Relating DNA Computing and Splitting/Fusion Grammars

Hans-Jörg Kreowski, Sabine Kuske, and Aaron Lye(✉)

Department of Computer Science and Mathematics, University of Bremen, P.O.Box 33 04 40, 28334 Bremen, Germany
{kreo,kuske,lye}@informatik.uni-bremen.de

Abstract. Splitting/fusion grammars were recently introduced as devices for the generation of hypergraph languages. Their rule application mechanism is inspired by basic operations of DNA computing. In this paper, we demonstrate that splitting/fusion grammars and well-known computational approaches based on DNA computing are closely related on a technical level beyond the mere motivation. This includes Adleman's seminal experiment, insertion-deletion systems, and extended iterated 2-splicing systems.

1 Introduction

Adleman demonstrated in his seminal experiment [1] that the NP-hard Hamiltonian path problem can be solved by a polynomial number of biochemical operations on DNA strands with high probability exploiting the parallelism of chemical reactions in tubes of molecules. This was the starting point of the area of DNA computing that has been intensely developed since then. Inspired by DNA computing, we introduced fusion grammars in [2] and splicing/fusion grammars in [3]. In this paper, we rename the latter by splitting/fusion grammars as the term "splicing" may be misleading.

The core of DNA computing is the biochemical processing on tubes of DNA molecules. In the framework of splitting/fusion grammars, we exploit similarities between hypergraphs and tubes of molecules, which are multisets from a mathematical point of view. Each hypergraph is the disjoint union of its connected components which corresponds to a multiset if one counts the isomorphic connected components. Therefore, connected components of hypergraphs can be seen as counterparts of DNA molecules. To emphasize this analogy, we call the connected components molecules. Furthermore, we reflect the Watson-Crick complementarity of DNA nucleotides and single DNA strands by a complementarity of hyperedges and the basic DNA operations ligation, restriction, duplication by polymerase chain reaction, and reading by gel electrophoresis by fusion, splitting, multiplication, and filtering of special connected components, respectively. In this paper, we show that the relation between DNA computing and splitting/fusion grammars goes far beyond mere motivation. We model three well-known DNA computing approaches in our framework. In Sect. 4, we

E. Guerra and F. Orejas (Eds.): ICGT 2019, LNCS 11629, pp. 159–174, 2019.
https://doi.org/10.1007/978-3-030-23611-3_10

recreate Adleman's experiment in terms of fusion grammars. In Sect. 5, insertion-deletion systems as one of the prominent (string) language generating devices based on DNA computing (cf., e.g., Chapter 6 of [4]) are transformed into splitting/fusion grammars. Another important DNA computing approach offers splicing systems in many variants (cf., e.g., Chapters 7 to 11 of [4]). In Sect. 6, we generalize extended iterated 2-splicing systems to 2-splicing grammars that are special regulated splitting/fusion grammars. Section 2 provides preliminaries for hypergraphs and the notion of splitting/fusion grammars is recalled in Sect. 3. Section 7 concludes the paper. The proofs of all stated correctness results are omitted because of the page limit.

2 Preliminaries

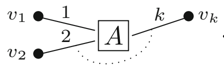
In this section, basic notions and notations of hypergraphs are recalled (see, e.g., [5]).

Let Σ be a label alphabet. A *hypergraph* over Σ is a system $H = (V, E, att, lab)$ where V is a finite set of *nodes*, E is a finite set of *hyperedges*, $att\colon E \to V^*$ is a function, called *attachment* (assigning a string of attachment nodes to each edge), and $lab\colon E \to \Sigma$ is a function, called *labeling*.

The length of the attachment $att(e)$ for $e \in E$ is called *type* of e, and e is called A-*hyperedge* if A is its label. Let $\Sigma' \subseteq \Sigma$ be a subalphabet of Σ and $type\colon \Sigma' \to \mathbb{N}$ a function, called *type function*. Then we require that every A-hyperedge with $A \in \Sigma'$ is of type $type(A)$. The components of $H = (V, E, att, lab)$ may also be denoted by V_H, E_H, att_H, and lab_H respectively. The class of all hypergraphs over Σ is denoted by $\mathcal{H}_\Sigma$.

A *(directed) graph* is a hypergraph $H = (V, E, att, lab)$ with $att(e) \in V^2$ for all $e \in E$. In this case, the hyperedges are called *edges*. If $att(e) = vv$ for some $v \in V$, then e is also called a *loop*. A graph is called *loop-free* if for all $e \in E$ $att(e) = vv'$ with $v \neq v'$. A graph is called *simple* if it is loop-free and no parallel edges exist.

The set $\{1, \ldots, k\}$ for some $k \in \mathbb{N}$ is denoted by $[k]$ which also denotes the discrete graph with the nodes $1, \ldots, k$ and an empty set of hyperedges.

In drawings, an A-hyperedge e with attachment $att(e) = v_1 \cdots v_k$ is depicted by 

. Moreover, a hyperedge of type 2 may be depicted as an edge by $\bullet\xrightarrow{A}\bullet$ instead of $\bullet\overset{1}{-}\boxed{A}\overset{2}{-}\bullet$. If there are two edges with the same label, but in opposite directions, we may draw them as an undirected edge. We assume the existence of a special label $* \in \Sigma$ that is omitted in drawings. We call a hypergraph *unlabeled* if $lab(e) = *$ for all $e \in E$.

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

Let $H \in \mathcal{H}_\Sigma$. Then a sequence of triples $(i_1, e_1, o_1) \ldots (i_n, e_n, o_n) \in (\mathbb{N} \times E_H \times \mathbb{N})^*$ is a *path* from $v \in V_H$ to $v' \in V_H$ if $v = att_H(e_1)_{i_1}, v' = att_H(e_n)_{o_n}$

and $att_H(e_j)_{o_j} = att_H(e_{j+1})_{i_{j+1}}$ for $j = 1, \ldots, n-1$ where, for each $e \in E_H$, $att_H(e)_i = v_i$ for $att_H(e) = v_1 \cdots v_k$ and $i = 1, \ldots, k$. In the case of simple graphs, a path may be denoted by the sequence of visited nodes as the involved edges are uniquely determined.

H is *connected* if each two nodes are connected by a path. A connected subgraph M of H is called a *molecule* of H if it is maximal meaning that $M \subseteq M' \subseteq H$ for a connected M' implies $M = M'$. The set of molecules of H is denoted by $\mathcal{M}(H)$.

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 $att_{H'}(g_E(e)) = g_V^*(att_H(e))$ 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 $v_1 \cdots v_n \in V_H^*$. H and H' are *isomorphic*, denoted by $H \cong H'$, if there is an isomorphism $g\colon H \to H'$, i.e., a morphism with bijective mappings. Clearly, $H \subseteq H'$ implies that the two inclusions $V_H \subseteq V_{H'}$ and $E_H \subseteq E_{H'}$ define a morphism $incl\colon H \to H'$. Given a morphism $g\colon H \to H'$, the image of H in H' under g defines the subgraph $g(H) \subseteq H'$.

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 $H = H' - (V, E) = (V_{H'} - V, E_{H'} - E, att_H, lab_H)$ with $att_H(e) = att_{H'}(e)$ and $lab_H(e) = lab_{H'}(e)$ for all $e \in E_{H'} - E$ defines a subgraph $H \subseteq H'$ if $att_{H'}(e) \in (V_{H'} - V)^*$ for all $e \in E_{H'} - E$, i.e., no remaining hyperedge is attached to a removed node. This condition is called *dangling condition.* The dangling condition is fulfilled in the special case that only hyperedges are removed.

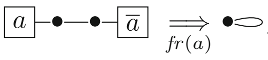
Given $H, H' \in \mathcal{H}_\Sigma$, the *disjoint union* of H and H' is denoted by $H + H'$. A special case is the disjoint union of H with itself k times, denoted by $k \cdot H$. Let $H \in \mathcal{H}_\Sigma$ and $m\colon \mathcal{M}(H) \to \mathbb{N}$ be a mapping, called *multiplicity.* Then the multiplication of m and H is defined by $m \cdot H = \sum\limits_{M \in \mathcal{M}(H)} m(M) \cdot M$. The disjoint union is unique up to isomorphism. It is easy to see that the disjoint union is commutative and associative. Moreover, there are injective morphisms $in_H\colon H \to H + H'$ and $in_{H'}\colon H' \to H + H'$ such that $in_H(H) \cup in_{H'}(H') = H + H'$ and $in_H(H) \cap in_{H'}(H') = \emptyset$. Each two morphisms $g_H\colon H \to Y$ and $g_{H'}\colon H' \to Y$ define a unique morphism $\langle g_H, g_{H'} \rangle\colon H + H' \to Y$ with $\langle g_H, g_{H'} \rangle \circ in_H = g_H$ and $\langle g_H, g_{H'} \rangle \circ in_{H'} = g_{H'}$. In particular, one gets $g = \langle g \circ in_H, g \circ in_{H'} \rangle$ for all morphisms $g\colon H + H' \to Y$ and $g + g' = \langle in_Y \circ g, in_{Y'} \circ g' \rangle\colon H + H' \to Y + Y'$ for morphisms $g\colon H \to Y$ and $g'\colon H' \to Y'$. The disjoint union is the coproduct in the category of hypergraphs.

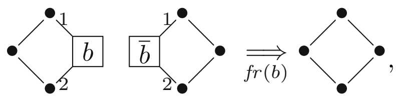
The merging of nodes is defined as a quotient by means of an equivalence relation $\equiv$ on the set of nodes V_H of H as follows: $H/\!\equiv\; = (V_H/\!\equiv, E_H, att_{H/\equiv}, lab_H)$ with $att_{H/\equiv}(e) = [v_1] \cdots [v_k]$ for $e \in E_H$, $att_H(e) = v_1 \cdots v_k$ where $[v]$ denotes the equivalence class of $v \in V_H$ and $V_H/\!\equiv$ is the set of equivalence classes. Given two sequences $d(1) \cdots d(k)$ and $d'(1) \cdots d'(k)$ of nodes in V_H for some $k \in \mathbb{N}$. Then the relation $d(i) \sim d'(i)$ for $i = 1, \ldots, k$ induces a particular equivalence relation (by the reflexive, symmetric and transitive closure of $\sim$) that is denoted by $d = d'$. This is employed in the next section to define the fusion of hyperedges. It is easy to see that $f\colon H \to H/{\equiv}$ given by $f_V(v) = [v]$ for all $v \in V_H$ and $f_E(e) = e$ for all $e \in E_H$ defines a *quotient morphism.*

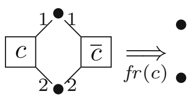
3 Splitting/Fusion Grammars

In this section, we recall the notion of splitting/fusion grammars which are called splicing/fusion grammars in [3]. The grammars are renamed because we think that "splitting" fits better. We begin with the concept of fusion considered as application of fusion rules, and then continue with splitting which is converse to fusion.

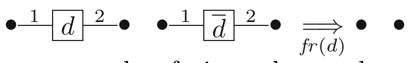
Definition 1. 1. Let $F \subseteq \Sigma$ be a finite set of labels and $k\colon F \to \mathbb{N}$ a type function.F is called *fusion alphabet*, its elements *fusion labels.* Let $\overline{a} \in \Sigma$ be a *complementary fusion label* for each $a \in F$ such that $\overline{a} \neq \overline{b}$ for all $a \neq b$. The set of complementary fusion labels is denoted by $\overline{F}$. The typing function and the complementarity are extended to $\overline{F}$ by $k(\overline{a}) = k(a)$ and $\overline{\overline{a}} = a$ for all $\overline{a} \in \overline{F}$. Let $H \in \mathcal{H}_\Sigma$ and $e, \overline{e} \in E_H$ with $a = lab_H(e) = \overline{lab_H(\overline{e})}$ for some $a \in F$. Then the *fusion of* e *and* $\overline{e}$ *in* H yields the hypergraph $H_{fuse(e,\overline{e})} = (H - (\emptyset, \{e, \overline{e}\}))/_{att_H(e)=att_H(\overline{e})}$.
2. Let $H, H' \in \mathcal{H}_\Sigma$. Then H *directly derives* H' through fusion wrt $a \in F$ if $H' \cong H_{fuse(e,\overline{e})}$ for some $e, \overline{e} \in E_H$ with $a = lab_H(e) = \overline{lab_H(\overline{e})}$. Here the fusion label $a \in F$ plays the role of a *fusion rule* indicated by the notation $fr(a)$. Its application is denoted by $H \underset{fr(a)}{\Longrightarrow} H'$.

Remark 1. 1. As each hyperedge belongs to a single molecule, the fusion of two hyperedges changes either one molecule or two molecules. Moreover, a fusion can have three different effects.
   – It may be a kind of folding, e.g., 

,
   – two molecules may be joint, e.g., 

, or
   – it can also result in disconnection, e.g., 

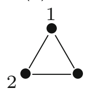
wrt one molecule

   or 

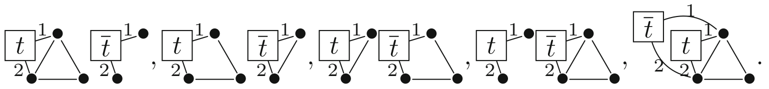
wrt two molecules.
2. It is easy to see that fusion rules can be applied in parallel if their matchings access pairwise different hyperedges (cf. [2]).

Definition 2. 1. Let $H' \in \mathcal{H}_\Sigma$ and $a \in F$. Then $H \in \mathcal{H}_\Sigma$ is a *splitting* of H' wrt a if there are $e, \overline{e} \in E_H$ with $a = lab_H(e) = \overline{lab_H(\overline{e})}$ such that $H' \cong H_{fuse(e,\overline{e})}$.
2. Such a splitting can be considered as a direct derivation $H' \underset{sr(a)}{\Longrightarrow} H$, where $sr(a)$ indicates that the label a is used as a splitting rule.

Remark 2. 1. To get more flexibility, we use other complementary labels for splitting than for fusion. For $a \in F$, the *complementary splitting label of* a is denoted by $\widehat{a}$ (instead of $\overline{a}$).

2. An application of $sr(a)$ to H' can be explicitly performed by (1) choosing a *matching morphism* $g'\colon [k(a)] \to H'$, i.e., a sequence of nodes $g'(1)\cdots g'(k(a))$ in H', (2) for $i = 1,\ldots,k(a)$, either splitting $g'(i)$ into two new nodes $d(i)$ and $\widehat{d}(i)$ or replacing it by one new node v with $d(i) = v = \widehat{d}(i)$, subject to the condition: $g'(i) = g'(j)$ for some $i \neq j$ if and only if $d(i) = d(j)$ and $\widehat{d}(i) = \widehat{d}(j)$, (3) constructing the hypergraph $I' = (V_{H'} \setminus \{g'(i) \mid i = 1,\ldots,k(a)\} + \{d(i), \widehat{d}(i) \mid i = 1,\ldots,k(a)\}, E_{H'}, att_{I'}, lab_{H'})$ with $att_{I'}(e')_j = d(i)$ or $att_{I'}(e')_j = \widehat{d}(i)$ for $j = 1,\ldots,k(a)$ provided that $att_{H'}(e')_j = g'(i)$ for some $i = 1,\ldots,k(a)$ and $att_{I'}(e') = att_{H'}(e')$ otherwise, (4) constructing H'' from I' by adding two new hyperedges e and $\widehat{e}$ with $att_{H''}(e) = d(1)\cdots d(k(a))$ and $att_{H'}(\widehat{e}) = \widehat{d}(1)\cdots\widehat{d}(k(a))$ as well as $lab_{H''}(e) = a$ and $lab_{H''}(\widehat{e}) = \widehat{a}$, and (5) renaming nodes and hyperedges of H'' optionally. I' in (3) is called *intermediate hypergraph.*
3. While the application of a fusion rule is unique up to isomorphism, splitting is highly nondeterministic in general because each tentacle of a hyperedge of H' that is attached to a matching node $g'(i)$ may be attached to $d(i)$ or $\widehat{d}(i)$ in H''. Consider, for example, a fusion symbol t of type 2 and a triangle 

where the numbered nodes define the matching. Although this is a very small graph with much symmetry, one gets five splittings:



As the high nondeterminism of splitting is not always desirable, we employ some variants of context conditions to cut the nondeterminism down.

Definition 3. Let F be a fusion alphabet and $a \in F$.

1. A *splitting rule with a fixed disjoint subcontext* is a triple $(sr(a), d\colon [k(a)] \to D, incl\colon D \to C)$ where $C, D \in \mathcal{H}_\Sigma$ are connected hypergraphs, d is a morphism, and $incl$ is an injective morphism. It can be applied to H' wrt the matching morphism $g'\colon [k(a)] \to H'$ if there is a morphism $f\colon C \to H'$ and the intermediate hypergraph I' can be chosen as $I'' + D$ with $d(i) \in V_{I''}$ and $\widehat{d}(i) \in V_D$ for $i = 1,\ldots,k(a)$ and $I'' \subseteq H'$ such that $g' = d \circ incl \circ f$.
2. A *splitting rule with double context* is a triple $(sr(a), c_1\colon [k(a)] \to C_1, c_2\colon [k(a)] \to C_2)$ where C_1 and C_2 are connected hypergraphs and c_1 and c_2 morphisms. It is applicable to H' wrt the matching morphism $g'\colon [k(a)] \to H'$ if there are morphisms $f_j\colon C_j \to H', j = 1,2$, with $f_1(C_1) \cap f_2(C_2) = g'([k(a)]) = f_1(c_1([k(a)])) = f_2(c_2([k(a)]))$ and the intermediate hypergraph I' can be chosen as $I_1 + I_2$ such that there are morphisms $f'_j\colon C_j \to I_j$ for $j = 1,2$ and injective morphisms $in_j\colon I_j \to H'$ for $j = 1,2$ such that $in_j \circ f'_j = f_j$ for $j = 1,2$.

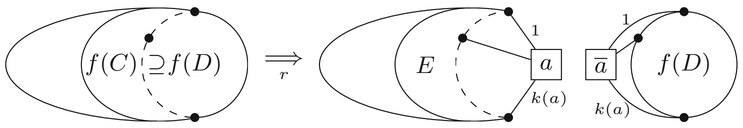


Fig. 1. Application of a splitting rule $r = (sr(a), d, incl)$ with fixed disjoint subcontext where $E = (f(C) - f(D)) + f([k(a)])$

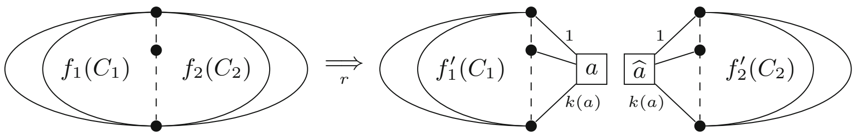


Fig. 2. Application of a splitting rule with double context $r = (sr(a), c_1, c_2)$

Remark 3. 1. The splitting rules with fixed disjoint subcontext are used in Sect. 5 together with a variant for the special types of graphs considered there. Splitting rules with fixed disjoint subcontext where context and subcontext coincide were already used in [3]. While the context is required to be present in the processed hypergraphs, the subcontext is required to be a disjoint component after splitting. Figure 1 illustrates such a splitting for a single molecule. This is not always possible as there may be a hyperedge in the processed hypergraphs that does not belong to the matching of the subcontext, but one of its tentacles is attached to an inner node of the subcontext, i.e., a node that is not matched by g'.

2. The splitting rules with double context are used in Sect. 6. An application of such a rule is only possible if the processed hypergraph can be cut in two parts which intersect in the splitting nodes only and where one context matches in one part and the other context in the other part. As the contexts are connected and intersect in the splitting nodes, only one molecule is cut while the other molecules remain unchanged. Figure 2 illustrates such a splitting for a single molecule.

Now we can define splitting/fusion grammars as used in this paper. Besides fusion and splitting rules, such a grammar provides a start hypergraph, a set of markers, and a set of terminal labels. The derivation process combines rule applications with multiplications. A terminal hypergraph belongs to the generated language if it is obtained by removing all marked hyperedges from a molecule that has at least one marked hyperedge and that is derived from the start hypergraph. The markers allow one to partition the molecules of the start hypergraph into marked and unmarked ones. As markers can not be generated, the unmarked molecules can only contribute to the generated language if they are fused with marked molecules so that they are of an auxiliary nature.

Definition 4. 1. A *splitting/fusion grammar* is a system $SFG = (Z, F, M, T, SR)$ where Z is a *start hypergraph*, $F \subseteq \Sigma$ is a fusion alphabet, $M \subseteq \Sigma$ with $M \cap (F \cup \overline{F}) = \emptyset$ is a finite set of *markers*, $T \subseteq \Sigma$ with $T \cap (F \cup \overline{F}) = \emptyset = T \cap M$

is a finite set of *terminal labels*, and SR is a finite set of splitting rules that may have some type of context conditions.

2. A *direct derivation* $H \Longrightarrow H'$ for some $H, H' \in \mathcal{H}_\Sigma$ is either a rule application $H \underset{r}{\Longrightarrow} H'$ for some rule in $SR \cup \{fr(a) \mid a \in F\}$ or a multiplication $H \underset{m}{\Longrightarrow} m \cdot H$ for some multiplicity m.
3. A *derivation* $H \overset{n}{\Longrightarrow} H'$ of length n is a sequence $H_0 \Longrightarrow H_1 \Longrightarrow \cdots \Longrightarrow H_n$ with $H = H_0$ and $H' = H_n$. One may write $H \overset{*}{\Longrightarrow} H'$.
4. $L(SFG) = \{rem_M(Y) \mid Z \overset{*}{\Longrightarrow} H, Y \in \mathcal{M}(H) \cap (\mathcal{H}_{T \cup M} - \mathcal{H}_T)\}$ is the *generated language* of SFG where $rem_M(Y)$ is the hypergraph obtained when removing all hyperedges with labels in M from Y.

Remark 4. 1. A splitting/fusion grammar where $SR = \emptyset$ is called a *fusion grammar* and SR can be omitted in the tuple.

2. The provision of markers is significant. In other generative devices initial objects to start a generation and rules to perform the generation are separated while in fusion grammars the corresponding information is integrated in the start hypergraph. But it is necessary in some cases to distinguish between marked connected components that can contribute directly to the generated language and other components that can contribute to the generated languages by fusion with marked components only (cf., for example, the transformation of hyperedge replacement grammars into fusion grammars in [2]).

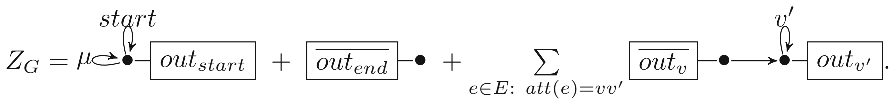
4 Adleman's Experiment

In this section, we adapt Adleman's famous experiment finding Hamiltonian paths by DNA computing to fusion grammars.

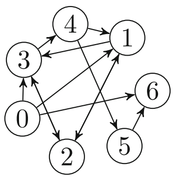
A graph G with designated nodes *start* and *end* is said to have a Hamiltonian path if and only if there exists a path from *start* to *end* that enters every node of the graph exactly once. The Hamiltonian path problem asks if a graph has a Hamiltonian path and has been proven to be NP-complete [6,7]. Adleman has proposed a transformation of the Hamiltonian path problem into a molecular biological process by encoding the graph by DNA molecules in order to generate a solution by massive parallelism. The computation is performed by standard DNA computing operations. In more detail, Adleman generated random paths by encoding every node of the input graph into a DNA strand such that two strands can be fused if and only if their corresponding nodes are connected via an edge in the respective direction. Afterwards filtering operations are applied to get rid of DNA strands that do not represent Hamiltonian paths from *start* to *end*. Finally emptiness is tested (see [1] for details).

For every unlabeled and loop-free graph, a fusion grammar can be constructed that generates paths from *start* to *end* by fusing smaller paths in parallel. Subsequent filter operations let only Hamiltonian paths remain. The start hypergraph of the fusion grammar contains a molecule for each edge of the input graph. Additionally, there is a molecule with a marker for constructing paths that begin with *start* and there is a molecule for the *end* node that allows to terminate paths fusions at *end*. The grammar is defined as follows.

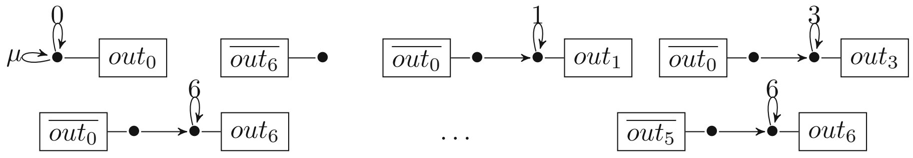
Definition 5. Let $G = (V, E, att, lab)$ be an unlabeled simple graph, and let $start, end \in V$ with $start \neq end$. Then $FG(G) = (Z_G, \{out_v \mid v \in V\}, \{\mu\}, V \cup \{*\})$ is the fusion grammar of G, where



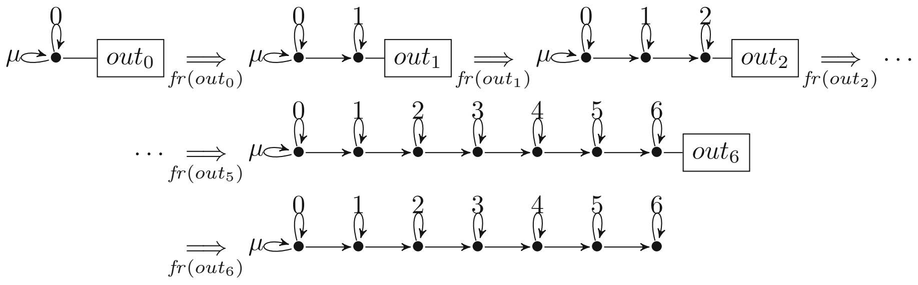
Example 1. Consider Adleman's sample graph, where bidirectional arrows represent two edges in opposite directions.



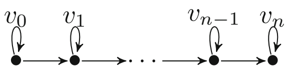
Select 0 as *start* and 6 as *end*. Then the start graph consists of the following molecules:



For example, the path starting from 0 and visiting then nodes 1 to 6 in this order can be sequentially generated as follows.


μ
0
out_0
$\Longrightarrow_{fr(out_0)}$
0 1
out_1
$\Longrightarrow_{fr(out_1)}$
0 1 2
out_2
$\Longrightarrow_{fr(out_2)}$
...
$\Longrightarrow_{fr(out_5)}$
0 1 2 3 4 5 6
out_6
$\Longrightarrow_{fr(out_6)}$
0 1 2 3 4 5 6

All involved fusions can be performed in parallel, because only pairwise distinct edges are fused where each such path appears with some probability depending on the number of copies of molecules of the start graph produced by a respective multiplication before the fusion step. Hence, this path (as well as all other paths) can be generated in a single step.

For each path $p = v_0 \cdots v_n$ we call the graph $pg(p) =$ 

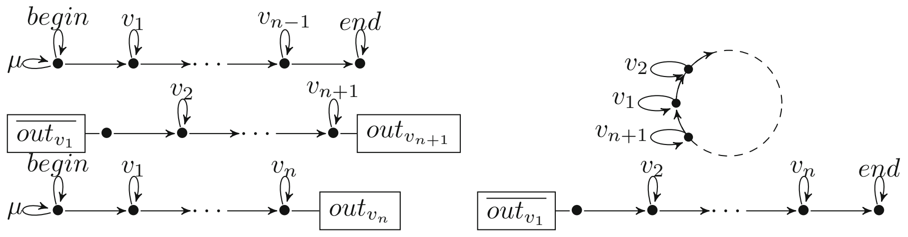
a *path graph* of p.

Theorem 1. $L(FG(G)) = \{pg(p) \mid p = start\ v_1 \cdots v_n\ end$ is a path in $G\}$.

In other words, the terminal graphs generated by $FG(G)$ are all path graphs from *start* to *end*.

The theorem follows directly from the following Lemma which can be proven by induction on the lengths of derivations.

Lemma 1. Let $Z_G \overset{*}{\Longrightarrow} H$ be a derivation in $FG(G)$. Let $M \in \mathcal{M}(H)$. Then M is either a molecule of Z_G or it has one of the following forms:



where $v_1, \ldots, v_{n+1} \in V, n \geq 1$.

In order to get only Hamiltonian paths, further filters are needed where the number of elements of a finite set X is denoted by $|X|$.

1. $length_k(L) = \{s \in L \mid |V_s| - 1 = k\}$ for $L \subseteq L(FG(G))$,
2. $simple(L) = \{s \in L \mid |V_s| = |lab_G(E_G)| - 1\}$ for $L \subseteq L(FG(G))$ where $lab_G(E) = \{lab_G(e) \mid e \in E\}$.

Corollary 1. The language $simple(length_{|V|-1}(L(FG(G))))$ is equal to the set of all path graphs $pg(p)$, where p is Hamiltonian.

The corollary indicates that it may be meaningsful to employ more general mechanisms to filter the members of the generated language from the derived hypergraphs. In the definition in Sect. 3, the generated language contains the terminal subhypergraph of a connected component of a derived hypergraph if it has some marked hyperedge, but no fusion hyperedges. This yields for our grammar that simulates Adleman's experiment graph representations of all paths from *begin* to *end* so that further filtering is needed to get the Hamiltonian paths. In general, other and further filter mechanisms may be employed if it is reasonable for a specific application.

5 Transformation of Insertion-Deletion Systems into Splitting/Fusion Grammars

Insertion-deletion systems are (string) language generating devices the basic operations of which are closely related to DNA computing (see, e.g., [8] and [4]). In this section, we transform insertion-deletion systems into splitting/fusion grammars. Our main result states that the transformation is correct meaning that the language generated by an insertion-deletion system equals the language of the corresponding splitting/fusion grammar up to the representation of strings by graphs.

5.1 Insertion-Deletion Systems

An *insertion-deletion system* is a quadruple $\gamma = (V, T, A, R)$ where V is a finite alphabet, $T \subseteq V$ is a subalphabet of *terminal symbols*, $A \subseteq V^*$ is a finite language of *axioms*, and R is a finite set of *rules* of the form $r = (u, \alpha/\beta, v)$ with $u, \alpha, \beta, v \in V^*$ such that either $\alpha = \lambda$ or $\beta = \lambda$. The application of rules to strings defines a binary relation of *computation steps*:

$$\textit{insertion} : w = xuvy \underset{r}{\longrightarrow} xu\beta vy \text{ for } w, x, y \in V^* \text{ and } r = (u, \lambda/\beta, v), \text{ and}$$

$$\textit{deletion} : w = xu\alpha vy \underset{r}{\longrightarrow} xuvy \text{ for } w, x, y \in V^* \text{ and } r = (u, \alpha/\lambda, v).$$

The reflexive and transitive closure of all computation steps is called *computation relation* and denoted by $\overset{*}{\underset{R}{\longrightarrow}}$, its elements are the *computations of* γ. The *generated language of* γ consists of all terminal strings that can be computed from axioms: $L(\gamma) = \{w \in T^* \mid z \overset{*}{\underset{R}{\longrightarrow}} w, z \in A\}$.

5.2 String Graphs

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 the elements of which are of type 2. 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 $sg(w) = (\{0\} \cup [n], [n], att_w, lab_w)$ with $att_w(i) = (i-1)i$ and $lab(i) = x_i$ for $i = 1, \dots, n$. The string graph of 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 $sg(\Sigma^*) = \{sg(w) \mid w \in \Sigma^*\}$.

Note that $u \in \Sigma^*$ is a substring of $w \in \Sigma^*$, i.e., $w = xuy$ for some $x, y \in \Sigma^*$ if and only if there is a graph morphism ${}_xu_y \colon sg(u) \to sg(w)$.

Each string graph $sg(w)$ for $w \in \Sigma^*$ gives rise to a special graph morphism $b_w \colon [2] \to sg(w)$ with $b_w(1) = 0$ and $b_w(2) = n$, where n is the length of w.

For technical reasons, we need the extension of a string graph by a labeled edge bending from the begin node to the end node. Consider $sg(w)$ for some $w \in \Sigma^*$, and let $s \in \Sigma$. Then the *s-handled string graph* $sg(w)_s$ contains $sg(w)$ as subgraph and the edge 0 with the attachment $0n$ and the label s, where n is the length of w.

5.3 The Transformation

Let $\gamma = (V, T, A, R)$ be an insertion-deletion system. The corresponding splitting/fusion grammar $SFG(\gamma)$ has a splitting rule for each rule of the insertion-deletion system, where the context u, v of the rule $(u, \alpha/\beta, v)$ is reflected by the context of the splitting rule. In the case of a deletion rule, the corresponding rule has a fixed disjoint subcontext reflecting the string to be deleted. In the case of an insertion rule, the corresponding rule has a fixed disjoint subcontext, too. But its application includes an additional node stretching within the intermediate hypergraph I'. Let $r = (u, \lambda/\beta, v)$ be an insertion rule and let $a \in \Sigma$. Then

$(sr(a), b_\lambda, {}_u\lambda_v)$ is a *splitting rule with node stretching*, and, for each $w = xuvy$, its application to $sg(w)_\mu$ with the matching $sg(uv) \xrightarrow{{}_x uv_y} sg(w) \subseteq sg(w)_\mu$ yields the molecules $sg(xuavy)_\mu$ and $sg(\lambda)_{\widehat{a}}$ meaning that the node separating $sg(u)$ and $sg(v)$ is stretched to an a-edge. Moreover, there is a fusion rule for each rule of the insertion-deletion system.

Definition 6. $SFG(\gamma) = (Z_\gamma, R, \{\mu\}, T, P_\gamma)$, where

- the rule set R is used as fusion alphabet with the complementary fusion alphabet $\overline{R}$ and the complementary splitting alphabet $\widehat{R}$ chosen such that $V, R, \overline{R}$ and $\widehat{R}$ are pairwise disjoint, the marker μ is an extra label, i.e., $\mu \notin V \cup R \cup \overline{R} \cup \widehat{R}$, μ and the fusion labels in R are of type 2,
- $Z_\gamma = \sum\limits_{z \in A} sg(z)_\mu + \sum\limits_{r=(u,\alpha/\beta,v) \in R} sg(\beta)_{\overline{r}}$, i.e., the start graph consists of the μ-handled string graphs of the axioms, the $\overline{r}$-handled string graphs of λ for each deletion rule r, and the $\overline{r}$-handled string graphs of the insertion string of each insertion rule r, and
- P_γ contains two types of splitting rules:
  1. for each deletion rule $r = (u, \alpha/\lambda, v) \in R$, there is a splitting rule with fixed disjoint subcontext $r_\gamma = (sr(r), b_\alpha, {}_u\alpha_v)$,
  2. for each insertion rule $r = (u, \lambda/\beta, v) \in R$, there is a splitting rule with node stretching $r_\gamma = (sr(r), b_\lambda, {}_u\lambda_v)$.

To see how these rules work, consider $sg(w)_\mu$. The rule r_γ for $r = (u, \alpha/\lambda, v)$ is applicable if $w = xu\alpha vy$ for some x, y using the matching $sg(u\alpha v) \xrightarrow{{}_x u\alpha v_y} sg(w) \subseteq sg(w)_\mu$. Then the splitting produces the two molecules $sg(xurvy)_\mu$ and $sg(\alpha)_{\widehat{r}}$. If the former molecule is fused with $sg(\lambda)_{\overline{r}}$ applying the fusion rule $fr(r)$, one gets the molecule $sg(xuvy)_\mu$. In other words, the application of r_γ to $sg(w)_\mu$ followed by the application of $fr(r)$ coincides with the computation step $w \underset{r}{\longrightarrow} xuvy$ in γ up to representation of strings by graphs. Similarly, the rule r_γ for $r = (u, \lambda/\beta, v)$ is applicable to $sg(w)_\mu$ if $w = xuvy$ for some x, y using the matching $sg(uv) \xrightarrow{{}_x u\lambda v_y} sg(w) \subseteq sg(w)_\mu$. The splitting with stretching yields the molecules $sg(xurvy)_\mu$ and $sg(\lambda)_{\widehat{r}}$. If the former molecule is fused with $sg(\beta)_{\overline{r}}$ applying the fusion rule $fr(r)$, then one gets $sg(xy\beta vy)_\mu$. This means that the application of r_γ to $sg(w)_\mu$ followed by the application of $fr(r)$ coincides with the computation step $w \underset{r}{\longrightarrow} xu\beta uv$ in γ.

These observations allow to prove the following lemma that links computations in an insertion-deletion system to special derivations in the corresponding splitting/fusion grammar. To formulate the lemma, we use three kinds of multiplication for a given insertion-deletion system $\gamma = (V, T, A, R)$:

1. $m(z)$ for $z \in A$ removes all molecules $sg(z')_\mu$ (via multiplication by 0) for $z' \in A$ with $z' \neq z$ and keeps all others.
2. $m(r)$ for $r = (u, \alpha/\beta, v) \in R$ duplicates the molecule $sg(\beta)_{\overline{r}}$ and keeps all others.
3. $m(\widehat{r})$ for $r = (u, \alpha/\beta, v) \in R$ removes the molecule $sg(\alpha)_{\widehat{r}}$ and keeps all others.

Lemma 2. Let $\gamma = (V, T, A, R)$ be an insertion-deletion system and $SFG(\gamma)$ the corresponding splitting/fusion grammar. Let $d = (w_0 \underset{r_1}{\rightarrow} w_1 \underset{r_2}{\rightarrow} \ldots \underset{r_n}{\rightarrow} w_n)$ be a computation in γ with $w_0 \in A$ and $r_i = (u_i, \alpha_i/\beta_i, v_i) \in R$. Then there is a derivation in $SFG(\gamma)$ of the following form:

$$d_\gamma = (Z_\gamma \underset{m(w_0)}{\Longrightarrow} sg(w_0)_\mu + X_\gamma \overset{4}{\Longrightarrow} sg(w_1)_\mu + X_\gamma \overset{4}{\Longrightarrow} \cdots \overset{4}{\Longrightarrow} sg(w_n)_\mu + X_\gamma),$$

where $X_\gamma = \sum\limits_{r=(u,\alpha/\beta,v)\in R} sg(\beta)_{\overline{r}}$. The sections $sg(w_{i-1})_\mu + X_\gamma \overset{4}{\Longrightarrow} sg(w_i)_\mu + X_\gamma$ for $i = 1, \ldots, n$ are defined by

$$\begin{aligned} sg(w_{i-1})_\mu + X_\gamma &\underset{(r_i)_\gamma}{\Longrightarrow} sg(x_i u_i r_i v_i y_i)_\mu + X_\gamma + sg(\alpha_i)_{\widehat{r}_i} \underset{m(\widehat{r}_i)}{\Longrightarrow} sg(x_i u_i r_i v_i y_i)_\mu + X_\gamma \\ &\underset{m(r_i)}{\Longrightarrow} sg(x_i u_i r_i v_i y_i)_\mu + sg(\beta_i)_{\overline{r}_i} + X_\gamma \underset{fr(r_i)}{\Longrightarrow} sg(w_i)_\mu + X_\gamma \end{aligned}$$

for some $x_i, y_i \in (V \cup R)^*$.

The derivation d_γ is called *insdel*-derivation of d.

Conversely, the derivations in $SFG(\gamma)$ can also be nicely related to the computations in γ.

Lemma 3. Let $\gamma = (V, T, A, R)$ be an insertion-deletion system and $SFG(\gamma)$ the corresponding splitting/fusion grammar. Let $D = (Z_\gamma \overset{*}{\Longrightarrow} H)$ be a derivation in $SFG(\gamma)$ and $sg(w)_\mu \in \mathcal{M}(H)$ for some $w \in V^*$. Then there is an insdel-derivation d_γ of some computation $d = (w_0 \overset{*}{\rightarrow} w)$.

Using these lemmata, one can prove that an insertion-deletion system and the corresponding splitting/fusion grammar generate the same language up to the representation of strings as string graphs.

Theorem 2. *Let $\gamma = (V, T, A, R)$ be an insertion/deletion system and $SFG(\gamma)$ its corresponding splitting/fusion grammar. Then*

$$sg(L(\gamma)) = \{sg(w) \mid w \in L(\gamma)\} = L(SFG(\gamma)).$$

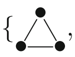
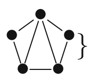
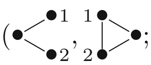
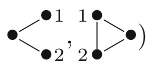
6 2-Splicing Grammars

In the literature, one encounters many variants of splicing systems as they are considered as a potential computational kernel of a future DNA computer (see e.g., [4] for a comprehensive survey). Typically, a rule of a splicing system has the form of a quadruple $(u_1, u_2; u_3, u_4)$ of four strings. It is applicable to two strings w and w' if $w = x_1u_1u_2x_2$ and $w' = x_3u_3u_4x_4$ for some strings x_1, x_2, x_3, and x_4. Such a rule application splits w and w' between u_1, u_2 and u_3, u_4 respectively and recombines the parts into $x_1u_1u_4x_4$ and $x_2u_2u_3x_3$. The operation is an 1-splicing if only the first result is further taken into account; it is a 2-splicing if both results are further considered. In an iterated splicing system, the splicing process is arbitrarily iterated on the resulting strings starting with a given set of strings, called axioms. Finally, an extended iterated splicing system has an additional alphabet of terminal symbols, and its generated language consists of all terminal strings that result from the splicing process. In this section, we introduce 2-splicing grammars generalizing extended iterated 2-splicing systems to hypergraphs as underlying structures.

Definition 7. 1. A *2-splicing grammar* is a system $\mathit{2SG} = (V, T, A, R)$ where V is a finite label alphabet, $T \subseteq V$ is a subalphabet of *terminal labels*, $A \subseteq \mathcal{H}_V$ is a finite set of connected hypergraphs, called *axioms*, and R is a finite set of *rules* of the form $r = (c_1, c_2; c_3, c_4)$ with $c_i \colon [k(r)] \to C_i$ where C_i is a connected hypergraph for each $i = 1, 2, 3, 4$ and some $k(r) \in \mathbb{N}$, called *type of* r.
2. The application of r to $H \in \mathcal{H}_V$ is defined by the application of the two splitting rules with double context $(sr(r_1), c_1, c_2)$ and $(sr(r_2), c_3, c_4)$ to two different molecules of H followed by the application of the fusion rules $fr(r_1)$ and $fr(r_2)$ where the complementary fusion and splitting labels satisfy the condition $\overline{r}_1 = \widehat{r}_2$ and $\overline{r}_2 = \widehat{r}_1$.
3. A *derivation* in *2SG* from H to H' is a sequence of rule applications and multiplications $H = H_0 \Longrightarrow H_1 \Longrightarrow \ldots \Longrightarrow H_n = H'$, shortly denoted by $H \overset{*}{\Longrightarrow} H'$.
4. The *generated language* of *2SG* consists of all terminal molecules of hypergraphs derived from the axioms: $L(\mathit{2SG}) = \{Y \in \mathcal{H}_T \mid \sum\limits_{Z \in A} Z \overset{*}{\Longrightarrow} H, Y \in \mathcal{M}(H)\}$.

Remark 5. 1. Focusing on rule application to the two changed molecules, it looks as in Fig. 3 where the subgraphs $f(C_j)$, $f'(C_j)$ and $f''(C_j)$ for $j = 1, 2$ coincide up to the distinguished nodes.
2. Without loss of generality, one can assume that none of the molecules involved in a derivation disappears because one may duplicate the two molecules that are cut by the rule application beforehand.
3. Note that the 2-splicing grammars are defined without markers. We refrain from their use because all the molecules of the start hypergraph can contribute to the generated language in the same way.

Example 2. To illustrate the concept of 2-splicing grammars, we specify a sample grammar *MOP* that generates a certain type of maximal outerplanar graphs. An undirected unlabeled graph is a *maximal outerplanar graph* (*mop* for short) if it consists of a simple cycle that visits all nodes and a maximum number of further edges such that the graph is planar, but any further edge would yield a non-planar graph.

$\mathit{MOP} = (\{*\}, \{*\}, \{$  ,  $\}, \{r_M = ($  ;  $)\})$

where the type of the single rule is 2 and the morphisms from the discrete graph with two nodes to the contexts are indicated by the numbered nodes.

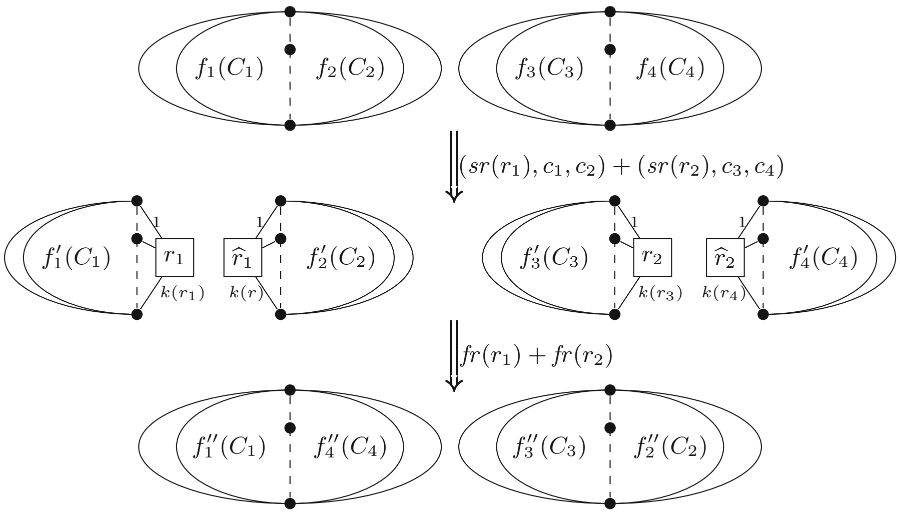
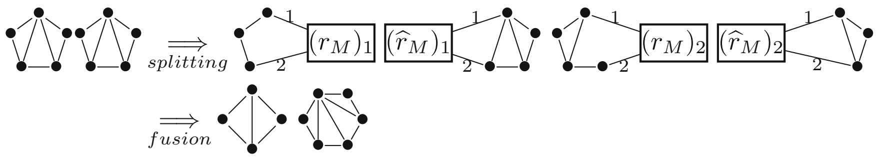
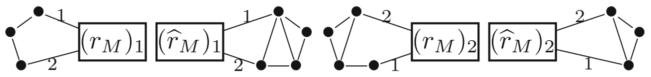


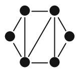
Fig. 3. 2-splicing wrt two molecules

The start graph joins the mops with 3 and 5 nodes. The splitting rule with the first two graphs of the rule as double context cannot be applied to the triangle as the embedding of the two context graphs are required to intersect in the distinguished nodes only. But the rule of *MOP* can be applied to two copies of the mop with 5 nodes in the following way:



An alternative way of the splitting yields



Then the recombination of the third and the second molecule yields .

This means that rule application in *MOP* can generate the mop with 4 nodes and two mops with 6 nodes. Iterating the rule application, one can generarate mops with an arbitrary number of nodes.

Let $\mathit{2SG}_{sg} = (V, T, A, R)$ be a 2-splicing grammar such that $A \subseteq sg(V^*)$ and each rule $r \in R$ has the form $(b_{u_1}, b_{u_2}; b_{u_3}, b_{u_4})$ for some $u_1, u_2, u_3, u_4 \in V^*$ (cf. Sect. 5.2). Then $\sigma = (V, T, A_\sigma, R_\sigma)$ with $A_\sigma = \{z \in V^* \mid sg(z) \in A\}$ and

$R_\sigma = \{(u_1, u_2; u_3, u_4) \mid (b_{u_1}, b_{u_2}; b_{u_3}, b_{u_4}) \in R\}$ defines an extended iterated 2-splicing system (on strings) that is also called extended H-system in [4]. This means that there is an obvious relation between the string case and the hypergraph case on the syntactic level. In contrast to this, the generated language of an extended iterated 2-splicing system is defined by means of an infinite iteration on sets of strings and not by derivations. The iteration process can be carried over to the hypergraph case.

Definition 8. Let $2SG = (V, T, A, R)$ be a 2-splicing grammar. Then its *iterated language* $L'(2SG)$ is defined as follows: $L'(2SG) = \sigma_2^*(A) \cap T^*$ with $\sigma_2^*(A) = \bigcup_{i \in \mathbb{N}} \sigma_2^i(A)$ where $\sigma_2^i(A)$ is inductively defined by $\sigma_2^0(A) = A$, and $\sigma_2^{i+1}(A) = \sigma_2(\sigma_2^i(A))$ for all $i \in \mathbb{N}$ with $\sigma_2(X) = \{M_3, M_4 \mid M_1 + M_2 \underset{r}{\Longrightarrow} M_3 + M_4, r \in R, M_1, M_2 \in X\}$ for all sets X of connected hypergraphs in $\mathcal{H}_V$.

Nicely enough, it turns out that the generated language and the iterated language of a 2-splicing grammar coincide.

Theorem 3. *Let 2SG be a 2-splicing grammar. Then* $L(2SG) = L'(2SG)$.

This means that the hypergraph case is a proper generalization of the string case on the semantic level, too.

7 Conclusion

In this paper, we have related three well-known DNA computing approaches to the framework of splitting/fusion grammars. First, we have recreated Adleman's seminal experiment that marks the origin of DNA computing by means of fusion grammars (where no splitting is needed). Secondly, we have transformed insertion-deletion systems into splitting/fusion grammars. And, thirdly, we have generalized extended iterated 2-splicing systems, that process strings as underlying data structure, by 2-splicing grammars that operate on hypergraphs. Future research in this context may head in various directions:

Further approaches to DNA computing like sticker systems (cf., e.g., [4]) may be related to splitting/fusion grammars.

In analogy to Adleman's experiment, we have solved the Hamiltonian path problem by fusion grammars with additional filter mechanisms in such a way that the decision takes one multiplication step and one parallel fusion step and is correct with high probability. It may be interesting to consider other NP-complete problems and to investigate the computational capability of splitting/fusion grammars in more depth.

In addition to the transformation of insertion-deletion systems, it may be worthwhile to generalize this kind of string processing to hypergraph processing with the hope that interesting examples can be modeled in this way.

Besides extended iterated 2-splicing systems, one encounters many variants based on splicing in the literature. To consider them from the point of view of splitting/fusion grammars may lead to new insights.

In [9], graph multiset transformation was introduced as a computational approach with massive parallelism inspired by DNA computing like splitting/fusion grammars. There the traditional double-pushout rules are used rather than the very special cases of splitting and fusion rules. As multisets and disjoint unions are very similar data structures, a comparison of the two approaches may be worthwhile.

Acknowledgment. We are grateful to the anonymous reviewers for their critical comments that encouraged us to add some more explanations.

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.: Splicing/fusion grammars and their relation to hypergraph grammars. In: Lambers, L., Weber, J. (eds.) ICGT 2018. LNCS, vol. 10887, pp. 3–19. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92991-0_1
4. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing—New Computing Paradigms. Springer, Heidelberg (1998)
5. Kreowski, H.-J., Klempien-Hinrichs, R., Kuske, S.: Some essentials of graph transformation. In: Ésik, Z., Martín-Vide, C., Mitrana, V. (eds.) Recent Advances in Formal Languages and Applications. Studies in Computational Intelligence, vol. 25, pp. 229–254. Springer, Heidelberg (2006). https://doi.org/10.1007/978-3-540-33461-3_9
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, London (1979)
7. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., (eds.) Proceedings of a Symposium on the Complexity of Computer Computations, The IBM Research Symposia Series, pp. 85–103. Plenum Press, New York (1972)
8. Kari, L., Thierrin, G.: Contextual insertions/deletions and computability. Inf. Comput. **131**(1), 47–61 (1996)
9. Kreowski, H.-J., Kuske, S.: Graph multiset transformation as a framework for massively parallel computation. In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) ICGT 2008. LNCS, vol. 5214, pp. 351–365. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87405-8_24