TUCS

Alexander Okhotin

Inverse homomorphic characterizations of conjunctive and Boolean grammars

Turku Centre *for* Computer Science

TUCS Technical Report
No 1080, June 2013

Inverse homomorphic characterizations of conjunctive and Boolean grammars

Alexander Okhotin
Department of Mathematics and Statistics, University of Turku, *and*
Turku Centre for Computer Science
Turku FI-20014, Finland
`alexander.okhotin@utu.fi`

TUCS Technical Report
No 1080, June 2013

Abstract

A well-known theorem by Greibach ("The hardest context-free language", *SIAM J. Comp.*, 1973) states that there exists such a "hardest" context-free language L_0, that every context-free language over any alphabet is representable as an inverse homomorphic image $h^{-1}(L_0)$, for a suitable homomorphism h. This paper establishes similar characterizations for conjunctive grammars and for Boolean grammars, that is, for context-free grammars extended with conjunction and negation operators.

Keywords: Conjunctive grammars, Boolean grammars, homomorphisms, hardest language.

TUCS Laboratory
FUNDIM, Fundamentals of Computing and Discrete Mathematics

1 Introduction

In 1973, Greibach [4] constructed a context-free language L_0 over an alphabet Σ_0, with the following property: for every context-free language L over any alphabet Σ there exists such a homomorphism $h \colon \Sigma \to \Sigma_0^*$, that a string $w \in \Sigma^*$ is in L if and only if its image $h(w)$ belongs to L_0. In other words, L is represented as an inverse homomorphic image $h^{-1}(L_0)$. This language L_0 is known as *the hardest context-free language*, for the reason that a decision procedure for L_0 can be used to recognize every context-free language using the same amount of resources, up to a constant factor.

The purpose of this paper is to establish a similar result for the families of *conjunctive grammars* [12] and *Boolean grammars* [15, 10]. These are variants of ordinary context-free grammars, which allow expressing conjunction and negation of syntactic conditions in the rules. Consider that a rule $A \to BC$ in an ordinary grammar states that if a string w is representable as BC—that is, as $w = uv$, where u has the property B and v has the property C—then w has the property A. In a conjunctive grammar, one can define a rule of the form $A \to BC \,\&\, DE$, which asserts that every string w representable *both* as BC (with $w = uv$) *and at the same time* as DE (with $w = xy$) therefore has the property A. A Boolean grammar further allows such rules as $A \to BC \,\&\, \neg DE$, which expresses all strings representable as BC, but not representable as DE. The importance of conjunctive and Boolean grammars is justified by two facts: on the one hand, they allow specifying a useful logical operation within natural inductive definitions of syntax, and on the other hand, they can be parsed by generally the same parsing algorithms as for ordinary context-free grammars [18, 17, 19]. Other results on these grammars include automaton characterizations [1, 14], normal form theorems [21], closure properties [11], an extensive study of the case of a one-symbol alphabet [5, 6, 8, 9, 23], several results on the parsing complexity [7, 20, 22], and the first developments on the stochastic case [2, 24].

Turning to the prospects of proving an analogue of Greibach's theorem for conjunctive grammars, the first thing to note is that Greibach's proof of her theorem for ordinary context-free grammars relies on using a grammar in the Greibach normal form [3], in which every rule is of the form

$$A \to a\alpha,$$

where a is a symbol of the alphabet. The construction transcribes all such rules in the image $h(a)$ of the symbol a, and the grammar for the language L_0 matches those transcriptions to the encodings of the symbols listed in α.

Unfortunately, no analogue of Greibach's normal form is known for conjunctive grammars. To be precise, the form itself can be generalized, so that all rules in a grammar are

$$A \to a\alpha_1 \,\&\, \ldots \,\&\, a\alpha_m,$$

but it is not known whether every grammar can be transformed to this form. As a work-around, this paper assumes a weaker form, upon which all the subsequent constructions are based. This is the *odd normal form* for conjunctive grammars, defined by Okhotin and Reitwießner [21], in which all rules are either of the form

$$A \to B_1 a_1 C_1 \,\&\, \ldots \,\&\, B_m a_m C_m,$$

or of the form $A \to a$ (not counting special rules of the form $S \to aA$ allowed for the initial symbol of the grammar).

Using this normal form instead of the Greibach normal form requires developing a new construction, which is quite different from the construction used by Greibach [3] for ordinary grammars. For a conjunctive grammar G, Section 3 defines a homomorphism h, in which the image $h(a)$ of a symbol a lists all conjuncts of the form BaC used in the grammar, along with all rules for B and C, all appropriately encoded. The base language L_0, defined by a conjunctive grammar, matches these encodings to each other, simulating the logic contained in the original grammar G.

In the last Section 4, the construction is adapted to prove the same kind of result for Boolean grammars.

The appendix reports on machine calculations, which were carried out in order to test the constructions. The method was found to work correctly on three different grammars.

2 Conjunctive grammars

Definition 1. *A conjunctive grammar is a quadruple $G = (\Sigma, N, R, S)$, in which:*

- *Σ is the* alphabet *of the language being defined;*
- *N is a finite set of symbols for the syntactic categories defined in the grammar, each representing a property that a string in Σ^* may have or not have;*
- *R is a finite set of* rules*, each of the form*

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_m, \tag{1}$$

where $A \in N$, $m \geqslant 1$ and $\alpha_1, \ldots, \alpha_m \in (\Sigma \cup N)^*$;

- *$S \in N$ is a symbol representing the property of being a syntactically well-formed sentence of the language.*

Each concatenation α_i in every rule (1) is called a *conjunct*. If a grammar has a unique conjunct in every rule, it is an ordinary context-free grammar.

Each rule (1) means that any string representable as each concatenation α_i therefore has the property A. This understanding can be formalized in several equivalent ways: by term rewriting [12], by parse trees [12], by logical deduction [16] and by language equations [13].

Consider one of these definitions, which extends Chomsky's definition of ordinary context-free grammars by string rewriting, using terms instead of strings.

Definition 2 ([12]). *Let $G = (\Sigma, N, R, S)$ be a conjunctive grammar, and consider terms over concatenation and conjunction, with symbols from $\Sigma \cup N$ and the empty string ε as atomic terms. The relation $\Longrightarrow$ of one-step rewriting on such is defined as follows:*

- *Using a rule $A \to \alpha_1 \& \ldots \& \alpha_m \in R$, any atomic subterm A of any term can be rewritten by the subterm $(\alpha_1 \& \ldots \& \alpha_m)$:*

$$\ldots A \ldots \Longrightarrow \ldots (\alpha_1 \& \ldots \& \alpha_m) \ldots$$

- *A conjunction of several identical strings in Σ^* can be rewritten by one such string: for every $w \in \Sigma^*$,*

$$\ldots (w \& \ldots \& w) \ldots \Longrightarrow \ldots w \ldots$$

The language generated by a term φ is is the set of all strings over Σ obtained from it in a finite number of rewriting steps:

$$L_G(\varphi) = \{\, w \mid w \in \Sigma^*,\ \varphi \Longrightarrow^* w \,\}.$$

The language generated by the grammar is the language generated by its initial symbol:

$$L(G) = L_G(S) = \{\, w \mid w \in \Sigma^*,\ S \Longrightarrow^* w \,\}.$$

The following normal form for conjunctive grammars is known as the *odd normal form*, because all strings generated by all symbols (except maybe the initial symbol), are of odd length.

Theorem A (Okhotin, Reitwießner [21]). *For every conjunctive grammar there exists and can be effectively constructed a conjunctive grammar $G = (\Sigma, N, R, S)$ generating the same language, which is in the* ***odd normal form****, that is, with all rules of the form*

$$\begin{array}{ll} A \to B_1 a_1 C_1 \& \ldots \& B_m a_m C_m & (m \geqslant 1,\ B_i, C_i \in N,\ a_i \in \Sigma) \\ A \to a & (a \in \Sigma) \end{array}$$

If S is never used in the right-hand sides of any rules, then the following two types of rules are also allowed:

$$\begin{aligned} &S \to aA && (a \in \Sigma,\ A \in N) \\ &S \to \varepsilon && \end{aligned}$$

3 Representation for conjunctive grammars

Theorem 1. *There exists such a conjunctive language L_0 over the alphabet $\Sigma_0 = \{a, b, c, d, \#\}$, that for every conjunctive language L over any alphabet Σ there is such a homomorphism $h\colon \Sigma \to \Sigma_0^*$, that $L = h^{-1}(L_0)$ if $\varepsilon \notin L$ and $L = h^{-1}(L_0 \cup \{\varepsilon\})$ if $\varepsilon \in L$.*

Let $G = (\Sigma, N, R, X)$ be any conjunctive grammar, in which every conjunct is of the form YsZ, Ys, sZ or s, where $Y, Z \in N$ and $s \in \Sigma$; in particular, every grammar in the odd normal form satisfies this condition. Let $\mathcal{C} = \{\alpha_1, \alpha_2, \ldots, \alpha_{|\mathcal{C}|}\}$, with $\alpha_i \in N\Sigma N \cup \Sigma N \cup N\Sigma \cup \Sigma$ be the set of all conjuncts used in the grammar, so that each rule is of the form

$$A \to \alpha_{i_1} \mathbin{\&} \ldots \mathbin{\&} \alpha_{i_m} \qquad (m \geqslant 1,\ i_1, \ldots, i_m \in \{1, \ldots, |\mathcal{C}|\}). \tag{2}$$

The symbols in the alphabet $\Sigma_0 = \{a, b, c, d, \#\}$ have the following meaning.

- The symbol a is used to represent the reference to each conjunct α_i as a^i.
- The symbol b is used to mark the definition of a conjunct $\alpha_i = XsY$ by b^i.
- The symbol c represents conjunction in the right-hand side of any rule. Each rule (2) has the following symmetric *left* and *right* representations:

$$\begin{aligned} \lambda(A \to \alpha_{i_1} \mathbin{\&} \ldots \mathbin{\&} \alpha_{i_m}) &= ca^{i_m} \ldots ca^{i_1}, \\ \rho(A \to \alpha_{i_1} \mathbin{\&} \ldots \mathbin{\&} \alpha_{i_m}) &= a^{i_1}c \ldots a^{i_m}c, \end{aligned}$$

- The symbol d is used to separate the definitions of any conjuncts with the same symbol s in the middle, so that each conjunct $\alpha_k \in \mathcal{C}$ is represented as follows:

$$\sigma(\alpha_k) = \begin{cases} \prod_{r \text{ is a rule for } X} \prod_{r' \text{ is a rule for } Y} \lambda(r) b^k \rho(r') d, & \text{if } \alpha_k = XsY, \\ \prod_{r \text{ is a rule for } X} \lambda(r) b^k d, & \text{if } \alpha_k = Xs, \\ \prod_{r' \text{ is a rule for } Y} b^k \rho(r') d, & \text{if } \alpha_k = sY, \\ b^k d, & \text{if } \alpha_k = s. \end{cases}$$

- The image $h(s)$ of any symbol $s \in \Sigma$ is concluded with the separator symbol $\# \in \Sigma_0$.

Define the image of $s \in \Sigma$ under h as

$$h_G(s) = \Bigg(\prod_{r \text{ is a rule for } S} \rho(r)d \Bigg) \cdot d \cdot \Bigg(\prod_{XsY \in \mathcal{C}} \sigma(XsY) \Bigg) \cdot \#.$$

Example 1. *Let $\Sigma = \{s, t\}$ and consider a grammar $G = (\Sigma, \{X, Y, Z\}, R, X)$ with the rules*

$$\begin{aligned} &X \to tY \\ &Y \to YsZ \,\&\, ZsZ \mid t \\ &Z \to t \end{aligned}$$

Let its conjuncts be numbered as $\alpha_1 = tY$, $\alpha_2 = YsZ$, $\alpha_3 = ZsZ$ and $\alpha_4 = t$. Then

$$\begin{aligned} h_G(s) &= acddca^2ca^3b^2a^4cdca^4b^2a^4cdca^4b^3a^4cd\# \qquad \textit{and} \\ h_G(t) &= acddba^4cdba^2ca^3cdb^4d\#, \end{aligned}$$

and accordingly the string $ttst \in L$ has the following image:

$$\begin{aligned} h_G(ttst) = {} & acddba^4cdba^2ca^3cdb^4d\#\ acddba^4cdba^2ca^3cdb^4d\# \\ & acddca^2ca^3b^2a^4cdca^4b^2a^4cdca^4b^3a^4cd\#\ acddba^4cdba^2ca^3cdb^4d\#. \end{aligned}$$

Figure 1 illustrates, how the parse tree of the string $ttst$ according to the grammar G can be reconstructed by following the pointers inside this image.

Given the image $h_G(ttst)$, consider the task of recognizing whether its pre-image $ttst$ is in $L(G)$. The recognition begins by looking at the image $h_G(t)$ of the first symbol and choosing any of the encoded rules for the initial symbol X, which are listed in the beginning of the image. There is only one such rule, $X \to \alpha_1$, where $\alpha_1 = tY$, and this rule is represented by the first symbols acd of the image. Here a is a reference to the conjunct α_1, the next symbol c marks the end of the conjunct's description, and the final d indicates that there are no further conjuncts in this rule.

The code a requests the conjunct $\alpha_1 = tY$, and should accordingly be matched to any code b representing its definition. Two such definitions are located in the image of every symbol t: one assuming further expansion of Y by the rule $Y \to t$, and the other assuming that Y is expanded by the rule $Y \to YsZ \,\&\, ZsZ$. Following the parse tree in Figure 1, in this case, the symbol t in tY should be the first symbol of the string $ttst$, and one should use the expansion of Y by the rule $Y \to YsZ \,\&\, ZsZ$. Hence, the recognition proceeds by matching this code a to the substring ba^2ca^3c located within the

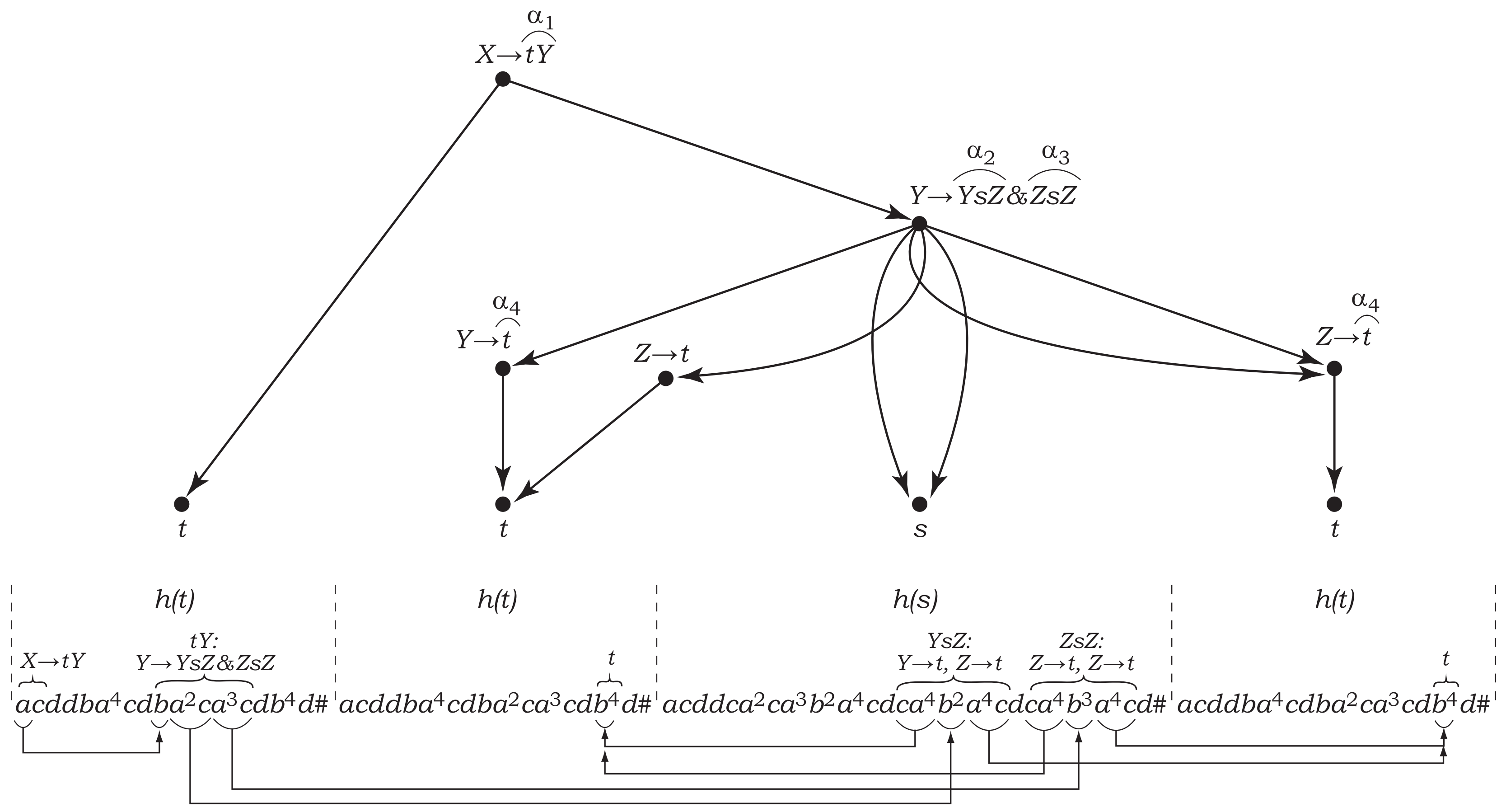
α1
X→tY
α2 α3
Y→YsZ&ZsZ
α4
Y→t
Z→t
α4
Z→t
t
t
s
t
h(t)
h(t)
h(s)
h(t)
X→tY
tY:
Y→YsZ&ZsZ
t
YsZ:
Y→t, Z→t
ZsZ:
Z→t, Z→t
t
acddba⁴cdba²ca³cdb⁴d# acddba⁴cdba²ca³cdb⁴d# acddca²ca³b²a⁴cdca⁴b²a⁴cdca⁴b³a⁴cd# acddba⁴cdba²ca³cdb⁴d#

Figure 1: How a parse tree of the string $w = ttst$ is reconstructed by analyzing its image $h(w)$.

image $h_G(t)$ of the first symbol t, as demonstrated at the bottom of Figure 1 by the leftmost arrow.

The substring ba^2ca^3c points to two conjuncts, $\alpha_2 = YsZ$ and $\alpha_3 = ZsZ$, which are to be checked independently. Consider the conjunct $\alpha_2 = YsZ$, for which the image $h_G(s)$ of some symbol s should contain the corresponding code b^2, surrounded by encodings of a rule for Y and a rule for Z. The pre-image of the input string in question contains a unique symbol s, and one should use the expansions of Y and Z by the rules $Y \to \alpha_4$ and $Z \to \alpha_4$, where $\alpha_4 = t$. This definition is encoded in a substring $ca^4b^2a^4c$, which points to a conjunct α_4 located to the left of the current symbol s (ca^4), and to another conjunct α_4 located to the right (a^4c). Finally, the encoding ca^4 of the conjunct on the left is matched to the string b^4 in the image $h_G(t)$ to the left, while a^4c is matched to the similar b^4 in the other image $h_G(t)$ to the right. These connections, along with the similar parse for the conjunct α_3, are illustrated in Figure 1.

The below conjunctive grammar implements this recognition.

$$\begin{aligned}
S &\to BdS \mid \overrightarrow{F_0}\overrightarrow{E} \,\&\, Ac\overrightarrow{E_0} \\
A &\to aA \mid a \\
B &\to aB \mid cB \mid a \mid c \\
C &\to aC \mid bC \mid cC \mid dC \mid \varepsilon \\
D &\to C\#D \mid \varepsilon \\
\overrightarrow{E} &\to \overrightarrow{F}\overrightarrow{E} \,\&\, Ac\overrightarrow{E_+} \mid dC\# \\
\overrightarrow{E_+} &\to \overrightarrow{F}\overrightarrow{E} \,\&\, Ac\overrightarrow{E_+} \mid dC\#D \\
\overrightarrow{F} &\to a\overrightarrow{F}b \mid acC\#\overleftarrow{E}b \\
\overleftarrow{E} &\to \overleftarrow{E}\overleftarrow{F} \,\&\, \overleftarrow{E_+}cA \mid Cd \\
\overleftarrow{E_+} &\to \overleftarrow{E}\overleftarrow{F} \,\&\, \overleftarrow{E_+}cA \mid DCd \\
\overleftarrow{F} &\to b\overleftarrow{F}a \mid b\overrightarrow{E}Cca \\
\overrightarrow{E_0} &\to \overrightarrow{F_0}\overrightarrow{E} \,\&\, Ac\overrightarrow{E_0} \mid dC\#D \\
\overrightarrow{F_0} &\to a\overrightarrow{F_0}b \mid acCdb \mid ac\overleftarrow{E}b
\end{aligned}$$

The main idea of this grammar is conveyed in the rules for the symbol $\overrightarrow{E}$. This symbol defines all strings of the form $x\#h_G(w)$, where $x\#$ is a suffix of the image $h_G(t)$ of some symbol t, while the rest of the string is the image of some string w. The suffix x begins with a list of conjuncts, and the purpose of $\overrightarrow{E}$ is to ensure that each of these conjuncts generates the string w.

$$\underbrace{a^{i_1}c \ldots a^{i_m}cdx\#}_{\text{suffix of } h_G(t)}\underbrace{\ldots}_{h_G(w)}, \qquad \text{where } w \in L_G(\alpha_{i_1}) \cap \ldots \cap L_G(\alpha_{i_m}).$$

The symbol $\overrightarrow{E}$ defines all such strings with $m \geqslant 1$, and if the sequence of conjuncts is empty ($m = 0$), then it ensures that $w = \varepsilon$. Its variant $\overrightarrow{E_+}$ does the same for all $m \geqslant 1$, but interprets an empty sequence of conjuncts as a sign that there is nothing to check. Either symbol invokes another symbol $\overrightarrow{F}$ to process the first conjunct, and at the same time refers to the symbol $\overrightarrow{E_+}$ in order to match the rest of the conjuncts by the same rules.

The symbol $\overrightarrow{F}$ first compares the conjunct code a^{i_1} to the corresponding code b^{i_1} for this conjunct's definition. This code is contained in the image of one of the symbols of w; accordingly, let $w = usv$, where s is the symbol in question. Then the conjunct α_{i_1} is of the form YsZ, and the code b^{i_1} is surrounded by the encoding of a rule for Y to the left, and the encoding of a rule for Z to the right.

$$\underbrace{a^{i_1}cx'\#}_{\text{suffix of } h_G(t)}\underbrace{\ldots}_{h_G(u)},\underbrace{\ldots yb^{i_1}z\ldots\#}_{h(s)}\underbrace{\ldots}_{h_G(v)},$$

The second rule for the symbol $\overrightarrow{F}$ refers to the symbol $\overleftarrow{E}$ (symmetric to $\overrightarrow{E}$) to specify the form of the string $h_G(u)\ldots y$, and thus ensure that the substring u is generated by Y in the original grammar. At the same time, when $\overrightarrow{F}$ was originally invoked in a rule for $\overrightarrow{E}$, this rule defined a concatenation $\overrightarrow{F}\overrightarrow{E}$, and now the latter symbol $\overrightarrow{E}$ specifies the form of $z\ldots\#h(v)$, ensuring that Z generates v.

The rules for the initial symbol S choose one of the rules for the initial symbol of G, and use a variant of $\overrightarrow{E}$, denoted by $\overrightarrow{E_0}$, to check that this rule generates the pre-image of the entire string.

Let L_0 be the language defined by the above grammar.

Lemma 1. *Let $G = (\Sigma, N, R, X)$ be a conjunctive grammar with all conjuncts of the form YsZ, Ys, sZ or s, where $Y, Z \in N$ and $s \in \Sigma$. Then, a string $w \in \Sigma^+$ is in $L(G)$ if and only if $h_G(w) \in L_0$.*

This preliminary version does not yet include a proper proof of this result, only a demagogic explanation of how it is supposed to work. To be sure that the grammar actually works as intended, the appendix reports on several tests.

4 Adapting to Boolean grammars

Boolean grammars further extend conjunctive grammars with a negation operator.

Definition 3. *A Boolean grammar is a quadruple $G = (\Sigma, N, R, S)$, where*

- *Σ is the alphabet;*

- *N is the set of symbols representing syntactic categories;*
- *R is a finite set of rules of the form*
$$A \to \alpha_1 \,\&\, \dots \,\&\, \alpha_m \,\&\, \neg\beta_1 \,\&\, \dots \,\&\, \neg\beta_n \tag{3}$$
with $A \in N$, $m, n \geqslant 0$, $m + n \geqslant 1$ and $\alpha_i, \beta_j \in (\Sigma \cup N)^*$;
- *$S \in N$ is the initial symbol.*

A rule (3) is meant to state that every string is representable in the form α_1, ..., α_m, but not representable in the form β_1, ..., β_n, therefore has the property A. This intuitive definition is formalized by using *language equations*, that is, by representing a grammar as a system of equations with formal languages as unknowns, and using a solution of this system as the language defined by the grammar.

Definition 4 (Okhotin [15])**.** *Let $G = (\Sigma, N, R, S)$ be a Boolean grammar, and consider the following system of equations in which every symbol $A \in N$ is an unknown language over Σ.*
$$A = \bigcup_{\substack{A \to \alpha_1 \,\&\dots\&\, \alpha_m \,\& \\ \&\, \neg\beta_1 \,\&\dots\&\, \neg\beta_n \in R}} \left[\bigcap_{i=1}^{m} \alpha_i \;\cap\; \bigcap_{j=1}^{n} \overline{\beta_j} \right] \tag{4}$$
Each symbol $B \in N$ used in the right-hand side of any equation is a reference to a variable, and each symbol $a \in \Sigma$ represents a constant language $\{a\}$.

Assume that for every integer $\ell \geqslant 0$ there exists a unique vector of languages $(\dots, L_A, \dots)_{A \in N}$ with $L_A \subseteq \Sigma^{\leqslant \ell}$, such that a substitution of L_A for A, for each $A \in N$, turns every equation (4) into an equality modulo intersection with $\Sigma^{\leqslant \ell}$. Then the system is said to have a strongly unique solution*, and, for every $A \in N$, the language $L_G(A)$ is defined as L_A from the unique solution of this system. The language generated by the grammar is $L(G) = L_G(S)$.*

The odd normal form can be extended to Boolean grammars.

Theorem 2. *For every Boolean grammar there exists and can be effectively constructed a Boolean grammar generating the same language, which is in the* ***odd normal form****, that is, with all rules of the form*

$$A \to B_1 a_1 C_1 \,\&\, \dots \,\&\, B_m a_m C_m \,\&\, \neg E_1 d_1 F_1 \,\&\, \dots \,\&\, \neg E_n d_n F_n$$
$$(m \geqslant 1,\ n \geqslant 0,\ B_i, C_i, E_j, F_j \in N,\ a_i, d_j \in \Sigma)$$
$$A \to a \qquad (a \in \Sigma)$$

If S is never used in the right-hand sides of any rules, then the following two types of rules are also allowed:

$$S \to aA \qquad (a \in \Sigma,\ A \in N)$$
$$S \to \varepsilon$$

Sketch of a proof. The transformation is carried out by the same method as in the case of conjunctive grammars [21].

Assume that the original Boolean grammar $G = (\Sigma, N, R, S)$ is in the *binary normal form* [15], that is, all rules in it are of the form

$$A \to B_1C_1 \& \ldots \& B_mC_m \& \neg D_1E_1 \& \ldots \& \neg D_nE_n \& \neg\varepsilon \qquad (m \geqslant 1,\ n \geqslant 0,\ B_i, C_i, D_j, E_j \in N), \tag{5a}$$

$$A \to a \qquad (a \in \Sigma). \tag{5b}$$

Construct another Boolean grammar $G' = (\Sigma, N' \cup \{S'\}, R', S')$, in which $N' = \{\, {}_xA_y \mid A \in N,\ x, y \in \Sigma \cup \{\varepsilon\}\,\}$, and the rules in R' are defined as follows.

For every pair $(B, C) \in N \times N$, define

$$\mu_{x,y}(BC) = \{{}_xB_a \cdot a \cdot {}_\varepsilon C_y \mid a \in \Sigma\} \cup \{{}_xB_\varepsilon \cdot a \cdot {}_aC_y \mid a \in \Sigma\} \cup \\ \cup \{{}_xB_\varepsilon \mid y \in L_G(C)\} \cup \{{}_\varepsilon C_y \mid x \in L_G(B)\}.$$

For every rule (5a) and for all $x, y \in \Sigma \cup \{\varepsilon\}$, let $\mu_{x,y}(B_iC_i) = \{\alpha_{i,1}, \ldots \alpha_{i,k_i}\}$ for each i-th positive conjunct, and let $\mu_{x,y}(D_jE_j) = \{\beta_{j,1}, \ldots \beta_{j,\ell_j}\}$ for each j-th negative conjunct. Then, for any choice $(i_1, \ldots, i_m)$ of numbers of positive conjuncts, the new grammar contains the rule

$${}_xA_y \to \alpha_{1,i_1} \& \cdots \& \alpha_{m,i_m} \& \neg\beta_{1,1} \& \cdots \& \neg\beta_{1,\ell_1} \cdots \& \neg\beta_{n,1} \& \cdots \& \neg\beta_{n,\ell_n},$$

which contains the chosen positive conjuncts and all negative conjuncts.

Secondly, for all $A \in N$, $x, y \in \Sigma^{\leqslant 1}$ and $a \in \Sigma$ satisfying $xay \in L_G(A)$, the new grammar contains the rule

$${}_xA_y \to a$$

Then, each symbol ${}_xA_y$ generates the language

$$L_{G'}({}_xA_y) = x^{-1}L_G(A)y^{-1} \cap \Sigma(\Sigma^2)^*,$$

Finally, define the rules for the initial symbol as follows:

$$\begin{aligned} S' &\to \varphi && \text{(for all } {}_\varepsilon S_\varepsilon \to \varphi \in R'\text{)}, \\ S' &\to a\, {}_aS_\varepsilon && \text{(for all } a \in \Sigma\text{)}. \end{aligned}$$

Then $L(G') = L(G)$, as desired. $\square$

This normal form allows replicating the construction in Theorem 1.

Theorem 3. *There exists such a Boolean language L_0 over the alphabet $\Sigma_0 = \{a, b, c, d, e, \#\}$, that for every Boolean language L over any alphabet Σ there is such a homomorphism $h\colon \Sigma \to \Sigma_0^*$, that $L = h^{-1}(L_0)$ if $\varepsilon \notin L$ and $L = h^{-1}(L_0 \cup \{\varepsilon\})$ if $\varepsilon \in L$.*

The transformation is very similar to the one in Theorem 1. The extra symbol e is used to mark negative conjuncts.

For a Boolean grammar $G = (\Sigma, N, R, S)$ in the odd normal form, let $\mathcal{C} = \{\gamma_1, \gamma_2, \ldots, \gamma_{|\mathcal{C}|}\}$, where $\gamma_i \in N\Sigma N \cup \Sigma N \cup N\Sigma \cup \Sigma$ be the set of all positive and negative conjuncts used in the grammar, so that each rule can be written in the form

$$A \to \gamma_{i_1} \,\&\, \ldots \,\&\, \gamma_{i_m} \,\&\, \neg\gamma_{i_{m+1}} \,\&\, \ldots \,\&\, \neg\gamma_{i_n} \tag{6}$$

for some $m \geqslant 1$, $n \geqslant 0$, and $i_1, \ldots, i_n \in \{1, \ldots, |\mathcal{C}|\}$.

Every such rule (6) is represented as follows:

$$\lambda(A \to \gamma_{i_1} \,\&\, \ldots \,\&\, \gamma_{i_m} \,\&\, \neg\gamma_{i_{m+1}} \,\&\, \ldots \,\&\, \neg\gamma_{i_n}) = ca^n e \ldots ca^{i_{m+1}} eca^{i_m} \ldots ca^{i_1},$$
$$\rho(A \to \gamma_{i_1} \,\&\, \ldots \,\&\, \gamma_{i_m} \,\&\, \neg\gamma_{i_{m+1}} \,\&\, \ldots \,\&\, \neg\gamma_{i_n}) = a^{i_1} c \ldots a^{i_m} cea^{i_{m+1}} c \ldots ea^n c.$$

The rest of the transformation is the same.

The "hardest" language L_0 is then defined by the following Boolean grammar, which extends the grammar given in Section 3 by adding several rules for handling the symbol e. These are the rules for the symbols $\overrightarrow{E_+}$, $\overleftarrow{E_+}$ and $\overrightarrow{E_0}$ (one for each) that actually implement the negation, as well as extra rules for B and C that take into account the extension of the alphabet.

$$\begin{aligned}
S &\to BdS \mid \overrightarrow{F_0}\overrightarrow{E} \,\&\, Ac\overrightarrow{E_0} \\
A &\to aA \mid a \\
B &\to aB \mid cB \mid eB \mid a \mid c \\
C &\to aC \mid bC \mid cC \mid dC \mid eC \mid \varepsilon \\
D &\to C\#D \mid \varepsilon \\
\overrightarrow{E} &\to \overrightarrow{F}\overrightarrow{E} \,\&\, Ac\overrightarrow{E_+} \mid dC\# \\
\overrightarrow{E_+} &\to \overrightarrow{F}\overrightarrow{E} \,\&\, Ac\overrightarrow{E_+} \mid \neg e\overrightarrow{F}\overrightarrow{E} \,\&\, eAc\overrightarrow{E_+} \mid dC\#D \\
\overrightarrow{F} &\to a\overrightarrow{F}b \mid acC\#\overleftarrow{E}b \\
\overleftarrow{E} &\to \overleftarrow{E}\overleftarrow{F} \,\&\, \overleftarrow{E_+}cA \mid Cd \\
\overleftarrow{E_+} &\to \overleftarrow{E}\overleftarrow{F} \,\&\, \overleftarrow{E_+}cA \mid \neg \overleftarrow{E}\overleftarrow{F}e \,\&\, \overleftarrow{E_+}cAe \mid DCd \\
\overleftarrow{F} &\to b\overleftarrow{F}a \mid b\overrightarrow{E}Cca \\
\overrightarrow{E_0} &\to \overrightarrow{F_0}\overrightarrow{E} \,\&\, Ac\overrightarrow{E_0} \mid \neg e\overrightarrow{F_0}\overrightarrow{E} \,\&\, eAc\overrightarrow{E_0} \mid dC\#D \\
\overrightarrow{F_0} &\to a\overrightarrow{F_0}b \mid acCdb \mid ac\overleftarrow{E}b
\end{aligned}$$

5 Any further characterizations?

Every family of languages closed under inverse homomorphisms can potentially have an analogue of Greibach's inverse homomorphic characterization. The question is, which families have it? Could it exist for linear, deterministic or unambiguous variants of ordinary (context-free) grammars? Could there be such a characterization for linear conjunctive grammars, unambiguous conjunctive grammars, etc.?

References

[1] T. Aizikowitz, M. Kaminski, "Conjunctive grammars and alternating pushdown automata", *Acta Informatica*, 50:3 (2013), 175–197.

[2] Z. Ésik, W. Kuich, "Boolean fuzzy sets", *International Journal of Foundations of Computer Science*, 18:6 (2007), 1197–1207.

[3] S. A. Greibach, "A new normal-form theorem for context-free phrase structure grammars", *Journal of the ACM*, 12 (1965), 42–52.

[4] S. A. Greibach, "The hardest context-free language", *SIAM Journal on Computing*, 2:4 (1973), 304–310.

[5] A. Jeż, "Conjunctive grammars can generate non-regular unary languages", *International Journal of Foundations of Computer Science*, 19:3 (2008), 597–615.

[6] A. Jeż, A. Okhotin, "Conjunctive grammars over a unary alphabet: undecidability and unbounded growth", *Theory of Computing Systems*, 46:1 (2010), 27–58.

[7] A. Jeż, A. Okhotin, "Complexity of equations over sets of natural numbers", *Theory of Computing Systems*, 48:2 (2011), 319–342.

[8] A. Jeż, A. Okhotin, "One-nonterminal conjunctive grammars over a unary alphabet", *Theory of Computing Systems*, 49:2 (2011), 319–342.

[9] A. Jeż, A. Okhotin, "Unambiguous conjunctive grammars over a one-letter alphabet", *Developments in Language Theory* (DLT 2013, Paris, France, 18–21 June 2013), LNCS 7907, to appear.

[10] V. Kountouriotis, Ch. Nomikos, P. Rondogiannis, "Well-founded semantics for Boolean grammars", *Information and Computation*, 207:9 (2009), 945–967.

[11] T. Lehtinen, A. Okhotin, “Boolean grammars and GSM mappings”, *International Journal of Foundations of Computer Science*, 21:5 (2010), 799–815.

[12] A. Okhotin, “Conjunctive grammars”, *Journal of Automata, Languages and Combinatorics*, 6:4 (2001), 519–535.

[13] A. Okhotin, “Conjunctive grammars and systems of language equations”, *Programming and Computer Science*, 28:5 (2002), 243–249.

[14] A. Okhotin, “On the equivalence of linear conjunctive grammars to trellis automata”, *RAIRO Informatique Théorique et Applications*, 38:1 (2004), 69–88.

[15] A. Okhotin, “Boolean grammars”, *Information and Computation*, 194:1 (2004), 19–48.

[16] A. Okhotin, “The dual of concatenation”, *Theoretical Computer Science*, 345:2–3 (2005), 425–447.

[17] A. Okhotin, “Generalized LR parsing algorithm for Boolean grammars”, *International Journal of Foundations of Computer Science*, 17:3 (2006), 629–664.

[18] A. Okhotin, “Recursive descent parsing for Boolean grammars”, *Acta Informatica*, 44:3–4 (2007), 167–189.

[19] A. Okhotin, “Fast parsing for Boolean grammars: a generalization of Valiant’s algorithm”, *Developments in Language Theory* (DLT 2010, London, Ontario, Canada, August 17–20, 2010), LNCS 6224, 340–351.

[20] A. Okhotin, “A simple P-complete problem and its language-theoretic representations”, *Theoretical Computer Science*, 412:1–2 (2011), 68–82.

[21] A. Okhotin, C. Reitwießner, “Conjunctive grammars with restricted disjunction”, *Theoretical Computer Science*, 411:26–28 (2010), 2559–2571.

[22] A. Okhotin, C. Reitwießner, “Parsing Boolean grammars over a one-letter alphabet using online convolution”, *Theoretical Computer Science*, 457 (2012), 149–157.

[23] A. Okhotin, P. Rondogiannis, “On the expressive power of univariate equations over sets of natural numbers”, *Information and Computation*, 212 (2012), 1–14.

[24] R. Zier-Vogel, M. Domaratzki, “RNA pseudoknot prediction through stochastic conjunctive grammars”, CiE 2013, to appear.

Appendix

In this preliminary report, the construction has been given without a proof. Until a proof is presented, some other evidence of the construction's correctness is required. This appendix gives such evidence in the form of calculations on several non-trivial grammars.

A The Dyck language

Consider the usual grammar for the Dyck language:

$$X \to XX \mid sXt \mid \varepsilon$$

Abandoning the empty string, this grammar can be transformed to following form:

$$\begin{aligned} &X \to sY \\ &Y \to YsY \mid Xt \mid t \end{aligned}$$

Under the enumeration of conjuncts as $\alpha_1 = sY$, $\alpha_2 = YsY$, $\alpha_3 = Xt$, $\alpha_4 = t$, the images of the symbols are:

$$\begin{aligned} h(s) &= acddbaacdbaaacdbaaaacdcaabbaacdcaabbaaacdcaabbaaaacd \\ &\quad caaabbaacdcaaabbaaacdcaaabbaaaacd \\ &\quad caaaabbaacdcaaaabbaaacdcaaaabbaaaacd\# \\ h(t) &= acddcabbbdbbbbd\# \end{aligned}$$

Each string in $\{s,t\}^*$ of length at most 12 was transformed by h and parsed according to the grammar in Section 3 using the Generalized LR parser. The images of all 196 non-empty strings belonging to the Dyck language were accepted, while the images of all remaining strings were rejected.

B A conjunctive grammar

Consider the language

$$L = \{\, t\,s^2t\,s^4t\,s^6t \ldots s^{2n}t \mid n \geqslant 0 \,\} = \{t, tsst, tsstsssst, \ldots\},$$

and a conjunctive grammar defining this language:

$$\begin{array}{lcl} X & \to & XY \,\&\, Zt \mid t \\ Y & \to & sY \mid t \\ Z & \to & YZss \mid Yss \end{array}$$

This grammar is transformed to the required form as follows.

$$\begin{array}{rcl} X & \to & XsY \mathbin{\&} Zt \mid t \\ Y & \to & sY \mid t \\ Z & \to & VsU \mid YsU \\ U & \to & s \\ V & \to & tZ \mid WtZ \\ W & \to & sW \mid s \end{array}$$

The enumeration of conjuncts is $\alpha_1 = XsY$, $\alpha_2 = Zt$, $\alpha_3 = t$, $\alpha_4 = sY$, $\alpha_5 = VsU$, $\alpha_6 = YsU$, $\alpha_7 = s$, $\alpha_8 = tZ$, $\alpha_9 = WtZ$, $\alpha_{10} = sW$.

Then the symbols have the following images:

$$\begin{aligned} h(s) = {} & acaacdaaacddcaacabaaaacdcaacabaaadcaaabaaaacdcaaabaaad \\ & bbbbaaaacdbbbbaaacdcaaaaaaaabbbbbaaaaaaaacdcaaaaaaaaabbbbbaaaaaaaacd \\ & caaaabbbbbbaaaaaaacdcaaabbbbbbaaaaaaacdbbbbbbd \\ & bbbbbbbbbbaaaaaaaaaacdbbbbbbbbbbaaaaaaaacd\# \\ h(t) = {} & acaacdaaacddcaaaaabbdcaaaaaabbdbbbdbbbbbbbbaaaaacdbbbbbbbbaaaaaacd \\ & caaaaaaaaaabbbbbbbbbaaaaacdcaaaaaaaaaabbbbbbbbbaaaaaacd \\ & caaaaaaabbbbbbbbbaaaaacdcaaaaaaabbbbbbbbbaaaaaacd\# \end{aligned}$$

The images of all strings of length up to 9 over the alphabet $\{s, t\}$ were parsed, and only the images of the strings t, $tsst$ and $tsstsssst$ were accepted.

C A Boolean grammar

Consider the following grammar, which defines the language $\{ a^m b^n \mid m \geqslant n \geqslant 1 \}$ by a vigorous use of nested negation:

$$\begin{array}{rcl} X & \to & sZ \mathbin{\&} \neg sY \\ Y & \to & Zt \mathbin{\&} \neg Xt \\ Z & \to & sZ \mid Zt \mid t \end{array}$$

For the enumeration of conjuncts $\alpha_1 = sZ$, $\alpha_2 = sY$, $\alpha_3 = Zt$, $\alpha_4 = Xt$, $\alpha_5 = t$, the images of the symbols are

$$\begin{aligned} h(s) &= aceaacddbacdbaaacdbaaaaacdbbaaaceaaaacd\# \\ h(t) &= aceaacddcabbbdcaaabbbdcaaaaabbbdcaaecabbbbdbbbbbd\# \end{aligned}$$

For every string in $\{s, t\}^*$ of length at most 12, its image was parsed according to the grammar in Section 4. The images of all 36 strings in $\{ a^m b^n \mid m \geqslant n \geqslant 1 \}$ of length at most 12 were accepted, while the rest of the images were rejected.

ISBN 978-952-12-2903-9
ISSN 1239-1891