Contents

Chapter 3

Grammars with Boolean operations

3.1 Conjunctive grammars

Conjunction corresponds to multiple conditions simultaneously. Not expressible in ordinary grammars. Augmenting the model with an explicit conjunction operation.

First mentioned in a Master's thesis by Szabari [6], supervised by Geffert. Later investigated by Okhotin [2].

Definition 3.1. *A* conjunctive grammar *is a quadruple $G = (\Sigma, N, R, S)$, in which:*

- Σ *is the* alphabet *of the language being defined, that is, a finite set of symbols, from which the strings in the language are built;*
- N *is a finite set of symbols representing syntactic categories defined by the grammar;*
- R *is a finite set of* grammar rules, *each of the form*

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_m, \tag{3.1}$$

  with $A \in N$, $m \geqslant 1$ and $\alpha_1, \ldots, \alpha_m \in (\Sigma \cup N)^*$;
- $S \in N$ *represents the set of syntactically well-formed sentences in the language.*

For every rule (3.1), each string α_i is called a *conjunct*, and if a grammar has a unique conjunct in every rule, it is an ordinary grammar. A collection of rules for a single nonterminal is written as

$$A \to \alpha_{1,1} \,\&\, \ldots \,\&\, \alpha_{1,n_1} \mid \ldots \mid \alpha_{n,1} \,\&\, \ldots \,\&\, \alpha_{n,m_n},$$

where the vertical lines are the disjunction.

Informally, a rule (3.1) states that if a string is representable as each concatenation α_i, then it has the property A.

Example 3.1. *The following conjunctive grammar generates the language $\{ a^n b^n c^n \mid n \geqslant 0\}$.*

$$\begin{array}{lcl} S & \to & AB \,\&\, DC \\ A & \to & aA \mid \varepsilon \\ B & \to & bBc \mid \varepsilon \\ C & \to & cC \mid \varepsilon \\ D & \to & aDb \mid \varepsilon \end{array}$$

The rules for the symbols A, B, C and D do not use conjunction, and have the same meaning as in an ordinary grammar. In particular, A and C generate the languages a^* and c^*, B defines $\{ b^n c^n \mid n \geqslant 0\}$ and D defines $\{ a^k b^k \mid k \geqslant 0\}$. Then the rule for S represents the strings in $a^*b^*c^*$ that have the same number of symbols b and c (ensured by AB), as well as the same number of symbols a and b (specified by DC). These are exactly all strings of the form $a^n b^n c^n$.

In other words, the grammar is based upon the representation of this language as an intersection of two ordinary languages:

$$\underbrace{\{\, a^i b^j c^k \mid j = k \}}_{L(AB)} \cap \underbrace{\{\, a^i b^j c^k \mid i = j \}}_{L(DC)} = \underbrace{\{\, a^n b^n c^n \mid n \geqslant 0 \}}_{L(S)}.$$

As in the case of ordinary grammars, the informal understanding of the meaning of a grammar can be formalized in four equivalent ways.

3.1.1 Definition by deduction

The first definition is by a formal deduction system, which represents the logical content of conjunctive grammars directly.

Definition 3.1(D). *For a conjunctive grammar $G = (\Sigma, N, R, S)$, consider elementary propositions (items) of the form "a string w has a property X", with $w \in \Sigma^*$ and $X \in \Sigma \cup N$, denoted by $X(w)$. The deduction system uses the following axioms:*

$$\vdash a(a) \qquad \textit{(for all } a \in \Sigma\textit{).}$$

Each rule $A \to X_{1,1} \ldots X_{1,\ell_1} \,\&\, \ldots \,\&\, X_{m,1} \ldots X_{m,\ell_m}$ in R, with $m \geqslant 1$, $\ell_i \geqslant 0$ and $X_{i,j} \in \Sigma \cup N$, is regarded as the following schema for deduction rules:

$$\big\{X_{i,j}(u_{i,j})\big\}_{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant j \leqslant \ell_i}} \vdash A(w) \qquad \textit{for all } u_{i,j} \in \Sigma^* \textit{ with } u_{1,1} \ldots u_{1,\ell_1} = \ldots = u_{m,1} \ldots u_{m,\ell_m} = w.$$

Whenever an item $X(w)$ can be deduced from the above axioms by the given deduction rules, this is denoted by $\vdash X(w)$. Define $L_G(X) = \{\, w \mid \vdash X(w)\}$ and $L(G) = L_G(S) = \{\, w \mid \vdash S(w)\}$.

Example 3.1(D). *Returning to the grammar from Example 3.1, consider the following derivation of the fact that the string abc is generated by the grammar.*

$$\begin{array}{rl}
\vdash a(a) & \textit{(axiom)} \\
\vdash b(b) & \textit{(axiom)} \\
\vdash c(c) & \textit{(axiom)} \\
\vdash A(\varepsilon) & \textit{(rule } A \to \varepsilon\textit{)} \\
a(a), A(\varepsilon) \vdash A(a) & \textit{(rule } A \to aA\textit{)} \\
\vdash B(\varepsilon) & \textit{(rule } B \to \varepsilon\textit{)} \\
b(b), B(\varepsilon), c(c) \vdash B(bc) & \textit{(rule } B \to bBc\textit{)} \\
\vdash D(\varepsilon) & \textit{(rule } D \to \varepsilon\textit{)} \\
a(a), D(\varepsilon), b(b) \vdash D(ab) & \textit{(rule } D \to aDb\textit{)} \\
\vdash C(\varepsilon) & \textit{(rule } C \to \varepsilon\textit{)} \\
c(c), C(\varepsilon) \vdash C(c) & \textit{(rule } C \to cC\textit{)} \\
A(a), B(bc), D(ab), C(c) \vdash S(abc) & \textit{(rule } S \to AB \,\&\, DC\textit{)}
\end{array}$$

3.1.2 Definition by term rewriting

The rewriting in conjunctive grammars is carried out generally in the same way as in Chomsky's definition of ordinary grammars. The only difference is in the objects being transformed: while rewriting in ordinary grammars operates with strings over $\Sigma \cup N$, which are terms over concatenation, rewriting in conjunctive grammars use terms over concatenation and conjunction.

Definition 3.1(R) (Szabari [6], Okhotin [2]). *Given a grammar G, consider terms over concatenation and conjunction with symbols from $\Sigma \cup N$ and the empty string ε as atomic terms. Assume that the symbols "(", "&" and ")" used to construct the terms are not in $\Sigma \cup N$. The relation $\Longrightarrow$ of one-step rewriting on the set of terms 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 relations of rewriting in zero or more steps, in one or more steps and in exactly ℓ steps are denoted by $\Longrightarrow^*$, $\Longrightarrow^+$ and $\Longrightarrow^\ell$, respectively. 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:

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

The language generated by the grammar is the language generated by the term S:

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

For simplicity, when a single-conjunct rule $A \to \alpha$ is applied, one can omit the parentheses, and rewrite A with α, rather than with (α).

Example 3.1(R). *Consider the grammar in Example 3.1. According to the definition by term rewriting, the string abc can be obtained by the following rewriting sequence.*

$S \Longrightarrow (AB \,\&\, DC) \Longrightarrow (aAB \,\&\, DC) \Longrightarrow (aB \,\&\, DC) \Longrightarrow (abBc \,\&\, DC) \Longrightarrow (abc \,\&\, DC) \Longrightarrow (abc \,\&\, aDbC) \Longrightarrow (abc \,\&\, abC) \Longrightarrow (abc \,\&\, abcC) \Longrightarrow (abc \,\&\, abc) \Longrightarrow abc.$

In essence, here two rewriting sequences, as in ordinary grammars, are carried out independently of each other, and both AB and DC have to be rewritten to the same string, in order to perform the last step of the rewriting.

3.1.3 Definition by parse trees

Parse trees for conjunctive grammars are, strictly speaking, directed acyclic graphs rather than trees.

Definition 3.1(T). *Let $G = (\Sigma, N, R, S)$ be a conjunctive grammar. Then a parse tree of a string $w \in \Sigma^*$ from $X \in \Sigma \cup N$ is any connected directed acyclic graph defined as follows. Each node of the tree is labelled either with a symbol from Σ or with a rule from R. All nodes labelled with symbols in Σ are linearly ordered, forming the string w; the set of Σ-labelled leaves in any subtree must form a contiguous substring of w There is a unique source node labelled with X or with a rule for X. A node labelled with $a \in \Sigma$ must have no sons.*

Each node x labelled with a rule $A \to X_{1,1} \ldots X_{1,\ell_1} \,\&\, \ldots \,\&\, X_{m,1} \ldots X_{m,\ell_m}$ must have sons labelled with $X_{1,1}, \ldots, X_{1,\ell_1}, \ldots, X_{m,1}, \ldots, X_{m,\ell_m}$, and these sons are linearly ordered. Let u be the substring of w formed by the leaves in the subtree of x. Then there should exist m partitions of u as $u = u_{1,1} \ldots u_{1,\ell_1} = \ldots = u_{m,1} \ldots u_{m,\ell_m}$, so that the subtree of each $X_{i,j}$ includes exactly the leaves corresponding to the symbols in $u_{i,j}$.

Consider the grammar from Example 3.1. Then, according to Definition 3.1(T), the tree in Figure 3.1 witnesses that the string *abc* is in the language defined by this grammar.

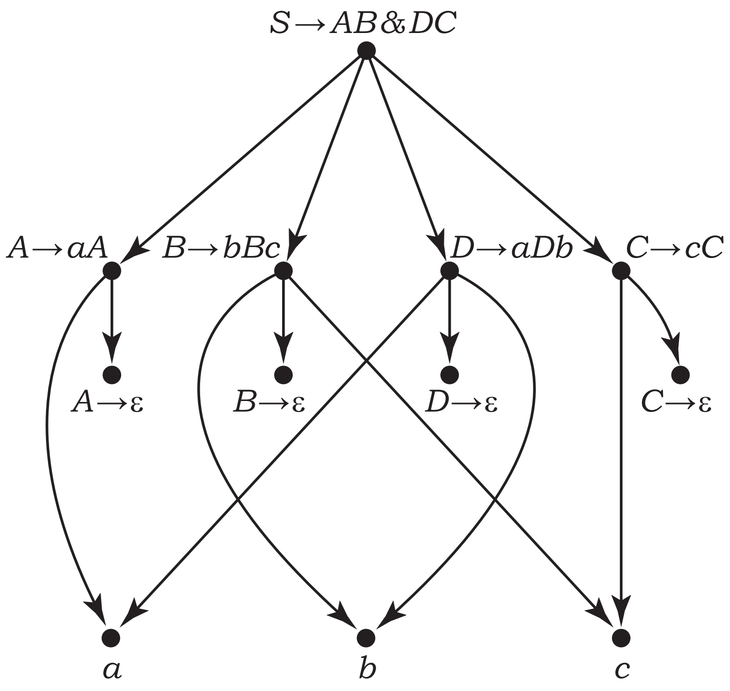

Figure 3.1: Parse tree of the string *abc* according to the conjunctive grammar for $\{\, a^n b^n c^n \mid n \geqslant 0\}$ given in Example 3.1.

3.1.4 Definition by language equations

Another equivalent definition of the language generated by a conjunctive grammar is by a solution of a system of language equations. The system corresponding to a grammar is defined similarly to the case of ordinary grammars, with the conjunction represented by the intersection operation. The resulting system is bound to have a least solution by the same basic lattice-theoretic argument as in the ordinary case with disjunction only, based only on the fact that the right-hand sides of the equations are monotone and continuous functions (a property shared by the intersection operation).

Definition 3.1(E) (Okhotin [3])**.** *For every conjunctive grammar $G = (\Sigma, N, R, S)$, the associated system of language equations is a system of equations in variables N, with each variable representing an unknown language over Σ, which contains the following equation for every variable A:*

$$A = \bigcup_{A\to\alpha_1 \& \ldots \& \alpha_m \in R} \ \bigcap_{i=1}^{m} \alpha_i \quad (\textit{for all } A \in N) \tag{3.2}$$

Each α_i in the equation is again a concatenation of variables and constant languages $\{a\}$ representing symbols of the alphabet, or constant $\{\varepsilon\}$ if α_i is the empty string. Let $(\ldots, L_A, \ldots)_{A\in N}$ be the least solution of this system. Then $L_G(A)$ is defined as L_A for each $A \in N$, and $L(G) = L_S$.

This least solution can be obtained as a limit of an ascending sequence of vectors of languages, with the first element $\bot = (\varnothing, \ldots, \varnothing)$, and with every next element obtained by applying the right-hand sides of the system (3.2) as a vector function $\varphi : \left(2^{\Sigma^*}\right)^{|N|} \to \left(2^{\Sigma^*}\right)^{|N|}$ to the previous element. Since this function is monotone with respect to the partial ordering $\sqsubseteq$ of componentwise inclusion, the resulting sequence $\{\varphi^k(\bot)\}_{k\to\infty}$ is ascending, and the continuity of φ implies that its limit (least upper bound) $\bigsqcup_{k\geqslant 0} \varphi^k(\bot)$ is the least solution.

Example 3.1(E). *According to the definition by language equations, the system corresponding to the grammar in Example 3.1 is*

$$\begin{cases} S &= AB \cap DC \\ A &= \{a\}A \cup \{\varepsilon\} \\ B &= \{b\}B\{c\} \cup \{\varepsilon\} \\ C &= \{c\}C \cup \{\varepsilon\} \\ D &= \{a\}D\{b\} \cup \{\varepsilon\}, \end{cases}$$

and this system has a unique solution with $S = \{ a^n b^n c^n \mid n \geqslant 0\}$, $A = a^*$, $B = \{ b^m c^m \mid m \geqslant 0\}$, $C = c^*$ and $D = \{ a^m b^m \mid m \geqslant 0\}$.

The membership of abc

$$\begin{array}{cccc} \begin{pmatrix} \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \end{pmatrix} & \begin{pmatrix} \varnothing \\ \{\varepsilon\} \\ \{\varepsilon\} \\ \{\varepsilon\} \\ \{\varepsilon\} \end{pmatrix} & \begin{pmatrix} \{\varepsilon\} \\ \{\varepsilon, a\} \\ \{\varepsilon, bc\} \\ \{\varepsilon, c\} \\ \{\varepsilon, ab\} \end{pmatrix} & \begin{pmatrix} \{\varepsilon, abc\} \\ \{\varepsilon, a, aa\} \\ \{\varepsilon, bc, bbcc\} \\ \{\varepsilon, c, cc\} \\ \{\varepsilon, ab, aabb\} \end{pmatrix} \quad \ldots \\ \perp & \varphi(\perp) & \varphi^2(\perp) & \varphi^3(\perp) \end{array}$$

3.1.5 Equivalence of the four definitions

To see that Definitions 3.1(R), 3.1(D), 3.1(T), 3.1(E) are equivalent.

Theorem 3.1. *Let $G = (\Sigma, N, R, S)$ be a conjunctive grammar, as in Definition 3.1. For every $X \in \Sigma \cup N$ and $w \in \Sigma^*$, the following four statements are equivalent:*

(R). $X \Longrightarrow^* w$,

(D). $\vdash X(w)$,

(T). there is a parse tree of w from X,

(E). $w \in \left[\bigsqcup_{k \geqslant 0} \varphi^k(\perp)\right]_X$.

3.2 Examples of conjunctive grammars

Consider the language $\{ wcw \mid w \in \{a, b\}^*\}$, which is among the most common examples of languages that have no ordinary grammar. This language represents such syntactic constructs as identifier checking in programming languages. As proved by Wotschke [7], it is not expressible as an intersection of finitely many ordinary languages. Constructing a conjunctive grammar for this language thus requires more than putting a conjunction on top of an ordinary grammar.

Example 3.2 (Okhotin [2]). *The following conjunctive grammar describe the language $\{ wcw \mid w \in \{a, b\}^*\}$.*

$$\begin{array}{lcl} S & \to & C \,\&\, D \\ C & \to & XCX \mid c \\ D & \to & aA \,\&\, aD \mid bB \,\&\, bD \mid cE \\ A & \to & XAX \mid cEa \\ B & \to & XBX \mid cEb \\ E & \to & XE \mid \varepsilon \\ X & \to & a \mid b \end{array}$$

The nonterminal symbol C defines the language $\{\, xcy | x, y \in \{a,b\}^*; |x| = |y|\}$ in the standard way, and thus the conjunction with C in the rule for S ensures that the string consists of two parts of equal length separated by a center marker. The other conjunct D checks that the symbols in corresponding positions are the same. The actual language generated by D is $L(D) = \{\, uczu \,|\, u, z \in \{a,b\}^*\}$, and it is defined inductively as follows: a string is in $L(D)$ if and only if

- either it is in $c\{a,b\}^*$ (the base case: no symbols to compare),
- or its first symbol is the same as the corresponding symbol on the other side, *and* the string without its first symbol is in $L(D)$ (that is, the rest of the symbols in the left part correctly correspond to the symbols in the right part).

The comparison of a single symbol to the corresponding symbol on the right is done by the nonterminals A and B, which generate the languages $\{\, xcvay \mid x, v, y \in \{a,b\}^*,\ |x| = |y|\}$ and $\{\, xcvby \mid x, v, y \in \{a,b\}^*,\ |x| = |y|\}$, respectively, and the above inductive definition is directly expressed in the rules for D, which recursively refer to D in order to apply the same rule to the rest of the string. Finally, the rule for S defines the set of strings of the form xcy with $|x| = |y|$ (ensured by C) and $x = y$ (imposed by D), and

$$\{\, xcy \mid x, y \in \{a,b\}^*,\ |x| = |y|\} \cap \{\, uczu \mid u, z \in \{a,b\}^*\} = \{\, wcw \mid w \in \{a,b\}^*\}.$$

It is important to note that the construction essentially uses the center marker, and therefore this method cannot be applied to constructing a conjunctive grammar for the language $\{\, ww \mid w \in \{a,b\}^*\}$. The question of whether $\{\, ww \mid w \in \{a,b\}^*\}$ can be generated by any conjunctive grammar remains an open problem.

The grammar in the next example defines the requirement of *declaration before use.*

Example 3.3. *The language*

$$\{\, s_1 a^{i_1} \ldots s_n a^{i_n} \mid n, i_1, \ldots, i_n \geqslant 0;\ \forall j \in \{1, \ldots, n\},\ \textit{if } s_i = c,\ \textit{then } \exists k < i : j_k = j_i\}.$$

is generated by the following grammar.

$$\begin{array}{lll} S & \to & SdA \mid ScA \,\&\, EdB \mid \varepsilon \\ A & \to & aA \mid \varepsilon \\ B & \to & aBa \mid Ec \\ E & \to & cAE \mid dAE \mid \varepsilon \end{array}$$

A substring of the form da^k represents a declaration of k, and every substring of the form ca^k is a reference to k, which requires an earlier declaration da^k.

The grammar applies generally the same technique of inductive definitions as in Example 3.2. The rule $S \to \varepsilon$ asserts that an empty sequence of declarations and references has the required property. The rule $S \to SdA$ appends a new declaration (dA) to a well-formed string with all references preceded by declarations (S). The other rule $S \to ScA \,\&\, EdB$ similarly appends a reference (cA), and at the same time ensures that this new reference has a preceding declaration (EdB). Here E defines an arbitrary sequence of declarations and references, and the concatenation EdB defines a suitable partition of the string, where the symbol d begins the appropriate declaration, and B ensures that the number of symbols a is the same in the declaration and in the reference.

The next example shows that conjunctive grammars over a one-symbol alphabet have a non-trivial expressive power, unlike the ordinary grammars, which are limited to regular unary languages. Though this does not exactly pertain to defining syntax, the demonstrated technique for constructing conjunctive grammars for unary languages has numerous theoretical implications to be explained later.

Example 3.4 (Jeż [1]). *The following conjunctive grammar with the start symbol A_1 generates the language $\{ a^{4^n} \mid n \geqslant 0\}$:*

$$\begin{array}{lcl} A_1 & \to & A_1A_3 \,\&\, A_2A_2 \mid a \\ A_2 & \to & A_1A_1 \,\&\, A_2A_6 \mid aa \\ A_3 & \to & A_1A_2 \,\&\, A_6A_6 \mid aaa \\ A_6 & \to & A_1A_2 \,\&\, A_3A_3 \end{array}$$

Each symbol A_i generates the language $\{ a^{i \cdot 4^n} \mid n \geqslant 0\}$.

This grammar is best explained in terms of base-4 notation of the length of the strings. Then each nonterminal A_i with $i \in \{1, 2, 3\}$ represents base-4 numbers $i\mathtt{0}\ldots\mathtt{0}$, or $\mathtt{120}\ldots\mathtt{0}$ for A_6. Substituting these four languages into the equation

$$A_1 = (A_1A_3 \cap A_2A_2) \cup \{a\},$$

the first concatenation A_1A_3 produces all numbers with the notation $\mathtt{10}^*\mathtt{30}^*$, $\mathtt{30}^*\mathtt{10}^*$ and $\mathtt{10}^+$, of which the latter is the intended set, while the rest are regarded as garbage. The second concatenation A_2A_2 yields $\mathtt{20}^*\mathtt{20}^*$ and $\mathtt{10}^+$. Though both concatenations contain some garbage, the garbage in the concatenations is disjoint, and is accordingly filtered out by the intersection, which produces exactly the numbers with the notation $\mathtt{10}^+$, that is, the language $\{ a^{4^n} \mid n \geqslant 1\}$. Finally, the union with $\{a\}$ yields the language $\{ a^{4^n} \mid n \geqslant 0\}$, and thus the first equation turns into an equality. The rest of the equations are verified similarly. Since the membership of each string in the languages $L(A_i)$ depends on the membership of strictly shorter strings, the grammar defines the correct languages by a proper induction. Proving that formally is a matter of technique [1].

Exercises

3.2.1. Construct a conjunctive grammar for the language $\{ a^m b^n c^m d^n \mid m, n \geqslant 0\}$.

3.2.2. Construct a conjunctive grammar for the language $\{ w \mid w \in \{a, b, c\}^*,\ |w|_a = |w|_b = |w|_c\}$.

3.2.3. Construct a conjunctive grammar for the language

$$\{ da^{i_1} \ldots da^{i_n} \mid n, i_1, \ldots, i_n \geqslant 0, \text{ and the numbers } i_1, \ldots, i_n \text{ are pairwise distinct}\},$$

which adopts the encoding from Example 3.3 and represents the condition of having no duplicate declarations.

3.2.4. Construct a conjunctive grammar for the language

$$L = \{ b\,ab\,a^3b\,a^7b\,a^{15}b \ldots a^{2^n-1}b \mid n \geqslant 0\} = \{b, bab, babaaab, babaaabaaaaaaab, \ldots\}.$$

3.2.5. Construct a conjunctive grammar for the language

$$L = \{ (a^n b)^n \mid n \geqslant 1\} = \{ab, aabaab, aaabaaabaaab, \ldots\}.$$

3.2.6. Construct a conjunctive grammar for the language $\{ (wc)^{|w|} \mid w \in \{a, b\}^*\}$.

3.2.7. Construct a conjunctive grammar for the language $\{ wcww \mid w \in \{a, b\}^*\}$.

3.2.8. Construct a conjunctive grammar for the language $\{ a^{5^n} \mid n \geqslant 0\}$.

Research problems

3.2.9. Does there exist a conjunctive grammar for the language $\{ ww \mid w \in \{a,b\}^* \}$?

3.2.10. Does there exist a conjunctive grammar for the language $\{ a^n b^{in} \mid n, i \geqslant 0 \}$?

3.2.11. Does there exist a conjunctive grammar for the language $\{ a^{n^2} \mid n \geqslant 0 \}$?

3.3 Normal forms for conjunctive grammars

Three normal forms. First, an extension of the *Chomsky normal form*, also called the binary normal form, in which all rules are of the form $A \to B_1C_1 \,\&\, \dots \,\&\, B_mC_m$ (the Chomsky normal form for ordinary grammars has $m = 1$) or $A \to a$. Secondly, a new normal form called the *odd normal form*, with rules $A \to B_1a_1C_1 \,\&\, \dots \,\&\, B_ma_mC_m$ or $A \to a$, in which all generated strings are of an odd length. Thirdly, a form with restricted disjunction, where every symbol A is defined as $A \to \alpha_1 \,\&\, \dots \,\&\, \alpha_m \mid w_1 \dots \mid w_n$, that is, there is only one rule of the general form, and the rest simply assert that the listed strings have the property A.

The transformation to the Chomsky normal form proceeds by first eliminating *null conjuncts*, that is, rules of the form $A \to \varepsilon \,\&\, \dots$ that can potentially generate the empty string. At the second step, *unit conjuncts*, that is, rules of the form $A \to B \,\&\, \dots$ that potentially cause circularities in the definition,

3.3.1 Eliminating null conjuncts

Similar to the case of ordinary grammars.

Lemma 3.1. *There exists an algorithm that, given a conjunctive grammar $G = (\Sigma, N, R, S)$, constructs the set* $\textsc{Nullable}(G) = \{ A \mid A \in N,\ \varepsilon \in L_G(A) \}$.

Proof. Nested sets:

$$\begin{aligned}
\textsc{Nullable}_0(G) &= \varnothing, \\
\textsc{Nullable}_{k+1}(G) &= \{A \in N \mid \text{there is a rule } A \to X_{1,1} \dots X_{1,\ell_1} \,\&\, \dots \,\&\, X_{m,1} \dots X_{m,\ell_m} \\
&\qquad\qquad \text{with } X_{i,j} \in \textsc{Nullable}_k(G)\}.
\end{aligned}$$

Correctness: as in the ordinary case, $A \in \textsc{Nullable}_k$ if and only if $\varepsilon \in [\varphi^k(\bot)]_A$. □

Lemma 3.2. *Let $G = (\Sigma, N, R, S)$ be a grammar and let $\textsc{Nullable}(G)$ be as in Lemma 3.1. Construct another grammar $G' = (\Sigma, N, R', S)$, where R' contains all rules of the form*

$$A \to X_{1,1} \dots X_{1,\ell_1} \,\&\, \dots \,\&\, X_{m,1} \dots X_{m,\ell_m},$$

for which there is a rule $A \to \theta_{1,0}X_{1,1}\theta_{1,1} \dots \theta_{1,\ell_1-1}X_{1,\ell_1}\theta_{1,\ell_1} \,\&\, \dots \,\&\, \theta_{m,0}X_{m,1}\theta_{m,1} \dots \theta_{m,\ell_m-1}X_{m,\ell_m}\theta_{m,\ell_m}$ in R, with $m \geqslant 1$, $\ell_i \geqslant 1$ and $\theta_{i,j} \in \textsc{Nullable}(G)^*$. Then, for every $A \in N$, $L_{G'}(A) = L_G(A) \setminus \{\varepsilon\}$.

3.3.2 Eliminating unit conjuncts

Lemma 3.3 (Substitution of unit conjuncts)**.** *Let $G = (\Sigma, N, R, S)$ be a conjunctive grammar. Let $A, B \in N$ and let*

$$A \to B \,\&\, \alpha_1 \,\&\, \dots \,\&\, \alpha_m \quad (m \geqslant 0,\ \alpha_i \in (\Sigma \cup N)^*) \tag{3.3}$$

be some rule for A that contains a unit conjunct B. Let $B \to \beta_{1,1} \& \ldots \& \beta_{1,\ell_1}$, $\ldots$, $B \to \beta_{k,1} \& \ldots \& \beta_{k,\ell_k}$ be all rules for B that do not contain the unit conjunct B ($\beta_{i,j} \neq B$).

Construct the grammar $G' = (\Sigma, N.R', S)$ by removing the rule (3.3) and, if $A \neq B$, adding the collection of rules

$$A \to \beta_{i,1} \& \ldots \& \beta_{i,\ell_i} \& \alpha_1 \& \ldots \& \alpha_m \qquad (1 \leqslant i \leqslant k).$$

Then $L_G(A) = L_{G'}(A)$ for all $A \in N$.

Let $G = (\Sigma, N, R, S)$ be a grammar without null conjuncts, and consider a directed graph of immediate reachability by unit conjuncts, with the set of nodes N and with the arcs $\{ (A, B) \mid \text{there is a rule } A \to B \& \ldots \text{ in } G\})$.

Lemma 3.4. *Let $\Gamma = (\{A_1 \ldots A_n\}, E)$ be a directed graph without multiple arcs. Then, using the transformation rules*

i. Any arc (A, B), with $A \neq B$, can be substituted with a possibly empty set of arcs $\{ (A, C) \mid (B, C) \in E,\ C \neq B\}$,

ii. Any loop (A, A) can be removed,

it is possible to remove all arcs from the graph in finitely many steps.

Proof. First, all the arcs (A_i, A_j), with $i \geqslant j$, are removed in the following order

$$\begin{array}{c} (A_1, A_1), \\ (A_2, A_1), (A_2, A_2) \\ (A_3, A_1), (A_3, A_2), (A_3, A_3), \\ \vdots \\ (A_n, A_1), \ldots, (A_n, A_{n-1}), (A_n, A_n). \end{array} \tag{3.4}$$

The removal of neither of them leads to restoration of any previously removed arc. The proof is by an induction on the position of the arc in the list (3.4). (***TBW***)

Once all the arcs (3.4) are removed, the resulting graph is acyclic, and it is possible to eliminate arcs leading to sink nodes one by one, until none remain in the graph. □

Theorem 3.2. *For each conjunctive grammar $G = (\Sigma, N, R, S)$ without null conjuncts there exists a conjunctive grammar $G' = (\Sigma, N, R', S)$ without null and unit conjuncts, which defines the same language.*

Proof. Note that grammar transformations carried out by Lemma 3.3 are equivalent to graph transformations done by Lemma 3.4. Since the latter provides a way to remove all arcs from the graph of immediate reachability by unit conjuncts in finitely many steps, there is a corresponding sequence of grammar transformations by Lemma 3.3, which leads to the removal of all unit conjuncts from the grammar. □

The transformation may lead to an exponential blow-up: if the given grammar contains the rules

$$\begin{array}{rcl} A & \to & B_1 \& \ldots \& B_k \\ B_1 & \to & \beta_1 \mid \beta_1' \\ & \vdots & \\ B_k & \to & \beta_k \mid \beta_k', \end{array}$$

where $|\beta_i|, |\beta_i'| \geqslant 2$, then the resulting grammar will have 2^k rules for A.

3.3.3 The Chomsky normal form

Definition 3.2. *A conjunctive grammar $G = (\Sigma, N, R, S)$ is said to be in the **Chomsky normal form**, if every rule of the form*

$$\begin{aligned} &A \to B_1C_1 \,\&\, \ldots \,\&\, B_mC_m && (m \geqslant 1,\ B_i, C_i \in N), \\ &A \to a && (a \in \Sigma), \\ &S \to \varepsilon, \end{aligned}$$

where the last rule is allowed only if S does not appear in the right-hand sides of any rules.

Theorem 3.3 (Okhotin [2])**.** *For every conjunctive grammar there exists and can be effectively constructed a conjunctive grammar in the Chomsky normal form generating the same language.*

Sketch of a proof. As in the case of ordinary grammars: first cut long rules, then eliminate null conjuncts, then eliminate unit conjuncts, and finally move every occurrence of a symbol from Σ in conjuncts of length 2 to a separate rule. ☐

Elimination of null conjuncts can be achieved with only a linear blowup in the size of the grammar, but the known procedure for eliminating unit conjuncts leads, in the worst case, to an exponential blowup.

3.3.4 The odd normal form

Definition 3.3. *A conjunctive grammar $G = (\Sigma, N, R, S)$ is said to be in the **odd normal form**, if all rules in R are of the form*

$$\begin{aligned} &A \to B_1a_1C_1 \,\&\, \ldots \,\&\, B_ma_mC_m && (n \geqslant 1,\ B_i, C_i \in N,\ a_i \in \Sigma) \\ &A \to a && (a \in \Sigma) \end{aligned}$$

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

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

For each $a \in \Sigma$, there is at most one rule $S \to aA$.

Theorem 3.4 (Okhotin, Reitwießner [5])**.** *For every conjunctive grammar there exists and can be effectively constructed a conjunctive grammar in the odd normal form generating the same language.*

The proof is based on the following construction, which produces a grammar generating the odd subset of the given language, and that grammar is *almost* in the odd normal form, with occasional unit conjuncts.

Lemma 3.5. *Let $G = (\Sigma, N, R, S)$ be a conjunctive grammar in the Chomsky normal form. Define the grammar $G' = (\Sigma, N', R', S')$, in which the set of nonterminals is $N' = \{\, {}_xA_y \mid A \in N,\ x, y \in \Sigma \cup \{\varepsilon\}\}$ with the initial symbol $S' = {}_\varepsilon S_\varepsilon$, and the set R' contains the following rules: first, for every rule $A \to B^{(1)}C^{(1)}\&\ldots\&B^{(m)}C^{(m)}$ in R, and for all $x, y \in \Sigma \cup \{\varepsilon\}$, the new grammar contains all rules*

$$ {}_xA_y \to \alpha^{(1)} \,\&\, \ldots \,\&\, \alpha^{(m)},$$

where each $\alpha^{(i)}$ is in

$$\{{}_xB_a^{(i)} \cdot a \cdot {}_\varepsilon C_y^{(i)} \mid a \in \Sigma\} \cup \{{}_xB_\varepsilon^{(i)} \cdot a \cdot {}_a C_y^{(i)} \mid a \in \Sigma\} \cup \{{}_xB_\varepsilon^{(i)} \mid y \in L_G(C^{(i)})\} \cup \{{}_\varepsilon C_y^{(i)} \mid x \in L_G(B^{(i)})\}.$$

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 $_xA_y$ generates the language

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

and, in particular, $L(G') = L_{G'}(_\varepsilon S_\varepsilon) = L(G) \cap \textsc{Odd}$.

The grammar constructed in Lemma 3.5 generates the odd subset of the given language. However, it actually encodes the entire information defined in the original grammar, and using the "even rules" allowed in the odd normal form one can generate the original language as it is.

Proof of Theorem 3.4. Let $L \subseteq \Sigma^*$ be conjunctive. Since every conjunctive language can be generated by a conjunctive grammar in Chomsky normal form (which can be obtained effectively), there is, by Lemma 3.5, a conjunctive grammar $G = (\Sigma, N, R, S)$ in odd normal form without even rules, such that for all $a \in \Sigma$,

$$L_G(S) = L \cap \textsc{Odd} \qquad \text{and} \qquad L_G(_aS_\varepsilon) = a^{-1}L \cap \textsc{Odd}.$$

Construct the grammar $G' = (\Sigma, N \cup \{S'\}, R \cup R', S')$, where S' is the new initial symbol with the following new rules:

$$\begin{array}{ll} S' \to \varphi & \text{(for all } S \to \varphi \in R\text{)}, \\ S' \to a\ {}_aS_\varepsilon & \text{(for all } a \in \Sigma\text{)}, \\ S' \to \varepsilon & \text{if } \varepsilon \in L. \end{array}$$

This grammar is in the odd normal form (with even rules) and it generates the language L:

$$\begin{aligned} L_{G'}(S') &= L_G(S) \cup \bigcup_{a\in\Sigma} aL_G(_aS_\varepsilon) \cup (L \cap \{\varepsilon\}) \\ &= (L \cap \textsc{Odd}) \cup \bigcup_{a\in\Sigma} a(a^{-1}L \cap \textsc{Odd}) \cup (L \cap \{\varepsilon\}) \\ &= (L \cap \textsc{Odd}) \cup \bigcup_{a\in\Sigma} (a(a^{-1}L) \cap a\textsc{Odd}) \cup (L \cap \{\varepsilon\}) \\ &= (L \cap \textsc{Odd}) \cup (L \cap \textsc{Even}) \cup (L \cap \{\varepsilon\}) \\ &= L. \end{aligned}$$

□

Theorem 3.5 (Okhotin, Reitwießner [5]). *For every conjunctive grammar $G = (\Sigma, N, R, S)$ and for every symbol $a \in \Sigma$, there exists and can be effectively constructed conjunctive grammars generating the languages $a^{-1} \cdot L(G) = \{\, w \mid aw \in L(G)\}$ and $L(G) \cdot a^{-1} = \{\, w \mid wa \in L(G)\}$.*

3.3.5 An analogue of Greibach normal form?

For ordinary grammars, there is another important normal form: the *Greibach normal form*, in which every rule is either $A \to a\alpha$ with $a \in \Sigma$ and $\alpha \in (\Sigma \cup N)^*$, or $A \to \varepsilon$. This definition naturally carries on to conjunctive grammars. It can be said that a conjunctive grammar $G = (\Sigma, N, R, S)$ is in Greibach normal form if every rule in R is of the form

$$\begin{aligned} &A \to a\alpha_1 \mathbin{\&} \ldots \mathbin{\&} a\alpha_n \quad (n \geqslant 1,\ a \in \Sigma,\ \alpha_i \in N^*) \quad \text{or} \\ &A \to \varepsilon. \end{aligned}$$

A transformation to this form is known only for the special case of a one-symbol alphabet. It can be inferred from Theorem 3.4 by first transforming a grammar to the odd normal form, and then using the commutativity of concatenation of unary languages.

Theorem 3.6 (Okhotin, Reitwießner [5]; independently obtained by Nomikos and Rondogiannis)**.** *For every conjunctive grammar over an alphabet $\Sigma = \{a\}$, there exists and can be effectively constructed a conjunctive grammar in the Greibach normal form generating the same language.*

A simple direct proof. Let $G = (\{a\}, N, R, S)$ be the original grammar and assume that it is in the Chomsky normal form. Construct a new grammar $G' = (\{a\}, N \cup \{S'\}, R', S)$ with the following rules:

$$\begin{array}{ll} S' \to aS & \\ A \to aB_1C_1 \mathbin{\&} \ldots \mathbin{\&} aB_mC_m & (A \to B_1C_1 \mathbin{\&} \ldots \mathbin{\&} B_mC_m \in R) \\ A \to \varepsilon & (A \to a \in R) \end{array}$$

Then $L_{G'}(A) = \{\, a^{n-1} \mid a^n \in L_G(A)\}$ for all $A \in N$, and hence $L(G') = L(G)$. □

Example 3.5. *Consider the grammar for the language $\{\, a^{4^n} \mid n \geqslant 0\}$ given in Example 3.4. It can be rewritten as follows.*

$$\begin{array}{rcl} S & \to & aA_1 \\ A_1 & \to & aA_1A_3 \mathbin{\&} aA_2A_2 \mid \varepsilon \\ A_2 & \to & aA_1A_1 \mathbin{\&} aA_2A_6 \mid a \\ A_3 & \to & aA_1A_2 \mathbin{\&} aA_6A_6 \mid aa \\ A_6 & \to & aA_1A_2 \mathbin{\&} aA_3A_3 \end{array}$$

Each symbol A_i generates the language $\{\, a^{i \cdot 4^n - 1} \mid n \geqslant 0\}$.

It remains unknown, whether every conjunctive grammar over an unrestricted alphabet can be transformed to that form.

Research problems

3.3.1. Is it possible to remove unit conjuncts from a conjunctive grammar with a less than exponential blow-up?

3.3.2. Is there a Greibach normal form for conjunctive grammars?

3.3.3. Is it possible to transform every conjunctive grammar to a grammar with restricted disjunction and without null conjuncts? See the unsuccessful attempt by Okhotin and Reitwießner [5].

3.4 Boolean grammars

3.4.1 Intuitive definition

Boolean grammars are ordinary grammars equipped with all propositional connectives, or, in other words, conjunctive grammars augmented with negation. One can say that conjunctive grammars are the monotone fragment of Boolean grammars, and ordinary grammars are their disjunctive fragment.

Definition 3.4 (Okhotin [4]). *A Boolean grammar is a quadruple $G = (\Sigma, N, R, S)$, in which*

- *Σ is the alphabet;*
- *N is the set of nonterminal symbols;*
- *P is a finite set of rules of the form*

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_m \,\&\, \neg\beta_1 \,\&\, \ldots \,\&\, \neg\beta_n \tag{3.5}$$

  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.*

The only difference from a conjunctive grammar is that some conjuncts can be negated: the conjuncts α_i and $\neg\beta_j$ are called *positive* and *negative* respectively, with the notation $\pm\alpha_i$ and $\pm\beta_j$ occasionally used to refer to conjuncts, without specifying whether they are positive or negative. A rule (3.5) can be read as follows: *“if a string is representable in the form α_1, …, α_m, but is not representable in the form β_1, …, β_n, then this string has the property A”.* This intuitive interpretation is not yet a formal definition, but this understanding is sufficient to construct grammars.

Example 3.6 (cf. Example 3.1). *The following Boolean grammar defines the language $\{ a^m b^n c^n \mid m, n \geqslant 0, m \neq n\}$:*

$$\begin{array}{rcl} S & \to & AB \,\&\, \neg DC \\ A & \to & aA \mid \varepsilon \\ B & \to & bBc \mid \varepsilon \\ C & \to & cC \mid \varepsilon \\ D & \to & aDb \mid \varepsilon \end{array}$$

The rules for the nonterminals A, B, C and D are ordinary, so, according to the intuitive semantics, they should generate the same languages as in Example 3.1. Then the propositional connectives in the rule for S specify the following combination of the conditions given by AB and DC:

$$\underbrace{\{ a^i b^j c^k \mid j = k\}}_{L(AB)} \cap \underbrace{\overline{\{ a^i b^j c^k \mid i = j\}}}_{L(DC)} = \{ a^i b^j c^k \mid j = k \text{ and } i \neq j\} = \underbrace{\{ a^m b^n c^n \mid m \neq n\}}_{L(S)}.$$

Example 3.7. *The following Boolean grammar generates the language $\{ ww \mid w \in \{a, b\}^*\}$:*

$$\begin{array}{rcl} S & \to & \neg AB \,\&\, \neg BA \,\&\, C \\ A & \to & XAX \mid a \\ B & \to & XBX \mid b \\ C & \to & XXC \mid \varepsilon \\ X & \to & a \mid b \end{array}$$

Again, according to the intuitive semantics, the nonterminals A, B, C and X should generate the appropriate ordinary languages, and

$$\begin{aligned} L(A) &= \{\, uav \mid u,v \in \{a,b\}^*,\ |u| = |v|\}, \\ L(B) &= \{\, ubv \mid u,v \in \{a,b\}^*,\ |u| = |v|\}. \end{aligned}$$

This implies

$$L(AB) = \{\, uavxby \mid u,v,x,y \in \{a,b\}^*, |u| = |x|, |v| = |y|\},$$

that is, $L(AB)$ contains all strings of even length with a mismatched a on the left and b on the right (in any position). Similarly,

$$L(BA) = \{\, ubvxay \mid u,v,x,y \in \{a,b\}^*, |u| = |x|, |v| = |y|\}$$

represents the mismatch formed by b on the left and a on the right. Then the rule for S defines the set of strings of even length without such mismatches:

$$L(S) = \overline{L(AB)} \cap \overline{L(BA)} \cap \{aa, ab, ba, bb\}^* = \{\, ww \mid w \in \{a,b\}^*\}.$$

Example 3.8. *The Boolean grammar*

$$\begin{array}{lcl} S & \to & \neg aA \,\&\, C \\ A & \to & \neg Sb \,\&\, C \\ C & \to & aC \mid Cb \mid \varepsilon \end{array}$$

generates the language $\{\, a^m b^n \mid m \leqslant n\}$.

The symbol C clearly defines a^*b^*. For each $j \geqslant 0$, the string b^j is always in $\overline{aA}$, regardless of the value of a, and therefore $b^* \subseteq L_G(S)$. For each $i \geqslant 1$, the string a^i is in $L_G(S)$ if and only if a^{i-1} is not in $L_G(A)$, which is false; therefore, $a^+ \cap L_G(S) = \varnothing$. Consider a string $a^i b^j$, with $i, j \geqslant 1$; according to the rules of the grammar,

$$a^i b^j \in L_G(S) \Leftrightarrow a^{i-1} b^j \notin L_G(A) \Leftrightarrow a^{i-1} b^{j-1} \in L_G(S),$$

and thus the membership of such a string in $L_G(S)$ is ultimately determined by whether it is reduced to a string from b^* or from a^+.

Though such a common-sense interpretation of Boolean grammars is clear for "reasonable" grammars, the use of negation can, in general, lead to logical contradictions (consider the grammar $S \to \neg S$), and for that reason giving a mathematically sound formal definition of Boolean grammars is far from being trivial. To be more precise, the dependence of statements of the form "string w has property A" becomes *non-monotone*, where the discovery of the fact that one statement is true may imply that another statement is false. Thus, a correct assignment of truth-values to statements is a solution of a certain infinite system of equations with Boolean unknowns. More natural and convenient formalizations are given by representing a grammar as a system of language equations, so that a particular distinguished solution of this system defines the language generated by a grammar.

There are two known definitions of Boolean grammars by equations. One of them uses equations with standard formal languages as unknowns, and avoids the resulting difficulties by imposing a restriction upon solutions of those equations. The other definition expands the model towards languages over three-valued logic and assigns a meaning to any grammar.

3.4.2 Definition by language equations

According to the simplest definition, a grammar is represented by a system of language equations defined analogously to the conjunctive case, with the negation represented by complementation.

This system is required to have a unique solution, and grammars with no solutions or multiple solutions are considered ill-formed. However, this does not yet guarantee that the membership of a string in the language depends only on the membership of shorter strings, which is essential for grammars to represent inductive definitions. Consider the following grammar, along with the associated system of language equations:

$$\begin{array}{lcl} S & \to & \neg S \,\&\, aA \\ A & \to & A \end{array} \qquad \begin{cases} S = \overline{S} \cap \{a\}A \\ A = A \end{cases}$$

The system has a unique solution $S = A = \varnothing$: indeed, supposing that there is a string $w \in A$, a contradiction of the form “*$aw \in S$ if and only if $aw \notin S$*” is obtained. Thus, in order to determine that $w \notin A$, one has to consider the string aw, which contradicts the principle of inductive definition. Furthermore, there is a theoretical result, that every recursive language is represented by a unique solution of a system of language equations associated to some Boolean grammar [**?**].

However, once an extra restriction is imposed upon these equations, a feasible definition of Boolean grammars can be obtained:

Definition 3.5 (Okhotin [4]). *Let $G = (\Sigma, N, R, S)$ be a Boolean grammar, and define the associated system of language equations*

$$A = \bigcup_{\substack{A \to \alpha_1 \& \ldots \& \alpha_m \& \\ \& \neg\beta_1 \& \ldots \& \neg\beta_n \in R}} \left[\bigcap_{i=1}^{m} \alpha_i \cap \bigcap_{j=1}^{n} \overline{\beta_j} \right] \tag{3.6}$$

Assume that for every integer $\ell \geqslant 0$ there exists a unique vector of languages $(\ldots, L_A, \ldots)_{A \in N}$ with $L_A \subseteq \Sigma^{\leqslant \ell}$, such that a substitution of L_A for A, for each $A \in N$, turns every equation (3.6) 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 $L(G) = L_G(S)$.*

Returning to the above grammar with the rules $S \to \neg S \,\&\, aA$ and $A \to A$, consider any number $\ell \geqslant 0$. Then the associated system (3.6) has two solutions modulo $\Sigma^{\leqslant \ell}$, namely, $(S = \varnothing, A = \varnothing)$ and $(S = \varnothing, A = \{a^\ell\})$. Therefore, the grammar is deemed invalid according to Definition 3.5.

Consider the Boolean grammar in Example 3.7. The corresponding system of language equations is

$$\begin{cases} S = \overline{AB} \cap \overline{BA} \cap C \\ A = XAX \cup \{a\} \\ B = XBX \cup \{b\} \\ C = XXC \cup \{\varepsilon\} \\ X = \{a\} \cup \{b\} \end{cases}$$

and the following assignment of languages to variables is its unique solution:

$$\begin{aligned} S &= \{\, ww \mid w \in \{a,b\}^* \}, \\ A &= \{\, uav \mid u, v \in \{a,b\}^*, |u| = |v| \}, \\ B &= \{\, ubv \mid u, v \in \{a,b\}^*, |u| = |v| \}, \\ C &= \{aa, ab, ba, bb\}^*, \\ X &= \{a, b\}. \end{aligned}$$

Furthermore, its solution modulo every $\Sigma^{\leqslant \ell}$ with $\ell \geqslant 0$ is unique, and hence $L(G)$ is well-defined as $\{\, ww \mid w \in \{a, b\}^* \}$.

3.4.3 Parse trees

Whenever a Boolean grammar generates a string, it defines one or more *parse trees.* These are, strictly speaking, finite acyclic graphs rather than trees, and they represent parses of a string according to positive conjuncts in the rules. A *parse tree* of a string $w = a_1 \ldots a_{|w|}$ from a nonterminal A has $|w|$ ordered leaves labelled by $a_1, \ldots, a_{|w|}$, and the rest of the vertices are labelled by rules from P. Each internal vertex of the tree corresponds to a substring $a_{i+1} \ldots a_j$, and if it is labelled by a rule

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_m \,\&\, \neg\beta_1 \,\&\, \ldots \,\&\, \neg\beta_n,$$

then it has exactly $|\alpha_1| + \ldots + |\alpha_m|$ sons corresponding to the symbols in the positive conjuncts, and for each nonterminal in each α_t, the corresponding son is labelled with some rule for that nonterminal, while for each terminal $a \in \Sigma$ in each α_t, the son is a leaf labelled with a. Furthermore, the descendants of the sons from each conjunct α_t encompass the same terminal string as their father, and in this way the tree represents multiple parses of the same substring.

Negative conjuncts are not represented in the tree, and thus a parse tree does not constitute a complete proof that the string is in $L(A)$. However, for some grammars, in which the negation is used judiciously, these trees illustrate some syntactic structures.

For each substring of w and for each $A \in N$, it is sufficient to have at most one subtree representing a parse of w from A, and hence the entire tree needs to have at most $|N| \cdot \frac{1}{2}|w|(|w|+1) + |w|$ nodes.

Bibliography

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

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

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

[3] A. Okhotin, "Conjunctive grammars and systems of language equations", *Programming and Computer Software*, 28:5 (2002), 243–249.

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

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

[6] A. Szabari, *Alternujúce zásobníkové automaty* (Alternating Pushdown Automata), in Slovak, diploma work (M.Sc. thesis), University of Košice, Czechoslovakia, 1991, 45 pp.

[7] D. Wotschke, "The Boolean closures of deterministic and nondeterministic context-free languages", In: W. Brauer (Ed.), *Gesellschaft für Informatik e. V., 3. Jahrestagung 1973*, LNCS 1, 113–121.

Index