Splicing/Fusion Grammars and Their Relation to (Chomsky and) Hypergraph Grammars

Hans-Jörg Kreowski [1], Sabine Kuske [1] and *Aaron Lye* [2]

University of Bremen
[1] Department of Computer Science, [2] Department of Mathematics
P.O.Box 33 04 40, 28334 Bremen, Germany
{kreo,kuske,lye}@informatik.uni-bremen.de

25.06.2018

11th International Conference on Graph Transformation (ICGT)

Motivation

We introduced fusion grammars at ICGT17.

Formal framework for fusion processes in:

- DNA computing
- chemistry
- tiling
- fractal geometry
- visual modeling
- etc

DNA computing

fusion

(ligation)

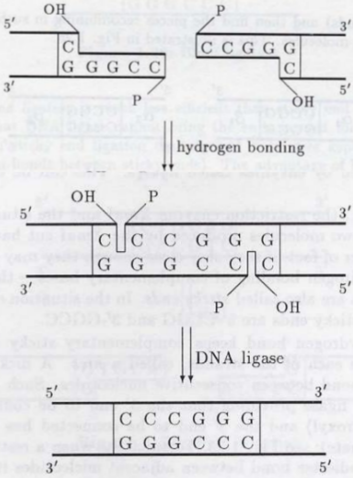


Figure 1.24: Ligation

DNA computing

fusion
(ligation)

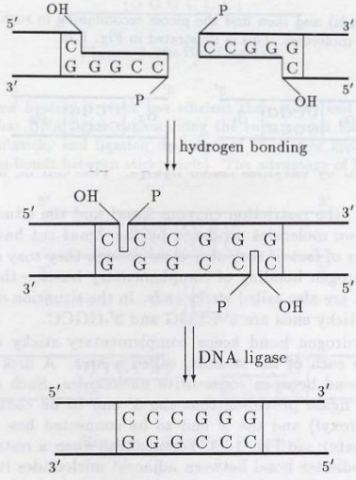
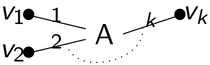


Figure 1.24: Ligation

splicing
(triggered by enzymes)

Hypergraph

We consider *hypergraphs* over Σ with *hyperedges* like



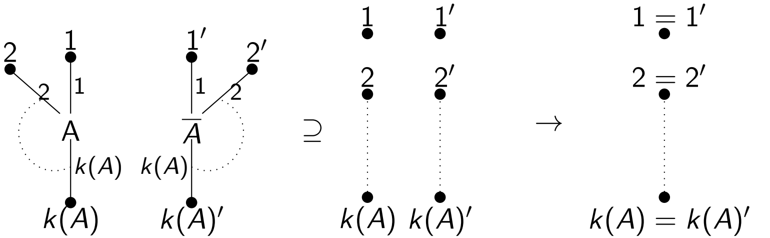
where $A \in \Sigma$.

The class of all hypergraphs over Σ is denoted by $\mathcal{H}_\Sigma$.

Fusion

Let $F \subseteq \Sigma$ be a fusion alphabet
with a type $k(A) \in \mathbb{N}$ for each $A \in F$ and
with a disjoint complementary copy $\overline{F} \subseteq \Sigma$ where $k(A) = k(\overline{A})$.

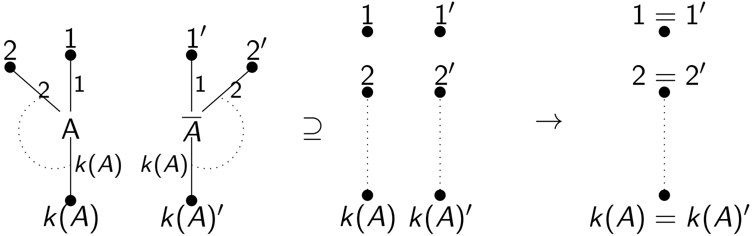
A fusion rule $fr(A)$ is defined as



Fusion

Let $F \subseteq \Sigma$ be a fusion alphabet
with a type $k(A) \in \mathbb{N}$ for each $A \in F$ and
with a disjoint complementary copy $\overline{F} \subseteq \Sigma$ where $k(A) = k(\overline{A})$.

A fusion rule $fr(A)$ is defined as



$$fr(A) = (A^\bullet + \overline{A}^\bullet \xleftarrow{in+\overline{in}} K + K \xrightarrow{\langle 1_K, 1_K \rangle} K)$$

where $K = [k(A)]$, in is the inclusion of K into $A^\bullet$ and $\overline{in}$ is the inclusion of K into $\overline{A}^\bullet$.

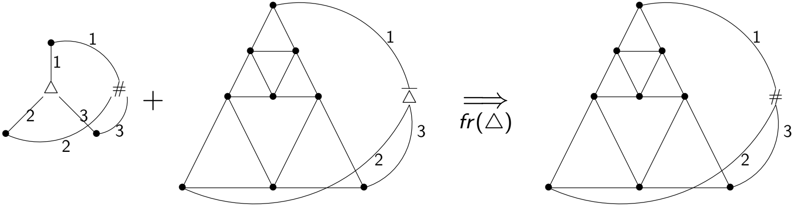
Fusion rule application

- The application of $fr(A)$ is defined by a double pushout

$$
\begin{array}{ccccc}
A^\bullet + \overline{A}^\bullet & \xleftarrow{in + \overline{in}} & K + K & \xrightarrow{\langle 1_K, 1_K \rangle} & K \\
\downarrow f & & \downarrow c & & \downarrow f' \\
H & \xleftarrow{e} & C & \xrightarrow{e'} & H'
\end{array}
$$

  where matching morphism $f : A^\bullet + \overline{A}^\bullet \to H$ satisfies the gluing condition always.
- C is unique up to isomorphism because $in + \overline{in}$ is injective.
- It is denoted by $H \underset{fr(A)}{\Longrightarrow} H'$.

Example



Fusion grammar $FG = (Z, F, M, T)$

- $Z \in \mathcal{H}_{F \cup \overline{F} \cup M \cup T}$ start hypergraph
  $F, M, T \subseteq \Sigma$, fusion, marker, terminal alphabet
  $M \cap (F \cup \overline{F}) = \emptyset$, $T \cap (F \cup \overline{F}) = \emptyset = T \cap M$

Fusion grammar $FG = (Z, F, M, T)$

- $Z \in \mathcal{H}_{F \cup \overline{F} \cup M \cup T}$ start hypergraph
  $F, M, T \subseteq \Sigma$, fusion, marker, terminal alphabet
  $M \cap (F \cup \overline{F}) = \emptyset$, $T \cap (F \cup \overline{F}) = \emptyset = T \cap M$
- A *direct derivation* is either

$$H \underset{fr(A)}{\Longrightarrow} H' \quad \text{for some } A \in F \text{ or}$$

$$H \underset{m}{\Longrightarrow} m \cdot H = \sum_{C \in \mathcal{C}(H)} m(C) \cdot C \quad \text{for some multiplicity } m \colon \mathcal{C}(H) \to \mathbb{N}$$

  where $\mathcal{C}(H)$ is the set of all connected components of H.
- A derivation is defined by the reflexive and transitive closure.

Fusion grammar $FG = (Z, F, M, T)$

- $Z \in \mathcal{H}_{F \cup \overline{F} \cup M \cup T}$ start hypergraph
  $F, M, T \subseteq \Sigma$, fusion, marker, terminal alphabet
  $M \cap (F \cup \overline{F}) = \emptyset$, $T \cap (F \cup \overline{F}) = \emptyset = T \cap M$
- A *direct derivation* is either

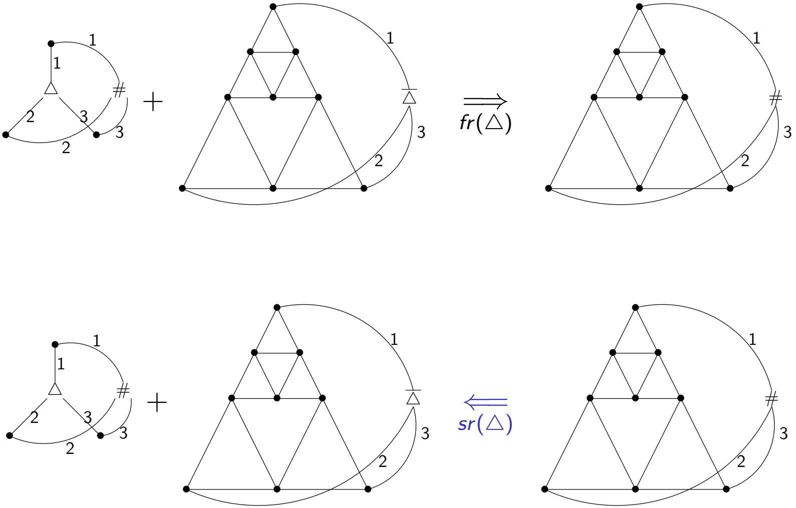
$$H \underset{fr(A)}{\Longrightarrow} H' \qquad \text{for some } A \in F \text{ or}$$

$$H \underset{m}{\Longrightarrow} m \cdot H = \sum_{C \in \mathcal{C}(H)} m(C) \cdot C \quad \text{for some multiplicity } m \colon \mathcal{C}(H) \to \mathbb{N}$$

  where $\mathcal{C}(H)$ is the set of all connected components of H.
- A derivation is defined by the reflexive and transitive closure.
- The generated language

$$L(FG) = \{rem_M(Y) \mid Z \overset{*}{\Longrightarrow} H, Y \in \mathcal{C}(H) \cap (\mathcal{H}_{T \cup M} - \mathcal{H}_T)\}.$$

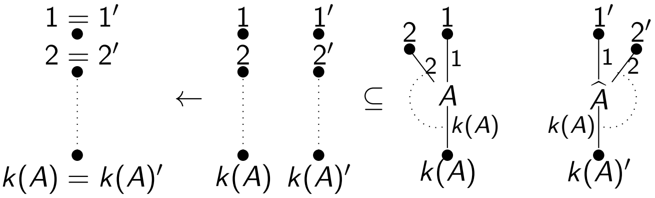
Example



Splicing

Let $F \subseteq \Sigma$ be a fusion alphabet
with a disjoint complementary copy $\widehat{F} \subseteq \Sigma$

A splicing rule $sr(A)$ is



$$K \xleftarrow{\langle 1_K, 1_K \rangle} K + K \xrightarrow{in + \widehat{in}} A^{\bullet} + \widehat{A}^{\bullet}$$

where $K = [k(A)]$, in is the inclusion of K into $A^{\bullet}$ and $\widehat{in}$ is the respective inclusion of K into $\widehat{A}^{\bullet}$.

Splicing rule application

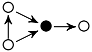
- The application of $sr(A)$ is defined by a double pushout

$$\begin{array}{ccccc} K & \xleftarrow{\langle 1_K, 1_K \rangle} & K + K & \xrightarrow{in + \widehat{in}} & A^\bullet + \widehat{A}^\bullet \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle c} & & \downarrow{\scriptstyle f'} \\ H & \xleftarrow{e} & C & \xrightarrow{e'} & H' \end{array}$$

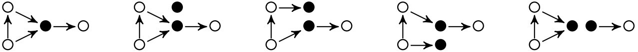
- C is not uniquely determined, because lefthand-side morphism is not injective.
- It is denoted by $H \underset{sr(A)}{\Longrightarrow} H'$.

Example

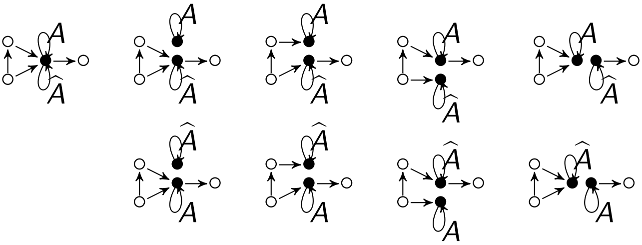
Consider $sr(A) = (\bullet \leftarrow \bullet \quad \bullet \rightarrow A \circlearrowright \bullet \quad \bullet \circlearrowleft \widehat{A})$.

Apply $sr(A)$ to  .

The pushout complement objects are:

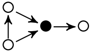


The derived graphs are:

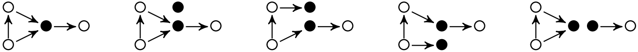


Example

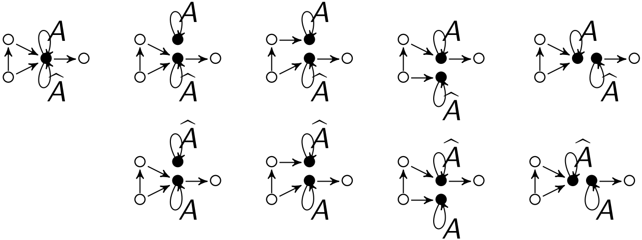
Consider $sr(A) = (\bullet \leftarrow \bullet \quad \bullet \rightarrow A\circlearrowleft\bullet \quad \bullet\circlearrowleft\widehat{A})$.

Apply $sr(A)$ to  .

The pushout complement objects are:



The derived graphs are:



This waste nondeterminism is often undesireable.

To cut it down, one may use context conditions.

Splicing rule with fixed disjoint context

$srfdc(A, a)$ consists of a splicing rule $sr(A)$ and a morphism $a\colon K \to X$ for some context X.

$$srfdc(A, a) = (X \xleftarrow{a} K \xleftarrow{\langle 1_K, 1_K \rangle} K + K \xrightarrow{in + \widehat{in}} A^\bullet + \widehat{A}^\bullet)$$

Splicing rule with fixed disjoint context

$srfdc(A, a)$ consists of a splicing rule $sr(A)$ and a morphism $a\colon K \to X$ for some context X.

$$srfdc(A, a) = (X \overset{a}{\leftarrow} K \xleftarrow{\langle 1_K, 1_K \rangle} K + K \xrightarrow{in + \widehat{in}} A^{\bullet} + \widehat{A}^{\bullet})$$

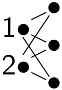
It is applicable to H if the pushout complement can be chosen in the following way

$$\begin{array}{ccc} K & \xleftarrow{\langle 1_K, 1_K \rangle} & K + K \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle y + a} \\ H & \xleftarrow{\langle m, b \rangle} & Y + X \end{array}$$

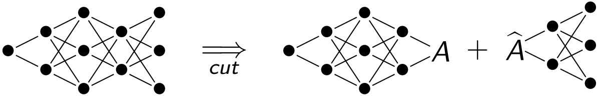
where $m\colon Y \to H$ is injective.

- The complement consists of two disjoint parts one of which is X.
- It is unique if it exists.
- X gets an $\widehat{A}$-hyperedge and Y get an A-hyperedge.

Example

$cut = ($ 

$\supseteq [2] \leftarrow [2] + [2] \subseteq A^{\bullet} + \widehat{A}^{\bullet})$



Splicing/fusion grammar $SFG = (Z, F, M, T, SR)$

extends fusion grammars by splicing rules with fixed disjoint context.

- $F, M, T \subseteq \Sigma$, fusion, marker, terminal alphabet
  $M \cap (F \cup \overline{F}) = \emptyset$, $T \cap (F \cup \overline{F}) = \emptyset = T \cap M$
  $M \cap (F \cup \widehat{F}) = \emptyset$, $T \cap (F \cup \widehat{F}) = \emptyset$

Splicing/fusion grammar $SFG = (Z, F, M, T, SR)$

extends fusion grammars by splicing rules with fixed disjoint context.

- $F, M, T \subseteq \Sigma$, fusion, marker, terminal alphabet
  $M \cap (F \cup \overline{F}) = \emptyset,\ T \cap (F \cup \overline{F}) = \emptyset = T \cap M$
  $M \cap (F \cup \widehat{F}) = \emptyset,\ T \cap (F \cup \widehat{F}) = \emptyset$
- A direct derivation is either

$H \underset{fr(A)}{\Longrightarrow} H'$ for some $A \in F$ or

$H \underset{m}{\Longrightarrow} m \cdot H = \sum_{C \in \mathcal{C}(H)} m(C) \cdot C$ for some $m \colon \mathcal{C}(H) \to \mathbb{N}$ or

$H \underset{srfdc(A,a)}{\Longrightarrow} H'$ for some $A \in F$ and $a \colon K \to X$.

Splicing/fusion grammar $SFG = (Z, F, M, T, SR)$

extends fusion grammars by splicing rules with fixed disjoint context.

- $F, M, T \subseteq \Sigma$, fusion, marker, terminal alphabet
  $M \cap (F \cup \overline{F}) = \emptyset$, $T \cap (F \cup \overline{F}) = \emptyset = T \cap M$
  $M \cap (F \cup \widehat{F}) = \emptyset$, $T \cap (F \cup \widehat{F}) = \emptyset$
- A direct derivation is either

$$H \underset{fr(A)}{\Longrightarrow} H' \qquad \text{for some } A \in F \text{ or}$$

$$H \underset{m}{\Longrightarrow} m \cdot H = \sum_{C \in \mathcal{C}(H)} m(C) \cdot C \qquad \text{for some } m\colon \mathcal{C}(H) \to \mathbb{N} \text{ or}$$

$$H \underset{srfdc(A,a)}{\Longrightarrow} H' \qquad \text{for some } A \in F \text{ and } a\colon K \to X.$$

- The generated language

$$L(SFG) = \{rem_M(Y) \mid Z \overset{*}{\Longrightarrow} H, Y \in \mathcal{C}(H) \cap (\mathcal{H}_{T \cup M} - \mathcal{H}_T)\}$$

Generative power

ICGT17:

- Fusion grammars can simulate hyperedge replacement grammars.
- Their membership problem is deciable.

How powerful are splicing/fusion grammars?

Transformation of Chomsky grammars into splicing/fusion grammars

Let (N, T, P, S) be a Chomsky grammar.
Let $p = (u_1 \dots u_k, v_1 \dots v_l) \in P$.
Let $x_1 \dots x_n = x_1 \dots x_{i-1} u_1 \dots u_k x_{i+k} \dots x_n$
Then

$$x_1 \dots x_{i-1} u_1 \dots u_k x_{i+k} \dots x_n \underset{p}{\Longrightarrow} x_1 \dots x_{i-1} v_1 \dots v_l x_{i+k} \dots x_n$$

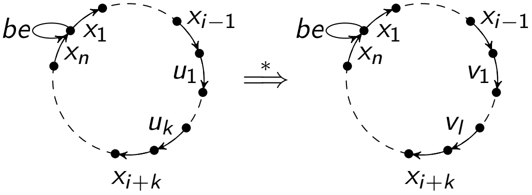
Transformation of Chomsky grammars into splicing/fusion grammars

Let (N, T, P, S) be a Chomsky grammar.
Let $p = (u_1 \dots u_k, v_1 \dots v_l) \in P$.
Let $x_1 \dots x_n = x_1 \dots x_{i-1} u_1 \dots u_k x_{i+k} \dots x_n$
Then

$$x_1 \dots x_{i-1} u_1 \dots u_k x_{i+k} \dots x_n \underset{p}{\Longrightarrow} x_1 \dots x_{i-1} v_1 \dots v_l x_{i+k} \dots x_n$$

Adapting a transformation of Chomsky grammars into iterated splicing systems (cf. [Păun,Rozenberg,Salomaa:1998]).

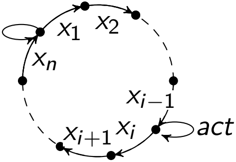
$cyc(x_1 \dots x_n) =$



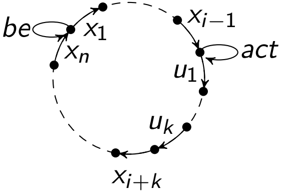
Simulating a direct derivation

$$p = (u_1 \dots u_k, v_1 \dots v_l) \in P$$

$$cyc(x_1 \dots x_n)^{act,i} =$$

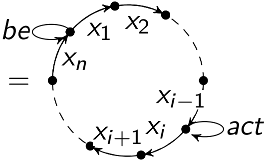


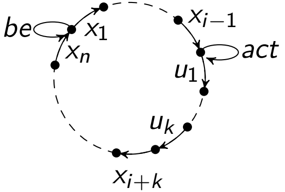
$$=$$



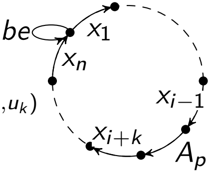
Simulating a direct derivation

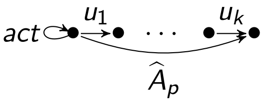
$p = (u_1 \dots u_k, v_1 \dots v_l) \in P$

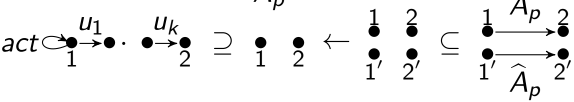
$cyc(x_1 \dots x_n)^{act,i} =$ 

$=$ 

1. splicing, i.e.,

$cyc(x_1 \dots x_n)^{act,i} \underset{sr(A_p, u_1 \dots, u_k)}{\Longrightarrow}$ 

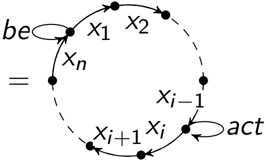
$+$ 

$sr(A_p, u_1 \dots u_k) =$ 

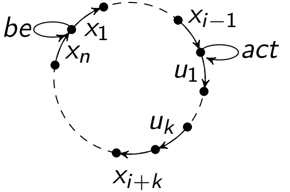
Simulating a direct derivation

$p = (u_1 \dots u_k, v_1 \dots v_l) \in P$

$cyc(x_1 \dots x_n)^{act,i} =$

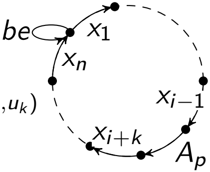


$=$

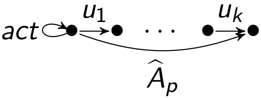


1. splicing, i.e.,

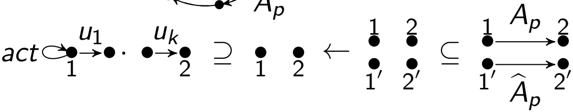
$cyc(x_1 \dots x_n)^{act,i} \underset{sr(A_p, u_1 \dots, u_k)}{\Longrightarrow}$



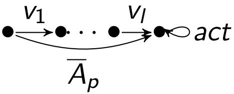
$+$



$sr(A_p, u_1 \dots u_k) =$

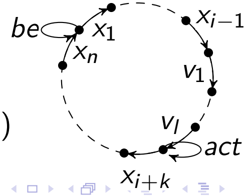


2. fusion, i.e.,



$\underset{fr(A_p)}{\Longrightarrow}$

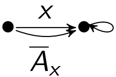
$+\ cyc(x_1 \dots x_{i-1} A_p x_{i+k} \dots x_n)$

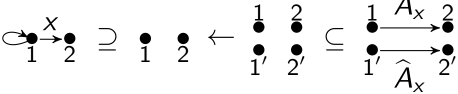


Moving the *act*-loop

$$Q = P \cup \{(x,x) \mid x \in N \cup T\}$$

i.e., for each $x \in N \cup T$:

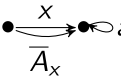
- $c_x =$ 

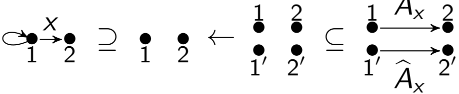
act, and
- $sr(A_x, x) = act$ 

Moving the *act*-loop

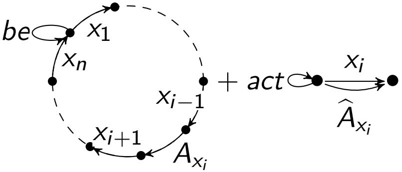
$$Q = P \cup \{(x,x) \mid x \in N \cup T\}$$

i.e., for each $x \in N \cup T$:

- $c_x =$ 

, and
- $sr(A_x, x) =$ 

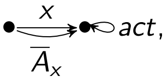
1. splicing, i.e.,

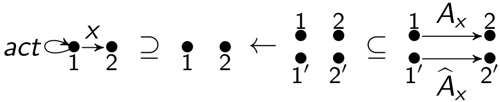
$cyc(x_1 \ldots x_n)^{act,i} \underset{sr(A_{x_i},x_i)}{\Longrightarrow}$ 

Moving the *act*-loop

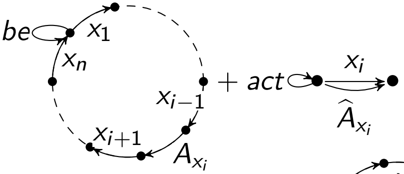
$$Q = P \cup \{(x, x) \mid x \in N \cup T\}$$

i.e., for each $x \in N \cup T$:

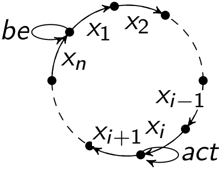
- $c_x =$ 

, and
- $sr(A_x, x) =$ 

1. splicing, i.e.,

   $cyc(x_1 \ldots x_n)^{act,i} \underset{sr(A_{x_i},x_i)}{\Longrightarrow}$ 

2. fusion, i.e.,

   $cyc(x_1 \ldots x_{i-1} A_{x_i} x_{i+k} \ldots x_n) + c_{x_i} \underset{fr(A_{x_i})}{\Longrightarrow}$ 

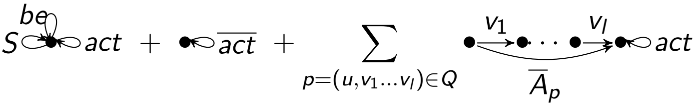
Transformation Chomsky grammars into splicing/fusion grammars

Given Chomsky grammar $CG = (N, T, P, S)$.
Let $Q = P \cup \{(x, x) \mid x \in N \cup T\}$.

$$SFG(CG) = (Z(CG), N(CG), \emptyset, T(CG), SR(CG))$$

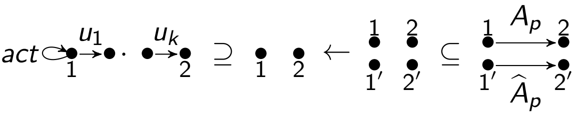
$$N(CG) = \{A_y \mid y \in N \cup T \cup P\} \cup \{act\}$$

$$T(CG) = T \cup \{be\}$$

$Z(CG) =$ 

$$SR(CG) = \{sr(A_p, u_1 \ldots, u_k) \mid p = (u_1 \ldots u_k, v) \in Q\}$$

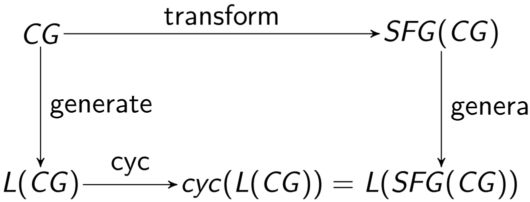
where

$sr(A_p, u_1 \ldots, u_k) =$ 

Theorem I

Let $CG = (N, T, P, S)$ be a Chomsky grammar
and $SFG(CG)$ the corresponding splicing/fusion grammar.
Then

$$cyc(L(CG)) = L(SFG(CG)).$$



Hypergraph grammar $HGG = (N, T, P, S)$

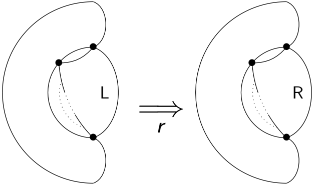
- $N, T \subseteq \Sigma$, $T \cap N = \emptyset$, $S \in N$,
  $A \in N$ has a *type* $k(A) \in \mathbb{N}$
  P is a finite set of rules of the form $r = (L \overset{a}{\leftarrow} K \overset{b}{\rightarrow} R)$
  where $L, K, R \in \mathcal{H}_\Sigma$, K discrete and a injective.
- A rule application $H \underset{r}{\Longrightarrow} H'$ (direct derivation) is defined by a double pushout

$$\begin{array}{ccccc} L & \xleftarrow{a} & K & \xrightarrow{b} & R \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle d} & & \downarrow{\scriptstyle h} \\ H & \xleftarrow{m} & I & \xrightarrow{m'} & H' \end{array}$$

  where matching morphism $g \colon L \to H$ is subject to the gluing condition.
- $L(HGG) = \{X \in \mathcal{H}_T \mid S^\bullet \overset{*}{\underset{P}{\Longrightarrow}} X\}$.

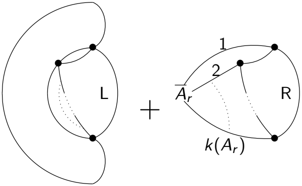
Simulating a direct derivation

$HGG : r = (L \supseteq K \xrightarrow{b} R)$



$\iff$

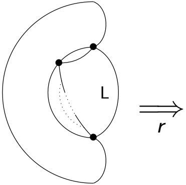
$SFG :$



R together with an additional $\overline{A}_r$-hyperedge attached to $b(1) \cdots b(k)$.

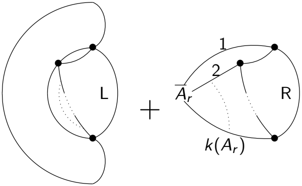
Simulating a direct derivation

$HGG : r = (L \supseteq K \xrightarrow{b} R)$



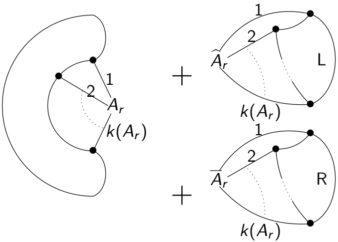
$\Longleftrightarrow$

$SFG :$



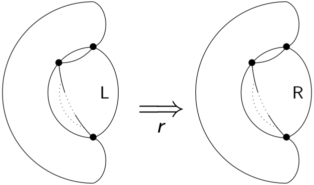
R together with an additional $\overline{A}_r$-hyperedge attached to $b(1) \cdots b(k)$.

$\underset{\substack{srfdc(A_r,a) \\ a:k(A_r)\to L}}{\Longrightarrow}$



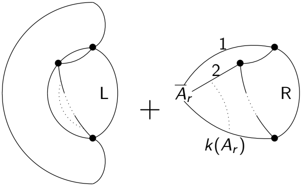
Simulating a direct derivation

$HGG : r = (L \supseteq K \xrightarrow{b} R)$



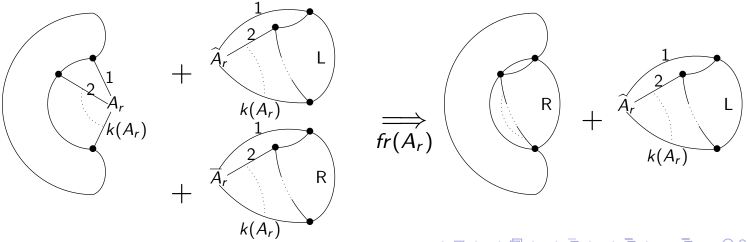
$\Longleftrightarrow$

$SFG :$



R together with an additional $\overline{A}_r$-hyperedge attached to $b(1)\cdots b(k)$.

$\Longrightarrow_{srfdc(A_r,a)}$
$a{:}k(A_r) \to L$



Transformation of hypergraph grammars into splicing/fusion grammars

Connective hypergraph grammar:
Each connected component contains some gluing node and for each $(L \supseteq K \xrightarrow{b} R)$: if $i, j \in V_K$ are connected in L, then $b(i), b(j)$ are connected in R.

Lemma

Connectedness is preserved.

Transformation of hypergraph grammars into splicing/fusion grammars

Connective hypergraph grammar:
Each connected component contains some gluing node and for each $(L \supseteq K \xrightarrow{b} R)$: if $i, j \in V_K$ are connected in L, then $b(i), b(j)$ are connected in R.

Lemma
Connectedness is preserved.

Theorem (2)
Let $HGG = (N, T, P, S)$ be a connective hypergraph grammar and $SFG(HGG)$ its corresponding splicing/fusion grammar. Then

$$L(HGG) = L(SFG(HGG)).$$

In other words, splicing/fusion grammars can simulate hypergraph grammars, but connectedness must be preserved.

Transformation of hypergraph grammars into splicing/fusion grammars

$$SFG(HGG) = (Z(HGG), F(HGG), M(HGG), T(HGG), SR(HGG))$$

- $F(HGG) = \{A_r \mid r = (L \supseteq [k] \to R) \in P, k(A_r) = k\}$, $M(HGG) = \{\mu\}$ with $k(\mu) = 0$, $T(HGG) = T$
- $F(HGG), \overline{F}(HGG), \widehat{F}(HGG), M(HGG), T$ are pairwise disjoint.
- $Z(HGG) = S^\bullet_\mu + \sum\limits_{r \in P} C(r)$

  where $C(r)$ for $r = (L \supseteq [k] \xrightarrow{b} R)$ is R together with an additional $\overline{A}_r$-hyperedge attached to $b(1) \cdots b(k)$.
- $SR(HGG) = \{srfdc(A_r, a) \mid r = (L \overset{a}{\supseteq} [k] \to R) \in P)$ where

$$srfdc(A_r, a) = (L \overset{a}{\supseteq} [k(A_r)] \xleftarrow{\langle 1_{[k(A_r)]}, 1_{[k(A_r)]} \rangle} [k(A_r)] + [k(A_r)] \xrightarrow{in + \widehat{in}} A^\bullet_r + \widehat{A}^\bullet_r$$

Pushout property

Lemma

Let C be a category with finite coproduct (denoted by $+$) and pushouts. Consider the following diagrams:

$$
\begin{array}{ccc}
L & \xleftarrow{\quad a \quad} & K \\
{\scriptstyle g}\downarrow & (1) & \downarrow{\scriptstyle d} \\
H & \xleftarrow[\quad m \quad]{} & I
\end{array}
\qquad
\begin{array}{ccc}
K & \xleftarrow{\quad \langle 1_K, 1_K \rangle \quad} & K + K \\
{\scriptstyle g \circ a}\downarrow & (2) & \downarrow{\scriptstyle d + a} \\
H & \xleftarrow[\quad \langle m, g \rangle \quad]{} & I + L
\end{array}
$$

Then the left diagram is a pushout if and only if the right diagram is a pushout.

Conclusion

- We have extended fusion grammars by splicing rules with fixed disjoint context.
- Transformation of Chomsky grammars into splicing/fusion grammars.
- Transformation of connective hypergraph grammars into splicing/fusion grammars.

Further work:

- Are there other meaningful conditions for splicing besides fixed disjoint context?
- How are DNA computing models related to splicing/fusion grammars?
- How can we overcome the limitation of generating connected components?
- Are there interesting examples where one can use all connected component resulting from splicing?

Thank you! Questions?