

Decidability and Complexity of the Membership and Emptiness Problem of Fusion Grammars

Aaron Lye

University of Bremen, Department of Mathematics
P.O.Box 33 04 40, 28334 Bremen, Germany

lye@math.uni-bremen.de

Abstract. Fusion grammars are a novel approach to the generation of hypergraphs. A fusion grammar is a hypergraph grammar which provides a start hypergraph of small connected components. To get large connected hypergraphs, they can be copied multiple times and can be fused by the application of fusion rules.

In this paper, we prove that the membership problem of fusion grammars is decidable in non-deterministic polynomial time. Moreover, fusion grammars exist for which deciding membership is NP-complete. We also analyze the emptiness problem for fusion grammars and give a reduction to finding a special non-negative solution for a homogenous linear diophantine equation system.

1 Motivation

Since the early 70's graph grammars as a language generating device are considered [1]. Since then several variants of graph grammars have been introduced and analyzed. Two major branches are hyperedge replacement (see, e.g., [2,3]) and node replacement grammars (see, e.g., [4]). Besides the generative power and structural properties, e.g. about the derivation, decision problems and their complexity are of particular interest. I.e., does there exist an algorithm for deciding whether or not a specific mathematical assertion does or does not have a proof? If yes, how efficient in terms of space and time is this algorithm? Besides the theoretical relevance, answering these questions also has practical implications, e.g. for parsing.

Fusion grammars [5] are a novel approach to the generation of hypergraph languages. They are motivated by the fact, that fusion processes occur in various scientific fields, like DNA computing, chemistry, tiling, fractal geometry, visual modeling, etc. A fusion grammar is a hypergraph grammar which provides a start hypergraph of small connected components. To get large connected hypergraphs, they can be copied multiple times and can be fused by the application of fusion rules.

In [5] it is shown that for substantial fusion grammars the membership problem is decidable. Nevertheless, its complexity and other decision problems, such as the question if a fusion grammar generates the empty language, was left open.

It is of particular interest to know if efficient deterministic algorithms exist for these problems. In this paper, we answer these questions.

The paper is structured as follows. Section 2 introduces basic notions and notations of hypergraphs as far as needed. Section 3 recalls the notion of fusion grammars. In Section 4, we prove that membership is decidable in non-deterministic polynomial time and, furthermore, fusion grammars exist for which deciding membership is NP-complete. Afterwards, in Section 5, we analyze the emptiness problem for fusion grammars and give a reduction to finding a special non-negative solution for a homogenous linear diophantine (i.e., integral) equation system. Section 6 contains a conclusion.

2 Preliminaries

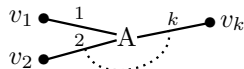
We consider hypergraphs the hyperedges of which are attached to a sequence of nodes and labeled in a given label alphabet Σ .

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*, $att: E \rightarrow V^*$ is a function, called *attachment*, where V^* is a sequence of vertices, and $lab: E \rightarrow \Sigma$ 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. 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}_Σ .

By $[k]$ we denote the discrete hypergraph with the nodes $1, \dots, k$ for some $k \in \mathbb{N}_{>0}$ and by A^\bullet we denote the hypergraph consisting of the nodes $1, \dots, k$ to which a single *A-hyperedge* is attached.

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



i.e., numbered tentacles connect the label with the corresponding attachment nodes. We assume the existence of a special label $*$ $\in \Sigma$ that is omitted in drawings. In this way, unlabeled hyperedges are represented by hyperedges labeled with $*$.

Given $H, H' \in \mathcal{H}_\Sigma$, the *disjoint union* of H and H' is denoted by $H + H'$. Further, $k \cdot H$ denotes the disjoint union of H with itself k times.

Given $H, H' \in \mathcal{H}_\Sigma$, a *hypergraph morphism* $g: H \rightarrow H'$ consists of two mappings $g_V: V_H \rightarrow V_{H'}$ and $g_E: E_H \rightarrow E_{H'}$ 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 $g_V^*: V_H^* \rightarrow V_{H'}^*$ 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 $v_1 \cdots v_n \in V_H^*$.

The *fusion* of nodes is defined as a quotient by means of an equivalence relation \equiv on the set of nodes V_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. It is

easy to see that $f: H \rightarrow H/\equiv$ 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*.

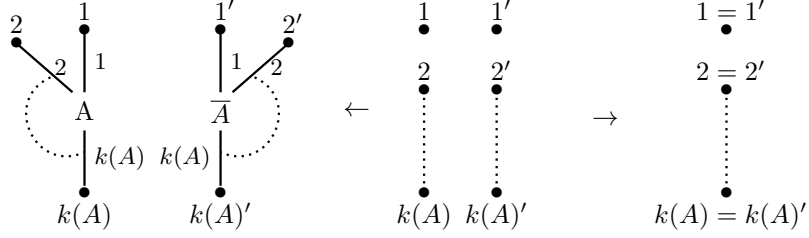
Let $H \in \mathcal{H}_\Sigma$. Then a sequence of triples $(i_1, e_1, o_1) \dots (i_n, e_n, o_n) \in (\mathbb{N} \times E_H \times \mathbb{N})^*$ is a *path* from $v \in V_H$ to $v' \in V_H$ if $v = \text{att}_H(e_1)_{i_1}$, $v' = \text{att}_H(e_n)_{o_n}$ and $\text{att}_H(e_j)_{o_j} = \text{att}_H(e_{j+1})_{i_{j+1}}$ for $j = 1, \dots, n-1$ where, for each $e \in E_H$, $\text{att}_H(e)_i = v_i$ for $\text{att}_H(e) = v_1 \dots v_k$ and $i = 1, \dots, k$. H is *connected* if each two nodes are connected by a path. A subgraph C of H , denoted by $C \subseteq H$, 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 $\mathcal{C}(H)$.

We use the *multiplication* of H defined by means of $\mathcal{C}(H)$ as follows. Let $m: \mathcal{C}(H) \rightarrow \mathbb{N}_{>0}$ be a mapping, called *multiplicity*, then $m \cdot H = \sum_{C \in \mathcal{C}(H)} m(C) \cdot C$.

3 Fusion Grammars

Besides a start hypergraph, a fusion grammar provides a set of fusion rules. The application of a fusion rule merges certain nodes which are given by two complementary hyperedges. Complementarity is defined on a set F of fusion labels that comes together with a complementary label \bar{A} for each $A \in F$. Given a hypergraph, the set of all possible fusions is finite as fusion rules never create anything. To overcome this limitation, we allow arbitrary multiplications of disjoint components within derivations. The language generated by a fusion grammar does not consist of all terminal hypergraphs that are derived from the start hypergraph, but are chosen in a slightly more complicated way. The problem is that the multiplications may also produce components that are not really needed. Therefore, we consider only terminal connected components of the derived hypergraphs as members of the generated language. Moreover, we use markers. They allow us to distinguish between wanted and unwanted terminal components; that is, markers identify components of the start hypergraph that contribute to the generation of a hypergraph. The language consists of all resulting connected components that contain no fusion symbols and at least one marker symbol, where marker symbols are removed in the end.

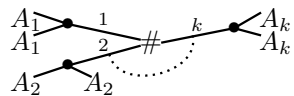
Definition 1. 1. Let $F \subseteq \Sigma$ and $k: F \rightarrow \mathbb{N}$ be a type function. Let $\bar{A} \notin F$ be the complementary label for each $A \in F$ such that $\bar{A} \neq \bar{B}$ for all $A \neq B$. Let $\bar{F} = \{\bar{A} \mid A \in F\}$. The typing is extended to complementary labels by $k(\bar{A}) = k(A)$ for all $A \in F$. Then F is called *fusion alphabet* and $A \in F$ specifies the following *fusion rule*, denoted by $fr(A)$:



The number at the nodes identify them so that the left-hand side inclusion and the right-hand side morphism are made visible. The morphism maps each attachment node and its primed counterpart to the same right-hand side node.

2. A *fusion grammar* is a system $FG = (Z, F, M, T)$ where $Z \in \mathcal{H}_{F \cup \bar{F} \cup T \cup M}$ is a *start hypergraph*, $F \subseteq \Sigma$ is a fusion alphabet, $M \subseteq \Sigma$ with $M \cap (F \cup \bar{F}) = \emptyset$ is a set of *markers*, and $T \subseteq \Sigma$ with $T \cap (F \cup \bar{F}) = \emptyset = T \cap M$ is a set of *terminal labels*.
3. A *derivation step* $H \Longrightarrow H'$ for some $H, H' \in \mathcal{H}_\Sigma$ is either a fusion rule application $H \xrightarrow{fr} H'$ or a multiplication $H \xrightarrow{m} m \cdot H$ for some multiplicity m .
4. A *derivation* is the reflexive and transitive closure of derivation steps, i.e., $H \xrightarrow{n} H'$ of length $n \geq 0$ is a sequence $H_0 \Longrightarrow H_1 \Longrightarrow \dots \Longrightarrow H_n$ with $H = H_0$ and $H' = H_n$. If the length does not matter, one may write $H \xrightarrow{*} H'$.
5. $L(FG) = \{rem_M(Y) \mid Z \xrightarrow{*} H, Y \in \mathcal{C}(H) \cap (\mathcal{H}_{T \cup M} - \mathcal{H}_T)\}$ is the *generated language* of FG where $rem_M(Y)$ is the hypergraph obtained when removing all hyperedges with labels in M from Y .

In order to generate hypergraphs with terminal hyperedges, the start hypergraph must contain at least one terminal hyperedge. Fusion grammars in which every component of the start hypergraph contains terminal edges are called *substantial*. In these grammars every multiplication increases the number of terminal hyperedges, every fusion of two disjoint connected components results in some connected component with more terminal hyperedges as the two source components. The fusion of two complementary hyperedges in the same connected component keeps the number of terminal hyperedges.

Example 1. Let $F = \{A_1, \dots, A_k\}$ with $type(A_i) = 1$ for all $A_i \in F$. Let $K = \{1, \dots, k\} \times \{1, \dots, k\}$ for some $k \in \mathbb{N}_{>0}$. Consider the fusion grammar $FG = (z_\# + \sum_{(i,j) \in K} z_{i,j}, F, \{\#\}, \{*\})$ where $z_\# =$  and $z_{i,j} =$

$\bar{A}_i - \bullet - \bullet - \bar{A}_j$. Then one may get symmetric structures like circles, e.g. like the one in Fig. 1a where $z_\#$ is fused with k connected components $z_{i,j}$ for respective i, j , or bipartite graphs like the one in Fig. 1b, where $z_\#$ is multiplied once and then k $z_{i,j}$ components are fused with both $z_\#$. In both cases π denotes some permutation on $1, \dots, k$. But one may also get also very irregular structures. Nevertheless, the language can be characterized as follows. All hypergraphs of the



Fig. 1: Derived hypergraphs of the fusion grammar in Example 1 before the markers are removed

language have in common that, after removing the $\#$ -hyperedges, all hyperedges are of type 2 and all vertices have degree 2.

Note that FG is not substantial. However, it can be easily transformed into a substantial grammar by replacing $z_{\#}$ by $\sum_{(i,j) \in K} z_{\#,i,j}$ where $z_{\#,i,j}$ is the result of the application of the fusion rule $fr(A_i)$ to $z_{\#}$ and $z_{i,j}$.

4 Membership

In [5] it is shown that for substantial fusion grammars the membership problem is decidable. In this section we focus on the complexity of this problem.

Theorem 1. *For every substantial fusion grammar FG , the membership problem is in NP (polynomial in the size of the asked hypergraph).*

Proof. Given a substantial fusion grammar (Z, F, M, T) and some hypergraph $H \in \mathcal{H}_{\Sigma}$. Because the fusion grammar is substantial, each connected component in the start hypergraph contains at least one terminal hyperedge. Hence, one needs at most the fusion of k connected components of the start hypergraph to get H where k is the number of terminal labeled hyperedges in H . Let $H = rem_M(H')$. Then by [5, Corollary 1] the derivation $Z \xrightarrow{*} H' + X$ can be rewritten as the most parallelized derivation where one may start with the multiplication, and then performs the necessary fusions, i.e., $Z \xrightarrow{m} m \cdot Z \xrightarrow{fr(F)_+} H' + X$, where X denotes the disjoint union of connected components which are present but not needed. The multiplicity m is bounded by k , i.e. $m \leq k$. By the fact that the number of possible fusions is finite, the length of the derivation is linear in the size of the graph generated.

In other words, for every hypergraph H of size k , if $H \in L(FG)$ then H has a derivation of length $\leq f(k)$ for some linear function f . Hence, the derivation and the selection of the connected component can be non-deterministically guessed by a non-deterministic Turing machine in polynomial time. Afterwards, a test

whether the generated hypergraph is isomorphic to H can be performed in non-deterministic polynomial time again. This second guess and the test takes linear time in k . \square

Theorem 2. *There is a fusion grammar FG such that deciding membership for $L(FG)$ is NP-complete.*

Proof. This follows from the fact that the result holds even for hyperedge replacement grammars (cf. [3, Theorem 2.7.2]), which fusion grammars can simulate [5, Theorem 2]. The transformation of hyperedge replacement grammars into fusion grammars presented in [5] takes linear time. More precisely, let (N, T, P, S) be some hyperedge replacement grammar where N and T are two finite sets of non-terminal and terminal symbols, P is a finite set of hyperedge replacement rules and $S \in N$ is a start symbol. Then $((S, \#)^\bullet + \sum_{r \in P} hgr(r), N, \{\#\}, T)$ with $\# \notin N \cup T$ is the constructed fusion grammar where $(S, \#)^\bullet$ denotes the discrete hypergraph of type S vertices equipped with two hyperedges labeled with S and $\#$, respectively, and $hgr(r)$ is the hypergraph representation of the rule which is the right-hand side of the rule equipped with some hyperedge complementary to the left-hand side. Consequently, the transformation is linear in P . \square

5 Emptiness

In this section we discuss the emptiness problem for fusion grammars. In contrast to the membership problem we do not restrict ourselves to substantial fusion grammars. We show that the non-emptiness problem for fusion grammars can be reduced to finding a special non-negative solution for a homogenous linear diophantine (i.e., integral) equation system.

Definition 2. Let $FG = (Z, F, M, T)$ be a fusion grammar, let c be the number of connected components in Z and let r be the size of the fusion alphabet F . Fix an ordering z_1, \dots, z_c for the connected components in Z and an ordering f_1, \dots, f_r for the fusion alphabet. Define a corresponding $\mathbb{Z}^{r \times c}$ matrix for the grammar as follows.

1. Each connected component can be represented as a vector over \mathbb{Z}^r and all together can be represented by a $\mathbb{Z}^{r \times c}$ matrix where columns represent connected components and each row concerns one fusion and its complementary label. The scalars of the matrix are the difference between the number of F -labeled hyperedges and the respective complementary ones, i.e.,

$$D = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1c} \\ d_{21} & d_{22} & \dots & d_{2c} \\ \vdots & \vdots & & \vdots \\ d_{r1} & d_{r2} & \dots & d_{rc} \end{pmatrix}, \quad d_{ij} = a_{ij} - b_{ij}$$

where a_{ij} is the number of f_i -labeled hyperedges and b_{ij} is the number of \bar{f}_i -labeled hyperedges in z_j .

2. Define the index set of significant indices by $I = \{j \in \{1, \dots, c\} \mid z_j \text{ has a marker}\}$.

Example 2. Consider the fusion grammar $FG = (\sum_{j=1}^5 z_i, \{A, B\}, \{\#\}, \{*\})$ with $\text{type}(A) = \text{type}(B) = 2$ and z_i for $i = 1, \dots, 5$ as follows.

$$z_1 = \# \begin{array}{c} A \\ \square \end{array} B \quad z_2 = \begin{array}{c} A \\ \square \\ \overline{A} \end{array} \quad z_3 = \begin{array}{c} \square \\ \overline{A} \end{array} \quad z_4 = \overline{B} \begin{array}{c} A \\ \square \square \end{array} B \quad z_5 = \overline{B} \square$$

The respective column vectors $d_1, \dots, d_5 \in \mathbb{Z}^2$ are:

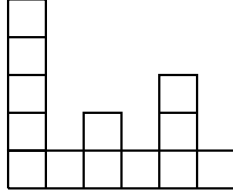
$$d_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, d_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, d_3 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, d_4 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, d_5 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

where the upper entry corresponds to the number of A -hyperedges and the lower entry corresponds to B -hyperedges. Together they form the matrix

$$D = \begin{pmatrix} 1 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

$I = \{1\}$ because only z_1 carries a marker.

It is easy to see that the language is not empty and generates graphs like the following.



Theorem 3. Let $FG = (Z, F, M, T)$ be a fusion grammar, let D be its matrix representation and let I be some corresponding index set of significant indices. We introduce some multiplicity variable $k_j \in \mathbb{N}$ for each z_j . Then $L(FG)$ is not empty if and only if there exist assignments to the multiplicity variables such that

$$D \cdot k = 0 \tag{1}$$

where $k = (k_1, \dots, k_c)^T$ and

$$\sum_{j \in I} k_j \geq 1. \tag{2}$$

Proof. In the following the two implications are proven.

Let $L(FG)$ be not empty, i.e., there exists some $H \in L(FG)$. Then there is a derivation $Z \xrightarrow{*} H' + X$ with $H = \text{rem}_M(H')$, H' connected and $H' \in \mathcal{H}_{T \cup M} - \mathcal{H}_T$. Moreover, H' is a fusion of connected components of Z . Let z_1, \dots, z_c be some ordering of the connected components in Z . Let m_j be the number of copies

of z_j needed to construct H' for $j = 1, \dots, c$. Let f_1, \dots, f_r be the elements of F in some order. Let a_{ij} be the number of f_i -labeled hyperedges and b_{ij} be the number of \bar{f}_i -labeled hyperedges in z_j for $i = 1, \dots, r$ and $j = 1, \dots, c$. As H' is the fusion of all these connected components and does not have any hyperedges labeled in $F \cup \bar{F}$, all those $F \cup \bar{F}$ -hyperedges disappear in the fusions. As a single fusion consumes exactly one pair f_i, \bar{f}_i for some i , the number of f_i -hyperedges and the number of \bar{f}_i -hyperedges for $i = 1, \dots, r$ in the involved connected components must be equal, i.e.,

$$\sum_{j=1}^c a_{ij} \cdot m_j - \sum_{j=1}^c b_{ij} \cdot m_j = \sum_{j=1}^c (a_{ij} - b_{ij}) \cdot m_j = 0.$$

This means that the vector (m_1, \dots, m_c) is a solution of the homogeneous equation system given by the matrix $(d_{ij})_{i=1, \dots, r, j=1, \dots, c}$ with $d_{ij} = a_{ij} - b_{ij}$ defined in Equation (1). Moreover, H' contains a marker hyperedge. Hence, $m_i > 0$ for some z_i with a marker hyperedge, i.e., Equation (2) is also satisfied.

Conversely, let $(m_1, \dots, m_c) \neq 0$ be a solution for the system in Equation (1) which satisfies Equation (2). Then this vector gives the multiplicity of the respective connected components of Z such that the number of f_i -hyperedges and the number of \bar{f}_i -hyperedges for $i = 1, \dots, r$ are equal. After respective multiplications the connected components $\sum_{j=1}^c m_j \cdot z_j$ (where z_j is omitted if $m_j = 0$) can be fused arbitrarily yielding one or several connected components. By the assumption that $m_i > 0$ for some z_i with a marker hyperedge at least one of the resulting connected components contributes to the language after removing the marker. Hence, the language is not empty. \square

Remark 1. The solution for Equation (1) is invariant under

- permutation of rows of D because the order of fusion variables is irrelevant;
- permutation of columns of D (with the same permutation applied on the column vector k) because this is simply fixing a different ordering on the connected components of the start hypergraph;
- addition of columns which corresponds to fusion of connected components.

Remark 2. If a column for some connected components with marker is initially the zero vector, then we can directly conclude that the system has a solution and the language is not empty.

Example 3. Consider the following matrix

$$\begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

where the third column vector corresponds to some connected component with marker. The zero vector may have been the result by the fact, that either the

connected component has no F -labeled hyperedges or the number of f_i - and \bar{f}_i -labeled hyperedges for all $i = 1, 2, 3$ are equal. In the latter case obviously the f_i - and \bar{f}_i -labeled hyperedges for all $i = 1, 2, 3$ can be fused arbitrarily resulting in some connected component the hyperedges of which are labeled only with terminal and marker symbols.

Remark 3. Solving the system corresponds to finding a linear dependence of some column corresponding to some connected component with marker with the remaining column vectors.

From this observation, we can directly conclude that if no column vector corresponding to some connected component is marker is the zero vector and some row contains no negative or no positive entries, then the system has no solution because the only possible assignment would be $k_1 = \dots = k_c = 0$ which violates Equation (2).

Example 4. Given four connected components z_1, \dots, z_4 over three fusion symbols in some start hypergraph where z_2 contains a marker. Let these components be such that

$$D = \begin{pmatrix} 1 & 3 & 1 & 2 \\ 0 & -1 & 1 & 0 \\ -2 & 3 & 0 & 0 \end{pmatrix}.$$

By the fact that the first row contains only positive entries, there are no multiplicities $k_1, \dots, k_4 \in \mathbb{N}$ such that $k_2 > 0$ and $k_1 + 3k_2 + k_3 + 2k_4 = 0$. Hence, the language is empty.

Remark 4. The two tests mentioned in Remark 2 and Remark 3 can be tested in linear time in the size of the matrix D .

Example 5 (Continue Example 2). Recall $D = \begin{pmatrix} 1 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix}$. Because only z_1 carries a marker hyperedge, we have to test $d_1 = (0, 0)^T$. But $d_1 = (1, 1) \neq (0, 0)$. Hence, the first test fails. Further, every row contains positive and negative scalars. Therefore, the second test fails as well. Hence, the two tests do not help in this particular case.

However, it is easy to see that the column vectors d_1, \dots, d_5 are in linear dependence: $k_1 - k_3 + k_4 = 0$ implies $k_3 - k_4 = k_1$ and $k_1 - k_5 = 0$ implies $k_1 = k_5$. Further, $k_1 \geq 1$ must be satisfied. Besides this, it is easy to see that k_4 can be neglected (e.g. $k_1 = 7, k_3 = 7, k_4 = 0$).

Assume k_1, \dots, k_5 being a solution for Equation (1) satisfying Equation (2). We elaborate the cases $k_1 = 1$ and $k_1 > 1$ a bit.

In the case $k_1 = 1$, we have due to the linear equations $k_5 = 1$ and $k_4 = k_3 - 1$. k_2 can be arbitrary. The case $k_2 = 4, k_3 = 3, k_4 = 2$ is depicted at the end of Example 2.

The case $k_1 > 1$ may need more explanation. Even though that there are several copies, precisely, k_1 copies of z_1 (i.e., several connected components with markers), this does not compromise the result, because every possible fusion of

all the connected components results in connected components which are labeled only with terminal and marker-symbols.

In contrast, if z_5 is removed from the start hypergraph, then there is no multiplicity such that both equations is satisfied because $k_1 = 0$ is required to satisfy Equation (1) but this violates Equation (2). The language is empty as there is no derivation resulting in connected components where connected components are only labeled with terminal and marker symbols.

- Remark 5.*
1. We are not asking if the columns of the system are linear dependent but if specific columns (at least one corresponding to a marker component) are linear combinations.
 2. Usually, a linear equation system can be solved in deterministic polynomial time using Gaussian elimination. In our case applying Gaussian elimination is not possible as we consider a free \mathbb{Z} module over \mathbb{Z} , i.e., our underlying ring for our scalars is no field but a principle ideal domain. Hence, multiplication of rows by a scalar is not possible, because this operation is not invertible. As a result we cannot apply Gaussian elimination to obtain the reduced row echelon form of the matrix D . But the Hermite normal form, which is also a triangular normal form, can be calculated.
 3. We are even more restricted as we are not searching for any solution but $k_1, \dots, k_c \in \mathbb{N}$.
 4. Further, it is well known that a homogeneous linear equation system has either exactly one solution (the trivial solution $k_1 = k_2 = \dots = k_n = 0$) or infinite many solutions (including the trivial one) [6, Section I.5]. But the trivial solution does not satisfy Equation (2). Thus, if Equation (1) has only the trivial solution, then the language is empty.

Fact 1 *$Ax = b$ has an integer solution x if and only if the system $Hy = b$ has an integer solution y where $Uy = x$, U the corresponding unimodular matrix U for A to obtain H and H is the column style Hermite normal form of H .*

Because H is triangular checking $Hy = b$ has an integer solution is easier than $Ax = b$. Further, in our case $b = 0$.

Votyakov and Frumkin [7] showed that for any system $Ax = 0$ of homogenous linear diophantine (i.e., integral) equations we can find a basis for the solution set in polynomial time. Further, the Hermite normal form H and the unimodular matrix U can be computed in deterministic polynomial time (cf. [8]).

Open Question: Does the basis for the solution set satisfy $k_1, \dots, k_c \in \mathbb{N}$?

Conjecture: This is the case and this can be checked in polynomial time.

6 Conclusion

In this paper we analyzed decidability and complexity of the membership and emptiness problem of fusion grammars.

Fusion grammars are closely related to hyperedge replacement grammars, but are also more powerful. This close relation results that membership can be computed in NP and also that there are languages for which deciding membership is NP-complete.

We analyzed the emptiness problem for fusion grammars and gave a reduction to finding a special non-negative solution for a homogenous linear diophantine (i.e., integral) equation system. These homogenous linear diophantine equation system are well understood. However, due to the fact, that we are not asking if the columns of the system are linear dependent but if specific columns (at least one corresponding to a marker component) are linear combinations and that we are even more restricted as we are not searching for any solution but $k_1, \dots, k_c \in \mathbb{N}$, the complexity of the decision procedure remains to show. We conjecture, even though that fusion grammars are more powerful the emptiness problem remains efficient computable.

Further research directions may be other decision problems like finiteness or equivalence of languages generated by fusion grammars. A closer analysis of the complexity of the membership problem for non-substantial may be interesting as well.

It is also very interesting that the proof for deciding emptiness relies only on a quantitative argument, i.e., it is irrelevant how the connected components are fused. As a consequence, it may be interesting to analyze this closure property of permutations further.

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

References

1. Hartmut Ehrig, Michael Pfender, and Hans-Jürgen Schneider. Graph grammars: An algebraic approach. In *IEEE Conf. on Automata and Switching Theory*, pages 167–180, Iowa City, 1973.
2. Annegret Habel. *Hyperedge Replacement: Grammars and Languages*, volume 643 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1992.
3. Frank Drewes, Annegret Habel, and Hans-Jörg Kreowski. Hyperedge replacement graph grammars. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations*, chapter 2, pages 95–162. World Scientific, 1997.
4. Joost Engelfriet and Grzegorz Rozenberg. Node replacement graph grammars. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations*, chapter 1, pages 1–94. World Scientific, 1997.
5. Hans-Jörg Kreowski, Sabine Kuske, and Aaron Lye. Fusion Grammars: A Novel Approach to the Generation of Graph Languages. In Detlef Plump and Juan de Lara, editors, *Proc. 10th International Conference on Graph Transformation (ICGT 2017)*, volume 10373 of *Lecture Notes in Computer Science*, pages 90–105. Springer, 2017.

6. Lothar Papula. *Mathematik für Ingenieure und Naturwissenschaftler. Band 2.* Vieweg & Teubner, 2009. 12. Auflage.
7. A. A. Votyakov and M.A. Frumkin. An algorithm for finding the general integer solution of a system of linear equations (in russian). In A.A. Fridman, editor, *Issledovaniya po diskretnoi optimizatsii [Studies in Discrete Optimization]*, pages 128–140, 1976.
8. Alexander Schrijver. *Theory of linear and integer programming.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1986.