particular, we are interested in the usual closure properties so as to know whether **IO-MCFLs** define a full abstract family of languages.

Definition 16. *A family of languages **L** is called a* (full)-abstract family of languages *(written* (full)-AFL*) if **L** is closed by homomorphism, inverse homomorphism, intersection with a regular set, union, concatenation and the Kleene star.*

In what follows, we show that most of this properties are verified.

Remark 3. A language $L \in MCFL$ can be written $L = L[x := L']_{IO}$, where x has no occurrence in L. The general form a **IO-MCFL** is therefore $L'[x := L'']_{IO}$

Definition 17. *Given an alphabet Σ, a function $\mathscr{H} : \Sigma \mapsto \Sigma$ is called a renaming.*

Lemma 3. *The family **IO-MCFL** is closed by renaming.*

Proof. Trivial.

The next closure properties are verified through direct proofs.

Theorem 7. *The family **IO-MCFL** is closed by homomorphism, union, intersection, concatenation and the Kleene star.*

Proof. Let us consider $L, L_1, L_2 \in$ **IO-MCFL**, languages built on the alphabet Σ:

- we consider a homomorphism $\mathscr{H} : \Sigma^* \mapsto \Sigma^*$, and show that $\mathscr{H}(L) \in$ **IO-MCFL** by induction on L. If L is a MCFL, the property is verified as the family of MCFLs is an AFL. Otherwise, there exist $L_1, L_2 \in IO - MCLF$ such that $L = L_1[x := L_2]_{IO}$; then $\mathscr{H}(L) = \mathscr{H}(\mathscr{L}[:= \mathscr{L}]_{\mathscr{J}\mathscr{O}})$. We built the $L'_1 = \mathscr{F}(L_1)$, where $\mathscr{F}$ is a symbol replacement such that $\mathscr{F}(x) = x'$ and $\mathscr{H}(x') = x'$; $\mathscr{F}(a) = a$ otherwise. Then $L_1[x := L_2]_{IO} = L'_1[x' := L_2]_{IO}$, and $\mathscr{H}(\mathscr{L}) = \mathscr{H}(L'_1[x' := L_2]_{IO}) = \mathscr{H}(L'_1)[x := \mathscr{H}(L_2)]$. By induction hypothesis, $\mathscr{H}(L'_1), \mathscr{H}(L_2) \in$ **IO-MCFL** and by construction, $\mathscr{H}(L) \in$ **IO-MCFL**.

– We show that $L_1 \cup L_2 \in$ **IO-MCFL**. We can consider the general form $L_i = L'_i[x_i := L_2]_{IO}$, for $i \in \{1,2\}$. Moreover, x_1 and x_2 can be renamed so that x_1 (*resp.* x_2) has no occurrence in L_2 (*resp.* L_1). Then

$$\begin{aligned} L_1 \cup L_2 &= L'_1[x_1 := L''_1]_{IO} \cup L'_2[x_2 := L''_2]_{IO} \\ &= ((L'_1 \cup L'_2)[x := L''_1]_{IO})[x := L''_2]_{IO} \\ &= ((L'_1 \cup L'_2)[x := L''_2]_{IO})[x := L''_1]_{IO} \end{aligned}$$

and $L'_1 \cup L'_2$ is a $IO-MCLF$ by induction hypothesis (the result is given in the case L'_1 and L'_2 are MCFLs), which implies $L_1 \cup L_2$ is a **IO-MCFL** by construction.
– We show that $L_1 \cap L_2 \in$ **IO-MCFL**, in a similar way:

$$\begin{aligned} L_1 \cap L_2 &= L'_1[x_1 := L''_1]_{IO} \cap L'_2[x_2 := L''_2]_{IO} \\ &= ((L'_1 \cap L'_2)[x := L''_1]_{IO})[x := L''_2]_{IO} \\ &= ((L'_1 \cap L'_2)[x := L''_2]_{IO})[x := L''_1]_{IO} \end{aligned}$$

and $L'_1 \cap L'_2$ is a $IO-MCLF$ by induction hypothesis, (the result is given in the case L'_1 and L'_2 are MCFLs) which implies $L_1 \cap L_2$ belongs to **IO-MCFL** by construction.
– we show $L_1 \cdot L_2$ belongs to **IO-MCFL** in a similar way:

$$\begin{aligned} L_1 \cdot L_2 &= L'_1[x_1 := L''_1]_{IO} \cdot L'_2[x_2 := L''_2]_{IO} \\ &= ((L'_1 \cdot L'_2)[x := L''_1]_{IO})[x := L''_2]_{IO} \\ &= ((L'_1 \cdot L'_2)[x := L''_2]_{IO})[x := L''_1]_{IO} \end{aligned}$$

and $L'_1 \cdot L'_2 \in IO-MCLF$ by induction hypothesis, (the result is given in the case L'_1 and L'_2 are MCFLs) which implies $L_1 \cdot L_2$ is a IO-MCFL by construction.
– the closure by Kleene star is a generalization of the previous result.

Property 1. The family **IO-MCFL** is closed by intersection with a regular set.

Proof. Let us consider a IO-MCFL L and a regular language L. We prove that $L \cap R$ belongs to **IO-MCFL**, by induction on L. If L is a MCFL, the result is given by the fat that MCFLs form a AFL. Let us suppose $L = L_1[x := L_2]_{IO}$. We consider $\mathbb{M}_R = (\mathscr{M}, \cdot, \epsilon)$ the syntactic monoid of R, where $\mathscr{M}$ is finite according to Myhill-Nerode. Then, there exists a homomorphism $\phi : \Sigma \mapsto \mathbb{M}_R$ and $\mathscr{N} \subseteq \mathbb{M}_R$ such that $R = \phi^{-1}(\mathscr{N})$. Given $m \in \mathscr{M}$ let us consider the homomorphism $\phi_m : \Sigma \cup \{x\} \mapsto \mathbb{M}_R$ defined by $\phi_m(x) = m$ and for every $a \in \Sigma$, $\phi_m(a) = \phi(a)$. We consider the regular set $R_m = \phi_m^{-1}(\mathscr{N})$. By induction hypothesis, $L_{1,m} = L_1 \cap R_m$ is in **IO-MCFL**. The language $L_{2,m} = L_2 \cap \phi^{-1}(\{m\})$ enjoys the same property, and by construction, $L_m = L_{1,m}[x := L_{2,m}]_{IO}$ belongs to **IO-MCFL**. Finally, because $L \cap R = \bigcup_{m \in \mathscr{M}} L_m$, which is a finite union, and **IO-MCFL** is closed by union, we can conclude that $L \cap R$ is a IO-MCFL.

While all these properties can be proved in quite easily, the closure under inverse homomorphism does not seem as trivial. Actually, we can conjecture that such a property does not stand.

5 Conclusion

References

[Bar84] Henk Barendregt. *λ-calculus: its syntax and semantics*. Elsevier Science Publishers Ltd., 1984.

[BS11] Pierre Bourreau and Sylvain Salvati. A Datalog recognizer for almost affine λ-CFGs. In Makoto Kanazawa, Andreàs Kornai, Marcus Kracht, and Hiroyuki Seki, editors, *MOL*, volume 6878 of *Lecture Notes in Artificial Intelligence*, pages 21–38. Springer, 2011.

[dG01] Philippe 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.

[dGP04] Philippe de Groote and Sylvain Pogodalla. On the expressive power of abstract categorial grammars: Representing context-free formalisms. *Journal of Logic, Language and Information*, 13(4):421–438, 2004.

[Fis68a] Michael J. Fischer. Grammars with macro-like productions. In *IEEE Conference Record of 9th Annual Symposium on Switching and Automata Theory*, pages 131–142. IEEE, 1968.

[Fis68b] Micheal J. Fischer. *Grammars with macro-like productions*. PhD thesis, Harvard University, 1968.

[JLT75] Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1):136–163, 1975.

[Jos85] Aravind K. Joshi. Tree-adjoining grammars: How much context-sensitivity is required to provide reasonable strucutral descriptions? *Natural Language Parsing: Psychological, Computational and Theoretical Perspectives*, pages 206–250, 1985.

[Kal10] Laura Kallmeyer. On mildly context-sensitive non-linear rewriting. *Research on Language and Computation*, 8(2):341–363, 2010.

[Kan06] Makoto Kanazawa. Abstract families of abstract categorial grammars. In Stanford University CSLI, editor, *Proceedings of WoLLIC*, 2006.

[Kan07] Makoto 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.

[Mus01] Reinhard 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.

[SMMK91] Hiroyuki Seki, Takashi Matsamura, Fujii Mamoru, and Tadao Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229, 1991.

[SU06] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*. Elsevier Science, 2006.

[Wei88] David Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania, 1988.

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