Fusion Grammars:
A Novel Approach to the Generation of Hypergraph Languages

Aaron Lye
joint work with Hans-Jörg Kreowski, Sabine Kuske

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

26.02.2017

Published at the International Conference for Graph Transformation 2017
Motivation

- DNA computing: Adleman’s experiment, sticker systems
- chemistry
- tiling
- fractal geometry
- visual modeling
- ...

Commonalities?
DNA computing

double strands of pairs \((A, T)\) and \((C, G)\)

Adleman’s experiment (1994): solution of the NP-hard Hamiltonian-path problem by a polynomial number of steps

constructing short DNA stands

- doubling by polymerase chain reaction: \(n\) repetitions yield \(2^n\) copies

fusion of complementary sticky ends

reading (sequencing): filtering of DNA molecules of certain lengths and with certain substrands

Figure 1.24: Ligation
BPMN diagrams, Sierpinski triangles

getorder A
produce A
produce B
getorder B

send A
send B

{}
Graph transformation

Graphs:

- sets of objects with relations between them
- static structure

Graph transformation:

- concepts of graphs and rules with various methods from the theory of formal languages
- the theory of concurrency
- a spectrum of applications
A hypergraph over $\Sigma$ is a system 
$H = (V, E, att: E \to V^*, lab: E \to \Sigma)$, $V, E$ finite sets

length of the attachment $att(e)$ for $e \in E$ is called type of $e$

$e$ is called $A$-hyperedge if $lab(e) = A$ 

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 $\Sigma$ is denoted by $\mathcal{H}_\Sigma$. 
Subgraph, removal, dangling condition

$H$ is a subhypergraph of $H'$ denoted by $H \subseteq H'$ if $V_H \subseteq V_{H'}$, $E_H \subseteq E_{H'}$, $\text{att}_H(e) = \text{att}_{H'}(e)$, and $\text{lab}_H(e) = \text{lab}_{H'}(e)$ for all $e \in E_H$. 
Subgraph, removal, dangling condition

$H$ is a subhypergraph of $H'$ denoted by $H \subseteq H'$ if $V_H \subseteq V_{H'}$, $E_H \subseteq E_{H'}$, $\text{att}_H(e) = \text{att}_{H'}(e)$, and $\text{lab}_H(e) = \text{lab}_{H'}(e)$ for all $e \in E_H$.

Let $H' \in \mathcal{H}_\Sigma$, $V \subseteq V_{H'}$, $E \subseteq E_{H'}$. Then the removal of $(V, E)$ from $H'$ given by $H = H' - (V, E) = (V_{H'} - V, E_{H'} - E, \text{att}_H, \text{lab}_H)$ with $\text{att}_H(e) = \text{att}_{H'}(e)$ and $\text{lab}_H(e) = \text{lab}_{H'}(e)$ for all $e \in E_{H'} - E$

$H' - (V, E)$ defines a subgraph $H \subseteq H'$ if $\text{att}_{H'}(e) \in (V_{H'} - V)^*$ for all $e \in E_{H'} - E$

dangling condition
Hypergraph morphism $g : H \rightarrow H'$

$g_V : V_H \rightarrow V_{H'}$ and $g_E : E_H \rightarrow E_{H'}$

such that $\forall e \in E_H :$

$lab_{H'}(g_E(e)) = lab_H(e)$ and

$att_{H'}(g_E(e)) = g_V^*(att_H(e))$

where $g_V^* : V_H^* \rightarrow V_{H'}^*$ is the canonical extension of $g_V$

$g_V^*(v_1 \cdots v_n) = g_V(v_1) \cdots g_V(v_n)$ for all $v_1 \cdots v_n \in V_H^*$. 
Hypergraph morphism \( g : H \rightarrow H' \)

\[ g_V : V_H \rightarrow V_{H'} \text{ and } g_E : E_H \rightarrow E_{H'} \]

such that \( \forall e \in E_H : \)
\[ lab_{H'}(g_E(e)) = lab_H(e) \text{ and } \]
\[ att_{H'}(g_E(e)) = g_V^*(att_H(e)) \]

where \( g_V^* : V_H^* \rightarrow V_{H'}^* \) is the canonical extension of \( g_V \)
\[ g_V^*(v_1 \cdots v_n) = g_V(v_1) \cdots g_V(v_n) \text{ for all } v_1 \cdots v_n \in V_H^*. \]

\( H \subseteq H' \) implies that the two inclusions \( V_H \subseteq V_{H'} \) and \( E_H \subseteq E_{H'} \)
define a morphism from \( H \rightarrow H' \).
Given a morphism \( g : H \rightarrow H' \), the image of \( H \) in \( H' \) under \( g \)
defines a subgraph \( g(H) \subseteq H' \).
Rule and rule application

rule: \( \gamma = (L \leftarrow K \rightarrow R) \quad L, K, R \in \mathcal{H}_\Sigma \)
\( L \supseteq K, K \to R \) injective on \( E_K \)

rule application: \( G \xrightarrow{\gamma} H \) (direct derivation)

\[
\begin{array}{c}
L \\ \downarrow g \\
G \\
\end{array} \quad \begin{array}{c}
K \\ \downarrow d \\
D \\
\end{array} \quad \begin{array}{c}
R \\ \downarrow h \\
H \\
\end{array}
\]

\[
\begin{array}{c}
\leftarrow l \\
\leftarrow c \\
\end{array} \quad \begin{array}{c}
r \\
f \\
\end{array}
\]

matching morphism \( g : L \to G \)
subject to an application condition (gluing condition)

- dangling condition
- identification condition
Rule and rule application

rule: \( \gamma = (L \leftarrow K \rightarrow R) \) \( L, K, R \in \mathcal{H}_\Sigma \)
\( L \supseteq K, K \rightarrow R \) injective on \( E_K \)

rule application: \( G \xrightarrow{\gamma} H \) (direct derivation)

\[
\begin{array}{ccc}
L & \xrightarrow{l} & K \\
\downarrow g & & \downarrow d \\
G & \xrightarrow{c} & D \\
\downarrow & & \downarrow f \\
\end{array}
\begin{array}{ccc}
 & & \\
 & & \\
 & & \\
R & \xrightarrow{r} & H \\
\end{array}
\]

matching morphism \( g : L \rightarrow G \)
subject to an application condition (gluing condition)

- dangling condition
- identification condition

derivation: reflexive and transitive closure
A hypergraph grammar is a system \( HG = (N, T, P, S) \)

\( N, T \subseteq \Sigma, T \cap N = \emptyset, S \in N, \)

\( A \in N \) has a type \( k(A) \in \mathbb{N} \)

\( P \) finite set of rules

\( L(HG) \) is defined as \( \{ X \in \mathcal{H}_T \mid S \overset{*}{\Rightarrow}_{P} X \} \).

hyperedge replacement grammar

rules of the form \( A \overset{*}{\Rightarrow} [k(A)] \subseteq R \)
Fusion

Fusion alphabet $F \subseteq \Sigma$

with a type $k(A)$ for each $A \in F$ and

with a disjoint complementary copy $\overline{F} \subseteq \Sigma$

fusion rule $fr(A)$

\[
\begin{align*}
fr(A) &= (A^\bullet + \overline{A}^\bullet) \xleftarrow{\langle in_1 + in_2 \rangle} [k(A)] + [k(A)] \xrightarrow{\langle 1[k(A)], 1[k(A)] \rangle} [k(A)]) \\
where \; \text{in}_1 + \text{in}_2 \; \text{is} \; \langle \text{in}_{A^\bullet} \circ \text{in}_1, \text{in}_{\overline{A}^\bullet} \circ \text{in}_2 \rangle \; \text{and} \; \text{in}_A \; \text{is} \; \text{the inclusion of} \; [k(A)] \; \text{into} \; A^\bullet \; \text{and} \; \text{in}_{\overline{A}} \; \text{is} \; \text{the inclusion of} \; [k(A)] \; \text{into} \; \overline{A}^\bullet.
\end{align*}
\]
Fusion grammar

\[ FG = (Z, F, M, T) \]

\[ F, M, T \subseteq \Sigma, \text{ fusion, marker, terminal alphabet} \]

\[ M \cap (F \cup \overline{F}) = \emptyset, \ T \cap (F \cup \overline{F}) = \emptyset = T \cap M \]

\[ Z \in \mathcal{H}_{FU\overline{F}UTUM} \]

direct derivation is either

- a rule application \( H \xrightarrow{r} H' \)
  for some parallel rule over \( fr(F) = \{ fr(f) \mid f \in F \} \) or
- a multiplication \( H \xrightarrow{m} m \cdot H \)
  for some multiplicity \( m : C(H) \rightarrow \mathbb{N} \).
Fusion grammar (generated language)

\[ FG = (Z, F, M, T) \]

\[ M = \emptyset : L(FG) = \{ Y \mid Z \xrightarrow{*} H, Y \in C(H) \cap H_T \} \]
Fusion grammar (generated language)

\[ FG = (Z, F, M, T) \]

\[ M = \emptyset: \]

\[ L(FG) = \{ Y \mid Z \Rightarrow^* H, Y \in C(H) \cap H_T \} \]

\[ M \neq \emptyset: \]

\[ L(FG) = \{ \text{rem}_M(Y) \mid Z \Rightarrow^* H, Y \in C(H) \cap (H_T \cup M - H_T) \} \]
Flexible parallelism

parallel independent iff no hyperedge is matched twice

always sequential independent

successive multiplications can be done simultaneously

fusion followed by multiplication can be interchanged

Corollary

the language can be generated by derivations of length 2
(1 multiplication; 1 massively parallel fusion)
Membership problem

A fusion grammar FG is *substantial* if none of the connected components of the start hypergraph consists of fusion and marker hyperedges only

Recall: Membership problem:
Given $H \in \mathcal{H}_T$ and FG.
Is $H \in L(FG)$?

**Theorem**

*The membership problem of substantial fusion grammars is solvable*
Relation: HRG to FG

hyperedge replacement grammars simulated by fusion grammars

HRG with connected right-hand sides

Theorem
\[ L(HRG) = L(FG(HRG)) \]
Relation: HRG to FG

hyperedge replacement grammars simulated by fusion grammars

HRG with connected right-hand sides

Theorem

\[ L(HRG) = L(FG(HRG)) \]

fusion grammars are more powerful than hyperedge replacement grammars, i.e.

Theorem

\[ HRG \subsetneq FG \]
Conclusion and further work

Fusion grammars are a new device for graph language generation with some first promising results

- flexible parallelism
- solvable membership problem
- $HRG \subsetneq FG$

Further work:

- splicing
- complexity analysis of membership and emptiness problem
- relation to monotone hypergraph grammars

Thank you! Questions?