Logic for Unambiguous Context-Free Languages

Yassine Hachaïchi

Logic for Unambiguous Context-Free Languages

Yassine Hachaïchi

LAMSIN - ENIT, Université de Tunis El Manar

Abstract

We give in this paper a logical characterization for unambiguous Context Free Languages, in the vein of descriptive complexity. A fragment of the logic characterizing context free languages given by Lautemann, Schwentick and Thérien based on implicit definability is used for this aim. We obtain a new connection between two undecidable problems, a logical one and a language theoretical one.

Key words: Descriptive complexity, logic and language theory, implicit definability on finite models.

1 Introduction

A language L over an alphabet A can be defined by several manners. The most famous are:

(1) a subset of A^* whose elements satisfy some given property; it is the analogous of comprehension schema in set theory,
(2) a subset of A^* whose elements are generated by some formal grammar,
(3) a subset of A^* whose elements are recognized by some model of computation.

In complexity theory, we try to classify languages according to the recognizer used (finite automata, push down automata, $\cdots$), or by the ressources (time, space, $\cdots$) neded by some model of computation (turing machine, random access machine, $\cdots$), see [21] for a detailed introduction to the field.

Email address: hachaichi.ens@gmail.com (Yassine Hachaïchi).

One of the aims of *descriptive complexity* [7,23] is to evaluate how easy or hard it is to express a given property defining some language (as in 1 above) in the language of logic.

The answer to this question got a meaning by the works of Büchi [4] and Elgot [9] who made the link between formal logic and formal language theory. This connection was made by identifying words to *finite* logical structures. Their result was that a word language is regular if, and only if it is the class of models of some Monadic Second-order sentence. Two questions were naturally asked:

The first one is:

(1) What is the expressive power of Monadic Second-order Logic on other structures than words, graphs and trees for example?

The other question is:

(2) Is there a logical description for each known class of words: star free, context free, ...?

Both directions were explored since, we will recall some results in the next section.

The logical description of the behaviour of computational models was also taken up in complexity theory. Starting with Fagin's work, it was shown that many complexity classes such as NP, P, LogSpace, NLogSpace, Pspace, $\cdots$ could be characterized by different varieties of second-order logic (involving for example fixed point logic or transitive closure operator). For an introduction to this field see Ebbinghaus and Flum's book [7].

In [12] and [13], I used some generalized quantifiers of comparison of cardinality, to get a new logical characterizations of the class of rudimentary languages in the scope of descriptive complexity. Lautemann, Schwentick and Thérien [18] gave recently a logical description of Context Free Languages. They used for this purpose the semantic quantifier of matching.

Our contribution in this paper is, in a first time, using a result of McNaughton and Papert [19] we will refine an algebraic normal form of Chomsky and Schützenberger which characterizes Context Free Languages using the Dyck languages.

The Second result of this paper is a description of Unambiguous Context Free Languages. This description uses a fragment of a logic built from first-order implicitely definable predicates introduced by Kolaitis [16]. This logic was motivated by the failure of the Beth property when we confine ourselves to finite structures.

This paper is organized as follows:

In the next section we give some background of language theory and logic, and we introduce some results of descriptive recognizability. We introduce in the last subsection the result of Lautemann and *al* [18].

In section 3 we refine an algebraic normal form given by Chomsky and Schützenberger [5], for describing Context Free Languages using the Dyck language.

In section 4 we give the logical characterization of unambiguous context free languages.

In the conclusion we try to link undecidability of unambiguity and undecidability of the logic IMP.

2 Notations and Background

We give here some definitions and results in formal language theory, logic and the connection between them.

For the rest of the section Σ will denote a finite vocabulary $\{c_1, \ldots, c_s\}$. A language is a subset of Σ^*, which is the set of finite words on Σ.

2.1 Formal Language Theory

We will recall in this section some notions of language theory, from the grammatical point of view, which we will use later in this paper, the curious reader can find more details on this area in Harrison's book [14].

A *context free grammar* is a 4-tuple $< \Sigma, N, S, P >$ such that:

- Σ and N are finite disjoint sets, called respectively the set of *terminal* and *non-terminal* symbols,
- S is a special symbol of N, called the *start symbol* or the *axiom* of the grammar,
- P is a *set of productions* of the form $X \to w$, where X is a non-terminal and $w \in (\Sigma \cup N)^*$.

If we replace each non-terminal symbol by a new symbol | not in $\Sigma \cup N$ in the right-hand side of a production we obtain a string called the *pattern* of the production.

A context free grammar is *regular* if all productions are of the form $X \to w|wY$ where X and Y are non-terminals and $w \in \Sigma^*$.

We define the (one step) *derivation rule* $\Rightarrow_G$ for a grammar G by

$$w_1 X w_2 \Rightarrow_G w_1 w w_2 \text{ is a derivation if and only if } X \to w \in P.$$

The reflexive and transitive closure of $\Rightarrow_G$ is denoted $\overset{*}{\Rightarrow}_G$.

A language $L \subseteq \Sigma^*$ is *context free* (resp. *regular*) if and only if there is a context free (resp. regular) grammar which derives it from S, i.e
$L = L(G) = \{w \in \Sigma | S \overset{*}{\Rightarrow}_G w\}$.

A language is *star free* if it is build from finite languages by only boolean operations and concatenation.

The *derivation tree* of a word w associates naturely to the derivations made from S until reaching w.

Formally, a *derivation tree of a word* $w \in L(G)$ is a tree so that :

- the root is labelled by the start symbol S,
- the leaves are labelled by terminals,
- the internal nodes are labelled by non terminals,
- the passage from an internal node to its sons corresponds to a production,
- the lecture of leaves from left to right give w.

Example Let's take the grammar

$$G =< \{a, b\}, \{X_0, X_1, X_2, X_3, X_4\}, X_0, P >$$

where P contains the following productions:

$P_{0,1} : X_0 \to aX_1X_2ba$

$P_{1,1} : X_1 \to aX_3X_2b$

$P_{2,1} : X_2 \to aab$

$P_{2,2} : X_2 \to ab$

$P_{3,1} : X_3 \to ab$

Let $w = aaababbaabba$ the word with derivation tree given in figure 1.

A context free grammar is *unambiguous* if every word in $L(G)$ has a unique derivation tree.

A context free language is *unambiguous* if it has an unambiguous context free

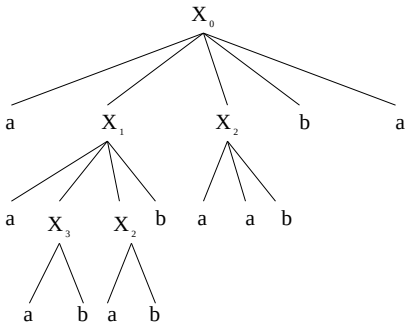

Fig. 1. A derivation tree of w

grammar which derives it.

2.2 Logic

As we mentioned in the introduction, we can identify words to finite models in a special logical signature $\tau_\Sigma = \{<, P_{c_1}, \ldots, P_{c_s}\}$.

In this introductory section to logic we define formally the word model and some logics that we will use in the forthcoming sections.

We will confine ourselves to finite structures and especially to words.

Let $\Sigma = \{c_1, \ldots, c_s\}$ be a finite vocabulary. We associate with a word $w = a_1 \ldots a_n$ over Σ, the *word model* S_w, namely the relational structure $S_w = (\{1, ..., n\}, <, P_{c_1}, \ldots, P_{c_s})$, where $<$ is the natural order on $\{1, ..., n\}$ and P_a is the unary predicate collecting the positions of w labelled a:
$P_a = \{i \in N | a_i = a\}$.

For example let's take the word $w = aabbabb$ on the vocabulary $\Sigma = \{a, b\}$.

The corresponding logical structure will be:

$S_w = (\{1, 2, 3, 4, 5, 6, 7\}, <, P_a^w = \{1, 2, 5\}, P_b^w = \{3, 4, 6, 7\})$.

The set of *first-order formulas* (on words), F.O., is built inductively from atomic formulas of the form:

$$x < y \text{ and } P_a(x) \text{ for } a \in \Sigma.$$

by means of connectives $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, *and* $\neg$ and the quantifiers $\exists$ *and* $\forall$.

Formally:

(1) If x, y are any variable symbol or constant symbol then $x < y$ and $P_a(x)$ for $a \in \Sigma$ are first-order formulas.

(2) If $\phi,\ \psi$ are first-order formulas then

$$\phi \wedge \psi,\ \phi \vee \psi,\ \phi \rightarrow \psi,\ \phi \leftrightarrow \psi,\ and \neg\phi$$

are also first-order formulas.

(3) If ϕ is a first-order formula and x is a variable then $\exists x\phi$, and $\forall x\phi$ are first-order formulas.

Monadic second-order logic, M.S.O., is built like first-order logic where we add $X_i(x)$ to atomic formulas (the first item in the first-order construction), for some unary variables $(X_i)_{i\in N}$ and can quantify over $X_i's$ (in the last item of the first-order construction).

A language L is said to be (explicitly) *definable in a logic* λ if and only if there exists a formula $\phi \in \lambda$ such that:

$$\forall w \in A^*(w \in L \Leftrightarrow S_w \models \phi).$$

Most classical model theoretical results fail when we confine ourselves to only finite models [7], especially Beth's theorem.

Let R be an n-ary relation symbol not in τ_Σ. An F.O. $[\tau_\Sigma \cup \{R\}]$ sentence, ϕ, *defines R implicitly* if every word structure has at most one expansion (S_w, R) to a $\tau_\Sigma \cup \{R\}$-structure satisfying ϕ.

The Beth's theorem says that a predicate is implicitly definable in F.O. if and only if it is explicitely definable in F.O., in the class of all structures (finite and infinite).

The failure of the Beth property for finite models, (see [7] for all properties of (classical) model theory that fail in finite model theory), stimulated Kolaitis [16] to define a logic of implicitly defined queries in some logic.

We will only use the case of first-order implicitely defined queries.

Remark. We use a different definition from the one used by Kolaitis in [16]. For a discussion see [7, page 213]

The logic $IMP = IMP(F.O.)$ is the set of formulas, that allows to define exactly those queries (properties) that are expressible in F.O. using first-order implicitely definable queries.

Formally:

An *IMP-formula* $\phi(\overline{x})$ is a tuple,

$$(\psi_1(R_1), \ldots, \psi_m(R_m), \psi(\overline{x}, R_1, \ldots, R_m))$$

where all $\psi_i's$ are first-order sentences on $\tau_\Sigma \cup \{R_i\}$, and ψ is first-order on $\tau_\Sigma \cup \{R_1, \ldots, R_m\}$
such that $\models_{fin} \exists! R_1 \ldots \exists! R_m(\psi_1(R_1) \wedge \ldots \wedge \psi_m(R_m))$.
Where $\models_{fin}$ means satisfiability on finite models, and $\exists!$ means there is a unique.

The meaning of $\phi(\overline{x})$ is fixed by requiring that:

$$S_w \models \forall X_1 \ldots \forall X_m(\psi_1(X_1) \wedge \ldots \wedge \psi_m(X_m))$$

$$\rightarrow \forall \overline{x}(\phi(\overline{x}) \leftrightarrow \psi(\overline{x}, X_1, \ldots, X_m))$$

This means: $S_w \models \phi(\overline{a}) \Leftrightarrow S_w \models \psi(\overline{a}, R_1, \ldots, R_m))$, where the $R_i's$ are uniquely determined by $S_w \models \psi_i(R_i)$.

If we use only unary predicates we will denote this logic $IMP(1)$.

2.3 Logic vs. Language theory

We will give in this section some results, in chronological order of their publications, on logical characterization of classes of languages, and the link between logical complexity and recognizability complexity.

Theorem 1 (Büchi [4], Elgot [9] and Lacroix [17]) *A language is regular if and only if it is definable in monadic second-order logic if and only if it is definable in IMP(1).*

Thomas improved the result of Büchi and Elgot to formulas of the form $\exists P\, F.O.$ where P is a monadic predicate.

The connection between trees and words was given by Mezei and Wright, for a comprehensive proof and a definition of regular tree languages see [11].

Theorem 2 (Mezei and Wright [20]) *A word language is context free if and only if the set of the derivation trees of its words is regular.*

And a new description of regular tree languages in the same vein as Büchi's result arises,

Theorem 3 (Doner, Thatcher and Wright [6,24]) *A tree language is regular if and only if it is definable in monadic second-order logic.*

First-order logic, the most natural sublogic of monadic second-order logic, on words was studied by McNaughton, for more details see [23].

Theorem 4 (McNaughton and Papert [19]) *A language is star free if and only if it is definable in first-order logic.*

Fagin studied the Existensial fragment of Second-order Logic and proved:

Theorem 5 (Fagin [10]) *Languages definable in existential second-order logic are exactly those computable in polynomial time by a non deterministic Turing machine.*

This result is true for all finite relational structures not only for word structures.

Theorem 6 (Thomas, Perrin and Pin [25]) *A star free set is of dot depth n if and only if it is definable in the boolean closure of Σ_n, Where Σ_n is the set of first-order formulas allowing n alternations of quantifiers (universal, existential).*

A recent result of Eiter and *al* [8], which says that every Existential Second-order prefix class either describes only regular languages or describes an NP-complete problem.

I suggest to curious readers to see the expository papers of Thomas [25], or Pin's one [22] for more details and results.

2.4 Matching vs. Context free languages

In this section we recall a result of Lautemann, Shwentick and Thérien for the description of Context Free Languages using the semantic quantifier of Matching.

Definition 7 *A binary relation M is called a matching if it satisfies the following conditions :*

(1) $\forall ij[(i,j) \in M \Rightarrow i < j]$.
(2) $\forall ij[(i,j) \in M \Rightarrow \forall k \neq i,j$
$((i,k), (k,i), (j,k), and (k,j)$ *are not in* $M)]$.
(3) $\forall ijkl[(i,j),(k,l) \in M \Rightarrow (i < k < j \rightarrow i < l < j)]$.

We will denote by $\psi(M)$, the conjunction of these three first-order items on $\tau \cup \{M\}$.

Let Match denote the class of matchings on word structures.

$\exists Match\, \phi$ means : there exists a relation $M \in Match$ such that $< S, M > \models \phi$.

In order to define any non regular language, one have to go beyond M.S.O. Logic. On the other hand, existential quantification over a simple binary relation express all context free languages and some $NP-$complete languages by the result of Eiter and *al* [8], their result says that any prefix class of Existential Second-order Logic either expresses only regular languages or expresses some $NP-$complete language.

Moreover they proved that NP-hardness is present with a sentence $\exists R\phi$ where R is a binary predicate and ϕ is first-order of the appropriate form.

Lautemann and *al* [18] choose a semantical approach for this purpose, in which they restrict the second-order quantifier to range over the class of matchings, *Match*. They define the class $\exists Match\, F.O.$ to consist of all those sets L of $\tau-$structures for which there is a first-order sentence ϕ *over* $\tau \cup \{M\}$ such that, For every $\tau-$structures S_w:
$S_w \in L$ if and only if, there is some matching M over S_w such that $< S_w, M > \models \phi$.

They proved the following:

Theorem 8 (Lautemann, Schwentick and Thérien [18]) *A language L is context free if and only if it is definable by a formula of the form $\exists Match\, \phi$ where $\phi \in F.O.$*

The result remains true also for $\phi \in M.S.O.$

For proving this result, the first step was to construct a first-order sentence over $\tau \cup \{M\}$ for each grammar G, which holds for a word structure S_w if and only if there is a G-derivation tree T of w, and there is an effective way to construct the matching from the tree, and vice versa.

For the other direction they combined results of Doner [6], Thatcher and Wright [24] and Mezei and Wright [20] to have the fact: *A language is context free if and only if the set of derivation trees of his words is a regular tree language.* (And this is independent of the grammar used.)

The last step was to construct trees from the matching, and prove that these trees satisfy some monadic second-order sentence. More details will be given in the proof of the theorem 10.

3 A new Chomsky and Schützenberger Normal form

In this section we reprove a stronger version of the Chomsky and Schützenberger theorem, by restricting the expression K, of theorem 8, to be only star free.

This result is also given in [1], but we reprove it by only logical arguments.

Theorem 9 (Chomsky and Schützenberger [5]) *A language L in Σ^* is context free if and only if $L = \psi(D_n \cap K)$ where D_n is the Dyck language on n "brackets", K a regular expression and ψ a monoïd homomorphism from $\Gamma \cup \overline{\Gamma}$ into Σ^*.*

We recall the Dyck language D_n on n brackets is the language generated by the grammar:

$G =< \{a_1, \ldots, a_n, \overline{a}_1, \ldots, \overline{a}_1\}, \{S\}, S, P >$ where P is the set of productions:

$S \to a_1 S \overline{a}_1 S | \ldots | a_n S \overline{a}_n S | \varepsilon$ where ε denotes the empty word.

If the $a_i's$ are assumed to be the opening brackets and $\overline{a}_i's$ the closing ones, D_n will be the set of well balanced brackets words.

Theorem 10 *A language L is context free if and only if $L = \psi(D_n \cap K)$ where D_n is the Dyck language on n "brackets", K a star-free expression and ψ a monoïd homomorphism from $\Gamma \cup \overline{\Gamma}$ into Σ^*.*

We recall the double Greibach normal form.

Lemma 11 *Every context free language is generated by a grammar $G =< N, \Sigma, S, P >$ which satisfies the following condition:*
all productions are of one of the forms:

(1) $S \to a$, $a \in \Sigma$ or
(2) $X \to aub$, $X \in N$, $a, b \in \Sigma$, and $u \in (\Sigma \cup N)^*$.

For a detailed proof we send the reader to the paper of Autebert and *al* [3].

Proof of the Theorem The way $L = \psi(D_n^* \cap K)$ for some star free expression K implies that L is a context free language derives obviously from the Chomsky and Schützenberger theorem because star free expressions are regular.

For the other way we will give some first-order conditions on a Dyck language to construct a set Z such that $L = \psi(Z)$. These conditions are intimately connected to the history of derivations.

Let L be a context free language. By the previous lemma we have a grammar in double Greibach normal form $G =< \Sigma, N, S, P >$ which derive it from S.

We enumerate first the non-terminal symbols, $X_0 = S, ..., X_N$.

After we label productions by ordered pairs $< i, j >$ where X_i is the left hand side non terminal of the production, and j enumerates injectively the

productions having X_i as left hand side non terminal. The elements of P are:

$P_{0,1} : X_0 \to w_{0,1}$

$\ldots$

$P_{0,i_0} : X_0 \to w_{0,i_0}$

$\ldots$

$P_{N,1} : X_N \to w_{N,1}$

$\ldots$

$P_{N,i_N} : X_N \to w_{N,i_N}$

and for each production we denote $c_{i,j}$ the total number of right hand side non terminals in the production $p_{i,j}$.

We construct now the set of brackets Γ. It is the set of tuples of integers $< a, b, c, d, e, f >$ where:

a, b are a production code such that $c_{a,b} \neq 0$ or $a = b = 0$.
c is 1 if $a = b = 0$, else $c = c_{a,b}$.
d is such that $1 \leq d \leq c$ and represents the range of the current non terminal in $p_{a,b}$, $1 \leq d \leq c$.
e, f are the next production code where e must be the code of the c^{th} non terminal in the right hand side of the production $p_{a,b}$ and $f \leq i_e$,or $e = 0$ if $a = b = 0$.

$\overline{\Gamma}$ will be the set of $\overline{< a, b, c, d, e, f >}$ for each element $< a, b, c, d, e, f > \in \Gamma$.

We give now the conditions on the words of the Dyck language on D_Γ to be in Z.

We will decide of the successor of each symbol in this word and give the range of the first and the last symbol.

(1) The first symbol in our word must be an opening bracket of a start configuration and the last one must close this bracket $\bigvee_{1 \leq i \leq i_0} (P_{<0,0,1,1,0,i>}(min) \wedge P_{\overline{<0,0,1,1,0,i>}}(max))$.
(2) $P_{<a,b,c,d,e,f>}(x)$ and $c_{e,f} = 0$ then we must close immediately our bracket $P_{\overline{<a,b,c,d,e,f>}}(x+1)$ because $p_{e,f}$ is a terminal production.
(3) $P_{<a,b,c,d,e,f>}(x)$ and $c_{e,f} > 0$ then we have $\bigvee_{e',f'} P_{<e,f,c_{e,f},1,e',f'>}(x+1)$ such that e' is the first non terminal in the right hand side of $p_{e,f}$.
(4) $P_{\overline{<a,b,c,d,e,f>}}(x)$ for some x and $c > d$ then we must have $\bigvee_{e',f'} P_{<a,b,c,d+1,e',f'>}(x+1)$ for some $f' < i'_e$ such that e' is the $d + 1^{st}$ non terminal in $p_{a,b}$.

(5) $P_{\overline{<a,b,c,d,e,f>}}(x)$ for some x and $c = d$ then we must have $\bigvee_{<a,b,c,d,e,f>} P_{\overline{<a,b,c,d,e,f>}}(x+1)$.

We are sure in item 5 to close the good type of parenthese because we are in a Dyck language.

Because of the finiteness of the set Γ these conditions are expressed by a first-order formula.

By McNaughton and Papert's theorem Z is a star free subset of D_Γ.

If $c_{a,b} \neq 0$ then the production $p_{a,b}$ have the form:

$p_{a,b} : X_a \to w_{(a,b,0)} X_{j_1} \dots w_{(a,b,c_{a,b}-1)} X_{j_{c_{a,b}}} w_{(a,b,c_{a,b})}$.

We now give the homorphism ϕ:

$\phi(< a, b, c, d, e, f >) = w_{(e,f,0)}$, and
$\phi(\overline{< a, b, c, d, e, f >}) = w_{(a,b,d)}$, and
$\phi(\overline{< 0, 0, 1, 1, 0, i >}) = \varepsilon$.

Where ε is the empty string.

By identifying the brackets to internal nodes of the spanning tree and the homomorphism images to leaves in the right place, we can trivially verify the eqality $L = \psi(Z)$. **Q.E.D**

Example Let's take the grammar

$$G =< \{a, b\}, \{S, Y, Z\}, S, P >$$

where P contains the following productions:

$S \to abba|aYabZba$

$Y \to aaYbaZbb|aZb$

$Z \to ab$

We first enumerate the non-terminals: $S = X_0,\ Y = X_1,\ and\ Z = X_2$. We can now enumerate productions:

$p_{0,1} : X_0 \to abba$

$p_{0,2} : X_0 \to aX_1abX_2ba$

$p_{1,1} : X_1 \to aaX_1baX_2bb$

$p_{1,2} : X_1 \to aX_2b$

$p_{2,1} : X_2 \to ab$

So we have:
$\Sigma = \{\langle 001101\rangle, \langle 001102\rangle, \langle 022111\rangle, \langle 022112\rangle, \langle 022221\rangle,$
$\langle 112111\rangle, \langle 112112\rangle, \langle 112221\rangle, \langle 121121\rangle\}$

Which we will denote later 1, 2, 3, 4, 5, 6, 7, 8, and 9.

The Dyck words must satisfy the formula

$$
\begin{aligned}
F \equiv\ & (((P_1(min) \wedge P_{\bar{1}}(max)) \vee (P_2(min) \wedge P_{\bar{2}}(max)))\wedge \\
& (P_1(x) \rightarrow P_{\bar{1}}(x+1))\wedge \\
& (P_2(x) \rightarrow (P_3(x+1) \vee P_4(x+1)))\wedge \\
& (P_3(x) \rightarrow (P_6(x+1) \vee P_7(x+1)))\wedge \\
& (P_4(x) \rightarrow P_9(x+1))\wedge \\
& (P_5(x) \rightarrow (P_{\bar{5}}(x+1))\wedge \\
& (P_6(x) \rightarrow (P_6(x+1) \vee P_7(x+1)))\wedge \\
& (P_7(x) \rightarrow P_9(x+1))\wedge \\
& (P_8(x) \rightarrow (P_{\bar{8}}(x+1))\wedge \\
& (P_9(x) \rightarrow (P_{\bar{9}}(x+1))\wedge \\
& (P_{\bar{1}}(x) \rightarrow x = max)\wedge \\
& (P_{\bar{2}}(x) \rightarrow x = max)\wedge \\
& (P_{\bar{3}}(x) \rightarrow (P_5(x+1))\wedge \\
& (P_{\bar{4}}(x) \rightarrow (P_5(x+1))\wedge \\
& (P_{\bar{5}}(x) \rightarrow \bigvee_{1 \leq i \leq 11} P_{\bar{i}}(x+1))\wedge \\
& (P_{\bar{6}}(x) \rightarrow (P_8(x+1))\wedge \\
& (P_{\bar{7}}(x) \rightarrow (P_8(x+1))\wedge \\
& (P_{\bar{8}}(x) \rightarrow \bigvee_{1 \leq i \leq 11} P_{\bar{i}}(x+1))\wedge \\
& (P_{\bar{9}}(x) \rightarrow \bigvee_{1 \leq i \leq 11} P_{\bar{i}}(x+1)).
\end{aligned}
$$

The homomorphism ϕ is defined by:

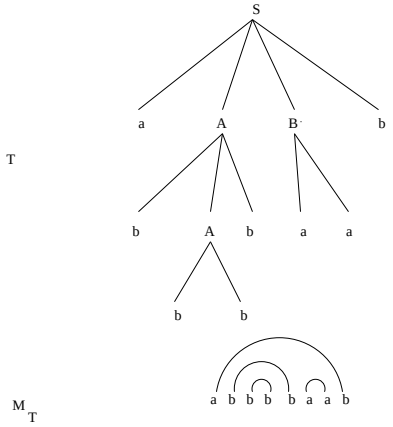

Fig. 2. Matching from derivation tree of w

$$\begin{aligned}
&\phi(1) = abba \quad , \quad \phi(2) = a \\
&\phi(3) = aa \quad , \quad \phi(4) = a \\
&\phi(5) = ab \quad , \quad \phi(6) = aa \\
&\phi(7) = a \quad , \quad \phi(8) = ab \\
&\phi(9) = ab \quad , \quad \phi(\overline{1}) = \varepsilon \\
&\phi(\overline{2}) = \varepsilon \quad , \quad \phi(\overline{3}) = ab \\
&\phi(\overline{4}) = ab \quad , \quad \phi(\overline{5}) = ba \\
&\phi(\overline{6}) = ba \quad , \quad \phi(\overline{7}) = ba \\
&\phi(\overline{8}) = bb \quad \textit{and } \phi(\overline{9}) = b
\end{aligned}$$

Let's take as example the word $w = aaabbababba$, we give in the figure below its derivation tree.

Then by extracting in a prefixed (first reach) way the opening brackets and at the same time in a postfixed (last reach) way the closing ones we get the word $w_D \in D^*$

$$w_D = 2\ 4\ 9\ \overline{9}\ \overline{4}\ 5\ \overline{5}\ \overline{2}$$

We then remark that $\phi(w_D) = w$.

The construction is closely connected to the derivation tree this is why we are sure of the equivalence.

4 A logic for unambiguous Context free languages

We give now a logic for unambiguous Context free languages. The main idea is that unambiguity needs unicity of existence.

Let the Logic IMP_2 be the sublogic of IMP where we use the implicit definition of only one predicate, which is binary.

$\exists Match\ F.O. \cap IMP_2$ will be the set of formulas of $\exists! Match\ F.O.$ where only one matching M satisfy the first-order formula.

Theorem 12 *A language is unambiguous context free if and only if it is definable in*
$\exists Match\ F.O. \cap IMP_2$.

One of the keysteps in the proof is:

Lemma 13 ([15]) *Every Unambiguous Context Free Language has an Unambiguous Context Free Grammar* $G =< N, \Sigma, S, P >$ *where all productions are of one of the forms:*

(1) $S \to a,\ a \in \Sigma$ *or*
(2) $X \to aub,\ X \in N,\ a, b \in \Sigma,$ *and* $u \in (\Sigma \cup N)^*$.

This lemma uses only the fact that the classical construction preserves unambiguity.

Proof of the theorem. The proof of this theorem is intimately connected to the one of Lautemann and *al* for giving a logic for context free languages, we only have to prove that unambiguity of the language implies uniqueness of the matching and vice versa.

By the previous lemma we have an unambiguous grammar in the normal form used in [18]. The processes:

(1) Eliminate all productions of the form $X \to \alpha$ for some $\alpha \in \Sigma$ by introducing a new production $Y \to u\alpha v$, for every production $Y \to uXv \in P$.
(2) Enumerate all non-terminals, $X_1 = S, ..., X_N$. Starting with $i = 2$ do the following for every i, as long as there is non-terminal production p= $X_i \to v$ whose pattern also appears as the pattern of a production with left-hand side $X_j,\ j < i$ replace p by all productions which can be obtained from it by substituting one of the non-terminals in v in all possible ways.

terminates and preserves unambiguity.

Then for every Unambiguous Context Free Language we have a Unambiguous Context Free Grammar in double Greibach Normal Form and any two non terminal productions have the same pattern iff they have the same left hand non terminal.

Let T be a derivation tree of w, the matching corresponding to T is M_T defined by:$(i,j) \in M_T$ if and only if i corresponds to the leftmost and j to the rightmost child of the same internal node of T.

We construct now the formula ψ_G over $< \Sigma, <, M >$ which holds for a string w with matching M iff there is a G derivation tree T for w such that $M = M_T$. It follows that there is a matching M on w with $< w, M > \models \psi_G$ iff w can be derived in G.

Let $(i,j) \in M_T$ an arch, the pattern of (i,j) is the string composed of their "brothers" written from left to right where internal nodes are replaced by|.

To be the matching constructed from a G derivation tree, the pattern must correspond to the pattern of a production in G.

For $p \equiv X_0 \to \alpha v_0 X_1 v_1 \dots X_s v_s \beta$ *where* $\alpha, \beta \in \Sigma,\ v_i \in \Sigma^*$, and $X_i \in N$ we construct a first-order formula: $\pi_p(x,y) =$
$P_\alpha(x) \wedge P_\beta(y) \wedge \exists x_1 y_1 \dots x_s y_s [(x < x_1 < y_1 < \dots < x_s < y_s < y)$
$\wedge(\psi_{v_0}(x,x_1) \wedge \psi_{v_1}(y_1,x_2) \wedge \dots \wedge \psi_{v_s}(y_s,y))$
$\wedge(M(x_1,y_1) \wedge \dots \wedge M(x_s,y_s))]$,

where $\psi_v(i,j)$ is the first-order formula $\bigwedge_{n=i}^{n=j} P_{w_{n-i}}(n)$ if $v = w_0 \dots w_r$, Which characterize the pattern between two positions x *and* y to correspond to some production p of G.

Let $\pi_X(x,y)$, for $x \in N$ be the disjunction of all the $\pi_p(x,y)$ whenever p has X as lefthand side.

We can write now the formula $\overline{\pi}_p(x,y) =$
$P_\alpha(x) \wedge P_\beta(y) \wedge \exists x_1 y_1 \dots x_s y_s [(x < x_1 < y_1 < \dots < x_s < y_s < y)$
$\wedge(\psi_{v_0}(x,x_1) \wedge \psi_{v_1}(y_1,x_2) \wedge \dots \wedge \psi_{v_s}(y_s,y))$
$\wedge(M(x_1,y_1) \wedge \dots \wedge M(x_s,y_s)) \wedge (\pi_{X_1}(x_1,y_1) \wedge \dots \wedge \pi_{X_s}(x_s,y_s)]$,

which restricts the pattern of the matching between x and y to correspond to the matching of a production having the appropriate non terminal as left hand side.

The formula ψ_G is then:

$$\bigvee_{S \to u \in P} (\psi_u(min,max)) \vee [\forall x \forall y (M(x,y) \to$$

$$\bigvee_{p \in P} \overline{\pi}_p(x,y)) \wedge (M(min,max) \wedge \pi_S(min,max))]$$

Since every production is uniquely determined by its pattern, this formula is appropriate for our aim.

For the other direction we remark that the construction of the tree is intimately connected to the matching. Then the uniqueness of the matching implies the uniqueness of the derivation tree for each word, this gives us, by definition, the unambiguity of the language. **Q.E.D.**

Note. As the property "*a binary relation is a matching*" can be expressed in first-order logic, we can construct a syntactic sublogic of IMP_2 which captures Unambiguous Context Free Languages. This can be done by the set of formulas $\phi \wedge \psi$, where ϕ defines a binary relation implicitely and ψ test if this relation is a matching. We gave in this paper the semantic definition rather than the syntactic one because of the simplicity of this notion in this case.

Corollary 14 *IMP_2 is undecidable.*

This is a simple consequence of undecidability of unambiguity.

5 Conclusion

We reproved in this paper an algebraic characterization of Context Free Languages by means of Dyck languages, using a result of McNaughton and Papert [19] for the logical description of star free expressions and the Double Greibach Normal Form. We could get a cleaner proof by using the Double Quadratic Greibach Normal Form.

Unambiguity of Context Free languages is relevant for compiling theory because if a program has two different derivations we can have different results for the same input.

This motivates me to try to describe Unambiguous Context Free Languages by logical means. But the undecidability of Unambiguity compels us to use an undecidable logic, which is IMP. For a proof of its undecidability see [16].

The undecidability of IMP is in the sense commonly understood. That is the set of IMP-formulas is *co-recursively enumerable complete*. But the undecidability of IMP_2 is in the sense that we can't decide if a given binary predicate, which is a matching, can be whether or not implicitly defined by a first-order formula.

The result of Eiter and *al* [8] discouraged me to look for some more syntactic logic for all classes between $N.P.$ and regular sets.

The result makes the link between two undecidable problems, a logical one and a language theoretic one.

The question which naturally arises after our result is:

Is there a logic for deterministic Context Free Languages?

References

[1] AUTEBERT, Jean Michel. *Théorie des langages et des automates*, Masson 1994.

[2] AUTEBERT, Jean Michel. *Personnal communication*, 1998.

[3] AUTEBERT, Jean-Michel, BERSTEL, Jean, et BOASSON, Luc. *Context-free languages and pushdown automata.* In : Handbook of formal languages. Springer Berlin Heidelberg, 1997. p. 111-174.

[4] BÜCHI, J. Richard. *Weak Second-Order Arithmetic and Finite Automata.* Mathematical Logic Quarterly, 1960, vol. 6, no 1-6, p. 66-92.

[5] SCHUTZENBERGER, M. P. *THE ALGEBRAIC THEORY OF CONTEXT-FREE LANGUAGES* N. CHOMSKY.* Computer programming and formal systems, 1963, vol. 28, p. 118.

[6] DONER, John. *Tree acceptors and some of their applications.* Journal of Computer and System Sciences, 1970, vol. 4, no 5, p. 406-451.

[7] EBBINGHAUS, Heinz-Dieter et FLUM, Jörg. *Finite model theory.* Springer Science & Business Media, 2005.

[8] EITER, Thomas, GOTTLOB, Georg, et GUREVICH, Yuri. *Existential second-order logic over strings.* Journal of the ACM (JACM), 2000, vol. 47, no 1, p. 77-131.

[9] ELGOT, Calvin C. *Decision problems of finite automata design and related arithmetics.* Transactions of the American Mathematical Society, 1961, vol. 98, no 1, p. 21-51.

[10] FAGIN, Ronald. *Generalized first-order spectra and polynomial time recognizable sets*, in RM Karp editor Complexity of computation, SIAM-AMS Proceedings 1974.

[11] GCSEG, Ferenc et STEINBY, Magnus. *Tree languages.* In : Handbook of formal languages. Springer Berlin Heidelberg, 1997. p. 1-68.

[12] HACHAÏCHI, Yassine. *A descriptive complexity approach to the linear hierarchy.* Theoretical computer science, 2003, vol. 304, no 1, p. 421-429.

[13] HACHAÏCHI, Yassine. *Fragments of monadic second-order logics over word structures.* Electronic Notes in Theoretical Computer Science, 2005, vol. 123, p. 111-123.

[14] HARRISON, Michael A. *Introduction to formal language theory.* Addison-Wesley Longman Publishing Co., Inc., 1978.

[15] HOTZ, Guenter. *Normal-form transformations of context-free grammars.* Acta Cybernetica, 1980, vol. 4, p. 65-84.

[16] KOLAITIS, Phokion G. *Implicit definability on finite structures and unambiguous computations.* In : Logic in Computer Science, 1990. LICS'90, Proceedings., Fifth Annual IEEE Symposium on e. IEEE, 1990. p. 168-180.

[17] LACROIX, Zoé. *Bases de données des relations implicites aux relations contraintes*, Ph.D. Université de Paris Sud 1996.

[18] LAUTEMANN, Clemens, SCHWENTICK, Thomas, et THÉRIEN, Denis. *Logics for context-free languages.* In : Computer science logic. Springer Berlin Heidelberg, 1994. p. 205-216.

[19] MCNAUGHTON, Robert et PAPERT, Seymour A. *Counter-Free Automata* (MIT research monograph no. 65). The MIT Press, 1971.

[20] MEZEI, Jorge et WRIGHT, Jesse B. *Algebraic automata and context-free sets.* Information and control, 1967, vol. 11, no 1, p. 3-29.

[21] PAPADIMITRIOU, Christos H. *Computational complexity.* John Wiley and Sons Ltd., 2003.

[22] PIN, Jean-Eric. *Logic, semigroups and automata on words* . Annals of Mathematics and Artificial Intelligence, 1996, vol. 16, no 1, p. 343-384.

[23] STRAUBING, Howard. *Finite automata, formal logic, and circuit complexity.* Springer Science & Business Media, 2012.

[24] THATCHER, James W. et WRIGHT, Jesse B.. *Generalized finite automata theory with an application to a decision problem of second-order logic.* Mathematical systems theory, 1968, vol. 2, no 1, p. 57-81.

[25] THOMAS, Wolfgang. *Languages, automata, and logic,* Handbook of formal languages, vol. 3: beyond words. 1997.