Context-Sensitive Fusion Grammars Are Universal

Aaron Lye(✉)

Department of Mathematics, University of Bremen,
P.O.Box 33 04 40, 28334 Bremen, Germany
lye@math.uni-bremen.de

Abstract. Context-sensitive fusion grammars are a special case of context-dependent fusion grammars where a rule has only a single positive context condition instead of finite sets of positive and negative context conditions. They generate hypergraph languages from start hypergraphs via successive applications of context-sensitive fusion rules and multiplications of connected components, as well as a filtering mechanism to extract terminal hypergraphs from derived hypergraphs in a certain way. The application of a context-sensitive fusion rule consumes two complementarily labeled hyperedges and identifies corresponding attachment vertices provided that the context condition holds. In this paper, we show that the Post correspondence problem can be formulated very intuitively by such a grammar. Furthermore, we prove that these grammars can generate all recursively enumerable string languages (up to representation of strings as graphs) and are universal in this respect.

Keywords: Graph transformation · Context-sensitive fusion grammars · Recursively enumerable languages · Chomsky grammar · Post correspondence problem

1 Introduction

In 2017 we introduced fusion grammars as generative devices on hypergraphs (cf. [2]). They are motivated by the observation, that one encounters various fusion processes in various scientific fields like DNA computing, chemistry, tiling, fractal geometry, visual modeling and others. The common principle is that a few small entities may be copied and fused to produce more complicated entities. Besides hypergraph language generation they can be used to model and solve interesting decision problems, e.g., in [3] it is shown that the Hamiltonian path problem can be solved efficiently by a respective fusion grammar due to the massive parallelism in a way that mimics Adleman's famous experiment in DNA computing (cf. [1]). In this paper, we show that the Post correspondence problem (PCP, cf. [6]), which is well-known to be undecidable, can be expressed very intuitively by means of fusion and its solvability by using context-sensitive fusion rules. Hence, undeciability results carry over to context-sensitive fusion grammars. Recently, we showed that context-dependent fusion grammars (introduced in [4]) are powerful enough to simulate Turing machines (cf. [5]). In this

A. Leporati et al. (Eds.): LATA 2020, LNCS 12038, pp. 275–286, 2020.
https://doi.org/10.1007/978-3-030-40608-0_19

paper, we show that one can do much better. We show that rules with a single positive context condition are sufficient. To prove this, a known result of formal language theory is used, which is, that each recursively enumerable string language is a (left) quotient of two linear languages. In our construction we employ the same recognition mechanism as the one for PCP. Throughout in the proofs we are actually operating on graphs. As graphs are a subclass of hypergraphs the results hold for the general case.

The paper is organized as follows. In Sect. 2, basic notions and notations of hypergraphs are recalled. Section 3 introduces the notions of context-sensitive fusion grammars. In Sect. 4 we present a reduction of the Post correspondence problem to the membership and emptiness problem for context-sensitive fusion grammars. Afterwards, we prove that context-sensitive fusion grammars can generate all recursively enumerable string languages (up to representation) in Sect. 5. Section 6 concludes the paper pointing out some open problems.

2 Preliminaries

A *hypergraph* over a given label alphabet Σ is a system $H = (V, E, s, t, lab)$ where V is a finite set of *vertices*, E is a finite set of *hyperedges*, $s, t \colon E \to V^*$ are two functions assigning to each hyperedge a sequence of *sources* and *targets*, respectively, and $lab \colon E \to \Sigma$ is a function, called *labeling*. The components of $H = (V, E, s, t, lab)$ may also be denoted by V_H, E_H, s_H, t_H, and lab_H respectively. The class of all hypergraphs over Σ is denoted by $\mathcal{H}_\Sigma$.

Let $H \in \mathcal{H}_\Sigma$, and let $\equiv$ be an equivalence relation on V_H. Then the *fusion of the vertices in H with respect to* $\equiv$ yields the (quotient) hypergraph $H/\!\equiv\, = (V_H/\!\equiv, E_H, s_{H/\equiv}, t_{H/\equiv}, lab_H)$ with the set of equivalence classes $V_H/\!\equiv\, = \{[v] \mid v \in V_H\}$ and $s_{H/\equiv}(e) = [v_1]\cdots[v_{k_1}]$, $t_{H/\equiv}(e) = [w_1]\cdots[w_{k_2}]$ for each $e \in E_H$ with $s_H(e) = v_1 \cdots v_{k_1}$, $t_H(e) = w_1 \cdots w_{k_2}$.

Given $H, H' \in \mathcal{H}_\Sigma$, a *hypergraph morphism* $g \colon H \to H'$ consists of two mappings $g_V \colon V_H \to V_{H'}$ and $g_E \colon E_H \to E_{H'}$ such that $s_{H'}(g_E(e)) = g_V^*(s_H(e))$, $t_{H'}(g_E(e)) = g_V^*(t_H(e))$ and $lab_{H'}(g_E(e)) = lab_H(e)$ for all $e \in E_H$, where $g_V^* \colon V_H^* \to V_{H'}^*$ is the canonical extension of g_V, given by $g_V^*(v_1 \cdots v_n) = g_V(v_1) \cdots g_V(v_n)$ for all $v_1 \cdots v_n \in V_H^*$.

Given $H, H' \in \mathcal{H}_\Sigma$, H is a *subhypergraph* of H', denoted by $H \subseteq H'$, if $V_H \subseteq V_{H'}$, $E_H \subseteq E_{H'}$, $s_H(e) = s_{H'}(e)$, $t_H(e) = t_{H'}(e)$, and $lab_H(e) = lab_{H'}(e)$ for all $e \in E_H$.

Let $H' \in \mathcal{H}_\Sigma$ as well as $V \subseteq V_{H'}$ and $E \subseteq E_{H'}$. Then the *removal* of (V, E) from H' given by $H = H' - (V, E) = (V_{H'} - V, E_{H'} - E, s_H, t_H, lab_H)$ with $s_H(e) = s_{H'}(e)$, $t_H(e) = t_{H'}(e)$ and $lab_H(e) = lab_{H'}(e)$ for all $e \in E_{H'} - E$ defines a subgraph $H \subseteq H'$ if $s_{H'}(e), t_{H'}(e) \in (V_{H'} - V)^*$ for all $e \in E_{H'} - E$. We will use removals of the form $(\emptyset, E)$ below.

Let $H \in \mathcal{H}_\Sigma$ and let $att(e)$ be the set of source and target vertices for $e \in E_H$. H is *connected* if for each $v, v' \in V_H$, there exists a sequence of triples $(v_1, e_1, w_1) \ldots (v_n, e_n, w_n) \in (V_H \times E_H \times V_H)^*$ such that $v = v_1, v' = w_n$ and $v_i, w_i \in att(e_i)$ for $i = 1, \ldots, n$ and $w_i = v_{i+1}$ for $i = 1, \ldots, n-1$. A subgraph

C of H is a *connected component* of H if it is connected and there is no larger connected subgraph, i.e., $C \subseteq C' \subseteq H$ and C' connected implies $C = C'$. The set of connected components of H is denoted by $\mathcal{C}(H)$.

Given $H, H' \in \mathcal{H}_\Sigma$, the *disjoint union* of H and H' is denoted by $H + H'$. It is defined by the disjoint union of the underlying sets (also denoted by $+$). The disjoint union of H with itself k times is denoted by $k \cdot H$. We use the *multiplication* of H defined by means of $\mathcal{C}(H)$ as follows. Let $m\colon \mathcal{C}(H) \to \mathbb{N}$ be a mapping, called *multiplicity*, then $m \cdot H = \sum\limits_{C \in \mathcal{C}(H)} m(C) \cdot C$.

A string is represented by a simple path where the sequence of labels along the path equals the given string. Let Σ be a label alphabet. Let $w = x_1 \dots x_n \in \Sigma^*$ for $n \geq 1$ and $x_i \in \Sigma$ for $i = 1, \dots, n$. Then the *string graph* of w is defined by $sg(w) = (\{0\} \cup [n], [n], s_w, t_w, lab_w)$ with $s_w(i) = i - 1, t_w(i) = i$ and $lab(i) = x_i$ for $i = 1, \dots, n$. The string graph of the empty string λ, denoted by $sg(\lambda)$, is the discrete graph with a single node 0. Obviously, there is a one-to-one correspondence between Σ^* and $sg(\Sigma^*) = \{sg(w) \mid w \in \Sigma^*\}$. For technical reasons, we need the extension of a string graph $sg(w)$ for some $w \in \Sigma^*$ by a s-labeled edge bending from the begin node 0 to the end node n, where n is the length of w. The resulting graph is denoted by $sg(w)_s$.

3 Context-Sensitive Fusion Grammars

In this section, we introduce context-sensitive fusion grammars. These grammars generate hypergraph languages from start hypergraphs via successive applications of context-sensitive fusion rules, multiplications of connected components, and a filtering mechanism. Such a rule is applicable if the positive context-condition holds. Its application consumes the two hyperedges and fuses the sources of the one hyperedge with the sources of the other as well as the targets of the one with the targets of the other.

Definition 1. *$F \subseteq \Sigma$ is a* fusion alphabet *if it is accompanied by a* complementary fusion alphabet $\overline{F} = \{\overline{A} \mid A \in F\} \subseteq \Sigma$, *where $F \cap \overline{F} = \emptyset$ and $\overline{A} \neq \overline{B}$ for $A, B \in F$ with $A \neq B$ and a* type function $type\colon F \cup \overline{F} \to (\mathbb{N} \times \mathbb{N})$ *with $type(A) = type(\overline{A})$ for each $A \in F$.*

For each $A \in F$, the fusion rule $fr(A)$ *is the hypergraph with $V_{fr(A)} = \{v_i, v_i' \mid i = 1, \dots, k_1\} \cup \{w_j, w_j' \mid j = 1, \dots, k_2\}$, $E_{fr(A)} = \{e, \overline{e}\}$, $s_{fr(A)}(e) = v_1 \cdots v_{k_1}$, $s_{fr(A)}(\overline{e}) = v_1' \cdots v_{k_1}'$, $t_{fr(A)}(e) = w_1 \cdots w_{k_2}$, $t_{fr(A)}(\overline{e}) = w_1' \cdots w_{k_2}'$, and $lab_{fr(A)}(e) = A$ and $lab_{fr(A)}(\overline{e}) = \overline{A}$.*

The application of $fr(A)$ to a hypergraph $H \in \mathcal{H}_\Sigma$ proceeds according to the following steps: (1) Choose a matching hypergraph morphism $g\colon fr(A) \to H$. *(2) Remove the images of the two hyperedges of $fr(A)$ yielding $X = H - (\emptyset, \{g(e), g(\overline{e})\})$. (3) Fuse the corresponding source and target vertices of the removed hyperedges yielding the hypergraph $H' = X/{\equiv}$ where $\equiv$ is generated by the relation $\{(g(v_i), g(v_i')) \mid i = 1, \dots, k_1\} \cup \{(g(w_j), g(w_j')) \mid j = 1, \dots, k_2\}$. The application of $fr(A)$ to H is denoted by $H \underset{fr(A)}{\Longrightarrow} H'$ and called a* direct derivation.

A context-sensitive fusion rule *is a tuple* $csfr = (fr(A), c\colon fr(A) \to C)$ *for some* $A \in F$ *where* c *is a hypergraph morphism with domain* $fr(A)$ *mapping into a finite context* C.

The rule $csfr$ *is applicable to some hypergraph* H *via a matching morphism* $g\colon fr(A) \to H$ *if there exists a hypergraph morphism* $h\colon C \to H$ *such that* h *is injective on the set of hyperedges and* $h \circ c = g$.

If $csfr$ *is applicable to* H *via* g, *then the direct derivation* $H \underset{csfr}{\Longrightarrow} H'$ *is the direct derivation* $H \underset{fr(A)}{\Longrightarrow} H'$.

Remark 1. 1. In this paper, we only make use of the case where every hyperedge has one source and one target vertex. Hence, fusion rules are of the form $\bullet\!\rightarrow\!\boxed{A}\!\rightarrow\!\bullet \;\; \bullet\!\rightarrow\!\boxed{\overline{A}}\!\rightarrow\!\bullet$. The *type* is therefore omitted throughout the paper.
2. The applications of $fr(A)$ and $(fr(A), id)$ are equivalent. We use the first as an abbreviation for the latter. We call these rules *context-free fusion rules.*

Example 1. Let $F = \{a_1, a_2, a_3\}$. Define $reduce(x) = (fr(x), fr(x) \to$
$$\begin{tikzcd}[column sep=small, row sep=tiny] & \boxed{\overline{x}} \arrow[r] & \overset{c_2}{\bullet} \\ \overset{c_1}{\bullet} \arrow[ur] \arrow[dr] & & \\ & \boxed{x} \arrow[r] & \overset{c_3}{\bullet} \end{tikzcd}$$
$)$ for each $x \in F$ where the morphism is uniquely defined by the labels and maps the vertices as follows: $v_1 \mapsto c_1, v_2 \mapsto c_1, w'_1 \mapsto c_2, w_1 \mapsto c_3$. Consider the graph $G =$
$$\begin{tikzcd}[column sep=small, row sep=tiny] & \boxed{\overline{a}_1} \arrow[r] & \overset{g_2}{\bullet} \arrow[r] & \boxed{\overline{a}_2} \arrow[r] & \overset{g_4}{\bullet} \arrow[r] & \boxed{\overline{a}_3} \arrow[r] & \overset{g_6}{\bullet} \\ \overset{g_1}{\bullet} \arrow[ur] \arrow[dr] & & & & & & \\ & \boxed{a_1} \arrow[r] & \overset{g_3}{\bullet} \arrow[r] & \boxed{a_3} \arrow[r] & \overset{g_5}{\bullet} \arrow[r] & \boxed{a_2} \arrow[r] & \overset{g_7}{\bullet} \end{tikzcd}$$
. Only $reduce(a_1)$ is applicable because the other complementarily labeled edges do not share a common source vertex. The matching morphism g maps the edges labeled $a_1, \overline{a}_1$, resp. in $fr(a_1)$ to the a_1-labeled (resp, $\overline{a}_1$-labeled) edges in G; vertices are mapped respectively: $v_1 \mapsto g_1, v_2 \mapsto g_1, w'_1 \mapsto g_2, w_1 \mapsto g_3$. The morphism $h\colon C \to G$ exists (inclusion morphism). Then $G \underset{reduce(a_1)}{\Longrightarrow}$
$$\begin{tikzcd}[column sep=small, row sep=tiny] & & \boxed{\overline{a}_2} \arrow[r] & \overset{[g_4]}{\bullet} \arrow[r] & \boxed{\overline{a}_3} \arrow[r] & \overset{[g_6]}{\bullet} \\ \overset{[g_1]}{\bullet} & \overset{[g_2]}{\bullet} \arrow[ur] \arrow[dr] & & & & \\ & & \boxed{a_3} \arrow[r] & \overset{[g_5]}{\bullet} \arrow[r] & \boxed{a_2} \arrow[r] & \overset{[g_7]}{\bullet} \end{tikzcd}$$
where $g_2 \equiv g_3$. Afterwards, no further context-sensitive fusion rule is applicable.

Given a finite hypergraph, the set of all possible successive fusions is finite as fusion rules never create anything. To overcome this limitation, arbitrary multiplications of disjoint components within derivations are allowed. The generated language consists of the terminal part of all resulting connected components that contain no fusion symbols and at least one marker symbol, where marker symbols are removed in the end. These marker symbols allow us to distinguish between wanted and unwanted terminal components.

Definition 2. *A* context-sensitive fusion grammar *is a system* $CSFG = (Z, F, M, T, P)$ *where* $Z \in \mathcal{H}_{F \cup \overline{F} \cup T \cup M}$ *is a* start hypergraph *consisting of a finite number of connected components,* F *is a finite fusion alphabet,* M *with* $M \cap (F \cup \overline{F}) = \emptyset$ *is a finite set of* markers, T *with* $T \cap (F \cup \overline{F}) = \emptyset = T \cap M$ *is a finite set of* terminal labels, *and* P *is a finite set of context-sensitive fusion rules.*

A direct derivation $H \Longrightarrow H'$ *is either a context-sensitive fusion rule application* $H \underset{csfr}{\Longrightarrow} H'$ *for some* $csfr \in P$ *or a multiplication* $H \underset{m}{\Longrightarrow} m \cdot H$ *for some multiplicity* $m \colon \mathcal{C}(H) \to \mathbb{N}$. *A* derivation $H \overset{n}{\Longrightarrow} H'$ *of length* $n \geq 0$ *is a sequence of direct derivations* $H_0 \Longrightarrow H_1 \Longrightarrow \ldots \Longrightarrow H_n$ *with* $H = H_0$ *and* $H' = H_n$. *If the length does not matter, we may write* $H \overset{*}{\Longrightarrow} H'$.

$L(CSFG) = \{rem_M(Y) \mid Z \overset{*}{\Longrightarrow} H, Y \in \mathcal{C}(H) \cap (\mathcal{H}_{T \cup M} \setminus \mathcal{H}_T)\}$ *is the* generated language *where* $rem_M(Y)$ *is the terminal hypergraph obtained by removing all hyperedges with labels in* M *from* Y.

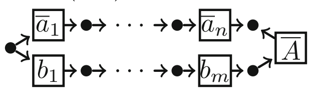
Remark 2. Let $CSFG = (Z, F, M, T, P)$ be a context-sensitive fusion grammar. If for every $A \in F$ a rule in P exists and every rule is context-free, then all rules are specified F and $CSFG$ is a fusion grammar as defined in [2]. P is obsolete.

4 A Context-Sensitive Fusion Grammar for the Post Correspondence Problem

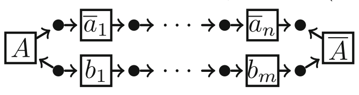
In this section, we model Post correspondence problems (PCPs) by means of context-sensitive fusion grammars in such a way that a PCP is solvable if the generated language of the corresponding grammar consists of a single vertex and that a PCP is not solvable if the language is empty. Therefore, it turns out that the emptiness problem and the membership problem for context-sensitive fusion grammars are undecidable.

The Post correspondence problem is defined as follow. Given a finite set of pairs $\{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$ with $u_i, v_i \in \Sigma^*$ for some finite alphabet Σ. Does there exist a sequence of indices $i_1 \cdots i_n$ with $n > 0$ such that $u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$? In terms of fusion, the pairs may be copied and fused in order to concatenate the strings. However, one needs a recognition mechanism to decide whether $u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$ or not. This recognition procedure is expressible by means of context-sensitive fusion.

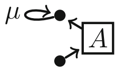
Construction 1. *Let* $S = \{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$ *with* $k \in \mathbb{N}$, $u_i, v_i \in \Sigma^*$ *be an instance of PCP. Let* $F = \Sigma + \{A\}$ *be a fusion alphabet with* $\overline{A} \notin \Sigma$. *Let* $P = \{fr(A)\} \cup \{reduce(x) \mid x \in \Sigma\}$ *where* $reduce(x)$ *be as in Example 1. For each* $(a, b) \in \Sigma^* \times \Sigma^*$ *where* $a = a_1 \cdots a_n$ *and* $b = b_1 \cdots b_m$ *define* $init(a, b) =$



and $cont(a, b) =$



. *Let* $A_\mu =$

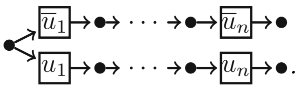


and $Z_S = \sum_{(a,b) \in S} init(a, b) + cont(a, b) + A_\mu$. *Then* $CSFG(S) = (Z_S, F, \{\mu\}, \emptyset, P)$ *is the to* S *corresponding context-sensitive fusion grammar.*

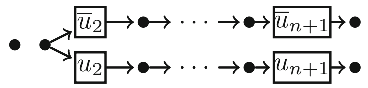
Theorem 1. *1.* $\bullet \in L(CSFG(S))$ *if and only if there exists a solution to* S.
2. $L(CSFG(S))$ *is either* $\{\bullet\}$ *or* $\emptyset$.

Corollary 1. *The membership and the emptiness problem for context-sensitive fusion grammars are undecidable.*

The proof of the theorem is based on the following lemmata.

Lemma 1. *Let $G = dsg(u_1 \cdots u_n, \overline{u}_1 \cdots \overline{u}_n)$ be the hypergraph consisting of two string graphs $sg(u_1 \cdots u_n)$ and $sg(\overline{u}_1 \cdots \overline{u}_n)$ with $u_1, \ldots, u_n \in \Sigma$ where the first vertex of both string graphs is the same. i.e.,*  *. Then $G \overset{n}{\Longrightarrow} [n+1]$ by applying $reduce(u_1), \ldots, reduce(u_n)$, where $[n+1]$ denotes the discrete graph with $n+1$ vertices and no edges.*

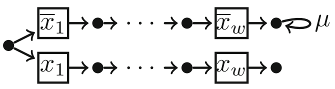
Proof. Induction base: $n = 0$. $dsg(\lambda, \lambda) = [1]$ because by definition $sg(\lambda)$ is the discrete graph $[1]$ by construction of dsg these two vertices are identified yielding the discrete graph $[1]$. Hence, $dsg(\lambda, \lambda) \overset{0}{\Longrightarrow} [1]$.

Induction step: Given $G = dsg(u_1 \cdots u_{n+1}, \overline{u}_1 \cdots \overline{u}_{n+1})$. Then $reduce(u_1)$ can be applied because by construction of $dsg(u_1 \cdots u_{n+1}, \overline{u}_1 \cdots \overline{u}_{n+1})$ the two complementary u_1- and $\overline{u}_1$-labeled hyperedges share a common source vertex yielding $G' =$  $= [1] + dsg(u_2 \cdots u_{n+1}, \overline{u}_2 \cdots \overline{u}_{n+1})$. Then by induction hypothesis $G' \overset{n}{\Longrightarrow} [1] + [n+1] = [n+2]$. □

Lemma 2. *Let $X_1 \underset{reduce(x)}{\Longrightarrow} X_2 \underset{fr(A)}{\Longrightarrow} X_3$ be a derivation in $CSFG(S)$. Then the two direct derivations can be interchanged yielding $X_1 \underset{fr(A)}{\Longrightarrow} X_2' \underset{reduce(x)}{\Longrightarrow} X_3$ for some X_2'.*

Proof. The statement follows directly from the fact that the two rules do not share fusion symbols such that they matches are hyperedge disjoint and that the context conditions of $reduce(x)$ only requires a commonly shared source for the two hyperedges. □

Proof (of Theorem 1). Let $S = \{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$. Let $i_1 \cdots i_n$ be a solution to S, i.e., $u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$. Let $m_1, \ldots, m_k$ be the number of occurrences of (u_j, v_j) in the sequence except the first. Then there exists a derivation $Z_S \underset{m}{\Longrightarrow} init(u_{i_1}, v_{i_1}) + cont(u_{i_2}, v_{i_2}) + \ldots + cont(u_{i_n}, v_{i_n}) + A_\mu$

$\overset{n-1}{\underset{fr(A)}{\Longrightarrow}} init(u_{i_1} u_{i_2} \cdots u_{i_n}, v_{i_1} v_{i_2} \cdots v_{i_n}) + A_\mu \underset{fr(A)}{\Longrightarrow}$ 

$\underset{reduce(x_1)}{\Longrightarrow} \cdots \underset{reduce(x_w)}{\Longrightarrow} [w] +$

where (1) $m(c) = 1$ for $c \in \{init(u_{i_1}, v_{i_1}), A_\mu\}, m(cont(u_j, v_j)) = m_j$ for $1 \leq j \leq k$ and $m(c) = 0$ otherwise; (2) the order in which the connected components are fused by applications of $fr(A)$ does not matter; (3) $x_1 \cdots x_w = u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$ with $x_j \in \Sigma$ because $i_1 \ldots i_n$ is a solution to S; and (4) the two connected complementary strings graphs can be erased by successive applications of $reduce(x)$ for suitable x due to Lemma 1. Hence, $\bullet \in L(CSFG(S))$.

Now let $\bullet \in L(CSFG(S))$. Then there exists a derivation $Z_S \overset{*}{\Longrightarrow} X + \mu \circlearrowright \bullet$ for some hypergraph X. A_μ is the only connected component with marker in the start hypergraph, therefore, $\mu \circlearrowright \bullet$ must stem from A_μ. The only possibility to get rid of the A-hyperedge without attaching a new one is the application of $fr(A)$ to A_μ and some $init(x_1 x_2 \cdots x_{w_1}, y_1 y_2 \cdots y_{w_2})$ with $x_j, y_j \in \Sigma$ where the latter connected component is obtained from respective multiplications and the successive fusion wrt $fr(A)$ to some $init(u_{i_1}, v_{i_1}) + cont(u_{i_2}, v_{i_2}) + \ldots + cont(u_{i_n}, v_{i_n})$ for some n and possibly applications of $reduce(x)$ for suitable x. Due to Lemma 2 all the applications of $reduce(x)$ can be shifted behind the applications of $fr(A)$ and due to [2, Corollary 1] all the multiplications can be done as initial derivation step. To obtain $\mu \circlearrowright \bullet$ the two connected complementary strings graphs must be erased by successive applications of $reduce(x_1), \ldots, reduce(x_{w_1})$. If $x_1 \cdots x_{w_1}$ is a proper prefix of $y_1 \cdots y_{w_2}$, i.e., $y_1 \cdots y_{w_2} = x_1 \cdots x_{w_1} y_{w_1+1} \cdots y_{w_2}$, then one gets $\mu \circlearrowright \bullet \rightarrow \boxed{y_{w_1+1}} \rightarrow \bullet \rightarrow \cdots \rightarrow \bullet \rightarrow \boxed{y_{w_2}} \rightarrow \bullet$, and analogously if $y_1 \cdots y_{w_2}$ is a proper prefix of $x_1 \cdots x_{w_1}$, then one gets $\bullet \rightarrow \boxed{\overline{x}_{w_2+1}} \rightarrow \bullet \rightarrow \cdots \rightarrow \bullet \rightarrow \boxed{\overline{x}_{w_1}} \rightarrow \bullet \circlearrowleft \mu$. This implies $w_1 = w_2$ and $y_i = x_i$ for $1 \le i \le w_1$ must hold. Because $x_1 \cdots x_{w_1} = u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$ and $n > 0$, $i_1 \cdots i_n$ is a solution to S.

The second statement is a direct consequence of the first. Other connected components do not contribute to the language due to the lack of μ-hyperedges. $\square$

5 Transformation of Chomsky Grammars into Context-Sensitive Fusion Grammars

In this section, we prove that context-sensitive fusion grammars can generate all recursively enumerable string languages. For every Chomsky grammar one can construct a corresponding context-sensitive fusion grammar such that the generated languages of the corresponding grammars coincide up to representation.

Construction 2. *Let $G = (N, T, P, S)$ be a Chomsky grammar. Let $T' = \{t' \mid t \in T\}$. Then $CSFG(G) = (Z, \{Y_0, Y_1, X_0, X_1, X_2, X_3, c\} + N + T', \{\mu\}, T, R)$ is the corresponding context-sensitive fusion grammar where*

$$Z = dsg(X_0, Y_0)_\mu + Z_= + Z_P,\ dsg(X_0, Y_0)_\mu = \bullet \leftarrow \boxed{Y_0} \leftarrow \bullet \rightarrow \boxed{X_0} \rightarrow \bullet \circlearrowleft \mu,$$
$$Z_= = sg(Y_1 \overline{ccc})_{\overline{Y_0}} + sg(\overline{c}\overline{S}\overline{cc})_{\overline{Y_1}} + \sum_{x \in N \cup T \cup \{c\}} sg(\overline{x} Y_1 \overline{x})_{\overline{Y_1}},$$

$$Z_P = \sum_{i=0}^{1} \sum_{x \in T} sg(x' X_i x)_{\overline{X_i}} + \sum_{\substack{u::=v \in P, v \in T^* \\ u = u_1 \cdots u_n \\ v = v_1 \cdots v_m}} sg(u_1 \cdots u_n X_1 v_m \cdots v_1)_{\overline{X_0}}$$

$$+ sg(cX_2ccc)_{\overline{X_1}} + \sum_{i=2}^{3} \sum_{x \in N \cup T} sg(xX_i x)_{\overline{X_i}} + \sum_{\substack{u::=v \in P \\ u = u_1 \cdots u_n \\ v = v_1 \cdots v_m}} sg(u_1 \cdots u_n X_3 v_m \cdots v_1)_{\overline{X_2}}$$

$$+ sg(cX_3c)_{\overline{X_2}} + sg(cc)_{\overline{X_3}} \quad \textit{and}$$
$$R = \{fr(A) \mid A \in \{Y_0, Y_1, X_0, X_1, X_2, X_3\}\} \cup \{reduce(x) \mid x \in N \cup T' \cup \{c\}\}.$$

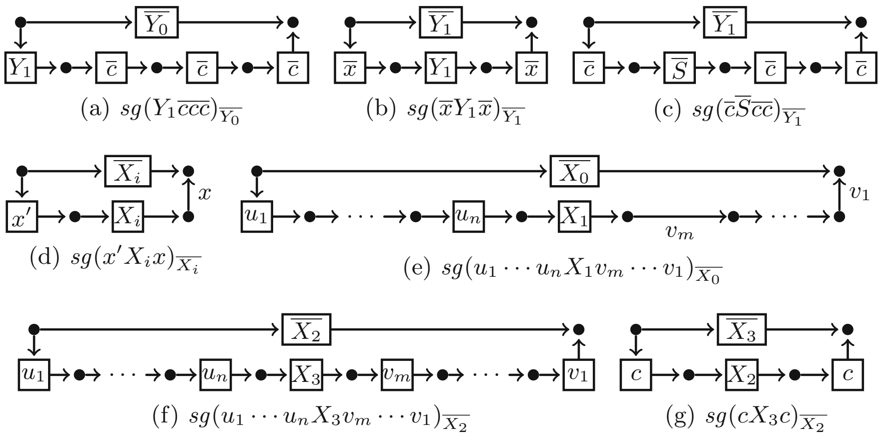

$\overline{Y_0}$
Y_1
$\overline{c}$
$\overline{c}$
$\overline{c}$
(a) $sg(Y_1\overline{ccc})_{\overline{Y_0}}$
$\overline{Y_1}$
$\overline{x}$
Y_1
$\overline{x}$
(b) $sg(\overline{x}Y_1\overline{x})_{\overline{Y_1}}$
$\overline{Y_1}$
$\overline{c}$
$\overline{S}$
$\overline{c}$
$\overline{c}$
(c) $sg(\overline{c}\overline{S}\overline{cc})_{\overline{Y_1}}$
$\overline{X_i}$
x'
X_i
x
(d) $sg(x'X_ix)_{\overline{X_i}}$
$\overline{X_0}$
u_1
u_n
X_1
v_m
v_1
(e) $sg(u_1\cdots u_nX_1v_m\cdots v_1)_{\overline{X_0}}$
$\overline{X_2}$
u_1
u_n
X_3
v_m
v_1
(f) $sg(u_1\cdots u_nX_3v_m\cdots v_1)_{\overline{X_2}}$
$\overline{X_3}$
c
X_2
c
(g) $sg(cX_3c)_{\overline{X_2}}$

Fig. 1. Schematic drawings of some connected components of the start hypergraph of CSFG(G)

Schematic drawings of some connected components of the start hypergraph are depicted in Fig. 1.

Theorem 2. *$L(CSFG(G)) = \{sg(w) \mid w \in L(G)\}$.*

The proof is based on the following fact. We recall some details of the proof because we will refer to them in the proof of Theorem 2.

Fact 1. *Any recursively enumerable string language L_0 is left quotient of two linear languages $L_P, L_=$, i.e., $L_0 = L_P \backslash L_= = \{x \mid y \in L_P \wedge yx \in L_=\}$ (cf. [7, Theorem 3.13.]).*

Remark 3. $L_0 = rev(rev(L_0)) = rev(L(G)) = L_= \backslash L_P$, where $rev(L_0) = \{r(w) \mid w \in L_0\}$ where $r(w) = x_n \cdots x_1$ for $w = x_1 \cdots x_n$ and $G = (N, T, P, S)$ is a Chomsky grammar with $L(G) = rev(L_0)$.

$$L_= = \{z_m c \ldots cz_1 cSccr(z_1)c \ldots cr(z_m)ccc \mid m \geq 1, z_i \in (N \cup T)^*, i = 1, \ldots, m\}$$
$$L_P = \{x_n u_n y_n c \ \ldots \ cx_1 u_1 y_1 ccr(y_1)r(v_1)r(x_1)c \ \ldots$$
$$\ldots \ cr(y_{n-1})r(v_{n-1})r(x_{n-1})cccr(y_n)r(v_n)r(x_n) \mid$$
$$n \geq 2, x_i, y_i \in (N \cup T)^*, u_i ::= v_i \in P, i = 1, \ldots, n-1, x_n v_n y_n \in T^*\}$$

where $c \notin N \cup T$. The basic idea is that for each $w \in L(G)$ exists a derivation $S = w_1 \rightarrow w_2 \rightarrow \cdots \rightarrow w_{n-1} \rightarrow w_n \rightarrow w_{n+1} = w$ with $w_i = x_i u_i y_i$ and

$w_{i+1} = x_i v_i y_i$ where $u_i ::= v_i \in P$ for $i = 1, \ldots, n$, i.e., $S = x_1 u_1 y_1 \rightarrow x_1 v_1 y_1 = x_2 u_2 y_2 \rightarrow \cdots \rightarrow x_{n-1} v_{n-1} y_{n-1} = x_n u_n y_n \rightarrow x_n v_n y_n = w$. $L_=$ captures the relation $x_i v_i y_i = x_{i+1} u_{i+1} y_{i+1}$ and L_P captures the relation $x_i u_i y_i \rightarrow x_i v_i y_i$.[1]

$L_=$ and L_P are linear. The following grammars generate them.

$$\begin{aligned}
G_= &= (\{Y_0, Y_1\}, N \cup T \cup \{c\}, P_=, Y_0) \text{ with}\\
P_= &= \{Y_0 ::= Y_1 ccc, Y_1 ::= cScc\} \;\cup\; \{Y_1 ::= xY_1x \mid x \in N \cup T \cup \{c\}\}\\
G_P &= (\{X_0, X_1.X_2, X_3\}, N \cup T \cup \{c\}, P_P, X_0) \text{ with}\\
P_P &= \{X_0 ::= xX_0x \mid x \in T\} \;\cup\; \{X_0 ::= uX_1r(v) \mid u ::= v \in P, v \in T^*\}\\
&\quad \cup \{X_1 ::= xX_1x \mid x \in T\} \;\cup\; \{X_1 ::= cX_2ccc\}\\
&\quad \cup \{X_2 ::= xX_2x \mid x \in N \cup T\} \;\cup\; \{X_2 ::= uX_3r(v) \mid u ::= v \in P\}\\
&\quad \cup \{X_3 ::= xX_3x \mid x \in N \cup T\} \;\cup\; \{X_3 ::= cX_2c, X_3 ::= cc\}.
\end{aligned}$$

Example 2. Let $G = (\{A\}, \{a.b\}, \{(A ::= aAb), (A ::= ab)\}, A)$. Then

$$\begin{aligned}
G_= &= (\{Y_0, Y_1\}, \{A, a, b, c\}, P_=, Y_0)\\
P_= &= \{Y_0 ::= Y_1 ccc, Y_1 ::= cAcc \mid aY_1a \mid bY_1b \mid AY_1A \mid cY_1c\}\\
G_P &= (\{X_0, X_1.X_2, X_3\}, \{A, a, b, c\}, P_P, X_0)\\
P_P &= \{X_0 ::= aX_0a \mid bX_0b \mid AX_1ba\}\\
&\quad \cup \{X_1 ::= aX_1a \mid bX_0b \mid cX_2ccc\}\\
&\quad \cup \{X_2 ::= aX_2a \mid bX_2b \mid AX_2A \mid AX_3bAa \mid AX_3ba\}\\
&\quad \cup \{X_3 ::= aX_3a \mid bX_3b \mid AX_3A \mid cX_2c \mid cc\}
\end{aligned}$$

Two derivations may be $X_0 \Longrightarrow aX_0a \Longrightarrow aAX_1baa \Longrightarrow aAbX_1bbaa \Longrightarrow aAbcX_2cccbbaa \Longrightarrow aAbcAX_3bAacccbbaa \Longrightarrow aAbcAccbAacccbbaa = d$ and $Y_0 \Longrightarrow Y_1ccc \Longrightarrow aY_1accc \Longrightarrow aAY_1Aaccc \Longrightarrow aAbY_1bAaccc \Longrightarrow aAbcAccbAaccc = z$. Removing the prefix z from d yields $bbaa$.

Every context-free string grammars can be transformed into fusion grammars generating the same language up to representation of strings as graphs as the following construction shows.

Construction 3. *Let* $G = (N, T, P, S)$ *be a context-free string grammar. Then* $FG(G) = (sg_\mu(S) + \sum_{r \in P} hgr(r), N, \{\mu\}, T)$ *with* $sg_\mu(S) = \mu\circlearrowleft\bullet\longrightarrow\boxed{S}\longrightarrow\bullet$*,* $hgr(r) = sg(u)_{\overline{A}}$ *for* $r = (A ::= u) \in P$ *and* $\mu \notin N \cup T$ *is the corresponding fusion grammar.*

[1] 1. W.l.o.g. assume $(S ::= S) \in P$ such that each derivation is of length ≥ 2.
2. For technical reasons each word contains the derivation twice, the middle is separated by cc, w_{n+1} is separated by ccc, the first is in reverse order and the second is reversed. This yields $w_n c w_{n-1} c \ldots c w_2 c w_1 c c r(w_2) c r(w_3) c \ldots c r(w_n) c c c r(w_{n+1})$. String in L_P are of the form $d = (w_n c w_{n-1} c \ldots c w_2 c w_1 c c r(w_2) c \ldots c r(w_n) c c c r(w_{n+1}))$, where $n \geq 2, w_i \rightarrow w_{i+1}$ in G and $w_{n+1} \in T^*$; and strings in $L_=$ are of the form $z = (z_m c \ldots c z_2 c S c c r(z_2) c \ldots c r(z_m) c c c)$, where $z_i \in (N \cup T)^*$. Therefore, $d = zz'$ for some z' if and only if $n = m + 1, S = w_1, z_i = w_i$ for $i = 1, \ldots, m$ and $z' = r(w_{n+1})$. Consequently, $r(w) = r(w_{n+1}) = r(y_n) r(v_n) r(x_n) \in L_= \backslash L_P$.

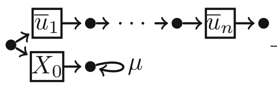
Example 3. Let $G = (\{A\}, \{a.b\}, \{r_1, r_2\}, A)$ with $r_1 = (A{::} = aAb)$ and $r_2 = (A{::} = ab)$. Then the rules are represented by $hgr(r_1) = sg(aAb)_{\overline{A}}$ and $(Z, \{A\}, \{\mu\}, \{a, b\})$ with $Z = sg_{\mu}(A) + sg(aAb)_{\overline{A}} + sg(ab)_{\overline{A}}$ is the corresponding fusion grammar.

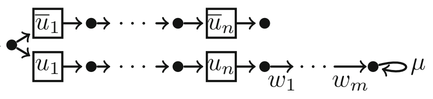
Lemma 3. *1. $L(FG(G)) = L(G)$.*
2. A derivation $w_1 \xrightarrow[r_1]{} \ldots \xrightarrow[r_{n-1}]{} w_n$ in G exists if and only if a derivation $Z \underset{m}{\Longrightarrow} m \cdot Z = sg_{\mu}(w_1) + hgr(r_1) + \ldots + hgr(r_{n-1}) \Longrightarrow sg_{\mu}(w_2) + hgr(r_2) + \ldots + hgr(r_{n-1}) \Longrightarrow \ldots \Longrightarrow sg_{\mu}(w_n)$ in $FG(G)$ exists.

Proof. 1. Each context-free string grammar G can be transformed into a hyperedge replacement grammar with connected right hand sides. Hence, the transformation of hyperedge replacement grammars into fusion grammars (cf. [2]) can be applied yielding $FG(G)$.
2. Proof by induction on the length of the derivation. □

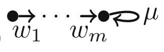
Remark 4. The connected components in the start hypergraphs of the context-sensitive fusion grammar in Construction 2 are hypergraph representation of the rules of the two linear string grammars (cf. Construction 3) slightly modified. The connected components in $Z_=$ are constructed for the linear rules in $G_=$ such that each symbol in $N \cup T \cup \{c\}$ is complemented and for each T-symbol the primed copy is used instead. The connected components for the linear rules in G_P containing X_0 and X_1 are constructed such that they contain fusion symbols left and terminal symbols right of the X_i-labeled hyperedge. Again for each terminal symbol the primed copy is used instead. The other connected components use the standard construction and are therefore only fusion symbol labeled (replacing also terminal symbols by their primed copy).

Proof (of Theorem 2). Let $w \in L(G)$. Then $w \in L_= \backslash L_P$ by Fact 1 and there are derivations in $G_=$ and G_P with $Y_0 \overset{*}{\to} u$ and $X_0 \overset{*}{\to} uw$ with $u = u_1 \cdots u_n$ and $w = w_1 \cdots w_m$. For each of these derivations exists by Lemma 3 a derivation in the corresponding fusion grammar ($FG(G_=)$, $FG(G_P)$, resp. where $G_=$ and G_P are defined in Remark 3). Because the nonterminal alphabets of $G_=$ and G_P are disjoint and the connected component $dsg(X_0, Y_0)_{\mu}$ contains two hyperedges one labeled with each start symbol of the two linear string grammars there is a derivation

$Z \overset{*}{\Longrightarrow}$ 

$+Z_P \overset{*}{\Longrightarrow}$ 

$= H$

applying context-free fusion rules[2]. Then the two complementary strings graphs can be erased by successive applications of $reduce(x)$ for suitable x due to Lemma 1, i.e., $H \underset{reduce(u_1)}{\Longrightarrow} \cdots \underset{reduce(u_n)}{\Longrightarrow}$ 

$+ [n]$. Consequently, $sg(w_1 \cdots w_m) \in L(CSFG(G))$.

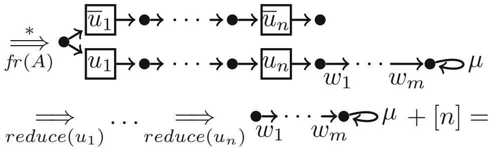
[2] Applying first $fr(A)$ with $A \in \{Y_0, Y_1\}$ and then $A \in \{X_0, X_1, X_2, X_3\}$ is arbitrary. The rules may be applied in any order.

Now, let $X \in L(CSFG(G))$. Then there is a derivation $Z \overset{*}{\Longrightarrow} H$ with $X = rem_M(Y), Y \in \mathcal{C}(H) \cap (\mathcal{H}_{T\cup M} \setminus \mathcal{H}_T)\}$. Because only $dsg(X_0, Y_0)_\mu$ contains a μ-hyperedge this connected component is substantial for some derived connected component contributing to the generated language. W.l.o.g. one can assume that $dsg(X_0, Y_0)_\mu$ is never multiplied due to the following reasoning. Let C be a connected component derivable from Z. Let $\#_\mu \colon \mathcal{H}_\Sigma \to \mathbb{N}$ be a mapping of hypergraphs over Σ to the number of μ-labeled hyperedges in the respective hypergraph. Then $\#_\mu(C) \leq 1$, i.e., no two or more copies of $dsg(X_0, Y_0)_\mu$ contribute to C as the following reasoning indicates. For each $C \in \mathcal{C}(Z)$ $\#_\mu(C) \leq 1$ by construction. For each $C \notin \mathcal{C}(Z)$ assume $Z \overset{*}{\Longrightarrow} C_1 + C_2 + [k] \underset{r}{\Longrightarrow} C + [l]$ for some $k, l \in \mathbb{N}$ where C_1 and C_2 are two connected components and $\#_\mu(C_i) \geq 1$ for $i = 1, 2$. $\#_\mu(C_i) \geq 1$ implies $C_i \neq [1]$. Hence, $Z \overset{*}{\Longrightarrow} C_i$, $i = 1, 2$. Further, r must be a context-free fusion rule because the positive context conditions of $reduce(x)$ restrict that both hyperedges must be attached to a common source vertex which is not possible if C_1 and C_2 are two connected components. Let $fr(A)$ be the applied context-free fusion rule, $A \in \{Y_0, Y_1, X_1, X_2, X_3, X_4\}$. W.l.o.g. let A be the label of the hyperedge in C_1 and let $\overline{A}$ be the label of the hyperedge in C_2. Furthermore, it is sufficient to analyze the case $\#_\mu(C_i) = 1$ for $i = 1, 2$. However, $\#_\mu(C_i) = 1$ implies that $dsg(X_0, Y_0)_\mu$ contributes to C_i but because the linear structure of the rules in $P_=$ and P_P carries over to the connected components C_2 cannot contain both a μ- and a $\overline{A}$-labeled hyperedge. Hence, the assumption must be false.

The fusion rules wrt $Y_0, Y_1, X_0, X_1, X_2, X_3$ are context free and thus one connected component or two connected components with two complementarily labeled hyperedges from this subset can be fused arbitrarily. This may produce connected components without markers where all the hyperedges labeled with $Y_0, \overline{Y_0}, Y_1, \ldots, X_3, \overline{X_3}$ are fused. E.g. $sg(\overline{x}Y_1\overline{x})_{\overline{Y_1}}$ may be multiplied several times and all the complementary Y_1- and $\overline{Y_1}$-hyperedges can be fused yielding two circles. However this connected component is not fusible to some other connected component because now it is only labeled with fusion symbols $\{N \cup T \cup \{c\}\}$ but for these symbols the fusion is restricted to take only place if the two complementary hyperedges are attached to the same vertex. A similar argument can be applied to other cases wrt connected components with X_i-hyperedges.

The direct derivations steps can be interchanged[3] in such a way that one gets a derivation of the following form:

$$Z \underset{m}{\Longrightarrow} dsg(X_0, Y_0)_\mu + m' \cdot Z_= + m'' \cdot Z_P \qquad \text{for some multiplicities } m', m''$$



with $A \in \{Y_0, Y_1, X_0, X_1, X_2, X_3\}$

[3] For the case of two context-free fusion rules see [2]; for the case involving *reduce* see Lemma 2. All multiplications can be done initially (using the same argument as in [2]).

Hence, $Y = sg(w_1 \cdots w_m)_\mu$. The linear structure of the connected components gives us $w_1 \cdots w_m \in L(G)$. □

6 Conclusion

In this paper, we have continued the research on context-dependent fusion grammars. We have introduced context-sensitive fusion grammars and have showed that the Post correspondence problem can be formulated very intuitively by such a grammar. Afterwards, we have showed that every Chomsky grammar can be simulated by a corresponding context-sensitive fusion grammar. Hence, they can generate all recursively enumerable string languages (up to representation of strings as graphs). This improves the previous result presented in [5] showing that context-dependent fusion grammars (with positive and negative context-conditions) are another universal computing model. However, further research is needed including the following open question. Is it true, that fusion grammars without context-conditions are not universal? Are also only negative context conditions powerful enough to simulate Chomsky grammars? If so is also a single negative context-condition sufficient? One may also investigate fusion grammar with other regulations like priorities or regular expressions.

Acknowledgment. We are grateful to Hans-Jörg Kreowski and Sabine Kuske for valuable discussions. We also thank the reviewers for their valuable comments.

References

1. Adleman, L.M.: Molecular computation of solutions to combinatorial problems. Science **266**, 1021–1024 (1994)
2. Kreowski, H.-J., Kuske, S., Lye, A.: Fusion grammars: a novel approach to the generation of graph languages. In: de Lara, J., Plump, D. (eds.) ICGT 2017. LNCS, vol. 10373, pp. 90–105. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61470-0_6
3. Kreowski, H.-J., Kuske, S., Lye, A.: Relating DNA computing and splitting/fusion grammars. In: Guerra, E., Orejas, F. (eds.) ICGT 2019. LNCS, vol. 11629, pp. 159–174. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23611-3_10
4. Kreowski, H.-J., Kuske, S., Lye, A.: Transformation of petri nets into context-dependent fusion grammars. In: Martín-Vide, C., Okhotin, A., Shapira, D. (eds.) LATA 2019. LNCS, vol. 11417, pp. 246–258. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-13435-8_18
5. Lye, A.: Transformation of turing machines into context-dependent fusion grammars. In: Post-Proceedings of 10th International Workshop on Graph Computation Models, (GCM 2019). Electronic Proceedings in Theoretical Computer Science (EPTCS) (2019). https://doi.org/10.4204/EPTCS
6. Post, E.L.: A variant of a recursively unsolvable problem. Bull. Am. Math. Soc. **52**, 264–269 (1946). https://doi.org/10.1090/s0002-9904-1946-08555-9
7. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing – New Computing Paradigms. Springer, Heidelberg (1998). https://doi.org/10.1007/978-3-662-03563-4