TUCS

Mikhail Barash | Alexander Okhotin

Linear grammars with one-sided contexts and their automaton representation

Turku Centre *for* Computer Science

TUCS Technical Report
No 1088, September 2013

Linear grammars with one-sided contexts and their automaton representation

Mikhail Barash
mikbar@utu.fi
Department of Mathematics, University of Turku, *and*
Turku Centre for Computer Science
Turku FI–20014, Finland
Alexander Okhotin
alexander.okhotin@utu.fi
Department of Mathematics, University of Turku, *and*
Turku Centre for Computer Science
Turku FI–20014, Finland

TUCS Technical Report
No 1088, September 2013

Abstract

The paper considers a family of formal grammars that extends linear context-free grammars with an operator for referring to the left context of a substring being defined, as well as with a conjunction operation (as in linear conjunctive grammars). These grammars are proved to be computationally equivalent to an extension of one-way real-time cellular automata with an extra data channel. The main result is the undecidability of the emptiness problem for grammars restricted to a one-symbol alphabet, which is proved by simulating a Turing machine by a cellular automaton with feedback. The same construction proves the Σ^0_2-completeness of the finiteness problem for these grammars and automata.

Keywords: Context-free grammars, conjunctive grammars, context-sensitive grammars, cellular automata, undecidability.

TUCS Laboratory
FUNDIM, Fundamentals of Computing and Discrete Mathematics

1 Introduction

The idea of defining context-free rules applicable only in certain contexts dates back to the early work of Chomsky. However, the mathematical model improvised by Chomsky, which he named a "context-sensitive grammar", turned out to be too powerful for its intended application, as it could simulate a space-bounded Turing machine. Recently, the authors [3] made a fresh attempt on implementing the same idea. Instead of the string-rewriting approach from the late 1950s, which never quite worked out for this task, the authors relied upon the modern understanding of formal grammars as a first-order logic over positions in a string, discovered by Rounds [16]. This led to a family of grammars that allows such rules as $A \to BC \,\&\, \lhd D$, which asserts that all strings representable as a concatenation BC and preceded by a left context of the form D have the property A. The semantics of such grammars are defined through logical deduction of items of the form *"a substring v written in left context u has a property A"* [3], and the resulting formal model inherits some of the key properties of formal grammars, including parse trees, an extension of the Chomsky normal form [3, 4], a form of recursive descent parsing [2] and a variant of the Cocke–Kasami–Younger parsing algorithm that works in time $O\big(\frac{n^3}{\log n}\big)$ [14].

This paper aims to investigate the *linear subclass* of grammars with one-sided contexts, where linearity is understood in the sense of Chomsky and Schützenberger, that is, as a restriction to concatenate nonterminal symbols only to terminal strings. An intermediate family of *linear conjunctive grammars*, which allows using the conjunction operation, but no context specifications, was earlier studied by the second author [12, 13]. Those grammars were found to be computationally equivalent to *one-way real-time cellular automata* [6, 17], also known under a proper name of *trellis automata* [5, 7].

This paper sets off by developing an analogous automaton representation for linear grammars with one-sided contexts. The proposed *trellis automata with feedback*, defined in Section 4, augment the original cellular automaton model by an extra communication channel, which adds exactly the same power as context specifications do in grammars. This representation implies the closure of this language family under complementation, which, using grammars alone, would require a complicated construction.

The main contribution of the paper is a method for simulating a Turing machine by a trellis automaton with feedback processing an input string over a one-symbol alphabet. This method subsequently allows uniform undecidability proofs for linear grammars with contexts, which parallels the recent results for conjunctive grammars due to Jeż [8] and Jeż and Okhotin [9, 10, 11], but is based upon an entirely different underlying construction.

The new construction developed in this paper begins in Section 6 with a simple example of a 3-state trellis automaton with feedback, which recognizes the language $\{ a^{2^k-2} \mid k \geqslant 2 \}$. To compare, ordinary trellis automata over a one-symbol alphabet recognize only regular languages [5]. The next Section 7 presents a simulation of a Turing machine by a trellis automaton with feedback, so that the latter automaton, given an input a^n, simulates $O(n)$ first steps of the Turing machine's computation on an empty input, and accordingly can accept or reject the input a^n depending on the current state of the Turing machine.

This construction is used in the last Section 8 to prove the undecidability of the emptiness problem for linear grammars with one-sided contexts over a one-symbol alphabet. The finiteness problem for these grammars is proved to be complete for the second level of the arithmetical hierarchy.

2 Grammars with one-sided contexts

Grammars with contexts were introduced by the authors [3, 4] as a model capable of defining context-free rules applicable only in contexts of a certain form.

Definition 1 ([3]). *A grammar with left contexts is a quadruple $G = (\Sigma, N, R, S)$, where*

- *Σ is the alphabet of the language being defined;*
- *N is a finite set of auxiliary symbols ("nonterminal symbols" in Chomsky's terminology), disjoint with Σ, which denote the properties of strings defined in the grammar;*
- *R is a finite set of grammar rules, each of the form*

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\, \lhd\beta_1 \,\&\, \ldots \,\&\, \lhd\beta_m \,\&\, \trianglelefteqslant\gamma_1 \,\&\, \ldots \,\&\, \trianglelefteqslant\gamma_n, \qquad (1)$$

  with $A \in N$, $k \geqslant 1$, $m, n \geqslant 0$ and $\alpha_i, \beta_i, \gamma_i \in (\Sigma \cup N)^*$;
- *$S \in N$ represents syntactically well-formed sentences of the language.*

Every rule (1) is comprised of *conjuncts* of three kinds. Each conjunct α_i specifies the form of the substring being defined, a conjunct $\lhd\beta_i$ describes the form of its left context, while a conjunct $\trianglelefteqslant\gamma_i$ refers to the form of the left context concatenated with the current substring. To be precise, let $w = uvx$ with $u, v, x \in \Sigma^*$ be a string, and consider defining the substring v by a rule (1). Then, each conjunct α_i describes the form of v, each *left context* $\lhd\beta_i$ describes the form of u, and each *extended left context* $\trianglelefteqslant\gamma_i$, describes the form of uv. The conjunction means that all these conditions must hold at the same time for this rule to be applicable.

If no context specifications are used in the grammar, that is, if $m = n = 0$ in each rule (1), then this is a *conjunctive grammar* [12]. If, furthermore, only one conjunct is allowed in each rule ($k = 1$), this is an ordinary context-free grammar. A grammar is called *linear*, if every conjunct refers to at most one nonterminal symbol, that is, $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_m, \gamma_1, \ldots, \gamma_n \in \Sigma^* N \Sigma^* \cup \Sigma^*$.

The language generated by a grammar with left contexts is defined by deduction of elementary statements of the form "a substring $v \in \Sigma^*$ in the left context $u \in \Sigma^*$ has the property $X \in \Sigma \cup N$", denoted by $X(u\langle v\rangle)$. A full definition applicable to every grammar with left contexts is presented in the authors' previous paper [3, 4]; this paper gives a definition specialized for linear grammars.

Definition 2. *Let $G = (\Sigma, N, R, S)$ be a linear grammar with left contexts, and consider deduction of items of the form $X(u\langle v\rangle)$, with $u, v \in \Sigma^*$ and $X \in N$. Each rule $A \to w$ defines an axiom scheme*

$$\vdash_G A(x\langle w\rangle),$$

for all $x \in \Sigma^*$. Each rule of the form

$$A \to x_1 B_1 y_1 \,\&\, \ldots \,\&\, x_k B_k y_k \,\&\, \lhd x'_1 D_1 y'_1 \,\&\, \ldots \,\&\, \lhd x'_m D_m y'_m \,\& \\ \trianglelefteqslant x''_1 E_1 y''_1 \,\&\, \ldots \,\&\, \trianglelefteqslant x''_n E_n y''_n$$

defines the following scheme for deduction rules for all $u, v \in \Sigma^*$:

$$\big\{B_i(ux_i\langle v_i\rangle)\big\}_{i\in\{1,\ldots,k\}}, \big\{D_i(x'_i\langle u_i\rangle)\big\}_{i\in\{1,\ldots,m\}}, \big\{E_i(x''_i\langle w_i\rangle)\big\}_{i\in\{1,\ldots,n\}} \vdash_G A(u\langle v\rangle),$$

where $x_i v_i y_i = v$, $x'_i u_i y'_i = u$ and $x''_i w_i y''_i = uv$. Then the language defined by a nonterminal symbol A is

$$L_G(A) = \{\, u\langle v\rangle \mid u, v \in \Sigma^*, \vdash_G A(u\langle v\rangle) \,\}.$$

The language defined by the grammar G is the set of all strings with an empty left context defined by S:

$$L(G) = \{\, w \mid w \in \Sigma^*, \vdash_G S(\varepsilon\langle w\rangle) \,\}.$$

This definition is illustrated in the grammar below.

Example 1. The following grammar defines the singleton language $\{abac\}$:

$$\begin{array}{lcl} S & \to & aBc \\ B & \to & bA \,\&\, \lhd A \\ A & \to & a \end{array}$$

The string *abac* is generated as follows:

$$\begin{aligned} &\vdash A(\varepsilon\langle a\rangle) && (A \to a)\\ &\vdash A(ab\langle a\rangle) && (A \to a)\\ A(ab\langle a\rangle), A(\varepsilon\langle a\rangle) &\vdash B(a\langle ba\rangle) && (B \to bA \,\&\, \lhd A)\\ B(a\langle ba\rangle) &\vdash S(\varepsilon\langle abac\rangle) && (S \to aBc) \end{aligned}$$

The next example defines a language that is known to have no linear conjunctive grammar [19].

Example 2 (Törmä [18])**.** The following linear grammar with contexts defines the language $\{\, a^n b^{in} \mid i, n \geqslant 1 \,\}$:

$$\begin{aligned} S &\to aSb \mid B \,\&\, \trianglelefteq S \mid \varepsilon\\ B &\to bB \mid \varepsilon \end{aligned}$$

The rule $S \to B \,\&\, \trianglelefteq S$ appends as many symbols b as there are as in the beginning of the string.

3 Linear grammars and normal form

It is known [3, 4, 14], that every grammar with contexts can be transformed to a certain normal form, which extends the Chomsky normal form for ordinary context-free grammars. While the original Chomsky normal form has all rules of the form $A \to BC$ and $A \to a$, this extension allows using multiple conjuncts BC and context specifications $\lhd D$.

A similar normal form shall now be established for the linear subclass of grammars. The transformation is carried out along the same lines as in the general case [3]. The first step is elimination of *null conjuncts*, that is, any rules of the form $A \to \varepsilon \,\&\, \ldots$. This is followed by elimination of *null contexts* $\lhd\varepsilon$, and of *unit conjuncts*, as in the rules $A \to B \,\&\, \ldots$. The final step is elimination of *extended left contexts* $\trianglelefteq E$, which are all expressed through proper left contexts $\lhd D$ [14]. Each step applies to linear grammars with contexts and preserves their linearity.

Theorem 1. *For every linear grammar with left contexts, there exists another linear grammar with left contexts that defines the same language and has all rules of the form*

$$A \to bB_1 \,\&\, \ldots \,\&\, bB_\ell \,\&\, C_1c \,\&\, \ldots \,\&\, C_kc \tag{2a}$$

$$A \to a \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m, \tag{2b}$$

where $A, B_i, C_i, D_i \in N$, $a, b, c \in \Sigma$, $\ell + k \geqslant 1$ *and* $m \geqslant 0$.

Let $G = (\Sigma, N, R, S)$ be an arbitrary linear grammar with contexts. Similarly to the general case of the grammars with contexts, its transformation to the normal form starts with a preprocessing phase: long conjuncts are cut until all of them are of the form bB, Cc or a, and every context specification $\lhd\gamma$ or $\trianglelefteq\gamma$ with $\gamma \in \Sigma$ or $|\gamma| > 1$ is restated as $\lhd X_\gamma$ or $\trianglelefteq X_\gamma$, respectively, where X_γ is a new nonterminal with a unique rule $X_\gamma \to \gamma$.

This results in a grammar $G_1 = (\Sigma, N_1, R_1, S)$ with the rules of the following kind:

$$\begin{aligned} A &\to bB && \text{(3a)} \\ A &\to Cc && \text{(3b)} \\ A &\to a && \text{(3c)} \\ A &\to B_1 \,\&\, \dots \,\&\, B_k \,\&\, \lhd D_1 \,\&\, \dots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \dots \,\&\, \trianglelefteq E_n && \text{(3d)} \\ A &\to \varepsilon, && \text{(3e)} \end{aligned}$$

where $a, b, c \in \Sigma$ and $A, B_i, D_i, E_i \in N$.

Then, *null conjuncts* in rules of the form $A \to \varepsilon \,\&\, \dots$ are eliminated using the method of Barash and Okhotin [3, 4]. First, one has to determine, which nonterminals generate the empty string, and in which contexts they generate it. This is done by constructing a set $\textsc{Nullable}(G) \subseteq 2^N \times N$ [3, 4] of nonterminals capable of generating the empty string in certain contexts. Intuitively, a pair $(\{K_1, \dots, K_t\}, A) \in \textsc{Nullable}(G)$ with $A, K_1, \dots, K_t \in N$ means that the nonterminal A generates the empty string in the context u (that is, $u\langle\varepsilon\rangle$), and u can be described by every nonterminal K_i (that is, $\varepsilon\langle u\rangle \in L_G(K_i)$).

Using the set $\textsc{Nullable}(G)$, a new grammar $G_2 = (\Sigma, N_1, R_2, S)$ without null conjuncts can be constructed as follows.

1. The rules of the form (3a)–(3d) are copied to the new grammar.

2. For every rule of the form (3a) and for every pair $(B, \{K_1, \dots, K_t\}) \in \textsc{Nullable}(G)$, a rule $A \to b \,\&\, \trianglelefteq K_1 \,\&\, \dots \,\&\, \trianglelefteq K_t$ is added to the new grammar.

3. For every rule of the form (3b) and for every pair $(C, \{K_1, \dots, K_t\}) \in \textsc{Nullable}(G)$, the new grammar has the rule $A \to c \,\&\, \lhd K_1 \,\&\, \dots \,\&\, \lhd K_t$. Moreover, if $\varepsilon\langle\varepsilon\rangle \in L_G(K_i)$ for all $i \in \{1, \dots, t\}$, then a rule $A \to c \,\&\, \lhd\varepsilon$ should be added.

4. For every rule of the form (3d), a rule $A \to B_1 \,\&\, \dots \,\&\, B_k \,\&\, E_1 \,\&\, \dots \,\&\, E_n \,\&\, \lhd\varepsilon$ shall be added to the new grammar, if $\varepsilon\langle\varepsilon\rangle \in L_G(D_i)$ for all $i \in \{1, \dots, m\}$.

Correctness Claim 1. *Let $A \in N$, $u, v \in \Sigma^*$. Then a string $u\langle v\rangle$ is in $L_{G_2}(A)$ if and only if $v \neq \varepsilon$ and $u\langle v\rangle \in L_{G_1}(A)$.*

After this step, the rules of the grammar can be of the following form:

$$A \to a \tag{4a}$$
$$A \to bB \tag{4b}$$
$$A \to Cc \tag{4c}$$
$$A \to B_1 \,\&\, \dots \,\&\, B_k \,\&\, \lhd D_1 \,\&\, \dots \,\&\, \lhd D_m \,\&\, \trianglelefteqslant E_1 \,\&\, \dots \,\&\, \trianglelefteqslant E_n \tag{4d}$$
$$A \to B_1 \,\&\, \dots B_k \,\&\, \lhd \varepsilon \tag{4e}$$
$$A \to b \,\&\, \trianglelefteqslant K_1 \,\&\, \dots \,\&\, \trianglelefteqslant K_t \tag{4f}$$
$$A \to c \,\&\, \lhd K_1 \,\&\, \dots \,\&\, \lhd K_t \tag{4g}$$
$$A \to c \,\&\, \lhd \varepsilon \tag{4h}$$

Null contexts $\lhd\varepsilon$, added to the grammar by the above construction, can be removed by the method of Barash and Okhotin [4]. Construct a new grammar $G_3 = (\Sigma, N_3, R_3, S_3)$, with $N_3 = N_1 \cup \{\, \widetilde{A} \mid A \in N_1 \,\}$ and $S_3 = \widetilde{S}$. Every nonterminal A has two copies, one with a non-empty left context, denoted by A, and the other with the empty left context, called $\widetilde{A}$.

1. Each rule of the form (4a) is added to the new grammar along with an extra rule $\widetilde{A} \to a$.

2. Each rule of the form (4b) is added to the new grammar. Moreover, the new grammar contains a rule $\widetilde{A} \to bB$.

3. For each rule (4c) of the original grammar, the new grammar has the rules $A \to Cc$ and $\widetilde{A} \to \widetilde{C}c$.

4. For each rule (4d), the new grammar has a rule $A \to B_1 \,\&\, \dots \,\&\, B_k \,\&\, \lhd \widetilde{D}_1 \,\&\, \dots \,\&\, \lhd \widetilde{D}_m \,\&\, \trianglelefteqslant \widetilde{E}_1 \,\&\, \dots \,\&\, \trianglelefteqslant \widetilde{E}_n$, and, if $m = 0$, a rule $\widetilde{A} \to \widetilde{B}_1 \,\&\, \dots \,\&\, \widetilde{B}_k \,\&\, \widetilde{E}_1 \,\&\, \dots \,\&\, \widetilde{E}_n$.

5. For each rule (4e) in the original grammar, the new grammar has a rule $\widetilde{A} \to \widetilde{B}_1 \,\&\, \dots \,\&\, \widetilde{B}_k$.

6. For every rule (4f), the new grammar has a rule $\widetilde{A} \to b \,\&\, \trianglelefteqslant \widetilde{K}_1 \,\&\, \dots \,\&\, \trianglelefteqslant \widetilde{K}_t$.

7. For each rule (4g), the corresponding rule of the new grammar is $\widetilde{A} \to c \,\&\, \lhd \widetilde{K}_1 \,\&\, \dots \,\&\, \lhd \widetilde{K}_t$.

8. For every rule (4h), the new grammar has a rule $\widetilde{A} \to c$.

Correctness Claim 2. *For all $A \in N$ and $u, v \in \Sigma^+$:*

- *$u\langle v\rangle \in L_{G_3}(A)$ if and only if $u\langle v\rangle \in L_{G_2}(A)$;*

- $\varepsilon\langle v\rangle \in L_{G_3}(\widetilde{A})$ *if and only if* $\varepsilon\langle v\rangle \in L_{G_2}(A)$.

The next step of the transformation is *elimination of unit conjuncts* in rules of the form $A \to B \,\&\, \ldots$. The construction of a new grammar $G_4 = (\Sigma, N_3, R_4, S_3)$ free of unit conjuncts can be done similarly to the cases of conjunctive grammars and grammars with contexts [12, 3, 4], by substituting all rules for the nonterminal B into each rule containing this unit conjunct.

The next step is to transform the grammar in such a way, that quantified conjuncts are only allowed in the rules of the form $A \to a \,\&\, \trianglelefteq E_1 \,\&\, \ldots \,\&\, \trianglelefteq E_n$, with $a \in \Sigma$ [14]. For a linear grammar with contexts G_4, this method produces a grammar $G_5 = (\Sigma, N_5, R_5, S_3)$ with the set of nonterminals $N_5 = N_3 \cup \{\, A' \mid A \in N_3 \,\} \cup \{\, A'' \mid A \in N_3 \,\} \cup \{\, A_a \mid A \in N_3,\, a \in \Sigma \,\}$. Every symbol D' (E'') should define all possible strings with the left context D (extended left context E, respectively), as it would be done by the context operator $\lhd D$ ($\trianglelefteq E$, respectively). The rules for every nonterminal D' and E'' reduce the length of the current substring to a single symbol, to which one can apply the extended left context operator. The construction itself is as follows:

1. For every rule of the form $A \to bB_1 \,\&\, \ldots \,\&\, bB_\ell \,\&\, C_1c \,\&\, \ldots C_kc \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \ldots \,\&\, \trianglelefteq E_n$, the new grammar has the rule

$$A \to bB_1 \,\&\, \ldots \,\&\, bB_\ell \,\&\, C_1c \,\&\, \ldots C_kc \,\&\, D'_1 \,\&\, \ldots \,\&\, D'_m \,\&\, E''_1 \,\&\, \ldots \,\&\, E''_n$$

2. For every rule of the form $A \to a \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \ldots \,\&\, \trianglelefteq E_n$, the new grammar has the rule

$$A \to a \,\&\, D'_1 \,\&\, \ldots \,\&\, D'_m \,\&\, E''_1 \,\&\, \ldots \,\&\, E''_n$$

3. For every conjunct $\lhd D$, with $D \in N$ in the right-hand sides of the rules of the original grammar, R_5 has the following rules:

$$\begin{aligned} D' &\to D'a && \text{(for all } a \in \Sigma) \\ D' &\to a \,\&\, \trianglelefteq D_a && \text{(for all } a \in \Sigma) \\ D_a &\to Da && \text{(for all } a \in \Sigma) \end{aligned}$$

4. For every conjunct $\trianglelefteq E$, with $E \in N$ in the right-hand sides of the rules of the original grammar, the following rules are added to R_5:

$$\begin{aligned} E'' &\to aE'' && \text{(for all } a \in \Sigma) \\ E'' &\to a \,\&\, \trianglelefteq E && \text{(for all } a \in \Sigma) \end{aligned}$$

Thus far, the grammar has rules of the following kind:

$$A \to a \,\&\, \trianglelefteq E_1 \,\&\, \ldots \,\&\, \trianglelefteq E_n \tag{5a}$$

$$A \to bB_1 \,\&\, \ldots \,\&\, bB_\ell \,\&\, C_1c \,\&\, \ldots \,\&\, C_kc \tag{5b}$$

Correctness Claim 3. *In the grammar G_5,*

$$\begin{aligned} L_{G_5}(D') &= \{\, u\langle v\rangle \mid \varepsilon\langle u\rangle \in L_{G_4}(D),\ v \in \Sigma^+ \,\}, \\ L_{G_5}(E'') &= \{\, u\langle v\rangle \mid \varepsilon\langle uv\rangle \in L_{G_4}(E) \,\}, \\ L_{G_5}(A_a) &= \{\, u\langle va\rangle \mid u\langle v\rangle \in L_{G_4}(A) \,\}. \end{aligned}$$

Furthermore, Okhotin has shown [14] that all *extended left contexts can be effectively converted to left ones.*

Given a grammar G_5, the construction of the new grammar $G_6 = (\Sigma, N_6, R_6, S_6)$ with $N_6 = N_5 \cup \{\, A^{a,X} \mid A \in N_5,\ a \in \Sigma,\ X \subseteq N \,\}$ is as follows.

1. For every rule of the form (5a) in the original grammar the new grammar has a rule
$$A \to a \,\&\, \lhd D_1^{a,X_1} \,\&\, \ldots \,\&\, \lhd D_n^{a,X_n}, \tag{6a}$$
where each X_i is a set of direct descendants of the corresponding D_i in a connected directed acyclic graph with a set of nodes $\{D_1, \ldots, D_n\} \supseteq \{E_1, \ldots, E_m\}$ with the set of sources $\{E_1, \ldots, E_m\}$. Moreover, the new grammar has a rule of the form
$$A^{a,\{E_1,\ldots,E_n\}} \to \varepsilon. \tag{6b}$$

2. For every rule (5b) in the original grammar, the new grammar has a copy of it, as well as extra rules of the form:
$$A^{a,X_1\cup\ldots\cup X_k} \to bB_1^{a,X_1} \,\&\, \ldots \,\&\, bB_\ell^{a,X_\ell} \,\&\, C_1c \,\&\, \ldots \,\&\, C_kc, \tag{6c}$$
with $a \in \Sigma$, $X_1, \ldots, X_k \subseteq N$.

Correctness Claim 4. *As a result of this construction, $L_{G_6}(A) = L_{G_5}(A)$, for all $A \in N$. Every language $L_{G_6}(A^{a,X})$ contains all such strings $u\langle v\rangle$, that the item $A(u\langle va\rangle)$ can be deduced in the grammar G_5 out of the premises $F(\varepsilon\langle uva\rangle)$ (for all $F \in X$), and no items $K(\varepsilon\langle uva\rangle)$ (with $K \in N$) can be inferred during such deduction.*

Now all context operators in rules of the grammar G_6 are of the form $\lhd\beta$.

The constructed grammar G_6 has null rules of the form (6b), which have to be eliminated by the procedure of null conjuncts elimination described above. Since none of such rules have contexts, the elimination shall not introduce any new contexts in the grammar. However, as a result of such elimination, some of the rules (6a) may get null contexts and some of the rules (6c) may get unit conjuncts, which can be again eliminated by the corresponding procedures. Neither of these procedures appends any new extended left contexts ($\trianglelefteq\gamma$) to the grammar.

Thus, all rules of the grammar are of the form (2a)–(2b), as desired.

4 Automaton representation

Linear conjunctive grammars are known to be computationally equivalent to one of the simplest types of cellular automata: the *one-way real-time cellular automata*, also known under the proper name of *trellis automata*. This section presents a generalization of trellis automata, which similarly corresponds to linear grammars with one-sided contexts.

An ordinary trellis automaton processes an input string of length $n \geqslant 1$ using a uniform array of $\frac{n(n+1)}{2}$ nodes, as presented in Figure 1(left). Each node computes a value from a fixed finite set Q. The nodes in the bottom row obtain their values directly from the input symbols using a function $I \colon \Sigma \to Q$. The rest of the nodes compute the function $\delta \colon Q \times Q \to Q$ of the values in their predecessors. The string is accepted if and only if the value computed by the topmost node belongs to the set of accepting states $F \subseteq Q$.

Theorem A (Okhotin [13])**.** A language $L \subseteq \Sigma^+$ is defined by a linear conjunctive grammar if and only if L is recognized by a trellis automaton.

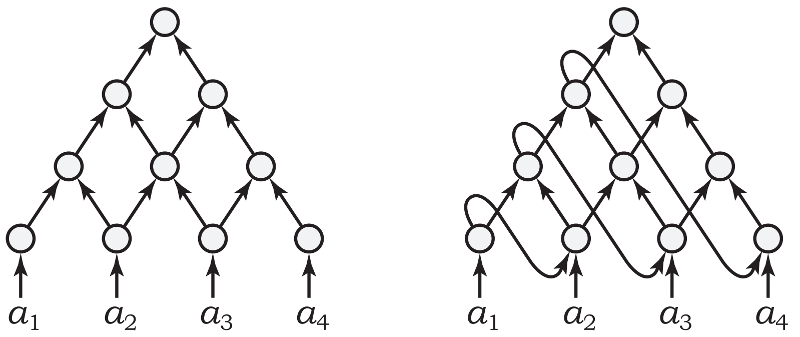


Figure 1: Trellis automata (left) and trellis automata with feedback (right).

In terms of cellular automata, every horizontal row of states in Figure 1(left) represents an automaton's configuration at a certain moment of time. An alternative motivation developed in the literature on trellis automata [5, 6, 7] is to consider the entire grid as a digital circuit with uniform structure of connections. In order to obtain a similar representation of linear grammars with left contexts, the trellis automaton model is extended with another type of connections, illustrated in Figure 1(right).

Definition 3. *A trellis automaton with feedback is a sextuple* $M = (\Sigma, Q, I, J, \delta, F)$*, in which:*

- Σ *is the input alphabet,*
- Q *is a finite non-empty set of states,*

- *$I\colon \Sigma \to Q$ is a function that sets the initial state for the first symbol,*

- *$J\colon Q \times \Sigma \to Q$ sets the initial state for every subsequent symbol, using the state computed on the preceding substring as a feedback;*

- *$\delta\colon Q \times Q \to Q$ is the transition function, and*

- *$F \subseteq Q$ is the set of accepting states.*

The behaviour of the automaton is described by a function $\Delta\colon \Sigma^* \times \Sigma^+ \to Q$, which defines the state $\Delta(u\langle v\rangle)$ computed on each string with a context $u\langle v\rangle$ by

$$\begin{aligned}
\Delta(\varepsilon\langle a\rangle) &= I(a),\\
\Delta(w\langle a\rangle) &= J\big(\Delta(\varepsilon\langle w\rangle), a\big),\\
\Delta(u\langle bvc\rangle) &= \delta\big(\Delta(u\langle bv\rangle), \Delta(ub\langle vc\rangle)\big).
\end{aligned}$$

The language recognized by the automaton is $L(M) = \{\, w \in \Sigma^+ \mid \Delta(\varepsilon\langle w\rangle) \in F \,\}$.

Theorem 2. *A language $L \subseteq \Sigma^+$ is defined by a linear grammar with left contexts if and only if L is recognized by a trellis automaton with feedback.*

The proof is by effective constructions in both directions.

Lemma 1. *Let $G = (\Sigma, N, R, S)$ be a linear grammar with left contexts, in which every rule is of the form*

$$\begin{aligned}
&A \to bB_1 \,\&\, \ldots \,\&\, bB_\ell \,\&\, C_1c \,\&\, \ldots \,\&\, C_kc && (b, c \in \Sigma,\ B_i, C_i \in N), && \text{(7a)}\\
&A \to a \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m && (a \in \Sigma,\ m \geqslant 0,\ D_i \in N), && \text{(7b)}
\end{aligned}$$

and define a trellis automaton with feedback $M = (\Sigma, Q, I, J, \delta, F)$ by setting $Q = \Sigma \times 2^N \times \Sigma$,

$$\begin{aligned}
I(a) &= (a, \{\, A \mid A \to a \in R \,\}, a)\\
J\big((b, X, c), a\big) &= \big(a, \{\, A \mid \exists \text{ rule (7b) with } D_1, \ldots, D_m \in X \,\}, a\big)\\
\delta\big((b, X, c'), (b', Y, c)\big) &= \big(b, \{\, A \mid \exists \text{ rule (7a) with } B_i \in X \text{ and } C_i \in Y \,\}, c\big)\\
F &= \big\{\, (b, X, c) \bigm| S \in X \,\big\}.
\end{aligned}$$

For every string with context $u\langle v\rangle$, let b be the first symbol of v, let c be the last symbol of v, and let $Z = \{\, A \mid u\langle v\rangle \in L_G(A) \,\}$. Then $\Delta(u\langle v\rangle) = (b, Z, c)$.

In particular, $L(M) = \{\, w \mid \varepsilon\langle w\rangle \in L_G(S) \,\} = L(G)$.

Proof. Induction on pairs $(|uv|, |v|)$, ordered lexicographically.

Basis: $\varepsilon\langle a\rangle$ with $a \in \Sigma$. The state computed on this string is $\Delta(\varepsilon\langle a\rangle) = I(a) = (a, Z, a)$ with $Z = \{\, A \mid A \to a \in R \,\}$. The latter set Z is the set of all symbols $A \in N$ with $\varepsilon\langle a\rangle \in L_G(A)$.

Induction step I: $u\langle a\rangle$ with $u \in \Sigma^*$ and $a \in \Sigma$. The state computed by the automaton on the string $u\langle a\rangle$ is defined as $\Delta(u\langle a\rangle)) = J(\Delta(\varepsilon\langle u\rangle), a)$. By the induction hypothesis, the state reached on the string $\varepsilon\langle u\rangle$ is $\Delta(\varepsilon\langle u\rangle) = (a, X, a)$, where a is the first symbol of u and $X \subseteq N$ is the set of symbols that generate $\varepsilon\langle u\rangle$. Substituting this value into the expression for the state reached on $u\langle a\rangle$ yields $\Delta(u\langle a\rangle) = J((a, X, a), a) = (a, Z, a)$, where

$$\begin{aligned} Z &= \{\, A \mid \text{there exists a rule (7b) with } D_1, \ldots, D_m \in X \,\} = \\ &= \{\, A \mid \text{there exists a rule (7b) with } \varepsilon\langle u\rangle \in L_G(D_i) \text{ for all } i \,\}. \end{aligned}$$

The latter condition means that Z is the set of all symbols $A \in N$ that generate the string $u\langle a\rangle$ using a rule of the form (7b). Since this string can only be generated by rules of that form, this is equivalent to $Z = \{\, A \mid u\langle a\rangle \in L_G(A) \,\}$, as claimed.

Induction step II: $u\langle bvc\rangle$ with $u, v \in \Sigma^*$ and $b, c \in \Sigma$. The state computed on such a string is $\Delta(u\langle bvc\rangle) = \delta(\Delta(u\langle bv\rangle), \Delta(ub\langle vc\rangle))$. By the induction hypothesis, the states reached by the automaton on the strings $u\langle bv\rangle$ and $ub\langle vc\rangle$ are respectively $\Delta(u\langle bv\rangle) = (b, X, b')$ and $\Delta(ub\langle vc\rangle) = (c', Y, c)$, where b is the last symbol of bv, c' is the first symbol of vc, $X \subseteq N$ is the set of nonterminal symbols generating $u\langle bv\rangle$ and $Y \subseteq N$ contains all such nonterminals that generate the string $ub\langle vc\rangle$.

Substituting the states reached on these shorter strings into the expression for the state computed on $u\langle bvc\rangle$ gives $\Delta(u\langle bvc\rangle) = \delta\big((b, X, b'), (c', Y, c)\big) = (b, Z, c)$, where

$$\begin{aligned} Z &= \big\{\, A \mid \text{there exists a rule (7a) with } B_i \in X \text{ and } C_i \in Y \,\} = \\ &= \big\{\, A \mid \text{there exists a rule (7a) with } u\langle bv\rangle \in L_G(B_i) \text{ and} \\ &\qquad\qquad\qquad\qquad ub\langle vc\rangle \in L_G(C_j), \text{ for all } i, j \,\}. \end{aligned}$$

That is, Z is exactly the set of nonterminals that generate the string $ub\langle vc\rangle$ by a rule of the form (7a). The string $u\langle bvc\rangle$ can only be generated by a rule of such a form, and, thus, $Z = \{\, A \mid u\langle bvc\rangle \in L_G(A) \,\}$, as desired.

□

Lemma 2. *Let $M = (\Sigma, Q, I, J, \delta, F)$ be a trellis automaton with feedback and define the grammar with left contexts $G = (\Sigma, N, R, S)$, where $N =$*

$\{ A_q \mid q \in Q \} \cup \{S\}$, and the set R contains the following rules:

$$\begin{array}{rlr}
A_{I(a)} \to a \,\&\, \lhd\varepsilon & (a \in \Sigma) & \text{(8a)}\\
A_{J(q,a)} \to a \,\&\, \lhd A_q & (q \in Q,\ a \in \Sigma) & \text{(8b)}\\
A_{\delta(p,q)} \to bA_q \,\&\, A_p c & (p, q \in Q,\ b, c \in \Sigma) & \text{(8c)}\\
S \to A_q & (q \in F) & \text{(8d)}
\end{array}$$

Then, for every string with context $u\langle v\rangle$, $\Delta(u\langle v\rangle) = r$ if and only if $u\langle v\rangle \in L_G(A_r)$. In particular, $L(G) = \{ w \mid \Delta(\varepsilon\langle w\rangle) \in F \} = L(M)$.

Proof. Induction on lexicographically ordered pairs $(|uv|, |v|)$.

Basis: $\varepsilon\langle a\rangle$ with $a \in \Sigma$. Then $\Delta(\varepsilon\langle a\rangle) = I(a)$. At the same time, $\varepsilon\langle a\rangle$ may only be generated by the rule of the form (8a), and such a rule for A_r exists if and only if $I(a) = r$.

Induction step I: $u\langle a\rangle$ with $u \in \Sigma^+$ and $a \in \Sigma$.

$\Rightarrow$ Let $\Delta(u\langle a\rangle) = r$. Then, $r = J(\Delta(\varepsilon\langle u\rangle), a)$. Let $q = \Delta(\varepsilon\langle u\rangle)$. By the induction hypothesis, $\varepsilon\langle u\rangle \in L_G(A_q)$. Since $J(q, a) = r$, the grammar contains a corresponding rule of the form (8b), which can be used to deduce the membership of $u\langle a\rangle$ in $L_G(A_r)$ as follows:

$$A_q(\varepsilon\langle u\rangle) \vdash_G A_r(u\langle a\rangle) \qquad (A_r \to a \,\&\, \lhd A_q). \tag{9}$$

$\Leftarrow$ Conversely, assume that $u\langle a\rangle \in L_G(A_r)$. Then its deduction must end with an application of a rule of the form (8b), as in (9). By construction, the existence of such a rule implies $r = J(q, a)$. Applying the induction hypothesis to $A_q(\varepsilon\langle u\rangle)$ yields $\Delta(\varepsilon\langle u\rangle) = q$. Then the automaton calculates as follows: $\Delta(u\langle a\rangle) = J(\Delta(\varepsilon\langle u\rangle), a) = J(q, a) = r$, as desired.

Induction step II: $u\langle bvc\rangle$ with $u, v \in \Sigma^*$ and $b, c \in \Sigma$.

$\Rightarrow$ Assume first that $\Delta(u\langle bvc\rangle) = r$. Then $r = \delta(p, q)$, where $p = \Delta(u\langle bv\rangle)$ and $q = \Delta(ub\langle vc\rangle)$. By the induction hypothesis, $u\langle bv\rangle \in L_G(A_p)$ and $ub\langle vc\rangle \in L_G(A_q)$. From this, using a rule of the form (8c), one can deduce

$$A_p(u\langle bv\rangle), A_q(ub\langle vc\rangle) \vdash_G A_r(u\langle bvc\rangle) \qquad (A_r \to bA_q \,\&\, A_p c), \tag{10}$$

that is, $u\langle bvc\rangle \in L_G(A_r)$, as claimed.

$\Leftarrow$ Conversely, if $u\langle bvc\rangle \in L_G(A_r)$, then the deduction establishing $A_r(u\langle bvc\rangle)$ must end as (10), using a rule of the form (8c). Then, by the construction, $r = \delta(p, q)$. Since the items $A_p(u\langle bv\rangle)$ and $A_q(ub\langle vc\rangle)$ are deduced in the grammar, by the induction hypothesis, $\Delta(u\langle bv\rangle) = p$ and $\Delta(ub\langle vc\rangle) = q$. Then $\Delta(u\langle bvc\rangle) = \delta(p, q) = r$. □

5 Closure properties

The automaton representation devised in previous section is useful for establishing some basic properties of linear grammars with contexts, which would be more difficult to obtain using grammars alone. For instance, one can prove their closure under complementation by taking a trellis automaton with feedback and inverting its set of accepting states.

Another closure result is the closure under concatenating a linear conjunctive language from the right.

Lemma 3. *Let $L \subseteq \Sigma^*$ be defined by a linear grammar with contexts, and let $K \subseteq \Sigma^*$ be a linear conjunctive language. Then the language $L \cdot K$ can be defined by a linear grammar with contexts.*

Proof. Let $G_1 = (\Sigma, N_1, R_1, S_1)$ and $G_2 = (\Sigma, N_2, R_2, S_2)$ be the grammars generating the languages L and K, respectively. Construct a linear conjunctive grammar with contexts $G = (\Sigma, N_1 \cup N_2 \cup \{S\}, R_1 \cup R_2 \cup R, S)$, where R contains the rules $S \to aS$ (for all $a \in \Sigma$) and $S \to S_2 \,\&\, \lhd S_1$. □

This, in particular, implies that the language

$$L = \{\, a^{i_1} b^{j_1} \dots a^{i_m} b^{j_m} \mid m \geqslant 2,\ i_t, j_t \geqslant 1,\ \exists \ell : i_1 = j_\ell \wedge i_{\ell+1} = j_m \,\},$$

used by Terrier [17] to show that linear conjunctive languages are not closed under concatenation, can be defined by a linear grammar with contexts.

By the same method as in Lemma 3, one can show that the Kleene star of any linear conjunctive language can be represented by a linear grammar with contexts.

Lemma 4. *Let L be a linear conjunctive language. Then the language L^* can be defined by a linear grammar with contexts.*

Proof. Let $G = (\Sigma, N, R, S)$ be a linear conjunctive grammar that defines L. Construct a linear grammar with contexts $G' = (\Sigma, N \cup \{S', X, Y, Z\}, R \cup R', S')$, with the following rules in R'.

$$\begin{array}{lcll}
S' & \to & aX & \text{(for all } a \in \Sigma) \\
S' & \to & \varepsilon & \\
X & \to & S \,\&\, \lhd Y & \\
Y & \to & aZ & \text{(for all } a \in \Sigma) \\
Z & \to & S \,\&\, \lhd S' &
\end{array}$$

Then $L(G') = L(G)^*$. □

6 Defining a non-regular unary language

Ordinary context-free grammars over a unary alphabet $\Sigma = \{a\}$ define only regular languages. Unary linear conjunctive languages are also regular, because a trellis automaton operates on an input a^n as a deterministic finite automaton [5]. The non-triviality of unary conjunctive grammars was discovered by Jeż [8], who constructed a grammar for the language $\{\, a^{4^k} \mid k \geqslant 0 \,\}$ using iterated conjunction and concatenation of languages.

This paper introduces a new method for constructing formal grammars for non-regular languages over a unary alphabet, which makes use of a left context operator, but does not rely upon non-linear concatenation. The simplest case of the new method is demonstrated by the following automaton, which can be transformed to a grammar by Lemma 2.

Example 3. Consider a trellis automaton with feedback $M = (\Sigma, Q, I, J, \delta, F)$ over the alphabet $\Sigma = \{a\}$ and with the set of states $Q = \{p, q, r\}$, where $I(a) = p$ is the initial state, the feedback function gives states $J(p, a) = q$ and $J(r, a) = p$, and the transition function is defined by $\delta(s, p) = p$ for all $s \in Q$, $\delta(q, q) = \delta(r, q) = q$, $\delta(p, q) = r$ and $\delta(p, r) = p$. The only accepting state is r. Then M recognizes the language $\{\, a^{2^k-2} \mid k \geqslant 2 \,\}$.

The computation of this automaton is illustrated in Figure 2. The state computed on each one-symbol substring $a^\ell\langle a\rangle$ is determined by the state computed on $\varepsilon\langle a^\ell\rangle$ according to the function J. Most of the time, $\Delta(\varepsilon\langle a^\ell\rangle) = p$ and hence $\Delta(a^\ell\langle a\rangle) = q$, and the latter continues into a triangle of states q. Once for every power of two, the automaton computes the state r on $\varepsilon\langle a^{2^k-2}\rangle$, which sends a signal through the feedback channel, so that J sets $\Delta(a^{2^k-2}\langle a\rangle) = p$. This in turn produces the triangle of states p and the next column of states r.

The following lemma states that the automaton in Example 3 indeed works as described.

Lemma 5. *Consider the automaton M from Example 3. Denote $M[i, j] = \Delta(a^{i-1}\langle a^{j-i+1}\rangle)$. Then:*

($\mathcal{U}_1$). *For all $i \in \{1, \ldots, 2^k\}$ and $j \in \{1, \ldots, 2^k\}$ satisfying $i + j = 2^k - 1$ and $i \leqslant j$, $M[i, j] = r$.*

($\mathcal{U}_2$). *For all $i \in \{2, \ldots, 2^k - 2\}$ and $j \in \{2^{k-1}, \ldots, 2^k - 2\}$ with $2^k \leqslant i + j \leqslant 2^{k+1} - 4$ and $i \leqslant j$, $M[i, j] = q$.*

($\mathcal{U}_3$). *For all $i \in \{1, \ldots, 2^k - 1\}$ and $j \in \{2^k - 1, \ldots, 2^{k+1} - 3\}$, such that $2^k \leqslant i + j \leqslant 2^{k+1} - 2$, $M[i, j] = p$.*

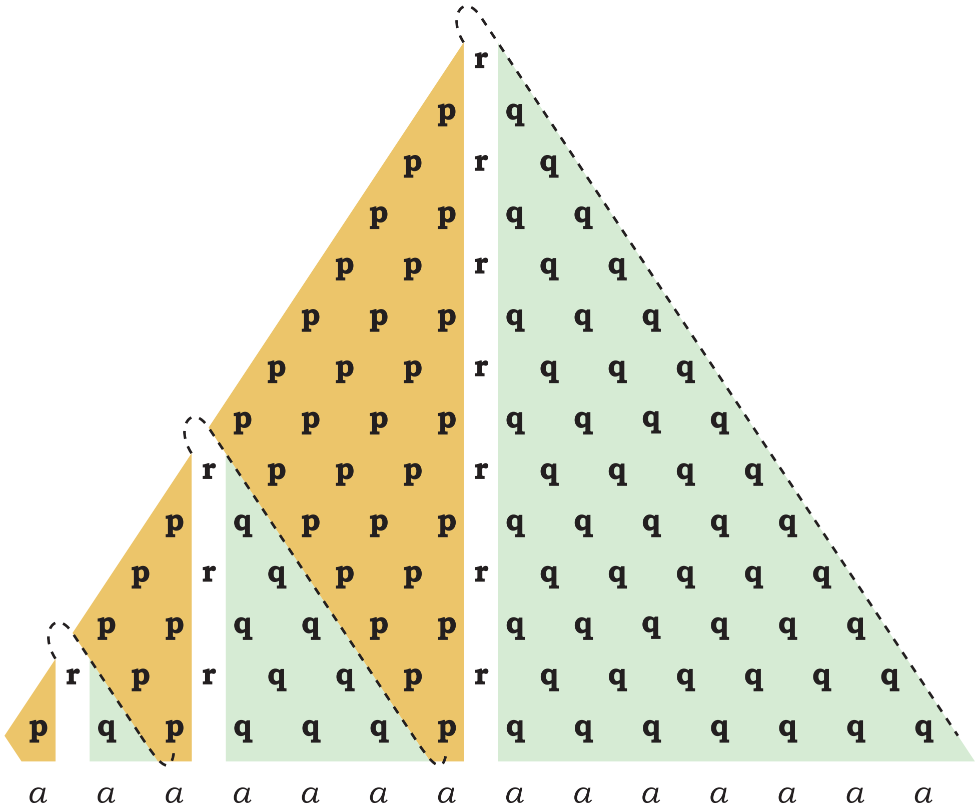


Figure 2: How the automaton in Example 3 recognizes $\{ a^{2^k-2} \mid k \geqslant 2 \}$.

It is now known that linear grammars with contexts over a one-symbol alphabet are non-trivial. How far does their expressive power go? For conjunctive grammars (which allow non-linear concatenation, but no context specifications), Jeż and Okhotin [9, 10, 11] developed a method for manipulating base-k notation of the length of a string in a grammar, which allowed representing the following language: for every trellis automaton M over an alphabet $\{0, 1, \ldots, k-1\}$, there is a conjunctive grammar generating $L_M = \{ a^\ell \mid \text{the base-}k \text{ notation of } \ell \text{ is in } L(M) \}$ [9]. This led to the following undecidability method: given a Turing machine T, one first constructs a trellis automaton M for the language $\text{VALC}(T) \subseteq \Sigma^*$ of computation histories of T; then, assuming that the symbols in Σ are digits in some base-k notation, one can define the unary version of $\text{VALC}(T)$ by a conjunctive grammar.

Linear grammars with contexts are an entirely different model, and the automaton in Example 3 has nothing in common with the basic unary conjunctive grammar discovered by Jeż [8], in spite of defining almost the same language. The new model seems to be unsuited for manipulating base-k digits, and the authors took another route to undecidability results, which is explained below.

7 Simulating a Turing machine

The overall idea is to augment the automaton in Example 3 to calculate some additional data, so that its computation on a unary string simulates any fixed Turing machine running on the empty input. Each individual cell $\Delta(a^k\langle a^\ell\rangle)$ computed by the automaton should hold some information about the computation of the Turing machine, such as the contents of a certain tape square at a certain time. Then the automaton can accept its input a^n depending on the state of the computation of the Turing machine at time $f(n)$.

Consider the computation in Figure 2, which is split into regions by vertical r-columns. The bottom line of states q in each region shall hold the tape contents of the Turing machine. The new automaton should simulate several steps of the Turing machine, and then transfer its resulting tape contents to the top diagonal border of this region. The transfer of each letter is achieved by sending a signal to the right, reflecting it off the vertical r-column, so that it arrives at the appropriate cell in the top border. From there, the tape contents shall be moved to the bottom line of the next region through the feedback data channel. Because of the reflection, the tape symbols shall arrive at the next region *in the reverse order.*

In order to simulate a Turing machine using this method, it is useful to assume a machine of the following special kind. This machine operates on an initially blank two-way infinite tape, and proceeds by making left-to-right and right-to-left sweeps over this tape, travelling a longer distance at every sweep. At the first sweep, the machine makes one step to the left, then, at the second sweep, it makes 3 steps to the right, then 7 steps to the left, 15 steps to the right, etc. In order to simplify the notation, assume that the machine always travels *from right to left* and flips the tape after completing each sweep.

Definition 4. *A* sweeping Turing machine *is a quintuple* $T = (\Gamma, \mathcal{Q}, q_0, \nabla, \mathcal{F})$*, where*

- Γ *is a finite* tape alphabet *containing a blank symbol* $\Box \in \Gamma$*;*
- $\mathcal{Q}$ *is a finite set of* states,
- $q_0 \in \mathcal{Q}$ *is the* initial state *and* $\mathcal{F} \subseteq \mathcal{Q}$ *is the set of* accepting states*;*
- $\nabla \colon \mathcal{Q} \times \Gamma \to \mathcal{Q} \times \Gamma$ *is a* transition function*;*
- $\mathcal{F}$ *is a finite set of* flickering states.

A configuration *of* T *is a string of the form* $[\![k]\!]uqav$*, where* $k \geqslant 1$ *is the number of the sweep, and* $uqav$ *with* $u, v \in \Gamma^*$*,* $a \in \Gamma$ *and* $q \in \mathcal{Q}$ *represents the tape contents* uav *with the head scanning the symbol* a *in the state* q*.*

The initial configuration of the machine is $\llbracket 1 \rrbracket \Box q_0 \Box$. Each k-th sweep deals with a tape with 2^k symbols, and consists of $2^k - 1$ steps of the following form:

$$\llbracket k \rrbracket ubqcv \vdash_T \llbracket k \rrbracket uq'bc'v \qquad\qquad (\nabla(q, c) = (q', c')).$$

Once the machine reaches the last symbol, it flips the tape, appends 2^k blank symbols and proceeds with the next sweep:

$$\llbracket k \rrbracket qcw \vdash_T \llbracket k+1 \rrbracket \Box^{2^k} w^R qc$$

A sweeping Turing machine never halts; at the end of each sweep, it may flicker *by entering a state from $\mathcal{F}$. Define the set of numbers accepted by T as $S(T) = \{\, k \mid \llbracket 1 \rrbracket \Box q_0 \Box \vdash_T^* \llbracket k \rrbracket q_{\mathrm{f}} cw \text{ for } q_{\mathrm{f}} \in \mathcal{F} \,\}$.*

A sweeping Turing machine is simulated by the following trellis automaton with feedback over a one-symbol alphabet.

Construction 1. Let $T = (\Gamma, \mathcal{Q}, q_0, \nabla, \mathcal{F})$ be a sweeping Turing machine. Construct a trellis automaton with feedback $M = (\{a\}, Q, I, J, \delta, F)$ as follows. Its set of states is $Q = \big\{ {}^{Z}\mathbf{p}^{x}_{y} \bigm| x, y \in \Gamma \cup \mathcal{Q}\Gamma,\ Z \in \{\circ, \bullet\} \big\} \cup \big\{ {}^{Z}\mathbf{q}^{x} \bigm| x \in \Gamma \cup \mathcal{Q}\Gamma,\ Z \in \{\circ, \bullet\} \big\} \cup \{\mathbf{r}\}$. Each superscript x represents a tape symbol at the current position, which is augmented with a state, if the head is in this position. Each subscript y similarly contains a symbol and possibly a state, representing the contents of some other tape square, which is being sent as a signal to the left. A bullet marker "$\bullet$" marks the beginning of the tape, whereas each state ${}^{Z}\mathbf{p}^{x}_{y}$ or ${}^{Z}\mathbf{q}^{x}$ with $Z = \circ$ shall be denoted by $\mathbf{p}^{x}_{y}$ and $\mathbf{q}^{x}$, respectively.

Let $I(a) = \mathbf{p}^{\Box}_{\Box q_0}$, $J(\mathbf{r}, a) = \mathbf{p}^{\Box}_{\Box}$, and $J({}^{Z}\mathbf{p}^{x}_{y}, a) = {}^{Z}\mathbf{q}^{y}$. For all $x, y, x', y' \in \Gamma \cup \mathcal{Q}\Gamma$ and $Z, Z' \in \{\circ, \bullet\}$, the following transitions are defined in δ:

$$\delta\left({}^{Z}\mathbf{q}^{x},\quad {}^{Z'}\mathbf{q}^{x'}\right) = {}^{Z}\mathbf{q}^{x} \qquad \text{(propagation; } x, x' \in \Gamma, \text{ and } x \in \mathcal{Q}\Gamma \text{ with } Z = \bullet)$$

$$\delta\left({}^{Z}\mathbf{q}^{x},\quad {}^{Z'}\mathbf{p}^{x'}_{y'}\right) = {}^{Z'}\mathbf{p}^{x}_{y'} \qquad \text{(propagation)}$$

$$\delta\left(\mathbf{p}^{x}_{y},\quad {}^{Z'}\mathbf{q}^{x'}\right) = \mathbf{r} \qquad \text{(r-column)}$$

$$\delta\left({}^{Z}\mathbf{p}^{x}_{y},\quad \mathbf{r}\right) = \mathbf{p}^{\Box}_{x} \qquad \text{(reflection)}$$

$$\delta\left({}^{Z}\mathbf{p}^{x}_{y},\quad {}^{Z'}\mathbf{p}^{x'}_{y'}\right) = {}^{Z'}\mathbf{p}^{x}_{y'} \qquad \text{(propagation)}$$

$$\delta\left(\mathbf{r},\quad {}^{Z'}\mathbf{p}^{x'}_{y'}\right) = {}^{\bullet}\mathbf{p}^{x'}_{y'} \qquad \text{(new region in top diagonal)}$$

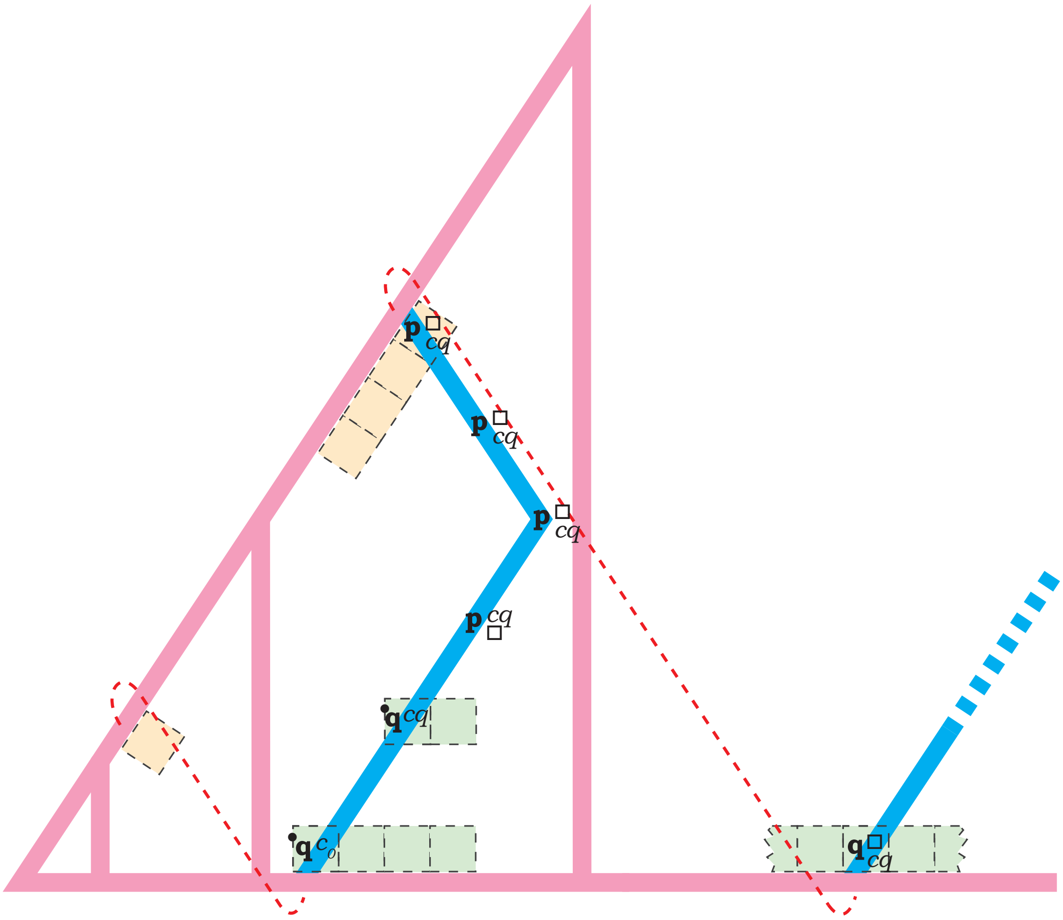
$$\delta\left(\mathbf{r},\quad {}^{Z'}\mathbf{q}^{x'}\right) = \mathbf{q}^{\Box} \qquad \text{(first q-column after r-column)}$$

A transition $\nabla(q, c) = (q', c)$ of the Turing machine is simulated as follows:

$$\begin{aligned} &\delta\,(\mathbf{q}^{cq}, \quad \mathbf{q}^{y}) = \mathbf{q}^{c'} && \text{(rewriting the symbol; } y \in \Gamma) \\ &\delta\,(\mathbf{q}^{x}, \quad \mathbf{q}^{cq}) = \mathbf{q}^{xq'} && \text{(moving the head; } x \in \Gamma) \end{aligned}$$

The set of accepting states is $F = \{\, \mathbf{p}^{\square}_{cq_{\mathrm{f}}} \mid c \in \Gamma,\ q_{\mathrm{f}} \in \mathcal{F} \,\}$.

The first thing to note about this construction is that if all attributes attached to the letters p, q, r are discarded, then the resulting automaton is exactly the one from Example 3. This ensures the overall partition of the computation into regions illustrated in Figure 2.



$\mathbf{p}^{\square}_{cq}$
$\mathbf{p}^{\square}_{cq}$
$\mathbf{p}^{\square}_{cq}$
$\mathbf{p}^{cq}_{\square}$
$\mathbf{q}^{cq}$
$\mathbf{q}^{c_0}$
$\mathbf{q}^{\square}_{cq}$

Figure 3: How a trellis automaton given by Construction 1 simulates a step of a Turing machine.

Each region corresponds to a sweep of the Turing machine. The bottom row of states contains the machine's configuration in the beginning of the sweep, where each state $\mathbf{q}^{x}$ holds the symbol in one square of the tape. The rightmost cell is $\mathbf{q}^{xq}$, and it contains the state of the Turing machine, while the leftmost cell is marked by a bullet ($^{\bullet}\mathbf{q}^x$). Each of the several rows above holds the tape contents after another step of computation. After $2^k - 1$ steps of simulation the head reaches the leftmost square, which marks the end of the current sweep.

Then, each tape symbol is propagated by a signal to the right using the states $\mathbf{p}_y^x$. Every such state holds two symbols: x is carried to the right, to be reflected off the right border, and y is a leftbound symbol that has already been reflected. As a result, the top diagonal border is filled with the states of the form $\mathbf{p}_y^x$, and their subscripts y form the resulting contents of the tape, reversed. These symbols are sent to the next region by the function J.

With this simulation running, the last state $q \in \mathcal{Q}$ reached by the Turing machine upon completing each k-th sweep shall always end up in a predefined position exactly in the middle of the top diagonal border. It will be $\Delta(\varepsilon\langle a^{2^{k+2}+2^{k+1}-2}\rangle) = \mathbf{p}_{cq}^{\square}$, and the trellis automaton with feedback accepts this string if and only if $q \in \mathcal{F}$.

Example 4. Consider a sweeping Turing machine $T = (\Gamma, \mathcal{Q}, q_0, \nabla, \mathcal{F})$, with $\Gamma = \{c, \square\}$, $\mathcal{Q} = \{q_0, q_1\}$ and $\mathcal{F} = \{q_1\}$. The transition function ∇ is defined as follows:

$$\begin{aligned}
\nabla(q_0, \square) &= (q_0, c) \\
\nabla(q_1, \square) &= (q_1, c) \\
\nabla(q_0, c) &= (q_1, c) \\
\nabla(q_1, c) &= (q_0, c)
\end{aligned}$$

The machine is started on a blank tape in the state q_0, that is, its initial configuration is $[\![1]\!]\square\mathbf{q_0}\square$. After the first move the configuration is

$$[\![1]\!]\square\mathbf{q_0}\square \vdash_T [\![1]\!]\mathbf{q_0}\square c.$$

After this step the machine flips its tape and makes three moves as follows:

$$\begin{gathered}
[\![2]\!]\square\square c\mathbf{q_0}\square \vdash_T \\
[\![2]\!]\square\square\mathbf{q_0}cc \vdash_T \\
[\![2]\!]\square\mathbf{q_1}\square cc \vdash_T \\
[\![2]\!]\mathbf{q_1}\square ccc.
\end{gathered}$$

Then the machine again flips its tape and makes 7 steps:

$$\begin{aligned}
\llbracket 3\rrbracket \Box\Box\Box\Box ccc\mathbf{q_1}\Box &\vdash_T \\
\llbracket 3\rrbracket \Box\Box\Box\Box cc\mathbf{q_1}c &\vdash_T \\
\llbracket 3\rrbracket \Box\Box\Box\Box c\mathbf{q_0}ccc &\vdash_T \\
\llbracket 3\rrbracket \Box\Box\Box\Box \mathbf{q_1}cccc &\vdash_T \\
\llbracket 3\rrbracket \Box\Box\Box\mathbf{q_0}\Box cccc &\vdash_T \\
\llbracket 3\rrbracket \Box\Box\mathbf{q_0}\Box ccccc &\vdash_T \\
\llbracket 3\rrbracket \Box\mathbf{q_0}\Box cccccc &\vdash_T \\
\llbracket 3\rrbracket \mathbf{q_0}\Box ccccccc.&
\end{aligned}$$

Thus, while traversing the tape, if the machine is scanning a blank, it rewrites it with a symbol c; and if it is scanning a symbol c, it switches its state (that is, changes from q_0 to q_1 and vice versa).

After every k-th sweep, that is, after each step with a number $2^n - 1$, the tape contains k symbols c and the head of the machine is scanning a blank symbol. If n is even, the machine is in the state q_0; otherwise the state is q_1. That is, the machine "flickers" after every second sweep.

Consider the trellis automaton with feedback, obtained by Construction 1 for the machine T.

The initial state of the automaton is $\mathbf{p}^{\Box}_{\Box q_0}$, which allows assigning the value $\mathbf{q}^{\Box q_0}$ to the second element in the bottom row of the trellis. This value represents the contents of the tape of T, which is being processed by the automaton.

In the first region, the entire configuration is $\mathbf{q_0}\Box$, that is, it consists of a unique square. Thus, the automaton cannot yet simulate a step of the machine, and all it has to do is to copy this one-square tape to the top diagonal border of the first region. The contents of this square are put in the state $\mathbf{p}^{\Box q_0}_{\Box}$ and then reflected off the vertical r-column in the state $\mathbf{p}^{\Box}_{\Box q_0}$, which is copied to the top border. This state is surrounded by two other states, ${}^{\bullet}\mathbf{p}^{\Box}_{\Box}$ and $\mathbf{p}^{\Box}_{\Box}$, which represent blank squares.

The final tape contents in the first region are then transferred to the second region through the feedback channel: the information from the state $\mathbf{p}^{\Box}_{\Box q_0}$ in position $(1, 4)$ is copied to the state $\mathbf{q}^{\Box q_0}$ in position $(5, 5)$, while another state ${}^{\bullet}\mathbf{p}^{\Box}_{\Box}$ in position $(1, 3)$ produces the state ${}^{\bullet}\mathbf{q}^{\Box}$ in position $(4, 4)$.

Now the trellis automaton can simulate a transition $\nabla(q_0, \Box) = (q_0, c)$ of T and the new contents of the tape are represented by the nodes ${}^{\bullet}\mathbf{q}^{\Box q_0}$ and $\mathbf{q}^{c}$ in positions $(4, 5)$ and $(5, 6)$, respectively. The former state has $Z = \bullet$, marking the left end of the tape, and hence no further steps of the Turing machine shall be simulated in this region (as all the transitions of the trellis automaton simulating the Turing machine require $Z = \circ$).

The data in the nodes in positions $(4, 5)$ and $(5, 6)$ are propagated first to the right, and, after reflecting off $\mathbf{r}$, to the left. The subsequent simulation

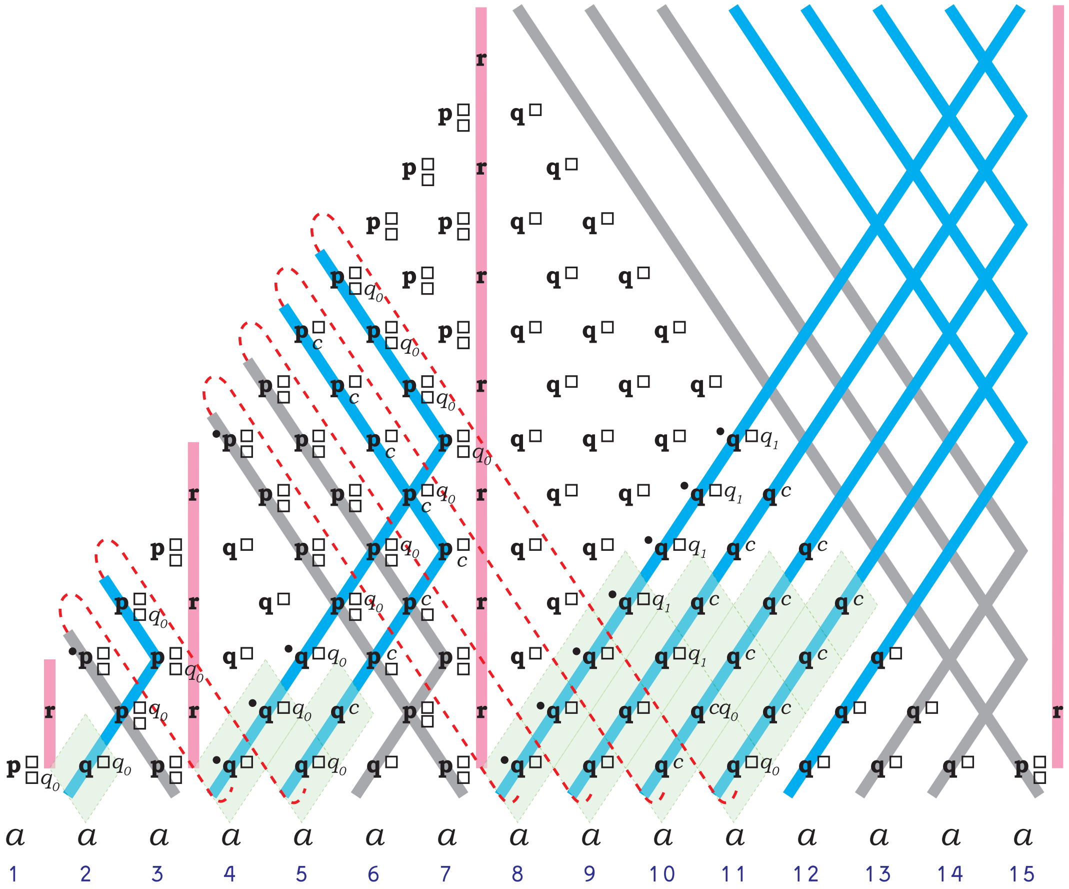


Figure 4: Computation of a trellis automaton M with feedback simulating the Turing machine T. The input string of M is a^{2^4-2}.

of the next sweep of the Turing machine in the third region is illustrated in the figure.

The correctness of Construction 1 is established in the following lemma, which describes the contents of each individual cell.

Lemma 6. *Let $T = (\Gamma, \mathcal{Q}, q_0, \nabla, \mathcal{F})$ be a sweeping Turing machine, and let $M = (\{a\}, Q, I, J, \delta, F)$ be a trellis automaton with feedback obtained in Construction 1.*

Consider some k-th sweep of M and denote $X_1^\ell \ldots X_{2^{k-2}}^\ell$ the tape contents after each $\ell \in \{1, \ldots, 2^{k-2}-1\}$ step in the sweep. Each symbol X_i^ℓ is either a symbol of the tape alphabet, or a pair (b, q), where $b \in \Gamma$ and $q \in \mathcal{Q}$, meaning that the head of the machine is scanning symbol b in state q.

Then:

- *For all $(i, j) \in \mathcal{U}_1$, $M[i, j] = \mathbf{r}$.*
- *For all $(i, j) \in \mathcal{U}_2$, $M[i, j] = {}^{Z}\mathbf{q}^{s}$, where $s = \square$ if $i \in \{2^{k-1}, \ldots, 3 \cdot 2^{k-2} - 1\}$, $j \in \{2^{k-1}, \ldots, 2^k - 2\}$ and $i \leqslant j$. Otherwise, $s = X_{i-2^{k-1}+1}^{t}$,*

with $t = j-i+1$ *if* $j-i < 2^{k-1}$*, and* $t = 2^{k-1}$ *if* $j-i \geqslant 2^{k-1}$*. Moreover,* $Z = \bullet$ *when* $i = 2^{k-1}$*.*

- *For all* $(i,j) \in \mathcal{U}_3$*,* $M[i,j] = {}^{Z}\mathbf{p}^{s}_{s'}$*, where* $s = s' = \square$ *for* $i \in \{1, \dots, 2^{k-1}-1\}$ *and* $j \in \{3 \cdot 2^{k-1}-2, \dots, 2^{k+1}-4\}$*, and* $s = X^{2^{k-2}-2}_{i-2^k+1}$*,* $s' = X^{2^{k-2}-2}_{j-5\cdot 2^{k-2}+2}$ *for the rest of pairs. Moreover,* $Z = \bullet$ *if* $i = 1$ *and* $j = 2^k - 1$*.*

Lemma 6 can be proved by a straightforward induction, inferring the state in each cell from the previously determined values of the neighbouring cells.

Theorem 3. *Let* $T = (\Gamma, \mathcal{Q}, q_0, \nabla, \mathcal{F})$ *be a sweeping Turing machine and let* $M = (\{a\}, Q, I, J, \delta, F)$ *be a trellis automaton with feedback obtained in Construction 1. Then* $L(M) = \{\, a^{2^{k+2}+2^{k+1}-2} \mid k \in S(T) \,\}$*.*

8 Implications

The simulation of Turing machines by a trellis automaton with feedback over a one-symbol alphabet is useful for proving undecidability of basic decision problems for these automata. Due to Theorem 2, the same undecidability results equally hold for linear grammars with contexts.

The first decision problem is testing whether the language recognized by an automaton (or defined by a grammar) is empty. The undecidability of the *emptiness problem* follows from Theorem 3. To be precise, the problem is complete for the complements of the r.e. sets.

Theorem 4. *The emptiness problem for linear grammars with left contexts over a one-symbol alphabet is* Π^0_1*-complete. It remains in* Π^0_1 *for any alphabets.*

Proof. The non-emptiness problem is clearly recursively enumerable. because one can simulate a trellis automaton with feedback on all inputs, accepting if it ever accepts. If the automaton accepts no strings, the algorithm does not halt.

The Π^0_1-hardness is proved by reduction from the Turing machine halting problem. Given a machine T and an input w, construct a sweeping Turing machine T_w, which first prints w on the tape (over $1 + \log|w|$ sweeps, using around $|w|$ states), and then proceeds by simulating T, using one sweep for each step of T. If the simulated machine T ever halts, then T_w changes into a special state q_{f} and continues moving its head until the end of the current sweep.

Construct a trellis automaton with feedback M simulating the machine T_w according to Theorem 3, and define its set of accepting states as $F = \{\, \mathbf{p}^{\square}_{cq_{\mathrm{f}}} \mid c \in \Sigma \,\}$. Then, by the theorem, M accepts some string a^{ℓ} if and only if T_w ever enters the state q_{f}, which is in turn equivalent to T's halting on w. □

The second slightly more difficult undecidability result asserts that testing the finiteness of a language generated by a given grammar is complete for the second level of the arithmetical hierarchy.

Theorem 5. *The finiteness problem for linear grammars with left contexts over a one-symbol alphabet is Σ_2^0-complete. It remains Σ_2^0-complete for any alphabet.*

Proof (a sketch). Reduction from the finiteness problem for a Turing machine, which is Σ_2^0-complete, see Rogers [15, §14.8]. Given a Turing machine T, construct a sweeping Turing machine T', which simulates T running on all inputs, with each simulation using a segment of the tape. Initially, T' sets up to simulate T running on ε, and then it regularly begins new simulations. Every time one of the simulated instances of T accepts, the constructed machine "flickers" by entering an accepting state in the end of one of its sweeps. Construct a trellis automaton with feedback M corresponding to this machine. Then $L(M)$ is finite if and only if $L(T)$ is finite. □

9 Conclusion

At the first glance, linear grammars with contexts seem like a strange model. However, they are motivated by the venerable idea of a rule applicable in a context, which is worth being investigated. Also, trellis automata with feedback at the first glance seem like a far-fetched extension of cellular automata. Its motivation comes from the understanding of a trellis automaton as a circuit with uniform connections [5], to which one can add a new type of connections. Both models are particularly interesting for being equivalent.

A suggested topic for future research is to investigate the main ideas in the literature on trellis automata [5, 6, 7, 17] and see whether they can be extended to trellis automata with feedback, and hence to linear grammars with contexts.

References

[1] T. Aizikowitz, M. Kaminski, "LR(0) conjunctive grammars and deterministic synchronized alternating pushdown automata", *Computer Science in Russia* (CSR 2011, St. Petersburg, Russia, 14–18 June 2011), LNCS 6651, 345–358.

[2] M. Barash, "Recursive descent parsing for grammars with contexts", *SOFSEM 2013 student research forum* (Špindlerův Mlýn, Czech Republic, 26-31 January, 2013), Local Proceedings II, 10–21, Institute of Computer Science AS CR, 2013.

[3] M. Barash, A. Okhotin, "Defining contexts in context-free grammars", *Language and Automata Theory and Applications* (LATA 2012, A Coruña, Spain, 5–9 March 2012), LNCS 7183, 106–118.

[4] M. Barash, A. Okhotin, "An extension of context-free grammars with one-sided context specifications", submitted (September 2013).

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

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

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

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

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

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

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

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

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

[14] A. Okhotin, "Improved normal form for grammars with one-sided contexts", *Descriptional Complexity of Formal Systems* (DCFS 2013, London, Ontario, Canada, 22-25 July 2013), LNCS 8031, 205–216.

[15] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability*, 1967.

[16] W. C. Rounds, “LFP: A logic for linguistic descriptions and an analysis of its complexity”, *Computational Linguistics*, 14:4 (1988), 1–9.

[17] V. Terrier, “On real-time one-way cellular array”, *Theoretical Computer Science*, 141:1–2 (1995), 331–335.

[18] I. Törmä, personal communication, February 2013.

[19] S. Yu, “A property of real-time trellis automata”, *Discrete Applied Mathematics*, 15:1 (1986), 117–119.

TURKU
CENTRE *for*
COMPUTER
SCIENCE

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics

Turku School of Economics
- Institute of Information Systems Sciences

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

ISBN 978-952-12-2928-2
ISSN 1239-1891