Theoretical Computer Science 412 (2011) 5132–5155

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Expressive power of LL(k) Boolean grammars☆

Alexander Okhotin*

Department of Mathematics, University of Turku, Turku FI–20014, Finland
Academy of Finland, Helsinki, Finland

ARTICLE INFO

Article history:
Received 2 April 2008
Received in revised form 4 May 2011
Accepted 9 May 2011
Communicated by D. Perrin

Keywords:
Boolean grammars
Conjunctive grammars
Context-free grammars
LL grammars
Language equations
Parsing
Recursive descent

ABSTRACT

The paper studies the family of *Boolean LL languages*, generated by Boolean grammars and usable with the recursive descent parsing. It is demonstrated that over a one-letter alphabet, these languages are always regular, while Boolean LL subsets of Σ^*a^* obey a certain periodicity property, which, in particular, makes the language $\{a^n b^{2^n} \mid n \geqslant 0\}$ non-representable. It is also shown that linear conjunctive LL grammars cannot generate any language of the form $L \cdot \{a, b\}$, with L non-regular, and that no languages of the form $L \cdot c^*$, with non-regular L, can be generated by any linear Boolean LL grammars. These results are used to establish a detailed hierarchy and closure properties of these and related families of formal languages.

1. Introduction

Boolean grammars [15] extend the definition of the context-free grammars by allowing explicit Boolean operations in the rules. While standard context-free grammars can combine syntactical conditions using only disjunction, effectively specified by multiple rules for a single symbol, *conjunctive grammars* [12] additionally allow conjunction, and *Boolean grammars* further support negation. At the same time, Boolean grammars preserve the most important property of the context-free grammars—that of defining the properties of strings inductively. Accordingly, the main context-free parsing algorithms, such as the Cocke–Kasami–Younger, the recursive descent and the generalized LR, can be extended to Boolean grammars without an increase in computational complexity [15,16,18]. The extended expressive power of Boolean grammars and their intuitive clarity make them a powerful tool for specifying languages, which can replace standard context-free grammars in some applications.

Though practical properties of Boolean grammars seem to be as good as in the case of standard context-free grammars, theoretical questions for Boolean grammars present a greater challenge. Already a formal definition of the semantics of Boolean grammars involves certain theoretical problems [10,15]. A major gap in the knowledge on these grammars is the lack of methods of proving non-representability of languages [15]. Even though the family generated by Boolean grammars was proved to be contained in $\mathit{DTIME}(n^{2.376}) \cap \mathit{DSPACE}(n)$ [15,21], there is still no proof that any particular language from $\mathit{DTIME}(n^{2.376}) \cup \mathit{DSPACE}(n)$ is not generated by any Boolean grammar, and no methods for constructing such proofs are known.

☆ A preliminary version of this paper was presented at the 16th International Symposium on Fundamentals of Computation Theory (FCT 2007) held in Budapest, Hungary on August 27–30, 2007.

* Corresponding address: Department of Mathematics, University of Turku, Turku FI–20014, Finland. Tel.: +358 2 333 5611; fax: +358 2 333 6595.
E-mail address: alexander.okhotin@utu.fi.

0304-3975/$ – see front matter
doi:10.1016/j.tcs.2011.05.013

Results of the latter kind are hard to obtain for many interesting classes of automata and grammars. Consider the family of *trellis automata* [4], also known as one-way real-time cellular automata, which have been studied since the 1970s, and which were eventually proved to be equal in power to a subclass of Boolean grammars [14]. No methods of establishing non-representability of languages in this family had been known for two decades, until the first such result by Yu [30], who established a pumping lemma for a special case. Only a decade later a general non-representability argument for trellis automata was discovered by Terrier [27], who used it to present the first context-free language not recognized by these automata. Another example is given by the growing context-sensitive languages, for which a method of proving non-representability was discovered by Jurdzinski and Loryś [9] twenty years after the model was proposed.

The purpose of this paper is to establish some limitations of the expressive power of the subcase of Boolean grammars, to which the recursive descent parsing is applicable: the *LL(k) Boolean grammars* [18]. Already for this class, obtaining non-representability proofs presents a challenge: consider that there exists an LL(1) linear conjunctive grammar for the language of computations of any Turing machine [17], which rules out a general pumping lemma. There also exists an LL(1) Boolean grammar for a P-complete language [22], which shows computational non-triviality of this class. This paper proposes several methods for proving non-representability of languages by these grammars, which become the first results of such a kind in the field of Boolean grammars.

Following a definition of Boolean grammars in Section 2, recursive descent parsers for Boolean grammars and their simple formal properties are described in Sections 3 and 4. In Section 5, it is proved that Boolean LL grammars over a unary alphabet generate only regular languages, in contrast with conjunctive grammars of the general form, which are known to have formidable expressive power in the domain of one-letter languages [5–8]. Section 6 considers subsets of Σ^*a^* representable by Boolean LL grammars and establishes a periodicity property of such languages, which, in particular, implies non-representability of the language $\{a^n b^{2^n} \mid n \geqslant 0\}$. Stronger non-representability results for two subclasses of Boolean LL grammars with linear concatenation are obtained in Sections 7 and 8. Based on these results, in Section 9, a detailed hierarchy of language families is obtained. Closure properties of these families are determined and compared to the known closure properties of context-free LL languages [26,29] in Section 10.

2. Boolean grammars and their non-left-recursive subset

Let Σ be a finite non-empty set used as an *alphabet*, let Σ^* be the set of all finite strings over Σ. For a string $w = a_1 \dots a_\ell \in \Sigma^*$ with $a_i \in \Sigma$, the *length* of the string is denoted by $|w| = \ell$. The unique string of length 0 is denoted by ε. For a string $w \in \Sigma^*$, for every partition $w = uv$, u is a *prefix* of w and v is its *suffix*; furthermore, for every partition $w = xyz$, the string y is a *substring* of w.

Any subset of Σ^* is a *language* over Σ. The common operations on languages are *concatenation* $K \cdot L = \{uv \mid u \in K,\ v \in L\}$ and the Boolean set operations: union $K \cup L$, intersection $K \cap L$ and complementation $\overline{L}$. *Boolean grammars* are a variant of the context-free grammars, in which all these operations can be explicitly specified.

Definition 1 ([15])**.** A Boolean grammar is a quadruple $G = (\Sigma, N, P, S)$, where Σ and N are disjoint finite non-empty sets of terminal and nonterminal symbols respectively; P is a finite set of rules of the form

$$A \to \alpha_1 \& \cdots \& \alpha_m \& \neg\beta_1 \& \cdots \& \neg\beta_n, \tag{1}$$

where $m + n \geqslant 1$, $\alpha_i, \beta_i \in (\Sigma \cup N)^*$; $S \in N$ is the initial symbol of the grammar.

In this paper, it is further assumed that $m \geqslant 1$ and $n \geqslant 0$ in every rule (1). If $m = 1$ and $n = 0$ in every such rule, then a *standard context-free grammar* is obtained. An intermediate family of *conjunctive grammars* [12] has $m \geqslant 1$ and $n = 0$ in every rule. *Linear* subclasses of Boolean, conjunctive and standard context-free grammars are defined by the additional requirement that $\alpha_i, \beta_i \in \Sigma^* \cup \Sigma^* N \Sigma^*$.

For each rule (1), the objects α_i and $\neg\beta_j$ (for all i, j) are called conjuncts, positive and negative respectively. The notation $\pm\alpha_i$ and $\pm\beta_j$ is used to refer to a conjunct of an unspecified sign.

The intuitive semantics of a Boolean grammar is fairly clear: a rule (1) specifies that every string that satisfies each of the conditions α_i and none of the conditions β_i is therefore generated by A. However, constructing a mathematical definition of a Boolean grammar has proved to be a relatively non-trivial task. Generally, a grammar is interpreted as a system of language equations in variables N, in which the equation for each $A \in N$ is

$$A = \bigcup_{A \to \alpha_1 \& \cdots \& \alpha_m \& \neg\beta_1 \& \cdots \& \neg\beta_n \in P} \left[\bigcap_{i=1}^{m} \alpha_i \ \cap \ \bigcap_{j=1}^{n} \overline{\beta_j} \right]. \tag{2}$$

The vector $(\dots, L_G(A), \dots)$ of languages generated by the nonterminals of the grammar is defined by a solution of this system. If a grammar is conjunctive or standard context-free, then the complementation operation is never used in this system, and hence the system is known to have a *least solution*: this solution is used to define the grammar. But for a Boolean grammar using negation, the system (2) may have no solutions or multiple solutions, and hence the definition requires more precise conditions, which have been a subject of research [10,15].