Alexander Okhotin

Nine open problems on conjunctive and Boolean grammars

Turku Centre *for* Computer Science

TUCS Technical Report
No 794, November 2006

Nine open problems on conjunctive and Boolean grammars

Alexander Okhotin

Academy of Finland, *and*
Department of Mathematics, University of Turku, Turku, Finland, *and*
Turku Centre for Computer Science, *and*
Department of Romance Philology, Rovira i Virgili University, Tarragona, Spain
alexander.okhotin@utu.fi

TUCS Technical Report

No 794, November 2006

Abstract

Conjunctive grammars are context-free grammars with an explicit conjunction operation in the formalism of rules; Boolean grammars are further equipped with an explicit negation. The paper briefly reviews these grammars and proposes 9 most interesting and important research problems for them. An award of $240 Canadian is offered for the first correct solution of each of these problems.

Keywords: Conjunctive grammars, Boolean grammars, language equations, parsing, complexity, automata

TUCS Laboratory
Discrete Mathematics for Information Technology

1 Introduction

A context-free grammar is the most obvious formalism for specifying syntax, and actually one of the first formal objects encountered by mankind: Pāṇini's treatise on Sanskrit written around 5th century B. C. used context-free productions to specify parts of the grammar. In the mid-20th century context-free grammars were rediscovered by Chomsky [1] and by the Algol 60 committee [16], and their mathematical study began, leading to what we know as formal language theory.

The formalism of context-free grammars contains one logical operation, the disjunction, which is represented by multiple rules for a single nonterminal. This makes it easy to express the set of all strings that satisfy one of the several given syntactical conditions: two rules $A \rightarrow \alpha$ and $A \rightarrow \alpha'$ essentially say that whatever satisfies the condition α *or* the condition α' therefore satisfies the condition represented by the nonterminal A. However, any other logical operations, conjunction and negation in particular, are not expressible using context-free grammars: if we want to specify all strings that satisfy a condition α and *at the same time* another condition β, this might be impossible (as the intersection of two context-free languages is not necessarily context-free), or the grammar for such an intersection might be immensely large and completely unlike the original grammar.

These logical operations are an essential part of any formal reasoning, and being able to express them is important for any mathematical model of syntax. Extending context-free grammars to support these operations can be regarded as *completing* the incomplete definition of context-free grammars. Some early attempts to do this were undertaken by Latta and Wall [12] and by Heilbrunner and Schmitz [8], who proposed formalisms for specifying Boolean combinations of context-free languages. Latta and Wall, in particular, argued for the relevance of their formalism for linguistics. However, the use of conjunction and negation in these grammars was heavily restricted, and one still could not use them as freely as the disjunction.

Conjunctive grammars [17] are a natural extension of context-free grammars, which partially fills this gap: the conjunction of two syntactical conditions can be directly expressed in the form of a rule

$$A \rightarrow \alpha \& \beta$$

Boolean grammars [27] further extend conjunctive grammars by allowing an explicit negation, that is, finally, every operation of Boolean logic is directly expressible in their formalism. For instance, the set of strings that satisfy a condition α and at the same time do not satisfy a condition β can be written as a rule

$$A \rightarrow \alpha \& \neg \beta$$

The language generated by a conjunctive grammar can be formally defined using derivations [17] that generalize context-free derivations, or, equiva-

lently, by means of language equations [21]. The semantics of Boolean grammars has originally been defined using language equations [24, 27], while some alternative approaches based upon ideas from logical programming have recently been proposed by Wrona [39] and by Kountouriotis et al. [10].

Boolean grammars can specify many abstract non-context-free languages, such as $\{a^n b^n c^n \mid n \geqslant 0\}$ [17], $\{ww \mid w \in \{a,b\}^*\}$ and $\{a^{2^n} \mid n \geqslant 0\}$ [27], the latter being outside of the Boolean closure of the context-free languages. Another evidence of their expressive power is given by a fairly compact grammar for the set of well-formed programs in a simple model programming language [30], which became the first specification of any programming language by a formal grammar from a computationally feasible class.

Though conjunctive and Boolean grammars have a greater expressive power than the context-free grammars, this increase in power does not lead to a complexity blowup: the languages generated by Boolean grammars are contained in $DTIME(n^3) \cap DSPACE(n)$. Practical parsing techniques for context-free grammars, such as the *recursive descent* and the *generalized LR*, have been extended first to conjunctive and then to Boolean grammars [18, 19, 23, 28, 32], and the algorithms have been implemented in an ongoing research-oriented parser generator project [20]. Recently Megacz [15] started the development of a practically oriented parser generator for Boolean grammars.

Context-free grammars can be regarded as a particular case of Boolean grammars, in which the set of allowed Boolean operations is restricted to disjunction only. The studies conducted so far show that the most essential practical properties of this particular case carry on to the general case. Some theoretical properties of conjunctive and Boolean grammars were established in the first papers [17, 27]. Later a characterization of a subcase of conjunctive grammars by cellular automata was obtained [25], as well as a characterization of Boolean grammars by derivations of a special kind [29]. However, most of the theoretical problems on these grammars are still open, and solving these problems will significantly contribute to an emerging theory of Boolean grammars. The extensive theory made for the restricted case of context-free grammars deserves to be developed in the general case!

This paper gives a brief overview of these grammars and states the most important open problems in the area. An award is offered for the first solution of any of these problems.

2 Definitions

2.1 Conjunctive grammars

Conjunctive grammars are context-free grammars with an explicit conjunction operation that has semantics of intersection of languages. In addition to

the implicit disjunction represented by multiple rules for a single nonterminal, which is the only logical operation expressible in context-free grammars, conjunctive grammars contain conjunction in the formalism of rules.

Definition 1. *A conjunctive grammar [17] is defined as a quadruple* $G = (\Sigma, N, P, S)$*, in which*

- Σ *and* N *are disjoint finite nonempty sets of terminal and nonterminal symbols respectively;*
- P *is a finite set of grammar rules, each of the form*

$$A \to \alpha_1 \& \ldots \& \alpha_n \quad (\text{where } A \in N,\ n \geqslant 1 \text{ and } \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*)$$

- $S \in N$ *is a nonterminal designated as the* start symbol.

For every rule $A \to \alpha_1 \& \ldots \& \alpha_n \in P$ *and for every* i $(1 \leqslant i \leqslant n)$*, an object* $A \to \alpha_i$ *is called a conjunct.*

A collection of rules for a single nonterminal can be written using the common notation

$$A \to \alpha_{11} \& \ldots \& \alpha_{1n_1} \mid \ldots \mid \alpha_{m1} \& \ldots \& \alpha_{mn_m},$$

in which the vertical line is, in essence, disjunction.

Similarly to the context-free case, a conjunctive grammar is called *linear conjunctive* if every rule it contains is either of the form $A \to u_1 B_1 v_1 \& \ldots \& u_n B_n v_n$, where $n \geqslant 1$, $u_i, v_i \in \Sigma^*$ and $B_i \in N$, or of the form $A \to w$, where $w \in \Sigma^*$.

The semantics of conjunctive grammars is defined using derivation, generally in the same way as in the context-free case. The only difference is in the objects being transformed: while context-free derivations operate with strings over $\Sigma \cup N$, which are terms over concatenation, a derivations in conjunctive grammars use terms over concatenation and conjunction.

Let us denote such terms as strings over an extended alphabet $\Sigma \cup N \cup \{\text{“(”}, \text{“\&”}, \text{“)”}\}$, assuming that none of the three special symbols is in $\Sigma \cup N$. The set of valid string representations is defined inductively as follows:

1. ε is a term.
2. Every symbol from $\Sigma \cup N$ is a term.
3. If $\mathcal{A}$ and $\mathcal{B}$ are nonempty terms, then $\mathcal{A}\mathcal{B}$ is a term.
4. If $\mathcal{A}_1, \ldots, \mathcal{A}_n$ $(n \geqslant 1)$ are terms, then $(\mathcal{A}_1 \& \ldots \& \mathcal{A}_n)$ is a term.

Definition 2. *Given a grammar* G*, define the relation* $\overset{G}{\Longrightarrow}$ *of immediate derivability on the set of terms:*

1. *A nonterminal can be rewritten by the body of some rule enclosed in parentheses, that is, for all* $s_1, s_2 \in (\Sigma \cup N \cup \{\text{“(”}, \text{“\&”}, \text{“)”}\})^*$ *and for all* $A \in N$*, if* $s_1 A s_2$ *is a term, then, for every rule* $A \to \alpha_1 \& \ldots \& \alpha_n \in P$*,*

$$s_1 A s_2 \overset{G}{\Longrightarrow} s_1(\alpha_1 \& \ldots \& \alpha_n) s_2 \tag{1}$$

2. *A conjunction of several identical terminal strings enclosed in parentheses can be replaced by one such string without the parentheses, that is, for all* $s_1, s_2 \in (\Sigma \cup N \cup \{\text{“(”}, \text{“\&”}, \text{“)”}\})^*$*, for all* $w \in \Sigma^*$ *and for all* $n \geqslant 1$*, if* $s_1(\underbrace{w\& \ldots \&w}_{n}) s_2$ *is a term, then*

$$s_1(\underbrace{w\& \ldots \&w}_{n}) s_2 \overset{G}{\Longrightarrow} s_1 w s_2 \tag{2}$$

Let $\overset{G}{\Longrightarrow}{}^*$ *be the reflexive and transitive closure of* $\overset{G}{\Longrightarrow}$ *.*

The language generated by a term $\mathcal{A}$ *is the set of all strings over* Σ *derivable from its start symbol in a finite number of steps:*

$$L_G(\alpha) = \{w \mid w \in \Sigma^*,\ \mathcal{A} \overset{G}{\Longrightarrow}{}^* w\} \tag{3}$$

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

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

Let us construct a conjunctive grammar for the most common example of a non-context-free language.

Example 1 ([17])**.** *The following conjunctive grammar generates the language* $\{a^n b^n c^n \mid n \geqslant 0\}$*:*

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

The grammar is based upon the representation of this language as an intersection of two context-free languages:

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

According to this grammar, the string abc can be derived in the following way:

$S \Longrightarrow (AB\&DC) \Longrightarrow ((aA)B\&DC) \Longrightarrow ((aA)(bBc)\&DC) \Longrightarrow$
$((aA)(bBc)\&(aDb)C) \Longrightarrow ((aA)(bBc)\&(aDb)(cC)) \Longrightarrow^4$
$((a())(b()c)\&(a()b)(c())) \Longrightarrow^4 ((a)(bc)\&(ab)(c)) \Longrightarrow^4 (abc\&abc) \Longrightarrow abc$

In essence, here two context-free derivations are done in parallel, and the same string has to be derived from AB and from DC in order to do the last step of the derivation.

An important property of conjunctive grammars is that every derivation can be represented in the form of a tree with shared leaves, which generalizes context-free parse trees. The tree corresponding to the above derivation is given in Figure 1, and one can clearly see how it combines two interpretations of the same string according to two conjuncts of the rule for S. A formal definition of such trees can be found in the literature [17, 19].

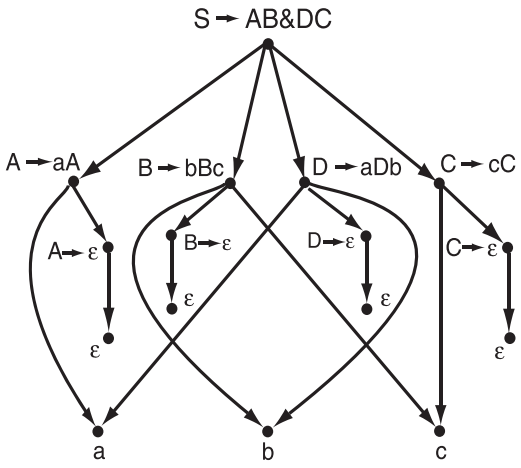

Figure 1: Parse tree of $abc \in L(G)$ according to the grammar in Example 1.

Another common example of a non-context-free language, $\{wcw \mid w \in \{a,b\}^*\}$, forms a more interesting case, because, as proved by Wotschke [38], it is not expressible as a finite intersection of context-free languages. Let us give a linear conjunctive grammar for this language and explain how it works.

Example 2 ([17])**.** *The following conjunctive grammar generates the language* $\{wcw \mid w \in \{a,b\}^*\}$*:*

$$
\begin{aligned}
S &\to C\&D\\
C &\to aCa \mid aCb \mid bCa \mid bCb \mid c\\
D &\to aA\&aD \mid bB\&bD \mid cE\\
A &\to aAa \mid aAb \mid bAa \mid bAb \mid cEa\\
B &\to aBa \mid aBb \mid bBa \mid bBb \mid cEb\\
E &\to aE \mid bE \mid \varepsilon
\end{aligned}
$$

The nonterminal C generates $\{xcy \mid x,y \in \{a,b\}^*; |x| = |y|\}$ and thus ensures that the string consists of two equal-length parts separated by a center marker. D takes one symbol from the left and uses A or B to compare

it to the corresponding symbol at the right. At the same time, D recursively refers to itself in order to apply the same rule to the rest of the string. Formally, A generates $\{xcvay \mid x,v,y \in \{a,b\}^*,\ |x| = |y|\}$, B generates $\{xcvby \mid x,v,y \in \{a,b\}^*,\ |x| = |y|\}$ and therefore D produces $\{uczu \mid u,z \in \{a,b\}^*\}$ (the last result may be obtained by a straightforward induction on the length of the string). Finally,

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

Let us construct a derivation of the string *abcab* and thus formally demonstrate that it is generated by the given grammar:

$S \Longrightarrow (C\&D) \Longrightarrow ((aCb)\&D) \Longrightarrow ((a(bCa)b)\&D) \Longrightarrow ((a(b(c)a)b)\&D) \Longrightarrow$
$((a(bca)b)\&D) \Longrightarrow ((abcab)\&D) \Longrightarrow (abcab\&D) \Longrightarrow (abcab\&(aA\&aD)) \Longrightarrow$
$(abcab\&(a(bAb)\&aD)) \Longrightarrow (abcab\&(a(b(cEa)b)\&aD)) \Longrightarrow$
$(abcab\&(a(b(c()a)b)\&aD)) \Longrightarrow (abcab\&(a(b(ca)b)\&aD)) \Longrightarrow$
$(abcab\&(a(bcab)\&aD)) \Longrightarrow (abcab\&(abcab\&aD)) \Longrightarrow$
$(abcab\&(abcab\&a(bB\&bD))) \Longrightarrow (abcab\&(abcab\&a(b(cEb)\&bD))) \Longrightarrow$
$(abcab\&(abcab\&a(b(c(aE)b)\&bD))) \Longrightarrow$
$(abcab\&(abcab\&a(b(c(a())b)\&bD))) \Longrightarrow$
$(abcab\&(abcab\&a(b(c(a)b)\&bD))) \Longrightarrow (abcab\&(abcab\&a(b(cab)\&bD))) \Longrightarrow$
$(abcab\&(abcab\&a(bcab\&bD))) \Longrightarrow (abcab\&(abcab\&a(bcab\&bD))) \Longrightarrow$
$(abcab\&(abcab\&a(bcab\&b(cE)))) \Longrightarrow (abcab\&(abcab\&a(bcab\&b(c(aE))))) \Longrightarrow$
$(abcab\&(abcab\&a(bcab\&b(c(a(bE)))))) \Longrightarrow$
$(abcab\&(abcab\&a(bcab\&b(c(a(b()))))))) \Longrightarrow$
$(abcab\&(abcab\&a(bcab\&b(c(a(b)))))) \Longrightarrow$
$(abcab\&(abcab\&a(bcab\&b(c(ab))))) \Longrightarrow (abcab\&(abcab\&a(bcab\&b(cab)))) \Longrightarrow$
$(abcab\&(abcab\&a(bcab\&bcab))) \Longrightarrow (abcab\&(abcab\&abcab)) \Longrightarrow$
$(abcab\&abcab) \Longrightarrow abcab$

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

Let us now consider a representation of conjunctive grammars by language equations, which generalizes the well-known characterization of the context-free grammars due to Ginsburg and Rice [5].

Definition 3. *For every conjunctive grammar $G = (\Sigma, N, P, S)$, the associated system of language equations [21] is a system of equations in variables N, in which each variable assumes a value of a language over Σ, and which contains the following equation for every variable A:*

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \in P} \bigcap_{i=1}^{m} \alpha_i \quad (\textit{for all } A \in N) \tag{4}$$

Each instance of a symbol $a \in \Sigma$ in such a system defines a constant language $\{a\}$, while each empty string denotes a constant language $\{\varepsilon\}$. A solution of such a system is a vector of languages $(\ldots, L_C, \ldots)_{C \in N}$, such that the substitution of L_C for C, for all $C \in N$, turns each equation (4) into an equality.

It is known that every such system has solutions, and among them the *least solution* with respect to componentwise inclusion, and this solution consists of exactly the languages generated by the nonterminals of the original conjunctive grammar: $(\ldots, L_G(C), \ldots)_{C \in N}$ [21].

This representation by language equations constitutes an equivalent semantics of conjunctive grammars, and it is this semantics, and not the fairly artificial derivation, that accounts for the intuitive clarity of conjunctive and context-free grammars.

2.2 Boolean grammars

Boolean grammars are context-free grammars augmented with all propositional connectives, or, in other words, conjunctive grammars with negation.

Definition 4. *A Boolean grammar [27] is defined as a quadruple $G = (\Sigma, N, P, S)$, where Σ and N are disjoint finite nonempty sets of terminal and nonterminal symbols respectively; P is a finite set of rules of the form*

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n \quad (A \in N,\ m+n \geqslant 1,\ \alpha_i, \beta_j \in (\Sigma \cup N)^*), \tag{5}$$

while $S \in N$ is the start symbol of the grammar.

For each rule (5), the objects $A \to \alpha_i$ and $A \to \neg\beta_j$ (for all i, j) are called conjuncts, positive and negative respectively. A conjunct with an unknown sign can be denoted $A \to \pm\gamma$, which means "$A \to \gamma$ or $A \to \neg\gamma$".

A Boolean grammar becomes a conjunctive grammar if negation is never used, that is, $n = 0$ for every rule (5); it degrades to a standard context-free grammar if neither negation nor conjunction are allowed, that is, $m = 1$ and $n = 0$ for all rules. As in the case of conjunctive grammars, let us adopt a short notation $A \to \varphi_1 \mid \ldots \mid \varphi_\ell$ for ℓ rules $A \to \varphi_i$ of the form (5) for a single nonterminal A.

Intuitively, a rule (5) can be read as *"if a string satisfies the syntactical conditions $\alpha_1, \ldots, \alpha_m$ and does not satisfy any of the syntactical conditions $\beta_1, \ldots, \beta_n$, then this string satisfies the condition represented by the nonterminal A".* This intuitive interpretation is not yet a formal definition, but this understanding is sufficient to construct grammars.

Example 3 (cf. Example 1)**.** *The following Boolean grammar generates the language* $\{a^m b^n c^n \mid m, n \geqslant 0, m \neq n\}$*:*

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

The rules for the nonterminals A, B, C and D are context-free, so, according to the intuitive semantics, they should generate the same languages as in Example 1. Then the propositional connectives in the rule for S specify the following combination of the conditions given by AB and DC (see Example 1):

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

Example 4. *The following Boolean grammar generates the language* $\{ww \mid w \in \{a, b\}^*\}$*:*

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

Again, according to the intuitive semantics, the nonterminals A, B, C and X should generate the appropriate context-free 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,

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

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

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

Though such a common-sense interpretation of Boolean grammars is clear for "reasonably written" grammars, the use of negation can, in general, lead to logical contradictions (consider the grammar $S \to \neg S$), and for that reason the task of defining a mathematically sound formal semantics for Boolean grammars is far from being trivial. All existing definitions of Boolean grammars [27, 39, 10] start with representing a grammar as a system of language equations with concatenation, union, intersection and complementation.

Definition 5 (cf. Definition 3)**.** *For every Boolean grammar $G = (\Sigma, N, P, S)$, the associated system of language equations is defined by analogy with the conjunctive case, with the following equations:*

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n \in P} \left[\bigcap_{i=1}^{m} \alpha_i \ \cap \ \bigcap_{j=1}^{n} \overline{\beta_j} \right] \tag{6}$$

In general, systems of language equations of the form (6) have a high expressive power and the associated undecidability results [24]. The notion of a least solution is no longer useful [27]. The class of languages represented by their unique solutions is exactly the class of recursive languages, and the way these languages are represented does not well correspond to the intuitive semantics of Boolean grammars defined above. However, different restrictions upon these equations lead to feasible semantics for Boolean grammars [27, 39]. Let us define the simplest of these restrictions:

Definition 6. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar, let (6) be the associated system of language equations. Suppose that for every finite substring-closed language $M \subset \Sigma^*$ (that is, for every $w \in M$ all substrings of w are also in M) there exists a unique vector of languages $(\ldots, L_C, \ldots)_{C \in N}$ ($L_C \subseteq M$), such that a substitution of L_C for C, for each $C \in N$, turns every equation (6) into an equality modulo intersection with M. Then G complies to the semantics of a strongly unique solution, and, for every $A \in N$, the language $L_G(A)$ can be defined as L_A from the unique solution of this system. The language generated by the grammar is $L(G) = L_G(S)$.*

A simple example of a grammar deemed invalid according to Definition 6 is the grammar $\{S \to \neg S \& aA, A \to A\}$. Here the associated system of language equations $\{S = \overline{S} \cap aA, A = A\}$ has a unique solution $S = A = \varnothing$: supposing $w \in A$, a contradiction of the form "*$aw \in S$ if and only if $aw \notin S$*" is obtained. However, this system has two solutions modulo every language $\{\varepsilon, a, \ldots, a^n\}$, namely, $(S = \varnothing, A = \varnothing)$ and $(S = \varnothing, A = \{a^n\})$. This makes it invalid.

The simplest example of a grammar that does not meet the condition of this definition is $S \to S$. However, it is valid according to alternative semantics for Boolean grammars, see Wrona [39], Kountouriotis et al. [10] and the author [27, 32]. The grammar $S \to \neg S$ is invalid according to

all known semantics. But it should be noted that, with the exception of Wrona's semantics [39], these semantics disagree only on extremal examples and ultimately define the same family of languages. It is also possible to extend the applicability of conjunctive parse trees to Boolean grammars.

Returning to Example 4, 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: $S = \{ww \mid w \in \{a,b\}^*\}$, $A = \{uav \mid u,v \in \{a,b\}^*, |u| = |v|\}$, $B = \{ubv \mid u,v \in \{a,b\}^*, |u| = |v|\}$, $C = \{aa, ab, ba, bb\}^*$, $D = \{a,b\}$. It is not hard to verify that the solution modulo every finite language, in the sense of the above definition, is unique, and hence $L(G) = \{ww \mid w \in \{a,b\}^*\}$.

Despite the increased descriptive power, the theoretical upper bound for the parsing complexity for Boolean grammars is still $O(n^3)$ [27], the same as in the context-free case, which is obtained by an extension of the Cocke–Kasami-Younger algorithm. This algorithm uses cubic time for every language generated by a Boolean grammar and on every input, and it requires that the grammar is transformed to the following extension of Chomsky normal form [27]:

Definition 7. *A Boolean grammar $G = (\Sigma, N, P, S)$ is in the binary normal form if every rule in P is of the form*

$$A \to B_1C_1\& \ldots \&B_mC_m\&\neg D_1E_1\& \ldots \&\neg D_nE_n\&\neg\varepsilon \quad (m \geqslant 1, n \geqslant 0)$$
$$A \to a$$
$$S \to \varepsilon \quad \text{(only if } S \text{ does not appear in right-hand sides of rules)}$$

Other algorithms that do not require grammar transformation have been proposed [28, 32, 39].

2.3 Linear conjunctive grammars and trellis automata

The family of languages defined by linear conjunctive grammars has actually been known for almost thirty years before these grammars were introduced [25]: this is the family defined by one of the simplest types of cellular automata. These are one-way real-time cellular automata, also known as *trellis automata*, studied by Dyer [4], Culik, Gruska and Salomaa [3], Ibarra and Kim [9], and others. Let us explain this concept following Culik, Gruska and Salomaa [3], who proposed it as a model of parallel computation in some electronic circuits.

A trellis automaton, defined as a quintuple $(\Sigma, Q, I, \delta, F)$, processes an input string of length n using a uniform array of $n(n+1)/2$ processor nodes, as in Figure 2. Each processor computes a value from a fixed finite set Q. The processors in the bottom row obtain their values directly from the input symbols using a function $I : \Sigma \to Q$. The rest of the processors compute the function $\delta : Q \times Q \to Q$ of the values in their predecessors. The string is accepted if and only if the value computed by the topmost processor belongs to the set of accepting states $F \subseteq Q$.

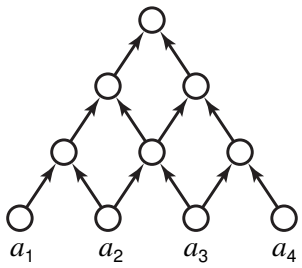

Figure 2: Computation done by a trellis automaton.

Evidently, trellis automata are one of the simplest computational models one can imagine, and they were proved to be computationally equivalent to conjunctive grammars:

Theorem 1 ([25])**.** *A language $L \subseteq \Sigma^+$ is generated by a linear conjunctive grammar if and only if L is recognized by a trellis automaton.*

One can attain complete equivalence by extending the definition of a trellis automaton by a single bit that determines whether ε is recognized or not. The proof of Theorem 1 is by an effective construction of a trellis automaton out of a grammar, and vice versa. Furthermore, the following refinement of the automaton-to-grammar construction is known:

Theorem 2 ([26])**.** *For every trellis automaton M there exists and can be effectively constructed a two-nonterminal linear conjunctive grammar generating the same language.*

The entire machinery of a trellis automaton is simulated using just one nonterminal, while the other decodes $L(M)$ from the language of the first nonterminal.

One can define linear Boolean grammars by the same restriction as for linear conjunctive grammars. These grammars actually define the same class of languages as linear conjunctive grammars and trellis automata [27].

2.4 Comparison of the families

Three families of languages have been considered. One of them, the linear conjunctive languages, turned out to be known, while the languages gener-

ated by conjunctive grammars and by Boolean grammars are new. Let us summarize the relation between these families shown in Figure 3.

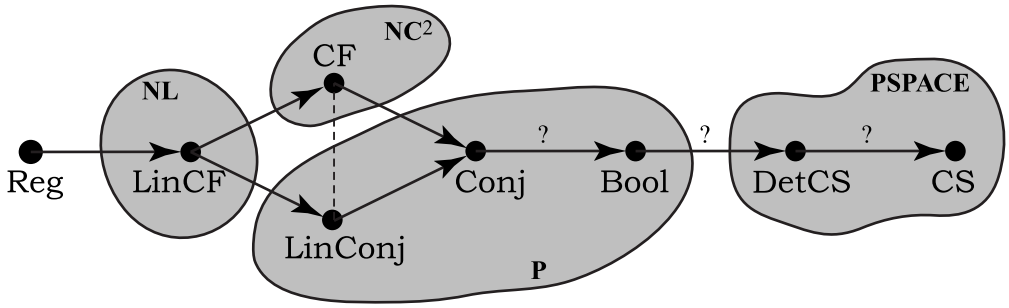

Figure 3: Hierarchy of languages.

Obviously, both the context-free languages (CF) and the linear conjunctive languages (LinConj) contain the linear context-free languages (LinCF). Both inclusions are strict, which is witnessed, for instance, by the languages $\{a^m b^{m+n} a^n \mid m, n \geqslant 0\}$ and $\{a^n b^n c^n \mid n \geqslant 0\}$. It is known from Terrier [36] that CF and LinConj are incomparable: there exists a linear context-free language L_T, such that L_T^2 (obviously a context-free language) is not recognized by any trellis automaton. The language L_T^2 also certifies proper containment of LinConj in the conjunctive languages (Conj).

	Reg	LinCF	CF	LinConj	Conj	Bool	DetCS	CS
Closure properties								
$\cup$	+	+	+	+	+	+	+	+
$\cap$	+	−	−	+	+	+	+	+
$\sim$	+	−	−	+	?	+	+	+
$\cdot$	+	−	+	−	+	+	+	+
$*$	+	−	+	−	+	+	+	+
h	+	+	+	−	−	−	−	−
h^{-1}	+	+	+	+	+	?	+	+
Decision problems								
Membership	+	+	+	+	+	+	+	+
Emptiness	+	+	+	−	−	−	−	−
Universality	+	−	−	−	−	−	−	−
Equivalence	+	−	−	−	−	−	−	−

Table 1: Closure properties and decidability of decision problems.

Since conjunctive grammars are a special case of Boolean grammars, the containment of one language family in the other is obvious. However, it is not known whether the inclusion is strict. Also, while it is known that the languages generated by Boolean grammars (Bool) are contained in DSPACE(n), or, in other words, they are all deterministic context-sensitive (DetCS) [27],

there is no proof of the strictness of this inclusion. Finally, it is a long-standing open problem whether DSPACE(n) is smaller than NSPACE(n), the latter also known as the family of context-sensitive languages (CS).

Decidability of most common decision problems for the families mentioned above, as well as the closure of the resulting families of languages under standard operations, are compared in Table 1.

3 The problems

3.1 Limitations of Boolean grammars

The limitations of the expressive power of the context-free grammars have been investigated quite well. Besides the complexity upper bounds, there exist direct techniques of proving non-context-freeness of particular languages, such as the pumping lemma and its variants, as well as Parikh's theorem, which show that some computationally very easy languages cannot be generated by any context-free grammar. Simple examples of non-context-free languages include $\{a^n b^n c^n \mid n \geqslant 0\}$, $\{wcw \mid w \in \{a,b\}^*\}$, $\{a^n b^{2^n} \mid n \geqslant 0\}$ and $\{a^{2^n} \mid n \geqslant 0\}$.

In contrast, no methods of proving nonrepresentability of languages by Boolean grammars are currently known, and this can be regarded as the most important theoretical problem on these grammars. Note that even for conjunctive grammars no techniques for proving non-existence of grammars for particular languages are known, and only for linear conjunctive grammars one can use fairly sophisticated counting arguments developed for the trellis automaton representation [36].

Of course, the known upper bounds on the complexity of parsing for Boolean grammars (deterministic cubic time or, using a different algorithm, deterministic linear space [27]) already imply that computationally harder languages are beyond their scope (by the time and the space hierarchies). The question is, whether any computationally easy languages cannot be specified by Boolean grammars, and how can one generally prove statements of this kind? Let us formally state this question as follows:

Problem 1. *Are there any languages recognized by deterministic linear bounded automata working in time $O(n^2)$ that cannot be specified by Boolean grammars?*

Following is quite an interesting candidate:

$$\{u_1 \dots u_n \mid \text{for every } i, \text{ either } u_i \in a^*c, \text{ or } \exists j,k\colon\ u_i = b^k c \text{ and } u_j = a^k c\}$$

This is an abstract language, which represents declaration of identifiers *before or after* their use; substrings of the form $a^k c$ represent declarations, while every substring $b^k c$ is a reference to a declaration of the form $a^k c$.

It is known that Boolean grammars can specify a related language, in which declarations must precede references, that is, the number j is required to be always less than i [30]. However, the corresponding grammar does not generalize for the given case.

Other languages possibly not representable by Boolean grammars can be sought for in the domain of unary languages: consider $\{a^{2^{2^n}} \mid n \geqslant 0\}$ or $\{a^{n^2} | n \geqslant 0\}$. Actually, some related nonrepresentability results are known for a certain subfamily of Boolean grammars [33]. The case of unary languages is the subject of the second problem.

3.2 Conjunctive grammars over a one-letter alphabet

It is well-known that context-free grammars over a one-letter alphabet generate only regular languages [5]. On the other hand, Boolean grammars can generate some nonregular unary languages, as shown by the following example:

Example 5. *The following Boolean grammar generates the language* $\{a^{2^n} \mid n \geqslant 0\}$*:*

$$\begin{aligned}
&S \to A\&\neg aA \mid aB\&\neg B \mid aC\&\neg C\\
&A \to aBB\\
&B \to \neg CC\\
&C \to \neg DD\\
&D \to \neg A
\end{aligned}$$

The rules for A, B, C and D represent the following language equation:

$$X = a\overline{\overline{\overline{X}^2}}^2$$

This equation was studied by Leiss [13], and its unique solution was found to be $\{a^n \mid \exists k \geqslant 0 : 2^{3k} \leqslant n < 2^{3k+2}\}$. Therefore, the corresponding nonterminals of the above grammar generate the following languages:

$$\begin{aligned}
L(A) &= \{a^n \mid \exists k \geqslant 0 :\ 2^{3k} \leqslant n < 2^{3k+2}\},\\
L(B) &= \{a^n \mid \exists k \geqslant 0 :\ 2^{3k} - 1 \leqslant n < 2^{3k+1}\},\\
L(C) &= \{a^n \mid \exists k \geqslant 0 :\ 2^{3k+1} \leqslant n < 2^{3k+2}\}, \text{ and}\\
L(D) &= \{a^n \mid \exists k \geqslant 0 :\ 2^{3k+2} \leqslant n < 2^{3k+3}\} \cup \{\varepsilon\}.
\end{aligned}$$

Then it is easy to verify that

$$L(S) = \underbrace{(L(A) \cap \overline{aL(A)})}_{\{a^{2^{3k}} \mid k \geqslant 0\}} \cup \underbrace{(aL(B) \cap \overline{L(B)})}_{\{a^{2^{3k+1}} \mid k \geqslant 0\}} \cup \underbrace{(aL(C) \cap \overline{L(C)})}_{\{a^{2^{3k+2}} \mid k \geqslant 0\}} = \{a^{2^n} \mid n \geqslant 0\}$$

Conjunctive grammars stand in the middle between context-free and Boolean grammars, and their expressive power in the case of a unary alphabet remains unknown. Do they generate only regular languages, like context-free grammars, or can they generate some nonregular language, like Boolean grammars? The Boolean grammar in Example 5 essentially uses negation, and it looks that there is no obvious way to replicate these constructions using conjunctive grammars. On the other hand, the regularity proof does not generalize from the context-free case, since it relies upon the pumping lemma.

Problem 2. *Do conjunctive grammars over a one-letter alphabet generate only regular languages?*

If they can generate any nonregular language, this would be a surprise. On the other hand, if only regular unary languages are generated, then some new idea would be needed for the proof. Perhaps one could first establish a pumping lemma for conjunctive grammars over a one-letter alphabet, but even this seems to be not an easy task.

3.3 Time complexity

The membership of a given string in a context-free language can be tested in time $\Theta(n^3)$ using the well-known Cocke–Kasami–Younger algorithm. At the same time, some asymptotically more efficient methods of context-free recognition are known: Valiant [37] reduced context-free membership problem to matrix multiplication, which allowed him to apply Strassen's [35] fast matrix multiplication algorithm to obtain a context-free recognizer working in time $O(n^{2.807})$. Using an asymptotically better matrix multiplication method due to Coppersmith and Winograd [2], the complexity of Valiant's recognizer can be improved to $O(n^{2.376})$.

However, already for conjunctive grammars there seems to be no way to reduce the membership problem to matrix multiplication. Therefore, the $DTIME(n^3)$ upper bound for the complexity given by the extension of the Cocke–Kasami–Younger algorithm remains the best known, and to improve this bound one would have to invent an entirely new algorithm. The question is, can this be done?

Problem 3. *Are the languages generated by Boolean grammars contained in $DTIME(n^{3-\varepsilon})$ for any $\varepsilon > 0$?*

3.4 Space complexity

The Cocke–Kasami–Younger algorithm for context-free grammars uses space $\Theta(n^2)$, and its generalization for Boolean grammars fits its data in the same amount of memory [27]. In both cases this is the best known upper bound for practically useful algorithms applicable to grammars of the general form. In the context-free case it was established by Lewis, Stearns and Hartmanis [14] that it is possible to trade time for space and use as little as $O(\log^2 n)$ memory.

Adapting the method of Lewis, Stearns and Hartmanis to Boolean grammars, and even to linear conjunctive grammars, does not seem to be possible, since $O(\log^2 n)$ space complexity is achieved by a binary search in a context-free derivation, while the generation of a string by a conjunctive or a Boolean grammar is not known to have such a structurally simple representation. Also, consider that already linear conjunctive grammars can specify a P-complete language [9], which makes polylogarithmic-space recognition not very likely.

The best known upper bound for the space complexity of Boolean grammars is $O(n)$ [27]. This suggests the task of improving this result, if that is possible:

Problem 4. *Are the languages generated by Boolean grammars contained in* $DSPACE(n^{1-\varepsilon})$ *for any* $\varepsilon > 0$*?*

If this were proved for any $\varepsilon > 0$, this would, in particular, separate Boolean grammars from the context-sensitive grammars. On the other hand, if the converse is the case, proving that would involve a lower bound technique interesting in itself.

As a simpler problem, consider separating the family generated by conjunctive grammars from $NSPACE(n)$.

3.5 Greibach normal form

A context-free grammar is said to be in Greibach normal form if every rule is either $A \to \varepsilon$, or $A \to a\alpha$ for some $a \in \Sigma$ and $\alpha \in (\Sigma \cup N)^*$. It is known from Greibach [6] that every context-free grammar can be transformed to an equivalent grammar with rules of this form.

A generalization for Boolean grammars will have rules of the form

$$A \to a\alpha_1 \& \ldots \& a\alpha_m \& \neg a\beta_1 \& \ldots \& \neg a\beta_n, \tag{7}$$

where $a \in \Sigma$, $m + n \geqslant 1$ and $\alpha_i, \beta_i \in (\Sigma \cup N)^*$. However, it is not known whether the family of languages generated by Boolean grammars in Greibach normal form is the same as the entire family generated by Boolean grammars. This is proposed as a research problem:

Problem 5. *Is it true that for every Boolean grammar there exists a Boolean grammar in Greibach normal form that generates the same language?*

For potentially simpler problems, consider the case of conjunctive grammars. Let us say that a conjunctive grammar is in Greibach normal form if its rules are

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

where $a \in \Sigma$, $m \geqslant 1$ and $\alpha_i \in (\Sigma \cup N)^*$. Can every conjunctive grammar be transformed to Greibach normal form?

In the same way one can define a linear conjunctive grammar in Greibach normal form, in which all rules must be of the form $A \to w$ ($w \in \Sigma^*$) or

$$A \to aB_1u_1 \& \ldots \& aB_mu_m,$$

where $a \in \Sigma$, $m \geqslant 1$ and $\alpha_i \in (\Sigma \cup N)^*$. Here it should be rather easy to prove that the linear conjunctive language $\{a^n b^{2^n} \mid n \geqslant 1\}$ [25] cannot be generated by any linear conjunctive grammar in Greibach normal form.

The question whether the language $\{a^n b^{2^n} \mid n \geqslant 1\}$ can be represented by a Boolean grammar in Greibach normal form might be a good starting point in approaching Problem 5. The answer to this question is likely negative, and a negative solution to the problem can be thus obtained.

3.6 Complementation of conjunctive grammars

The family of languages generated by Boolean grammars is closed under all Boolean operations and concatenation simply by virtue of having the corresponding operators as a part of the formalism. However, conjunctive grammars do not have an explicit negation operator, and the question whether for every conjunctive grammar G there exists a grammar for the complement of $L(G)$ is open:

Problem 6. *Is the family of conjunctive languages closed under complementation?*

If the answer is negative, a possible witness language is the one from Example 4: the language $\overline{\{ww \mid w \in \{a,b\}^*\}}$ is known to be context-free, while its complement might be nonrepresentable by conjunctive grammars.

Let us note that linear conjunctive languages *are* closed under complementation, which easily follows from their automaton representation [25], and can also be proved by an explicit construction [22].

3.7 Inherent ambiguity

Let us say that a Boolean grammar $G = (\Sigma, N, P, S)$ is *unambiguous* [34] if

1. For every nonterminal A and for every string w there exists at most one rule
$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n,$$
such that $w \in L_G(\alpha_1) \cap \ldots \cap L_G(\alpha_m) \cap \overline{L_G(\beta_1)} \cap \ldots \cap \overline{L_G(\beta_n)}$.

2. For every conjunct $A \to \pm s_1 \ldots s_\ell$ and for every string w there exists at most one factorization $w = u_1 \ldots u_\ell$, such that $u_i \in L_G(s_i)$ for all i.

A language L can be called inherently ambiguous with respect to Boolean grammars if every Boolean grammar generating it is ambiguous.

The ambiguity of the first type can be effectively eliminated by supplying every rule with an additional conjunct that expresses the condition of nonrepresentability by all other rules for this nonterminal. However, the ambiguity of the second type might be necessary to represent some languages. Is it truly necessary? This question can be stated as follows:

Problem 7. *Do there exist any inherently ambiguous languages with respect to Boolean grammars?*

There are strong reasons to expect a positive answer, that is, that there exist such languages. Note that an $O(n^2)$-time parsing algorithm for unambiguous Boolean grammars is known [34], and so if there are no inherently ambiguous languages, then the languages generated by Boolean grammars are contained in $DTIME(n^2)$ (see also Problem 3), which is an upper bound unheard of even for context-free languages of the general form. If the answer to Problem 7 is negative, this would be an extremely surprising result.

If we consider obtaining a positive answer, the languages $\{ww \mid w \in \{a, b\}^*\}$ and $\{a^{2^n} \mid n \geqslant 0\}$ are possible candidates for being inherently ambiguous. Let us show that the existing grammars given in Examples 4 and 5 are ambiguous. In Example 4, consider the string $w = aabb$ and the conjunct $S \to \neg AB$: there are two factorizations $w = a \cdot abb = aab \cdot b$, such that $a \in L(A)$, $abb \in L(B)$, $aab \in L(A)$ and $b \in L(B)$. For Example 5 it is sufficient to take $w = aa$ and $A \to aBB$: there exist factorizations $w = a \cdot \varepsilon \cdot a$ and $w = a \cdot a \cdot \varepsilon$, where $\varepsilon, a \in L(B)$. But perhaps there could still exist unambiguous Boolean grammars for these languages.

As a special case of this problem, one can consider ambiguity with respect to conjunctive grammars, and investigate whether there exist inherently ambiguous languages in this sense. Let us note that all linear conjunctive languages are unambiguous already with respect to linear conjunctive grammars, which easily follows from the form of grammars obtained from trellis automata [25].

3.8 Hierarchy of Boolean LL(k) languages

A certain subclass of Boolean grammars can be parsed using a generalized form of the well-known recursive descent method [18, 28].

Let $k \geqslant 1$ be the length of parser's lookahead, and let us say that a Boolean grammar $G = (\Sigma, N, P, S)$ is LL(k) if for every nonterminal A, for every lookahead string $u \in \Sigma^*$ ($|u| \leqslant k$), and for every sequence of conjuncts $B_0 \to \pm\eta_1 B_1 \theta_1$, ..., $B_{\ell-1} \to \pm\eta_\ell B_\ell \theta_\ell$, such that $B_0 = S$ and $B_\ell = A$, there exists at most one rule

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

such that

$$L_G((\alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n)\theta_\ell \ldots \theta_1)$$

contains u or, if $|u| = k$, any strings that begin with u.

The class of Boolean LL(k) grammars is more powerful than it seems. Following is a small and simple LL(1) Boolean grammar, which generates a P-complete language [31]; the proof of its P-completeness can be found in the cited technical report.

Example 6. *The following LL(1) Boolean grammar generates a P-complete language:*

$$\begin{aligned} S &\to B\&\neg AbS\&\neg CS \\ A &\to aA \mid \varepsilon \\ B &\to aB \mid bB \mid \varepsilon \\ C &\to aCAb \mid b \end{aligned}$$

It is known that all Boolean LL(k) grammars over a one-letter alphabet generate only regular languages [28], and therefore this subfamily is strictly weaker than the general family of Boolean grammars. The question is, do the languages generated by LL(k) Boolean grammars form an infinite hierarchy with respect to the length of the lookahead k, or does this hierarchy collapse at some point k_0?

Problem 8. *Does there exist a number $k_0 > 0$, such that, for all $k \geqslant k_0$, Boolean LL(k) grammars generate the same family of languages as Boolean LL(k_0) grammars?*

To compare, for LL(k) context-free grammars, an infinite hierarchy with respect to k was established by Kurki-Suonio [11].

One can also study the intermediate case of LL(k) conjunctive grammars, which probably have less expressive power than LL(k) Boolean grammars. In particular, no LL(k) conjunctive grammar for any P-complete language has been found.

3.9 Nonterminal complexity of Boolean grammars

It has long been known that n-nonterminal context-free languages form an infinite hierarchy [7], in the sense that for every $n \geqslant 2$ there exists a language representable using n nonterminals and not representable using $n-1$ nonterminals. Consider the families of languages generated by n-nonterminal Boolean grammars, for all $n \geqslant 1$. Do these families form an infinite hierarchy, or does this hierarchy collapse at some point?

Problem 9. *Does there exist a number $k > 0$, such that every language generated by any Boolean grammar can be generated by a k-nonterminal Boolean grammar?*

For conjunctive grammars the answer to this question is also unknown. On the other hand, for linear conjunctive grammars the hierarchy collapses, as 2 nonterminals are sufficient to describe every language from this family [26].

4 Conclusion and award announcement

Context-free grammars are a very natural and important particular case of Boolean grammars, which was discovered early in the history of mankind, and which has been studied rather extensively in the last fifty years. It turns out that several key properties of this particular case, in which only disjunction is allowed, are the same in the general case of grammars with all Boolean operations. This makes the general case of Boolean grammars both theoretically and practically important. It is time to study this general case!

To promote the research on conjunctive and Boolean grammars, I decided to offer awards for the first correct solutions of any of the nine problems proposed in this paper. The award is $240 Canadian per problem, which is equally distributed between the authors of the solution. If two papers solving the same question appear simultaneously, the award is split between them. Each author receives a handwritten certificate and an award cheque. Every lady among the authors additionally receives a flower. In most cases a solution must be published in a recognized journal or presented at a recognized conference to qualify for the award.

Acknowledgements

I am grateful to Artiom Alhazov, Michael Domaratzki, Alexander Krassovitskiy, Michal Kunc, Mark-Jan Nederhof, Kai Salomaa and Detlef Wotschke for their comments on the manuscript and for discussions related to the problems. Thanks are due to Oleg Krapilsky for producing Figure 1 in 2000 A. D.;

let me apologize for failing to acknowledge this until now, despite using the figure several times.

This work is supported by the Academy of Finland under grant 118540.

References

[1] N. Chomsky, *Syntactic Structures*, The Hague, Mouton, 1957.

[2] D. Coppersmith, S. Winograd, "Matrix multiplication via arithmetic progressions", *Journal of Symbolic Computation*, 9:3 (1990), 251–280.

[3] K. Culik II, J. Gruska, A. Salomaa, "Systolic trellis automata", I and II, *International Journal of Computer Mathematics*, 15 (1984), 195–212, and 16 (1984), 3–22.

[4] C. Dyer, "One-way bounded cellular automata", *Information and Control*, 44 (1980), 261–281.

[5] S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM*, 9 (1962), 350–371.

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

[7] J. Gruska, "Descriptional complexity of context-free languages", *Proceedings of MFCS 1973*, 71–83.

[8] S. Heilbrunner, L. Schmitz, "An efficient recognizer for the Boolean closure of context-free languages", *Theoretical Computer Science*, 80 (1991), 53–75.

[9] O. H. Ibarra, S. M. Kim, "Characterizations and computational complexity of systolic trellis automata", *Theoretical Computer Science*, 29 (1984), 123–153.

[10] V. Kountouriotis, Ch. Nomikos, P. Rondogiannis, "Well-founded semantics for Boolean grammars", *Developments in Language Theory* (DLT 2006, Santa Barbara, USA, June 26–29, 2006), LNCS 4036, 203–214.

[11] R. Kurki-Suonio, "Notes on top-down languages", *BIT*, 9 (1969), 225–238.

[12] M. Latta, R. Wall, "Intersective context-free languages", *Lenguajes Naturales y Lenguajes Formales IX*, Barcelona, 1993, 15–43.

[13] E. L. Leiss, "Unrestricted complementation in language equations over a one-letter alphabet", *Theoretical Computer Science*, 132 (1994), 71–93.

[14] P. M. Lewis III, R. E. Stearns, J. Hartmanis, "Memory bounds for recognition of context-free and context-sensitive languages", *IEEE Conference Record on Switching Circuit Theory and Logical Design*, 191–202, 1965.

[15] A. Megacz, "Scannerless Boolean parsing", LDTA 2006, *Electronic Notes in Theoretical Computer Science*, 164:2 (2006), 97–102.

[16] P. Naur (Ed.), J. W. Backus, J. H. Wegstein, A. van Wijngaarden, M. Woodger, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, "Revised report on the algorithmic language ALGOL 60", *Communications of the ACM*, 6:1 (1963), 1–17.

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

[18] A. Okhotin, "Top-down parsing of conjunctive languages", *Grammars*, 5:1 (2002), 21–40.

[19] A. Okhotin, "LR parsing for conjunctive grammars", *Grammars*, 5:2 (2002), 81–124.

[20] A. Okhotin, "Whale Calf, a parser generator for conjunctive grammars", *Implementation and Application of Automata* (CIAA 2002, Tours, France, July 3–5, 2002), LNCS 2608, 213–220.

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

[22] A. Okhotin, "On the closure properties of linear conjunctive languages", *Theoretical Computer Science*, 299 (2003), 663–685.

[23] A. Okhotin, "A recognition and parsing algorithm for arbitrary conjunctive grammars", *Theoretical Computer Science*, 302 (2003), 365–399.

[24] A. Okhotin, "Decision problems for language equations with Boolean operations", *Automata, Languages and Programming* (Proceedings of ICALP 2003, Eindhoven, The Netherlands, June 30–July 4, 2003), LNCS 2719, 239–251; journal version submitted.

[25] A. Okhotin, "On the equivalence of linear conjunctive grammars to trellis automata", *Informatique Théorique et Applications*, 38:1 (2004), 69–88; preliminary version presented at DLT 2002.

[26] A. Okhotin, "On the number of nonterminals in linear conjunctive grammars", *Theoretical Computer Science*, 320:2–3 (2004), 419–448.

[27] A. Okhotin, "Boolean grammars", *Information and Computation*, 194:1 (2004), 19–48; preliminary version presented at DLT 2003.

[28] A. Okhotin, "An extension of recursive descent parsing for Boolean grammars", Technical Report 2004–475, School of Computing, Queen's University, Kingston, Ontario, Canada; revised version submitted.

[29] A. Okhotin, "The dual of concatenation", *Theoretical Computer Science*, 345:2–3 (2005), 425–447; preliminary version presented at MFCS 2004.

[30] A. Okhotin, "On the existence of a Boolean grammar for a simple programming language", *Proceedings of AFL 2005* (May 17–20, 2005, Dobogókő, Hungary).

[31] A. Okhotin, "A simple P-complete language and its specification by language equations", TUCS TR 703, Turku Centre for Computer Science, Turku, Finland, August 2005.

[32] A. Okhotin, "Generalized LR parsing algorithm for Boolean grammars", *International Journal of Foundations of Computer Science*, 17:3 (2006), 629–664; preliminary version presented at DLT 2005.

[33] A. Okhotin, O. Yakimova, "On language equations with complementation", *Developments in Language Theory* (DLT 2006, Santa Barbara, USA, June 26–29, 2006), LNCS 4036, 420–432.

[34] A. Okhotin, "Unambiguous Boolean grammars", manuscript.

[35] V. Strassen, "Gaussian elimination is not optimal", *Numerische Mathematik*, 13 (1969), 354–356.

[36] V. Terrier, "On real-time one-way cellular array", *Theoretical Computer Science*, 141 (1995), 331–335.

[37] L. G. Valiant, "General context-free recognition in less than cubic time", *Journal of Computer and System Sciences*, 10 (1975), 308–315.

[38] 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.

[39] M. Wrona, "Stratified Boolean grammars", *Mathematical Foundations of Computer Science* (MFCS 2005, Gdansk, Poland, August 29–September 2, 2005), LNCS 3618, 801–812.

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematical Sciences

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 952-12-1805-3
ISSN 1239-1891