

# Applications of Weighted Tree Automata and Tree Transducers in Natural Language Processing

Andreas Maletti

Institute of Computer Science  
Universität Leipzig, Germany

Krippen — March 28, 2019

# Final Round of Jeopardy!

IBM Watson

\$36,681

Brad Rutter

\$5,400

Ken Jennings

\$2,400

Final Category

U.S. cities

Answer: Its largest airport was named for a World War II hero;  
its second largest, for a World War II battle.

# Final Round of Jeopardy!

IBM Watson

\$36,681

Brad Rutter

\$5,400

Ken Jennings

\$2,400

Final Category

U.S. cities

Answer: Its largest airport was named for a World War II hero;  
its second largest, for a World War II battle.

Watson (bet \$947)

What is Toronto?????  
(confidence  $\approx$  30%)

Brad (bet \$5,000)

What is Chicago?

Ken (bet \$2,400)

What is Chicago?

# Final Round of Jeopardy!



## Wikipedia page of Toronto Pearson International Airport

**Lester B. Pearson International Airport** [...] is the primary international airport serving Toronto, its metropolitan area, and surrounding region known as the Golden Horseshoe in the province of Ontario, Canada.

It is the largest and busiest airport in Canada, the second-busiest international air passenger gateway in the Americas, and the 31st-busiest airport in the world [...].

The airport is named in honour of Lester B. Pearson, Nobel Peace Prize laureate and 14th Prime Minister of Canada.

## Wikipedia page of Toronto Pearson International Airport

**Lester B. Pearson International Airport** [...] is the primary international airport serving Toronto, its metropolitan area, and surrounding region known as the Golden Horseshoe in the province of Ontario, Canada.

It is the largest and busiest airport in Canada, the second-busiest international air passenger gateway in the Americas, and the 31st-busiest airport in the world [...].

The airport is named in honour of Lester B. Pearson, Nobel Peace Prize laureate and 14th Prime Minister of Canada.

**Mention Detection** is the task to identify all instances of the same entity

# Mention Detection

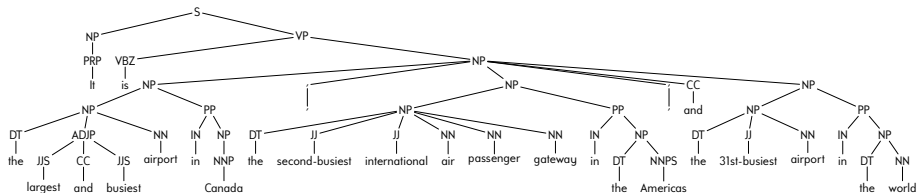
## Mention Detection

- 1 Identify all mentions (= noun phrases)
- 2 Identify which mentions reference the same entity

# Mention Detection

## Mention Detection

- 1 Identify all mentions (= noun phrases)
- 2 Identify which mentions reference the same entity

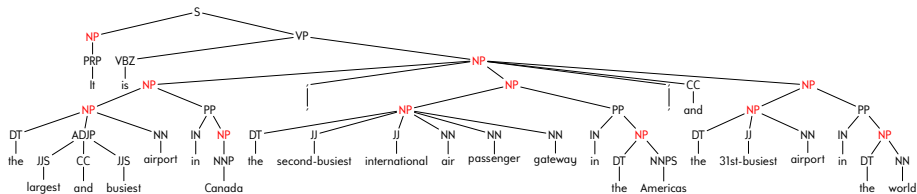




# Mention Detection

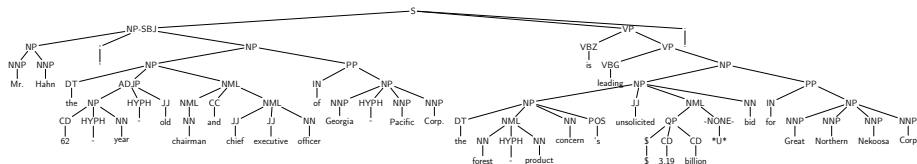
## Mention Detection

- 1 Identify all mentions (= noun phrases)
- 2 Identify which mentions reference the same entity



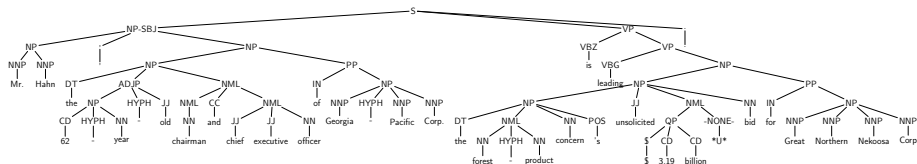
## Parsing

- determining the syntactic structure of a sentence
- subject to a given theory of syntax (encoded in the training data)
  - ▶ constituent syntax
  - ▶ dependency syntax
  - ▶ ...



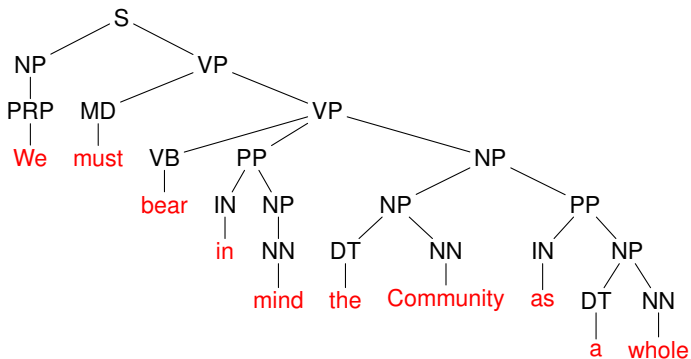
## Parsing

- determining the syntactic structure of a sentence
- subject to a given theory of syntax (encoded in the training data)
  - ▶ constituent syntax
  - ▶ dependency syntax
  - ▶ ...



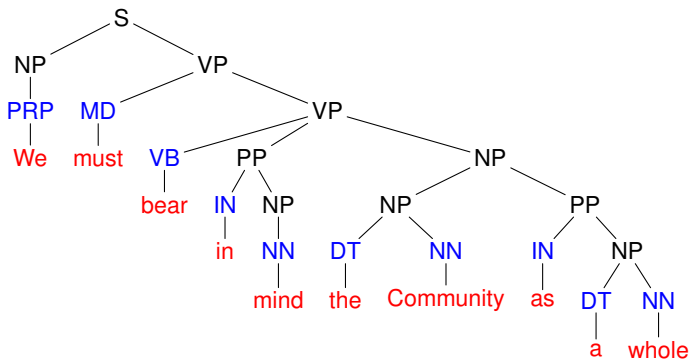
# Constituent Parsing

Example: We must bear in mind the Community as a whole



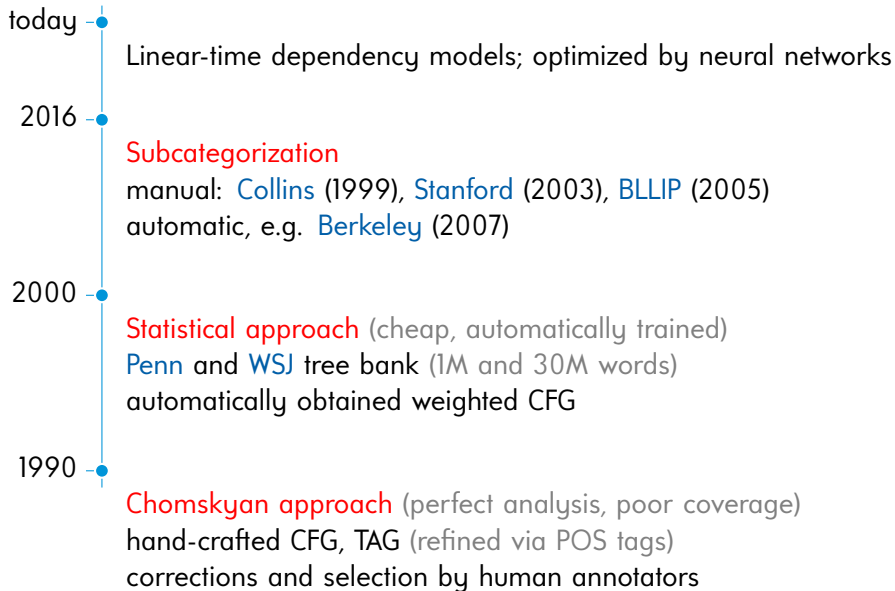
# Constituent Parsing

Example: We must bear in mind the Community as a whole



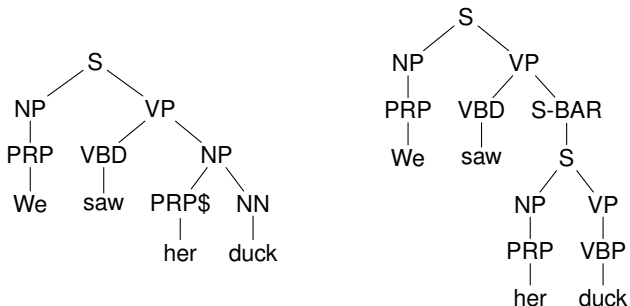
POS-tag: part-of-speech tag, “class” of a word

# Constituent Parsing



# Constituent Parsing

All models use weights for disambiguation:



## Definition (Commutative semiring)

Algebraic structure  $(S, +, \cdot, 0, 1)$  is **commutative semiring** if

- $(S, +, 0)$  and  $(S, \cdot, 1)$  are commutative monoids
- $\cdot$  distributes over finite sums  
( $s \cdot 0 = 0$  for all  $s \in S$ )



## Definition (Commutative semiring)

Algebraic structure  $(S, +, \cdot, 0, 1)$  is **commutative semiring** if

- $(S, +, 0)$  and  $(S, \cdot, 1)$  are commutative monoids
- $\cdot$  distributes over finite sums  
( $s \cdot 0 = 0$  for all  $s \in S$ )

## Examples

- semiring  $(\mathbb{N}, +, \cdot, 0, 1)$  of nonnegative integers
- semiring  $(\mathbb{Q}, +, \cdot, 0, 1)$  of rational numbers
- Viterbi semiring  $([0, 1], \max, \cdot, 0, 1)$  of probabilities

# Weighted Tree Automaton

Fix a commutative semiring  $(S, +, \cdot, 0, 1)$

Definition (Weighted tree automaton [Berstel, Reutenauer 1982])

Tuple  $(Q, \Sigma, I, R)$  is **weighted tree automaton** if

- finite set  $Q$  of **states**
- finite set  $\Sigma$  of **terminals**
- **initial states**  $I \subseteq Q$
- finite set  $R$  of **weighted rules** of the form  $q \xrightarrow{s} \sigma(q_1, \dots, q_k)$   
 $(s \in S, \sigma \in \Sigma, k \geq 0, q, q_1, \dots, q_k \in Q)$

# Weighted Tree Automaton

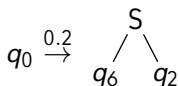
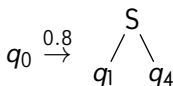
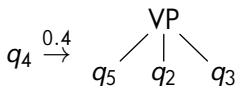
Fix a commutative semiring  $(S, +, \cdot, 0, 1)$

Definition (Weighted tree automaton [Berstel, Reutenauer 1982])

Tuple  $(Q, \Sigma, I, R)$  is **weighted tree automaton** if

- finite set  $Q$  of **states**
- finite set  $\Sigma$  of **terminals**
- **initial states**  $I \subseteq Q$
- finite set  $R$  of **weighted rules** of the form  $q \xrightarrow{s} \sigma(q_1, \dots, q_k)$   
( $s \in S, \sigma \in \Sigma, k \geq 0, q, q_1, \dots, q_k \in Q$ )

Example rules

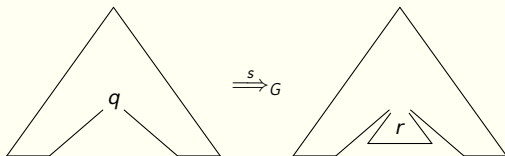


# Weighted Tree Automaton

## Definition (Derivation semantics and recognized tree language)

Let  $(Q, \Sigma, l, R)$  tree automaton

- for the state  $q \in Q$  labeling the leftmost state-labeled leaf position and every rule  $q \xrightarrow{s} r \in R$

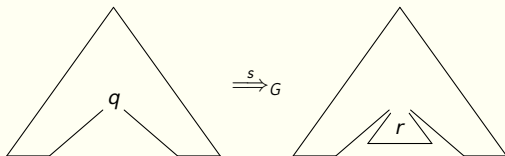


# Weighted Tree Automaton

## Definition (Derivation semantics and recognized tree language)

Let  $(Q, \Sigma, l, R)$  tree automaton

- for the state  $q \in Q$  labeling the leftmost state-labeled leaf position and every rule  $q \xrightarrow{s} r \in R$



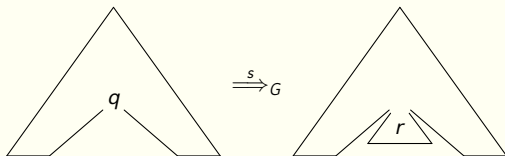
- derivation  $D: t \xrightarrow{s_1} \dots \xrightarrow{s_n} u$  from  $t$  to  $u$  with weight  $\prod_{i=1}^n s_i$

# Weighted Tree Automaton

## Definition (Derivation semantics and recognized tree language)

Let  $(Q, \Sigma, l, R)$  tree automaton

- for the state  $q \in Q$  labeling the leftmost state-labeled leaf position and every rule  $q \xrightarrow{s} r \in R$



- derivation  $D: t \xrightarrow{s_1} \dots \xrightarrow{s_n} u$  from  $t$  to  $u$  with weight  $\prod_{i=1}^n s_i$
- recognized weighted tree language  $L$

$$L(t) = \sum_{q \in I} \left( \sum_{\substack{D: \text{derivation} \\ \text{from } q \text{ to } t}} \text{weight}(D) \right)$$

# Constituent Parsing

grammar	$F_1$ -score	
	$ w  \leq 40$	full
CFG		62.7
TSG [Post, Gildea, 2009]	82.6	
TSG [Cohn et al., 2010]	85.4	84.7
CFG <sub>sub</sub> [Collins, 1999]	88.6	88.2
CFG <sub>sub</sub> [Petrov, Klein, 2007]	90.6	90.1
CFG <sub>sub</sub> [Petrov, 2010]		91.8
TSG <sub>sub</sub> [Shindo et al., 2012]	92.9	92.4

# Local Tree Languages

CFG (in tree-generation mode) = Local tree grammar

Definition (Local tree grammar [Gécseg, Steinby 1984])

**Local tree grammar**  $(\Sigma, I, P)$  is finite set  $P$  of weighted branchings  $\sigma \xrightarrow{s} w$  (with  $\sigma \in \Sigma$ ,  $s \in S$ , and  $w \in \Sigma^k$ ) together with a set  $I \subseteq \Sigma$  of root labels



# Local Tree Languages

## Example (with root label S)

$S \xrightarrow{0.3} NP_1 VP_2$

$NP_2 \xrightarrow{0.4} NP_2 PP$

$MD \xrightarrow{0.1} \text{must}$

$VP_2 \xrightarrow{0.6} MD VP_3$

$VP_3 \xrightarrow{0.2} VB PP NP_2$

...

# Local Tree Languages

## Example (with root label S)

$S \xrightarrow{0.3} NP_1 VP_2$

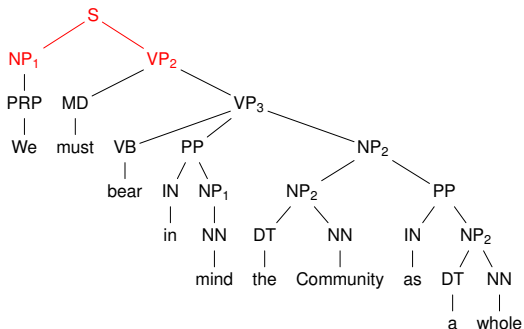
$NP_2 \xrightarrow{0.4} NP_2 PP$

$MD \xrightarrow{0.1} \text{must}$

$VP_2 \xrightarrow{0.6} MD VP_3$

$VP_3 \xrightarrow{0.2} VB PP NP_2$

...



# Local Tree Languages

## Example (with root label S)

$S \xrightarrow{0.3} NP_1 VP_2$

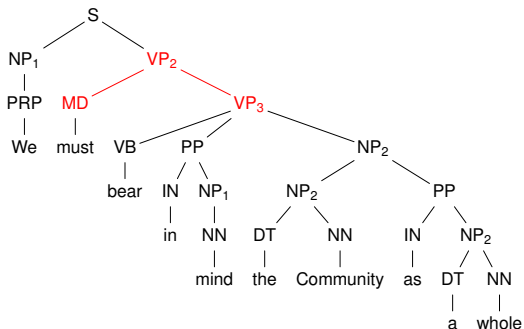
$NP_2 \xrightarrow{0.4} NP_2 PP$

$MD \xrightarrow{0.1} \text{must}$

$VP_2 \xrightarrow{0.6} MD VP_3$

$VP_3 \xrightarrow{0.2} VB PP NP_2$

...



# Local Tree Languages

## Example (with root label S)

$S \xrightarrow{0.3} NP_1 VP_2$

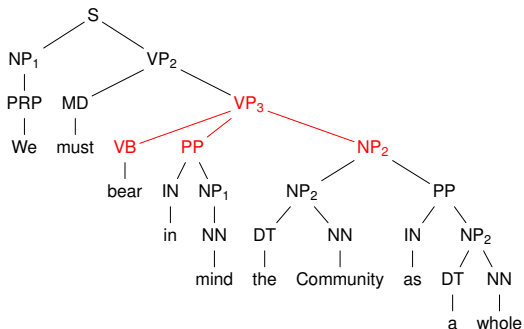
$VP_2 \xrightarrow{0.6} MD VP_3$

$NP_2 \xrightarrow{0.4} NP_2 PP$

$VP_3 \xrightarrow{0.2} VB PP NP_2$

$MD \xrightarrow{0.1} \text{must}$

...



# Local Tree Languages

## Example (with root label S)

$S \xrightarrow{0.3} NP_1 VP_2$

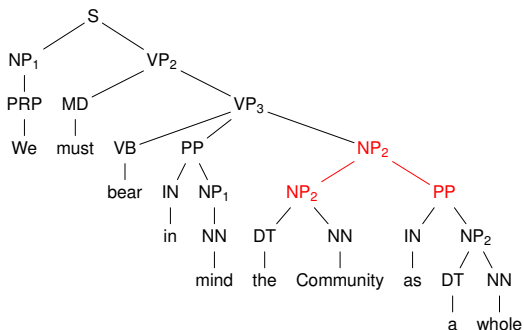
$NP_2 \xrightarrow{0.4} NP_2 PP$

$MD \xrightarrow{0.1} \text{must}$

$VP_2 \xrightarrow{0.6} MD VP_3$

$VP_3 \xrightarrow{0.2} VB PP NP_2$

...



# Local Tree Languages

## Example (with root label S)

$S \xrightarrow{0.3} NP_1 VP_2$

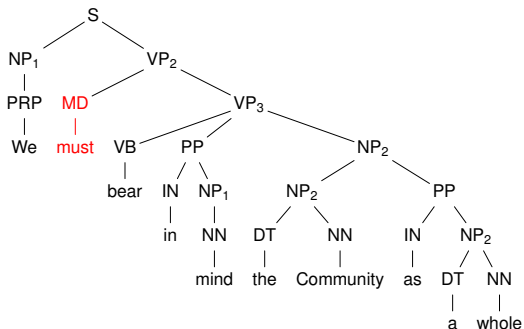
$VP_2 \xrightarrow{0.6} MD VP_3$

$NP_2 \xrightarrow{0.4} NP_2 PP$

$VP_3 \xrightarrow{0.2} VB PP NP_2$

$MD \xrightarrow{0.1} \text{must}$

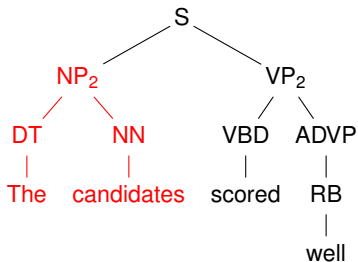
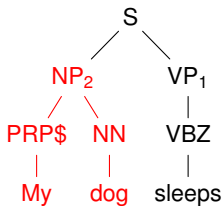
...



# Local Tree Languages

not closed under union

- these singletons are local

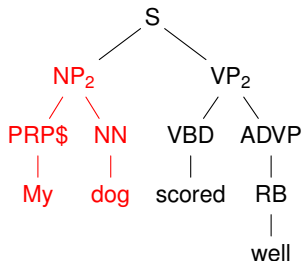
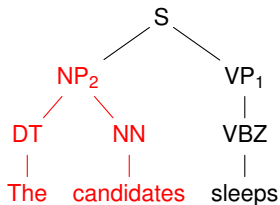


- but their union cannot be local

# Local Tree Languages

not closed under union

- these singletons are local



- but their union cannot be local  
(as we also generate these trees — overgeneralization)



# Local Tree Languages

## Question

Given a regular tree language  $L$ , determine whether  $L$  is local

# Local Tree Languages

## Question

Given a regular tree language  $L$ , determine whether  $L$  is local

## Answer

decidable

# Tree Substitution Languages

Definition (Tree substitution grammar [Joshi, Schabes 1997])

**Tree substitution grammar**  $(\Sigma, I, P)$  is finite set  $P$  of weighted fragments  $\text{root}(t) \xrightarrow{s} t$  (with  $s \in S$  and  $t \in \mathcal{T}_\Sigma$ ) together with a set of root labels  $I \subseteq \Sigma$

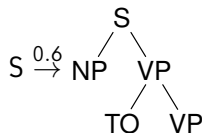
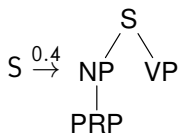
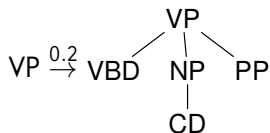
# Tree Substitution Languages

Definition (Tree substitution grammar [Joshi, Schabes 1997])

**Tree substitution grammar**  $(\Sigma, I, P)$  is finite set  $P$  of weighted fragments  $\text{root}(t) \xrightarrow{s} t$  (with  $s \in S$  and  $t \in T_\Sigma$ ) together with a set of root labels  $I \subseteq \Sigma$

Typical fragments

[Post 2011]



# Tree Substitution Languages

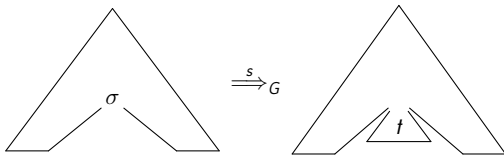
(Leftmost) Derivation step       $\xi \xrightarrow{s}_G \zeta$

- $\xi = c[\text{root}(t)]$  and  $\zeta = c[t]$  for some context  $c$  and  $\text{root}(t) \xrightarrow{s} t \in P$

# Tree Substitution Languages

(Leftmost) Derivation step       $\xi \xrightarrow{s}_G \zeta$

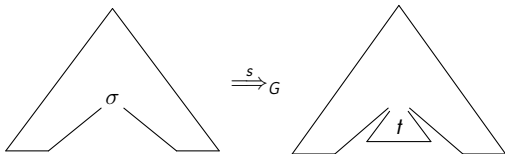
- $\xi = c[\text{root}(t)]$  and  $\zeta = c[t]$  for some context  $c$  and  $\text{root}(t) \xrightarrow{s} t \in P$
- for each fragment  $t \in P$  with root label  $\sigma$



# Tree Substitution Languages

(Leftmost) Derivation step       $\xi \xrightarrow{s}_G \zeta$

- $\xi = c[\text{root}(t)]$  and  $\zeta = c[t]$  for some context  $c$  and  $\text{root}(t) \xrightarrow{s} t \in P$
- for each fragment  $t \in P$  with root label  $\sigma$



- recognized weighted tree language  $L$

$$L(t) = \begin{cases} \sum_{D: \text{derivation from root}(t) \text{ to } t} \text{weight}(D) & \text{if } \text{root}(t) \in I \\ 0 & \text{otherwise} \end{cases}$$

# Tree Substitution Languages

## Fragments

$$S \xrightarrow{0.3} S(\text{NP}_1(\text{PRP}), \text{VP}_2)$$
$$\text{VP} \xrightarrow{0.1} \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2))$$
$$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$$
$$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$$

## Derivation

S



# Tree Substitution Languages

## Fragments

$S \xrightarrow{0.3} S(NP_1(PRPR), VP_2)$

$VP \xrightarrow{0.1} VP_2(MD, VP_3(VB, PP, NP_2))$

$PRP \xrightarrow{0.5} PRP(We)$

$MD \xrightarrow{0.2} MD(must)$

## Derivation

S

# Tree Substitution Languages

## Fragments

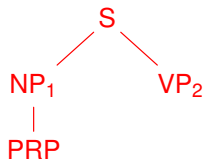
$S \xrightarrow{0.3} S(NP_1(PRP), VP_2)$

$VP \xrightarrow{0.1} VP_2(MD, VP_3(VB, PP, NP_2))$

$PRP \xrightarrow{0.5} PRP(We)$

$MD \xrightarrow{0.2} MD(must)$

## Derivation



# Tree Substitution Languages

## Fragments

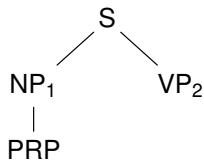
$S \xrightarrow{0.3} S(NP_1(\text{PRP}), VP_2)$

$VP \xrightarrow{0.1} VP_2(\text{MD}, VP_3(\text{VB}, \text{PP}, NP_2))$

$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$

$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$

## Derivation



# Tree Substitution Languages

## Fragments

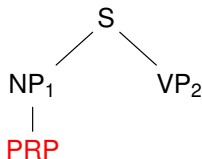
$S \xrightarrow{0.3} S(\text{NP}_1(\text{PRP}), \text{VP}_2)$

$\text{VP} \xrightarrow{0.1} \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2))$

$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$

$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$

## Derivation



# Tree Substitution Languages

## Fragments

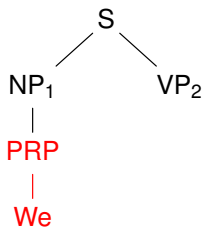
$S \xrightarrow{0.3} S(\text{NP}_1(\text{PRP}), \text{VP}_2)$

$\text{VP} \xrightarrow{0.1} \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2))$

$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$

$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$

## Derivation



# Tree Substitution Languages

## Fragments

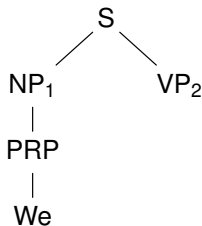
$S \xrightarrow{0.3} S(\text{NP}_1(\text{PRP}), \text{VP}_2)$

$\text{VP} \xrightarrow{0.1} \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2))$

$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$

$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$

## Derivation



# Tree Substitution Languages

## Fragments

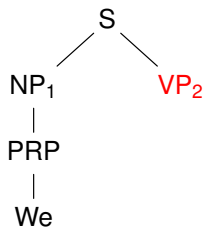
$S \xrightarrow{0.3} S(\text{NP}_1(\text{PRP}), \text{VP}_2)$

$\text{VP} \xrightarrow{0.1} \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2))$

$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$

$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$

## Derivation



# Tree Substitution Languages

## Fragments

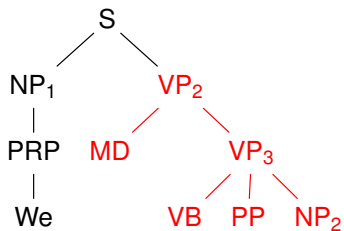
$S \xrightarrow{0.3} S(NP_1(PRP), VP_2)$

$VP \xrightarrow{0.1} VP_2(MD, VP_3(VB, PP, NP_2))$

$PRP \xrightarrow{0.5} PRP(We)$

$MD \xrightarrow{0.2} MD(must)$

## Derivation





# Tree Substitution Languages

## Fragments

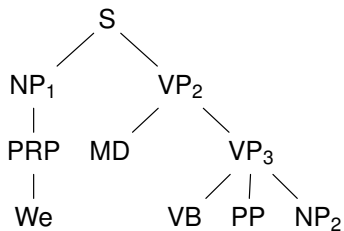
$S \xrightarrow{0.3} S(NP_1(\text{PRP}), VP_2)$

$VP \xrightarrow{0.1} VP_2(\text{MD}, VP_3(\text{VB}, \text{PP}, NP_2))$

$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$

$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$

## Derivation



# Tree Substitution Languages

## Fragments

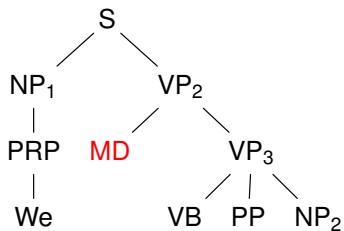
$S \xrightarrow{0.3} S(NP_1(\text{PRP}), VP_2)$

$VP \xrightarrow{0.1} VP_2(\text{MD}, VP_3(\text{VB}, \text{PP}, NP_2))$

$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$

$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$

## Derivation



# Tree Substitution Languages

## Fragments

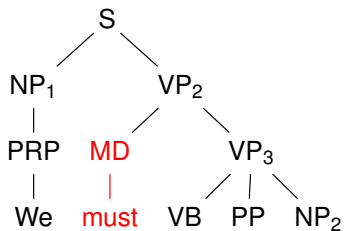
$S \xrightarrow{0.3} S(\text{NP}_1(\text{PRP}), \text{VP}_2)$

$\text{VP} \xrightarrow{0.1} \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2))$

$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$

$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$

## Derivation



# Tree Substitution Languages

## Fragments

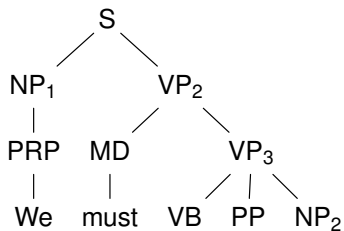
$S \xrightarrow{0.3} S(\text{NP}_1(\text{PRP}), \text{VP}_2)$

$\text{VP} \xrightarrow{0.1} \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2))$

$\text{PRP} \xrightarrow{0.5} \text{PRP}(\text{We})$

$\text{MD} \xrightarrow{0.2} \text{MD}(\text{must})$

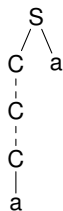
## Derivation



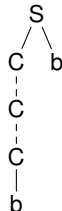
# Tree Substitution Languages

not closed under union

- these languages are tree substitution languages individually



$$L_1 = \{S(C^n(a), a) \mid n \in \mathbb{N}\}$$



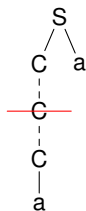
$$L_2 = \{S(C^n(b), b) \mid n \in \mathbb{N}\}$$

- but their union is not

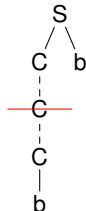
# Tree Substitution Languages

not closed under union

- these languages are tree substitution languages individually



$$L_1 = \{S(C^n(a), a) \mid n \in \mathbb{N}\}$$



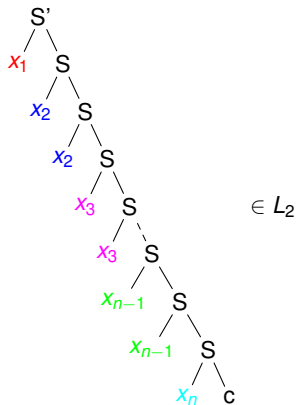
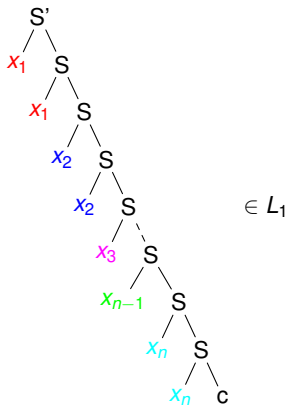
$$L_2 = \{S(C^n(b), b) \mid n \in \mathbb{N}\}$$

- but their union is not  
(exchange subtrees below the indicated cuts)

# Tree Substitution Languages

## not closed under intersection

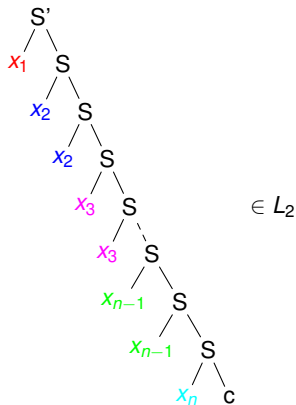
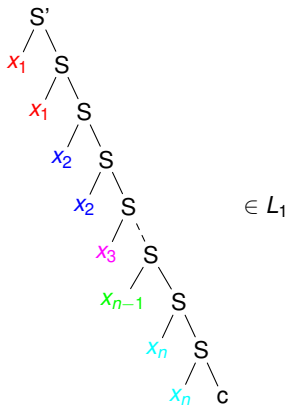
- these languages  $L_1$  and  $L_2$  are tree substitution languages individually for  $n \geq 1$  and arbitrary  $x_1, \dots, x_n \in \{a, b\}$



# Tree Substitution Languages

## not closed under intersection

- these languages  $L_1$  and  $L_2$  are tree substitution languages individually for  $n \geq 1$  and arbitrary  $x_1, \dots, x_n \in \{a, b\}$



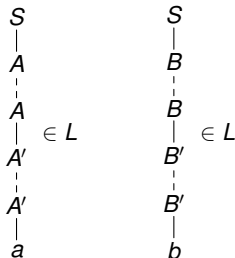
- but their intersection only contains trees with  $x_1 = x_2 = \dots = x_n$  and is not a tree substitution language



# Tree Substitution Languages

not closed under complement

- this language  $L$  is a tree substitution language

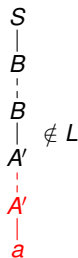
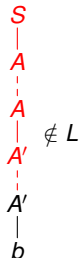
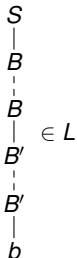


- but its complement is not

# Tree Substitution Languages

not closed under complement

- this language  $L$  is a tree substitution language



- but its complement is not  
(exchange as indicated in red)

# Tree Substitution Languages

## Question

Given regular tree language  $L$ , is it a tree substitution language

# Tree Substitution Languages

## Question

Given regular tree language  $L$ , is it a tree substitution language

## Answer

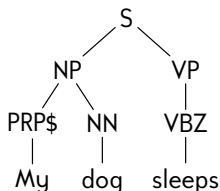
???

# Subcategorization

## Tags:

- official tags often conservative
  - ▶ **English:**  $\approx$  50 tags
  - ▶ **German:**  $\gg$  200 tags

ADJA-Sup-Dat-Sg-Fem



## Comparison:

- rule of subcategorized CFG vs. corresponding rule of tree automaton

$S-1 \rightarrow ADJP-2 \quad S-1$

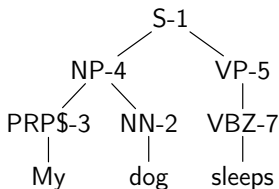
$S-1 \rightarrow S(ADJP-2, S-1)$

# Subcategorization

## Tags:

- official tags often conservative
  - English:  $\approx 50$  tags
  - German:  $\gg 200$  tags
- all modern parsers use refined tags  $\rightarrow$  subcategorization

ADJA-Sup-Dat-Sg-Fem



## Comparison:

- rule of subcategorized CFG vs. corresponding rule of tree automaton

$S-1 \rightarrow ADJP-2 \ S-1$

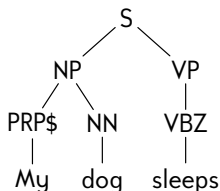
$S-1 \rightarrow S(ADJP-2, S-1)$

# Subcategorization

## Tags:

- official tags often conservative
  - English:  $\approx 50$  tags
  - German:  $\gg 200$  tags
- all modern parsers use refined tags  $\rightarrow$  **subcategorization**
- but return parse over official tags  $\rightarrow$  **relabeling**

ADJA-Sup-Dat-Sg-Fem



## Comparison:

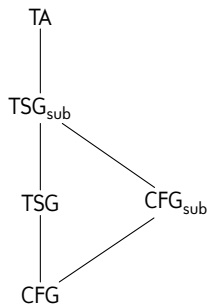
- rule of subcategorized CFG vs. corresponding rule of tree automaton

$S-1 \rightarrow ADJP-2 \quad S-1$

$S-1 \rightarrow S(ADJP-2, S-1)$

# Constituent Parsing

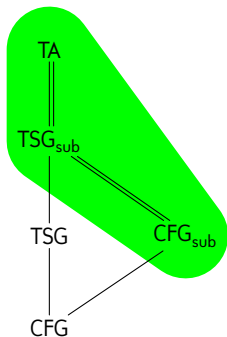
grammar	$F_1$ -score	
	$ w  \leq 40$	full
CFG		62.7
TSG [Post, Gildea, 2009]	82.6	
TSG [Cohn et al., 2010]	85.4	84.7
CFG <sub>sub</sub> [Collins, 1999]	88.6	88.2
CFG <sub>sub</sub> [Petrov, Klein, 2007]	90.6	90.1
CFG <sub>sub</sub> [Petrov, 2010]		91.8
TSG <sub>sub</sub> [Shindo et al., 2012]	<b>92.9</b>	<b>92.4</b>





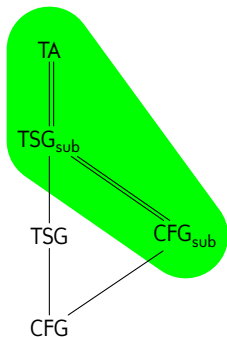
# Constituent Parsing

grammar	$F_1$ -score	
	$ w  \leq 40$	full
CFG		62.7
TSG [Post, Gildea, 2009]	82.6	
TSG [Cohn et al., 2010]	85.4	84.7
CFG <sub>sub</sub> [Collins, 1999]	88.6	88.2
CFG <sub>sub</sub> [Petrov, Klein, 2007]	90.6	90.1
CFG <sub>sub</sub> [Petrov, 2010]		91.8
TSG <sub>sub</sub> [Shindo et al., 2012]	<b>92.9</b>	<b>92.4</b>



# Constituent Parsing

grammar	$\bar{F}_1$ -score	
	$ w  \leq 40$	full
CFG		62.7
TSG [Post, Gildea, 2009]	82.6	
TSG [Cohn et al., 2010]	85.4	84.7
CFG <sub>sub</sub> [Collins, 1999]	88.6	88.2
CFG <sub>sub</sub> [Petrov, Klein, 2007]	90.6	90.1
CFG <sub>sub</sub> [Petrov, 2010]		91.8
TSG <sub>sub</sub> [Shindo et al., 2012]	92.9	92.4

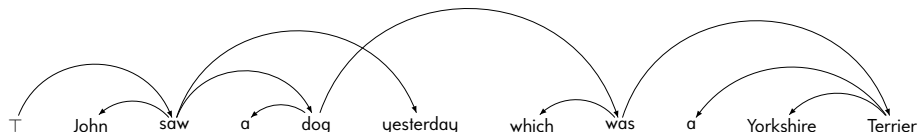


Hence:

- subcategorization = finite-state
- all modern models equivalent to tree automata in expressive power

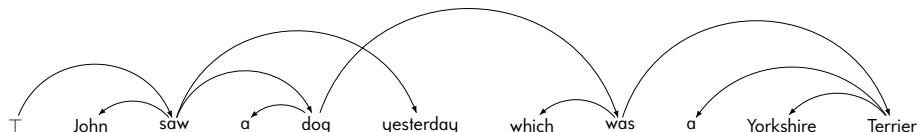
## Parsing

- determining the syntactic structure of a given sentence
- subject to a given theory of syntax (encoded in the training data)
  - ▶ constituent syntax
  - ▶ **dependency syntax**
  - ▶ ...



## Parsing

- determining the syntactic structure of a given sentence
- subject to a given theory of syntax (encoded in the training data)
  - ▶ constituent syntax
  - ▶ **dependency syntax**
  - ▶ ...



## Practical results:

- linear-time statistical parsers
- Google's "Parsey McParseface"  
94%  $F_1$ -score; linguists achieve 96-97%

[[Andor et al., 2016](#)]

# Combinatory Categorical Grammars

## Combinators (Compositions)

Composition rules of degree  $k$  are

$$ax/c, cy \rightarrow axy \quad (\text{forward rule})$$

$$cy, ax \setminus c \rightarrow axy \quad (\text{backward rule})$$

with  $y = |_1c_1|_2 \cdots |_kc_k$

# Combinatory Categorical Grammars

## Combinators (Compositions)

Composition rules of degree  $k$  are

$$ax/c, cy \rightarrow axy \quad (\text{forward rule})$$

$$cy, ax \setminus c \rightarrow axy \quad (\text{backward rule})$$

with  $y = |_1c_1|_2 \cdots |_kc_k$

## Examples

$$\underbrace{\frac{C \quad D/E/D \setminus C}{D/E/D}}_{\text{degree 0}}$$

$$\underbrace{\frac{D/E/D \quad D/E \setminus C}{D/E/E \setminus C}}_{\text{degree 2}}$$

## Combinatory Categorical Grammar (CCG)

$(\Sigma, A, k, I, L)$

- terminal alphabet  $\Sigma$  and atomic categories  $A$
- maximal degree  $k \in \mathbb{N} \cup \{\infty\}$  of composition rules
- initial categories  $I \subseteq A$
- lexicon  $L \subseteq \Sigma \times \mathcal{C}(A)$  with  $\mathcal{C}(A)$  categories over  $A$

## Combinatory Categorical Grammar (CCG)

$(\Sigma, A, k, I, L)$

- terminal alphabet  $\Sigma$  and atomic categories  $A$
- maximal degree  $k \in \mathbb{N} \cup \{\infty\}$  of composition rules
- initial categories  $I \subseteq A$
- lexicon  $L \subseteq \Sigma \times \mathcal{C}(A)$  with  $\mathcal{C}(A)$  categories over  $A$

### Notes:

- always all rules up to the given degree  $k$  allowed
- **$k$ -CCG** = CCG using all composition rules up to degree  $k$



# Combinatory Categorical Grammars

$$\begin{array}{cccccc}
 c & c & & d & & d & & e & e \\
 \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots \\
 \vdots & C & \frac{D/E/D \setminus C}{D/E/D} & & \vdots & & & \vdots & \vdots \\
 \vdots & & & & D/E \setminus C & & & \vdots & \vdots \\
 C & & \frac{D/E/E \setminus C}{D/E/E} & & & & & \vdots & \vdots \\
 \hline
 & & & & & & & E & \vdots \\
 & & & & & & & \hline
 & & & & D/E & & & & E \\
 & & & & \hline
 & & & & & & & & D
 \end{array}$$

2-CCG generates string language  $\mathcal{L}$  with  $\mathcal{L} \cap c^+d^+e^+ = \{c^i d^i e^i \mid i \geq 1\}$   
 for initial categories  $\{D\}$

$$L(c) = \{C\}$$

$$L(d) = \{D/E \setminus C, D/E/D \setminus C\}$$

$$L(e) = \{E\}$$

# Combinatory Categorical Grammars

- allow (deterministic) relabeling (to allow arbitrary labels)

## Question

Which tree languages are legal proof trees of CCG

# Machine Translation

Review translation [by [Google Translate](#) ]

- 1 The room it is not narrowly was a simple, bathtub was also attached.
- 2 Wi-fi, TV and I was available.
- 3 Church looked When morning awake open the curtain.
- 4 But was a little cold, morning walks was good.

## Review translation [by Google Translate ]

- ① The room it is not narrowly was a simple, bathtub was also attached.
- ② Wi-fi, TV and I was available.
- ③ Church looked When morning awake open the curtain.
- ④ But was a little cold, morning walks was good.

## Original [Japanese — © tripadvisor\* ]

- ① 部屋もシンプルでしたが狭くなく、バスタブもついていました。
- ② Wi-fi、テレビも利用出来ました。
- ③ 朝起きてカーテンを開けると教会が見えました。
- ④ ちょっと寒かったけれど、朝の散策はグッドでしたよ。

## Review translation [by Google Translate after 2016]

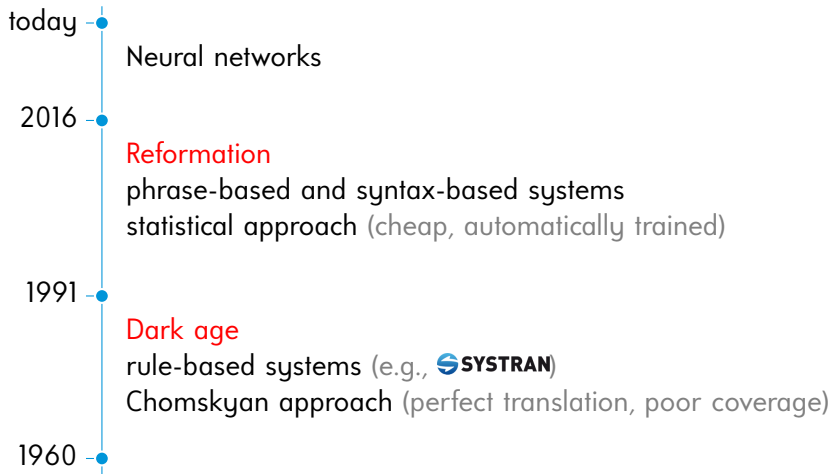
- 1 The room was simple, but it was not small, and the bathtub was also attached.
- 2 Wi-fi, TV was also available.
- 3 When I woke up in the morning and opened the curtain, I saw the church.
- 4 It was a bit cold, but walking in the morning was good.

## Original [Japanese — © tripadvisor]

- 1 部屋もシンプルでしたが狭くなく、バスタブもついていました。
- 2 Wi-fi、テレビも利用出来ました。
- 3 朝起きてカーテンを開けると教会が見えました。
- 4 ちょっと寒かったけれど、朝の散策はグッドでしたよ。

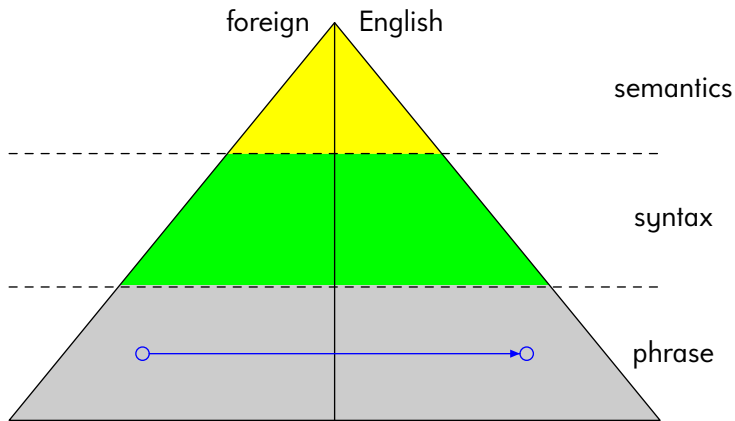
# Machine Translation

## Short History:



# Machine Translation

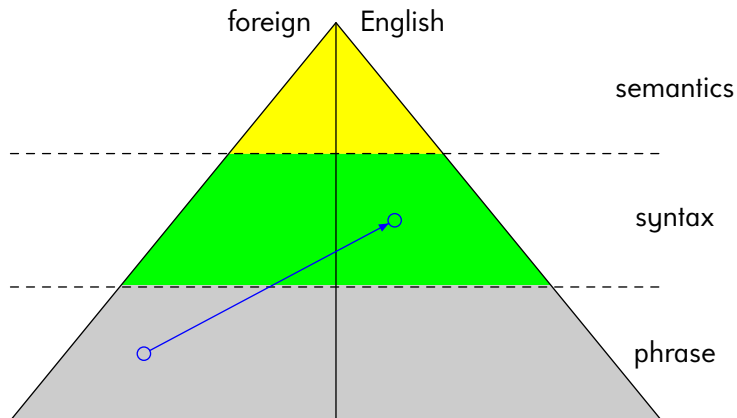
Vauquois triangle:



Translation model: [string-to-string](#)

# Machine Translation

Vauquois triangle:

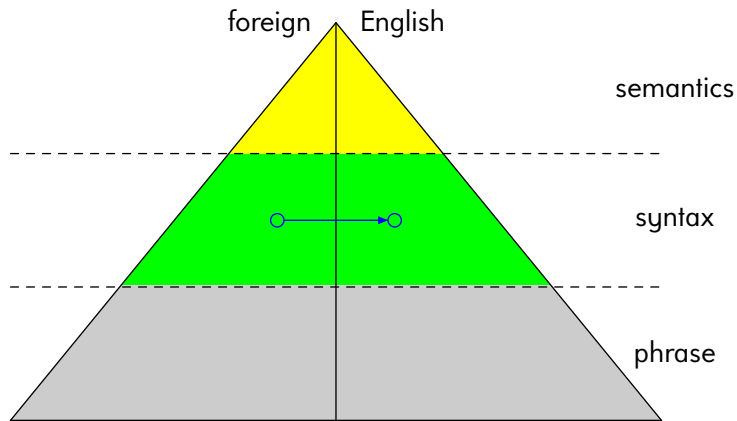


Translation model: [string-to-tree](#)



# Machine Translation

Vauquois triangle:



Translation model: [tree-to-tree](#)

# Machine Translation

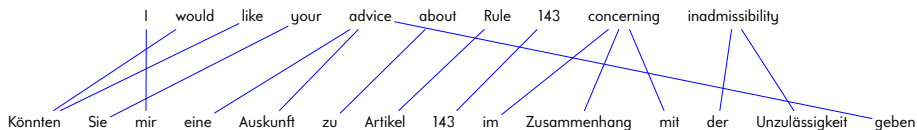
parallel corpus, word alignments, parse tree

I would like your advice about Rule 143 concerning inadmissibility

Könnten Sie mir eine Auskunft zu Artikel 143 im Zusammenhang mit der Unzulässigkeit geben

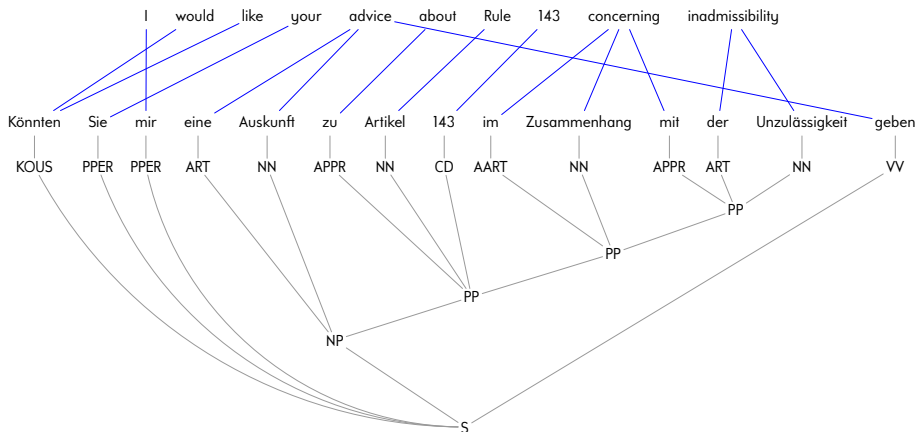
# Machine Translation

parallel corpus, **word alignments**, parse tree



# Machine Translation

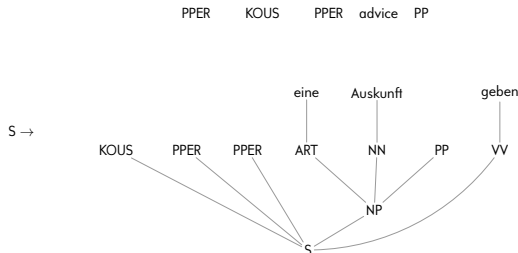
parallel corpus, word alignments, **parse tree**



# Weighted Synchronous Grammars

**Synchronous tree substitution grammar:** productions  $N \rightarrow (r, r_1)$

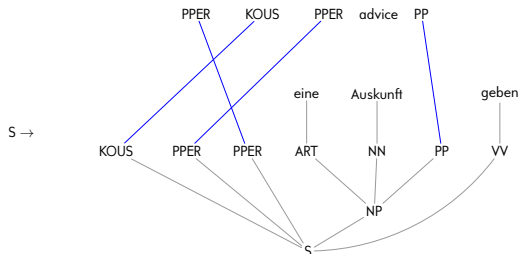
- nonterminal  $N$
- right-hand side  $r$  of context-free grammar production
- right-hand side  $r_1$  of tree substitution grammar production



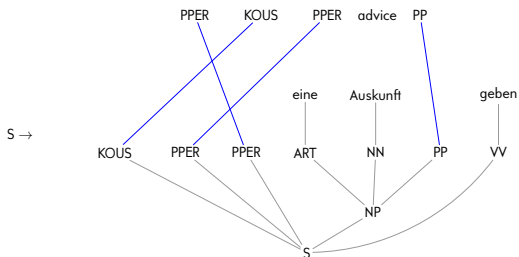
# Weighted Synchronous Grammars

**Synchronous tree substitution grammar:** productions  $N \rightarrow (r, r_1)$

- nonterminal  $N$
- right-hand side  $r$  of context-free grammar production
- right-hand side  $r_1$  of tree substitution grammar production
- (bijective) synchronization of nonterminals



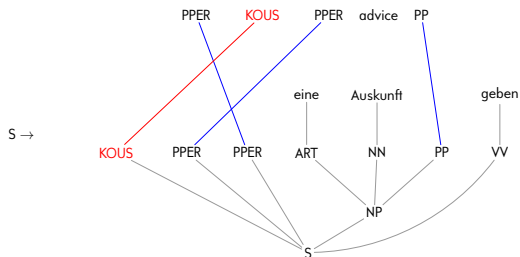
# Synchronous Grammars



Production application:

- 1 Selection of synchronous nonterminals

# Synchronous Grammars

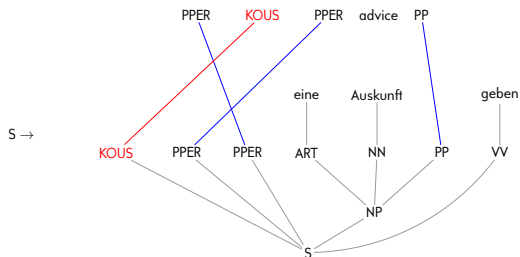


Production application:

- 1 Selection of synchronous nonterminals



# Synchronous Grammars



## Production application:

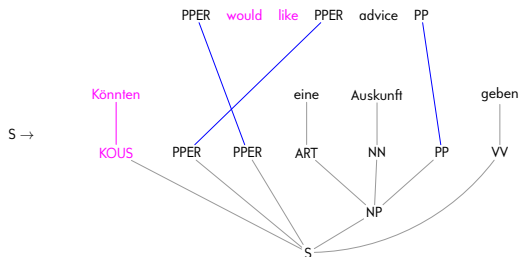
- 1 Selection of synchronous nonterminals
- 2 Selection of suitable production

would like

KOUS →

Könnten  
|  
KOUS

# Synchronous Grammars



## Production application:

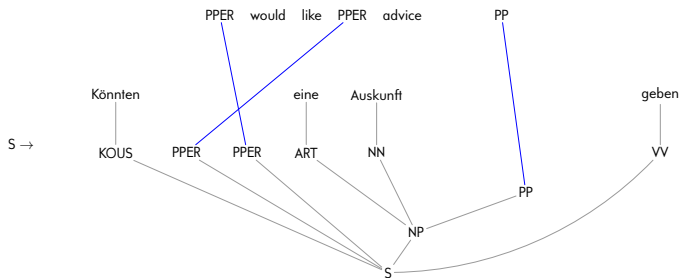
- 1 Selection of synchronous nonterminals
- 2 Selection of suitable production
- 3 Replacement on both sides

KOUS →

would like

Könnten  
KOUS

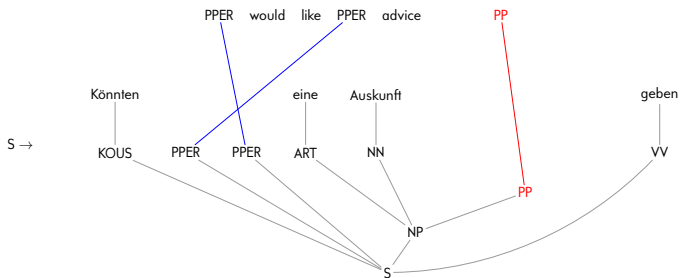
# Synchronous Grammars



Production application:

- 1 synchronous nonterminals

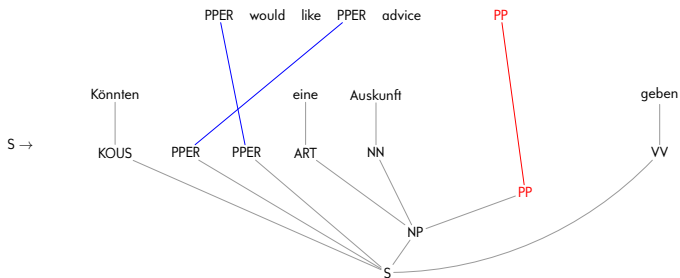
# Synchronous Grammars



Production application:

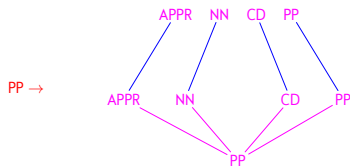
- 1 synchronous nonterminals

# Synchronous Grammars

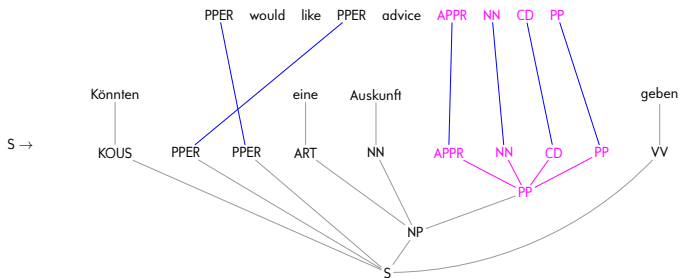


## Production application:

- 1 synchronous nonterminals
- 2 suitable production

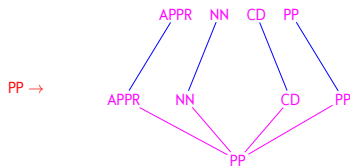


# Synchronous Grammars



## Production application:

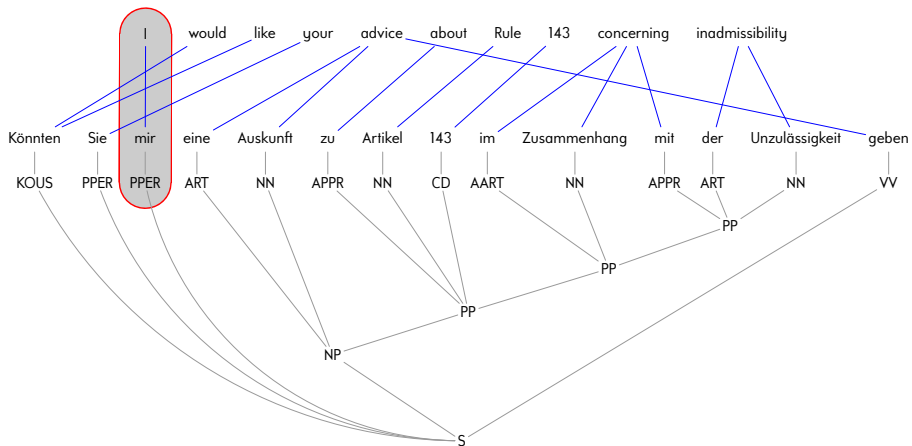
- 1 synchronous nonterminals
- 2 suitable production
- 3 replacement





# Production Extraction

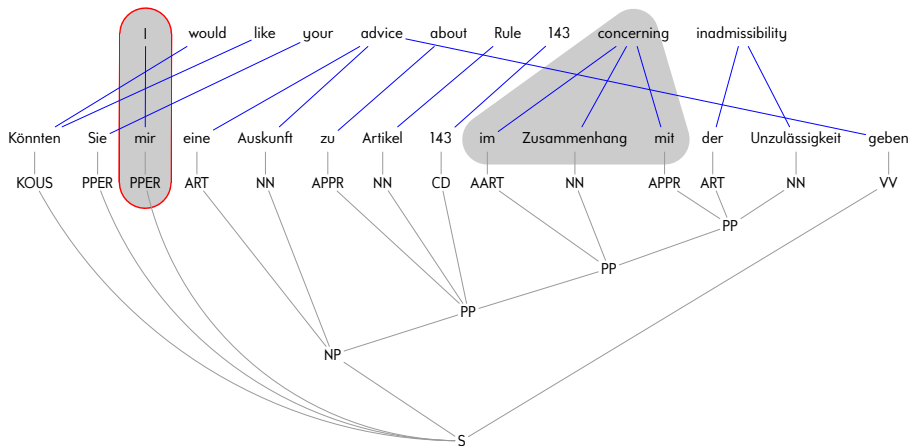
(extractable productions marked in red)





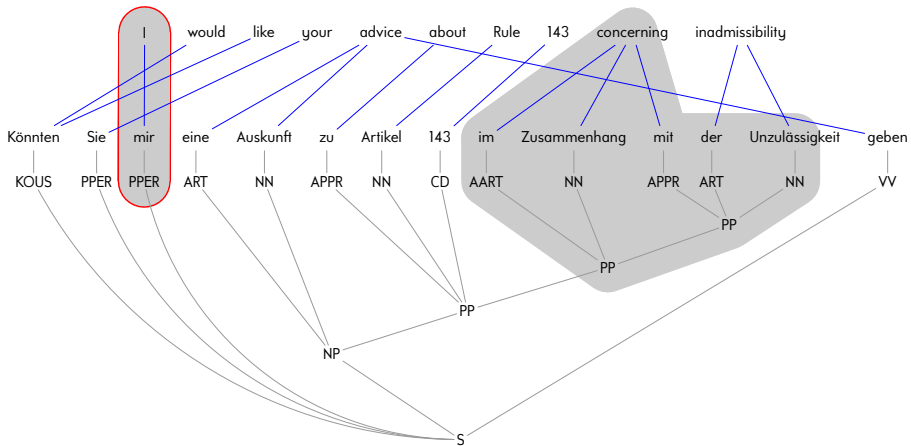
# Production Extraction

(extractable productions marked in red)



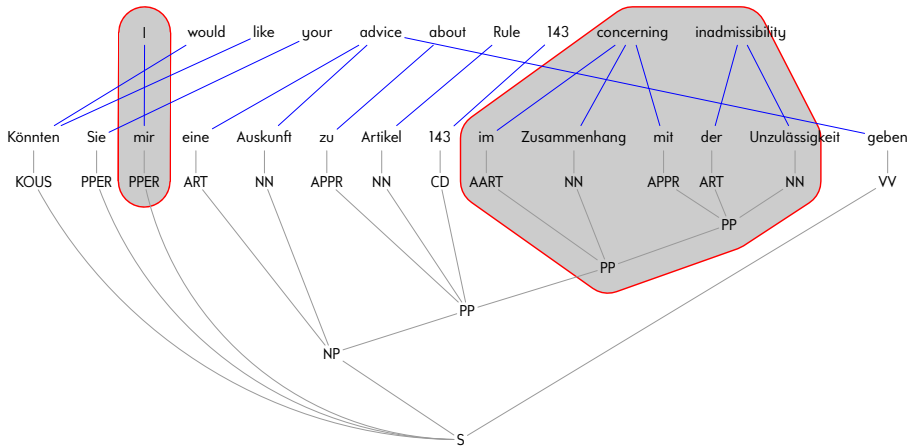
# Production Extraction

(extractable productions marked in red)



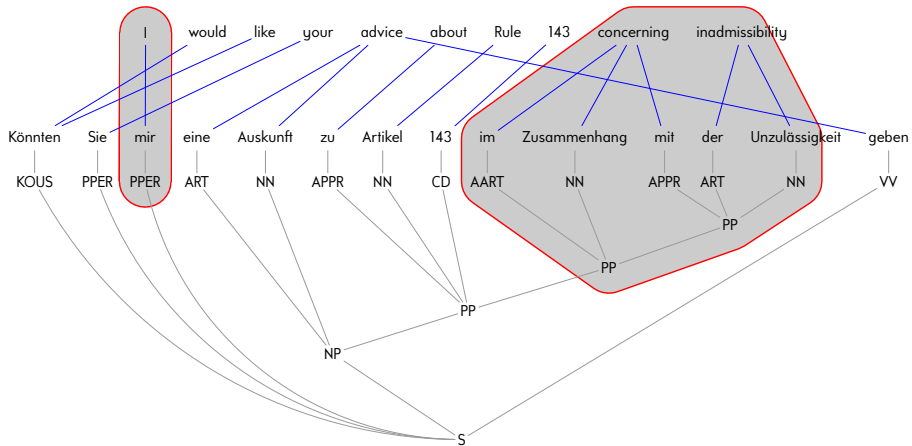
# Production Extraction

(extractable productions marked in red)



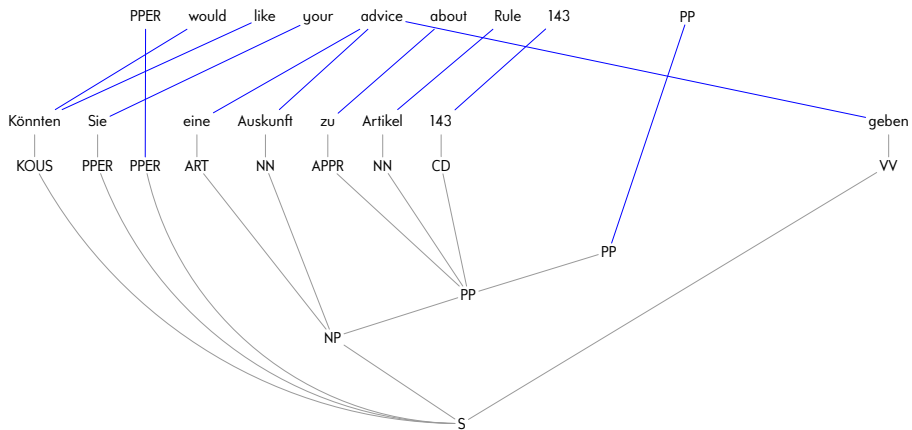
# Production Extraction

## Removal of extractable production:



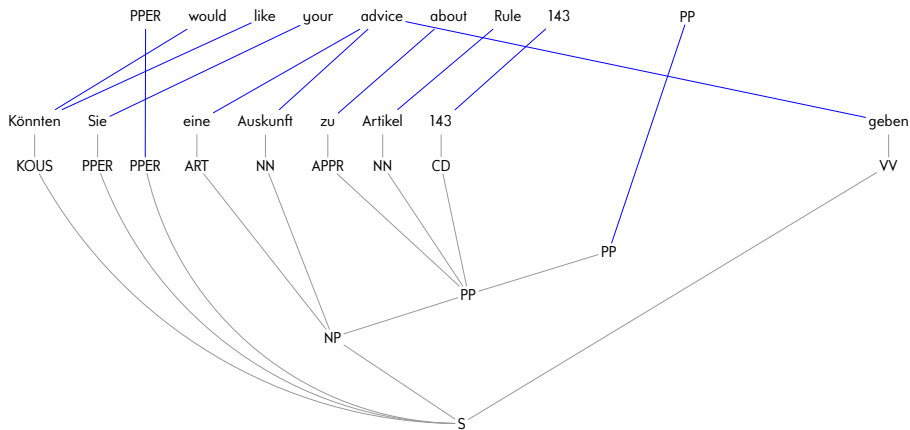
# Production Extraction

Removal of extractable production:



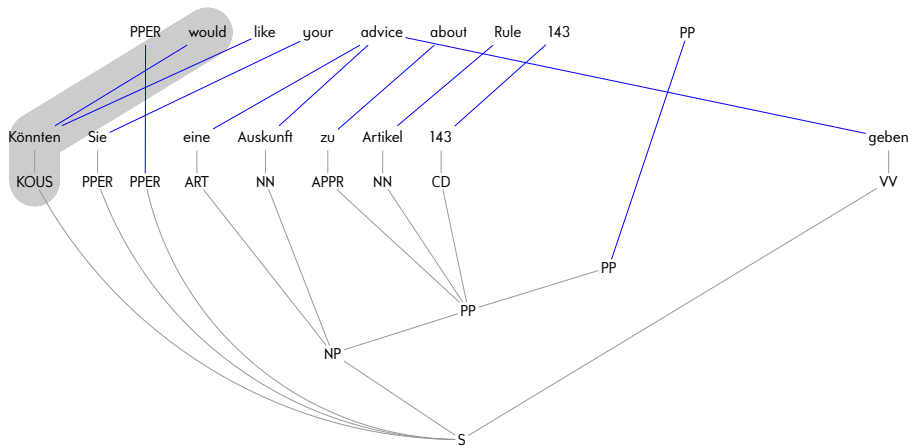
# Production Extraction

Repeated production extraction:



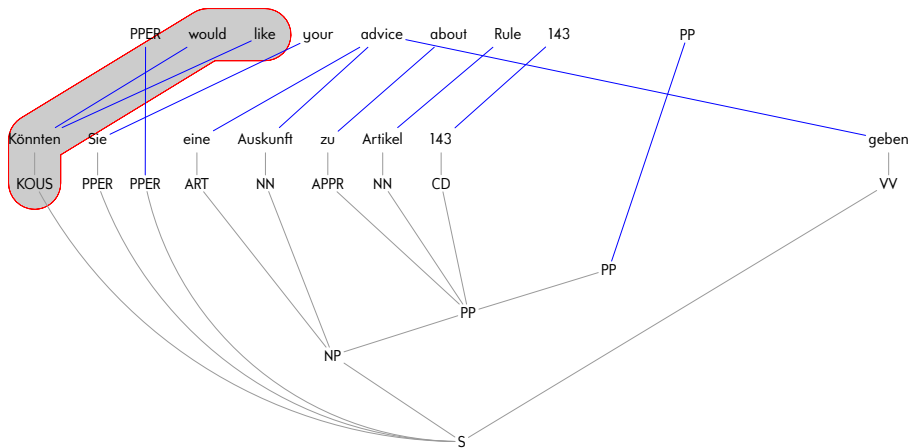
# Production Extraction

Repeated production extraction:



# Production Extraction

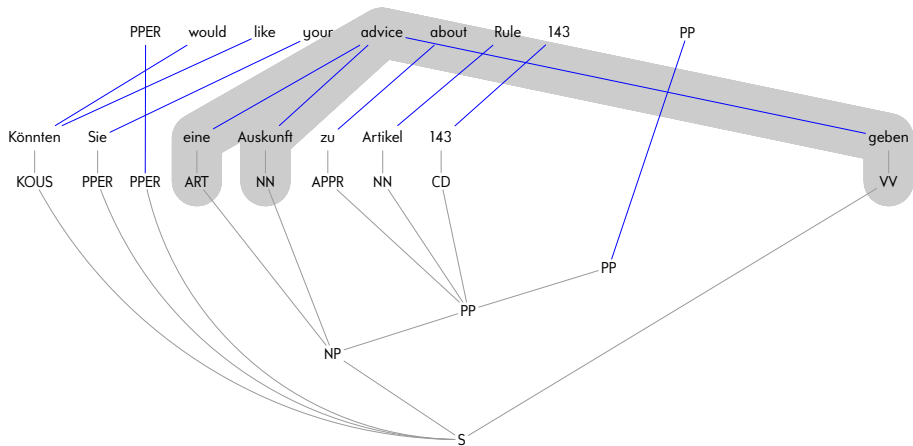
Repeated production extraction: (extractable productions marked in red)





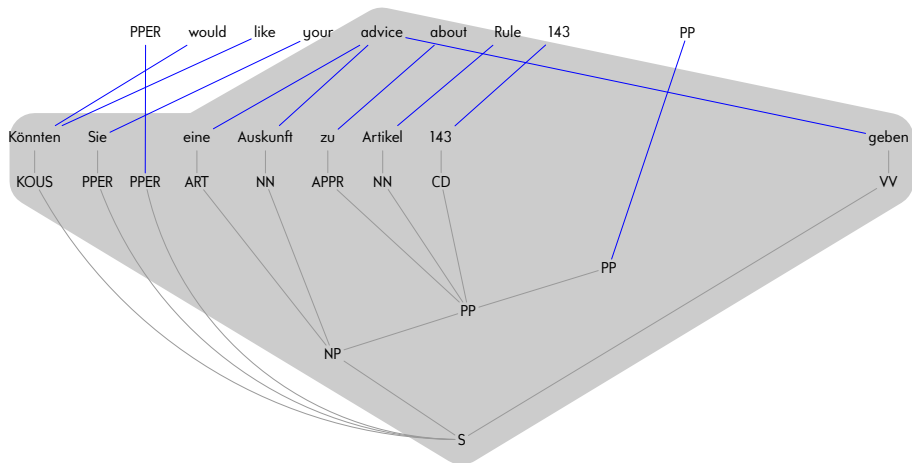
# Production Extraction

Repeated production extraction: (extractable productions marked in red)



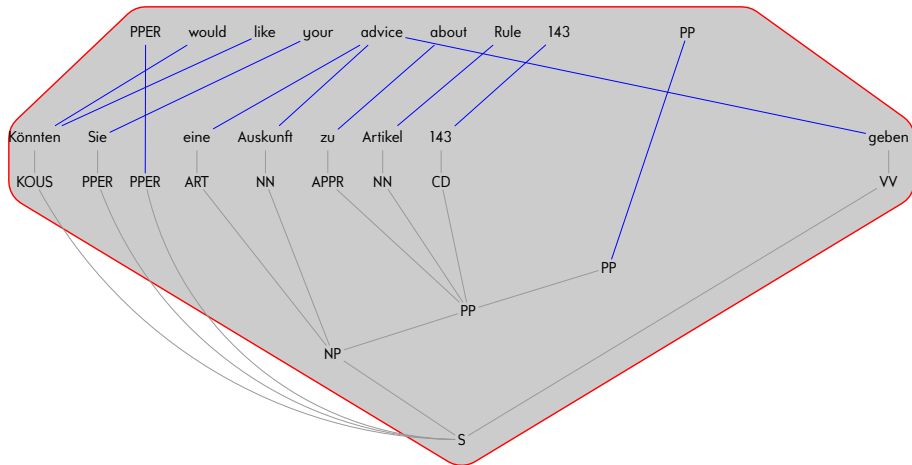
# Production Extraction

Repeated production extraction: (extractable productions marked in red)



# Production Extraction

Repeated production extraction: (extractable productions marked in red)



# Synchronous Tree Substitution Grammars

## Advantages:

- very simple
- implemented in framework ‘Moses’  
[[Koehn](#) et al.: Moses — Open source toolkit for statistical machine translation. *Proc. ACL*, 2007]
- “context-free”

# Synchronous Tree Substitution Grammars

## Advantages:

- very simple
- implemented in framework ‘Moses’  
[Koehn et al.: Moses — Open source toolkit for statistical machine translation. *Proc. ACL*, 2007]
- “context-free”

## Disadvantages:

- problems with discontinuities
- composition and binarization not possible  
[M., Graehl, Hopkins, Knight: The power of extended top-down tree transducers. *SIAM Journal on Computing* 39(2), 2009]  
[Zhang, Huang, Gildea, Knight: Synchronous Binarization for Machine Translation. *Proc. NAACL*, 2006]
- “context-free”

English → German translation task:

(higher BLEU is better)

Type	System	BLEU		
		vanilla	WMT 2013	WMT 2015
string-to-string	FST	16.8	20.3	25.2
string-to-tree	STSG	15.2	19.4	24.5
tree-to-tree	STSG	14.5	—	15.3

STSG = synchronous tree substitution grammar

English → German translation task:

(higher BLEU is better)

Type	System	BLEU		
		vanilla	WMT 2013	WMT 2015
string-to-string	FST	16.8	20.3	25.2
string-to-tree	STSG	15.2	19.4	24.5
tree-to-tree	STSG	14.5	—	15.3

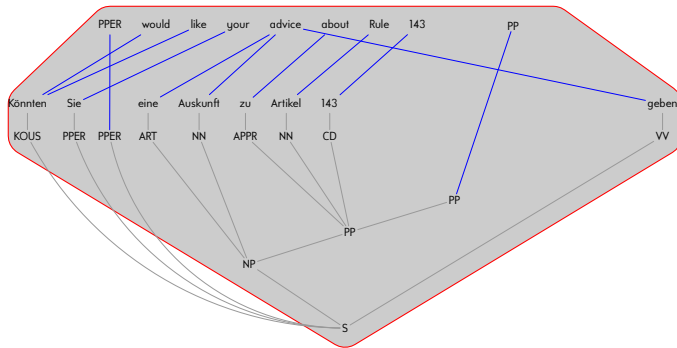
STSG = synchronous tree substitution grammar

## Observations:

- syntax-based systems competitive with manual adjustments
- much less so for vanilla systems
- very unfortunate situation (more supervision yields lower scores)

from [Seemann, Braune, M.: A systematic evaluation of MBOT in statistical machine translation. *Proc. MT-Summit*, 2015]  
and [Bojar et al.: Findings of the 2013 workshop on statistical machine translation. *Proc. WMT*, 2013]  
and [Bojar et al.: Findings of the 2015 workshop on statistical machine translation. *Proc. WMT*, 2015]

# Production Extraction



- very specific production
- every production for 'advice' contains sentence structure  
(syntax "in the way")



# Synchronous Grammars

**Synchronous multi tree substitution grammar:**  $N \rightarrow (r, \langle r_1, \dots, r_n \rangle)$

variant of [M: Why synchronous tree substitution grammars?. *Proc. NAACL*, 2010]

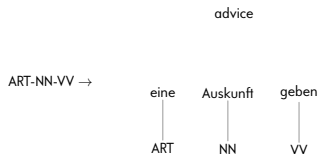
- nonterminal  $N$
- right-hand side  $r$  of context-free grammar production
- right-hand **sides**  $r_1, \dots, r_n$  of regular tree grammar production

# Synchronous Grammars

**Synchronous multi tree substitution grammar:**  $N \rightarrow (r, \langle r_1, \dots, r_n \rangle)$

variant of [M.: Why synchronous tree substitution grammars?. *Proc. NAACL*, 2010]

- nonterminal  $N$
- right-hand side  $r$  of context-free grammar production
- right-hand **sides**  $r_1, \dots, r_n$  of regular tree grammar production



# Synchronous Grammars

**Synchronous multi tree substitution grammar:**  $N \rightarrow (r, \langle r_1, \dots, r_n \rangle)$

variant of [M: Why synchronous tree substitution grammars?. *Proc. NAACL*, 2010]

- nonterminal  $N$
- right-hand side  $r$  of context-free grammar production
- right-hand **sides**  $r_1, \dots, r_n$  of regular tree grammar production

advice

ART-NN-VV  $\rightarrow$

eine	Auskunft	geben
ART	NN	VV

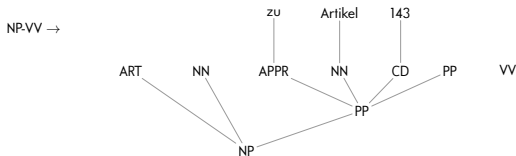
# Synchronous Grammars

**Synchronous multi tree substitution grammar:**  $N \rightarrow (r, \langle r_1, \dots, r_n \rangle)$

variant of [M.: Why synchronous tree substitution grammars?. *Proc. NAACL*, 2010]

- nonterminal  $N$
- right-hand side  $r$  of context-free grammar production
- right-hand **sides**  $r_1, \dots, r_n$  of regular tree grammar production

ART-NN-VV about Rule 143 PP

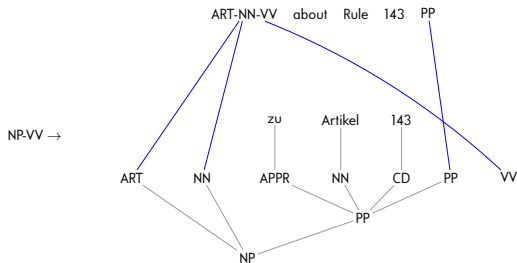


# Synchronous Grammars

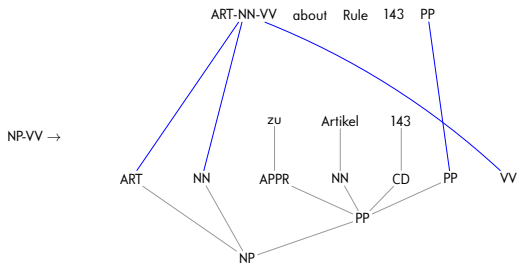
**Synchronous multi tree substitution grammar:**  $N \rightarrow (r, \langle r_1, \dots, r_n \rangle)$

variant of [M.: Why synchronous tree substitution grammars?. *Proc. NAACL*, 2010]

- nonterminal  $N$
- right-hand side  $r$  of context-free grammar production
- right-hand **sides**  $r_1, \dots, r_n$  of regular tree grammar production
- synchronization via map NT  $r_1, \dots, r_n$  to NT  $r$



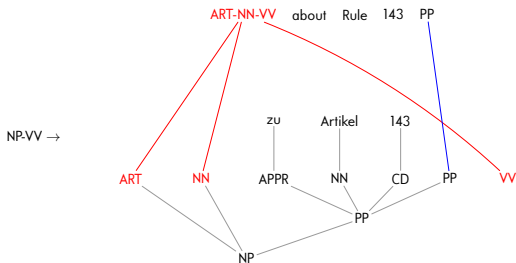
# Synchronous Grammars



Production application:

- 1 synchronous nonterminals

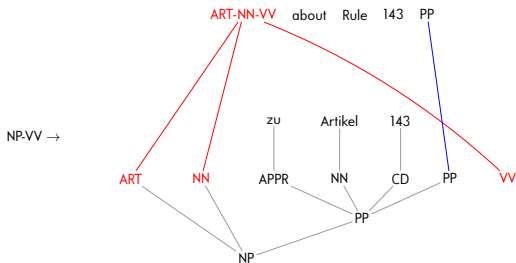
# Synchronous Grammars



Production application:

- 1 synchronous nonterminals

# Synchronous Grammars



## Production application:

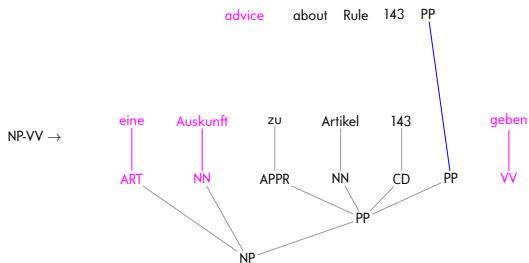
- 1 synchronous nonterminals
- 2 suitable production

ART-NN-VV →



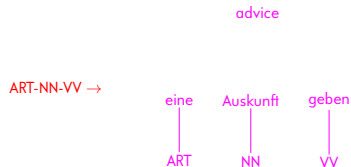


# Synchronous Grammars

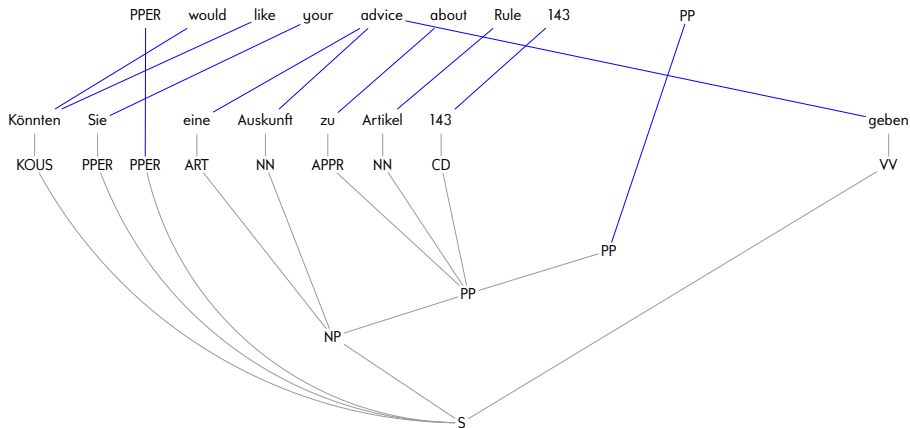


## Production application:

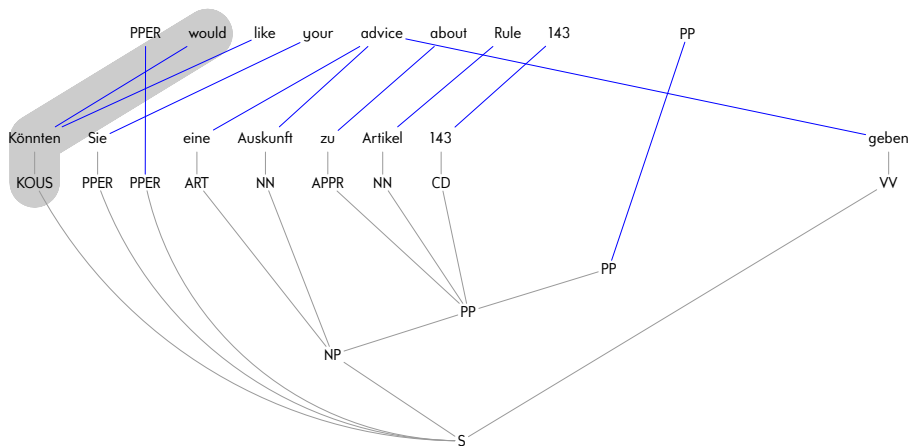
- 1 synchronous nonterminals
- 2 suitable production
- 3 replacement



# Production Extraction

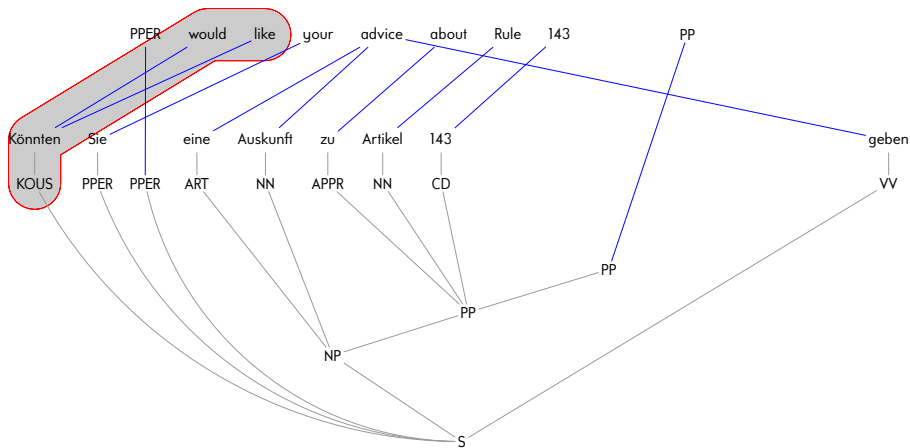


# Production Extraction



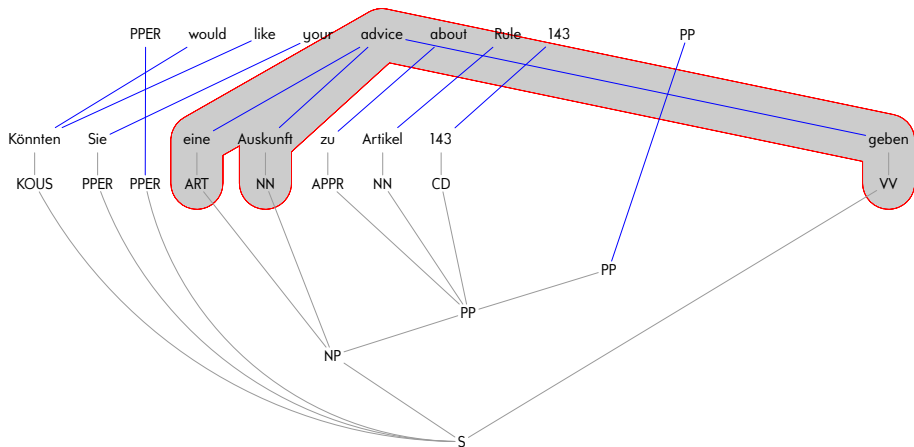
# Production Extraction

(extractable productions marked in red)



# Production Extraction

(extractable productions marked in red)



# Synchronous Multi Tree Substitution Grammars

## Advantages:

- complicated discontinuities
- implemented in framework 'Moses'

[Braune, Seemann, Quernheim, M.: Shallow local multi bottom-up tree transducers in *SMT. Proc. ACL*, 2013]

- binarizable, composable

# Synchronous Multi Tree Substitution Grammars

## Advantages:

- complicated discontinuities
- implemented in framework 'Moses'

[Braune, Seemann, Quernheim, M.: Shallow local multi bottom-up tree transducers in SMT. *Proc. ACL*, 2013]

- binarizable, composable

## Disadvantages:

- output non-regular (tree-level) or non-context-free (string-level)  
(in fact output is captured by MRTG = MCFTG without variables)
- not symmetric (input context-free; output not)

Task	BLEU	
	STSG	SMTSG
English → German	15.0	*15.5
English → Arabic	48.2	*49.1
English → Chinese	17.7	*18.4
English → Polish	21.3	*23.4
English → Russian	24.7	*26.1

STSG = synchronous tree substitution grammar

SMTSG = synchronous multi tree substitution grammar



# Evaluation

Task	BLEU		Productions	
	STSG	SMTSG	STSG	SMTSG
English → German	15.0	*15.5	14M	144M
English → Arabic	48.2	*49.1	55M	491M
English → Chinese	17.7	*18.4	17M	162M
English → Polish	21.3	*23.4	—	—
English → Russian	24.7	*26.1	—	—

STSG = synchronous tree substitution grammar

SMTSG = synchronous multi tree substitution grammar

Task	BLEU		Productions	
	STSG	SMTSG	STSG	SMTSG
English → German	15.0	*15.5	14M	144M
English → Arabic	48.2	*49.1	55M	491M
English → Chinese	17.7	*18.4	17M	162M
English → Polish	21.3	*23.4	—	—
English → Russian	24.7	*26.1	—	—

STSG = synchronous tree substitution grammar

SMTSG = synchronous multi tree substitution grammar

## Observations:

- consistent improvements
- 1 magnitude more productions
- SMTSG alleviate some of the problems of syntax-based systems

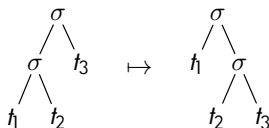
# Synchronous Grammars

## Evaluation properties:



rotations implementable?

(for arbitrary  $t_1, t_2, t_3$ )



symmetric?



domain regular?



range regular?



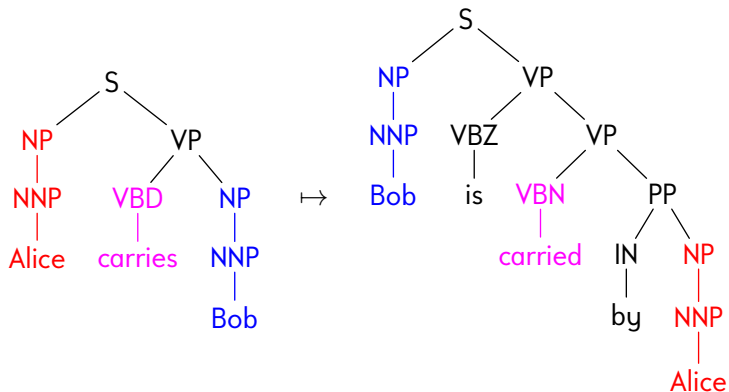
closed under composition?

following [Knight: Capturing practical natural language transformations. *Machine Translation* 21(2), 2007]  
and [May, Knight, Vogler: Efficient inference through cascades of weighted tree transducers. *Proc. ACL*, 2010]

Icons by interactivemania (<http://www.interactivemania.com/>) and UN Office for the Coordination of Humanitarian Affairs

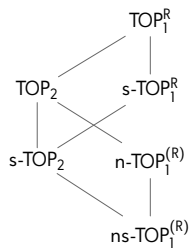
# Synchronous Grammars

Illustration of rotation:








# Top-down Tree Transducer

Hasse diagram:

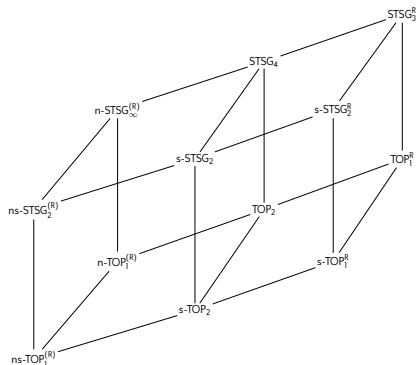


(composition closure in subscript)

Model	Property				
					
ns-TOP	X	X	✓	✓	✓
n-TOP	X	X	✓	✓	✓
s-TOP	X	X	✓	✓	X <sub>2</sub>
s-TOP <sup>R</sup>	X	X	✓	✓	✓
TOP	X	X	✓	✓	X <sub>2</sub>
TOP <sup>R</sup>	X	X	✓	✓	✓

# Synchronous Tree Substitution Grammars






Hasse diagram:



(composition closure in subscript)

composition closures by

[Engelfriet, Fülöp, M.: Composition closure of linear extended top-down tree transducers. *Theory of Computing Systems*, to appear 2016]

Model	Property				
					
n-TOP	X	X	✓	✓	✓
TOP	X	X	✓	✓	X <sub>2</sub>
TOP <sup>R</sup>	X	X	✓	✓	✓
ns-STSG	✓	✓	✓	✓	X <sub>2</sub>
n-STSG	✓	X	✓	✓	X <sub>∞</sub>
s-STSG <sup>(R)</sup>	✓	X	✓	✓	X <sub>2</sub>
STSG	✓	X	✓	✓	X <sub>4</sub>
STSG <sup>R</sup>	✓	X	✓	✓	X <sub>3</sub>

## Advantages of SMTSG

- always have regular look-ahead
- can always be made nondeleting & shallow
- closed under composition

# Synchronous Multi Tree Substitution Grammars

## Advantages of SMTSG

- always have regular look-ahead
- can always be made nondeleting & shallow
- closed under composition

## Disadvantages of SMTSG:

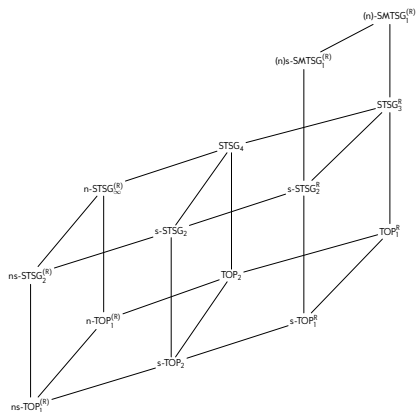
- non-regular range

(theoretically interesting?)








# Synchronous Multi Tree Substitution Grammars

Hasse diagram:



(composition closure in subscript)

Model					
n-TOP	X	X	✓	✓	✓
TOP	X	X	✓	✓	X <sub>2</sub>
TOP <sup>R</sup>	X	X	✓	✓	✓
ns-STSG	✓	✓	✓	✓	X <sub>2</sub>
n-STSG	✓	X	✓	✓	X <sub>∞</sub>
s-STSG <sup>(R)</sup>	✓	X	✓	✓	X <sub>2</sub>
STSG	✓	X	✓	✓	X <sub>4</sub>
STSG <sup>R</sup>	✓	X	✓	✓	X <sub>3</sub>
SMTSG	✓	X	✓	X	✓
reg. range	✓	X	✓	✓	✓
symmetric	✓	✓	✓	✓	✓

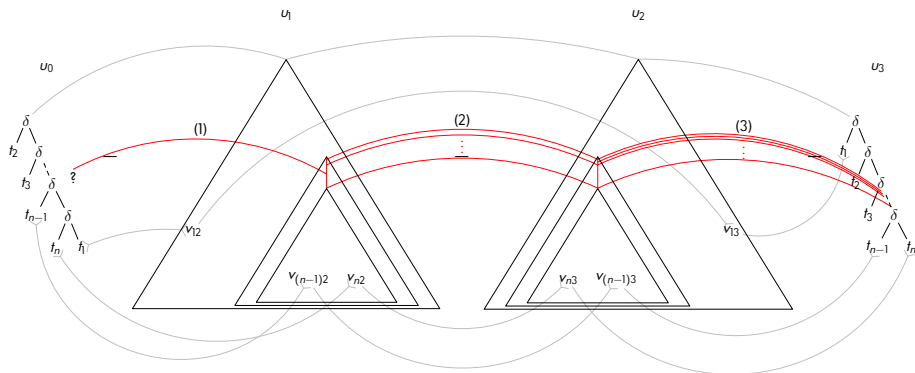
(string-level) range characterization by

[Gildea: On the string translations produced by multi bottom-up tree transducers. *Computational Linguistics* 38(3), 2012]

# Synchronous Multi Tree Substitution Grammars

## Theorem

$$(\text{STSG}^R)^3 \subsetneq \text{reg.-range SMTSG}$$



# Summary

## Parsing:

- tree automata = CFG with subcategorization  
(which are the state-of-the-art models for many languages)
- wealth of open problems for non-constituent parsing  
(alternative theories seem to be on the rise; “Parsey McParseface”)

# Summary

## Parsing:

- tree automata = CFG with subcategorization  
(which are the state-of-the-art models for many languages)
- wealth of open problems for non-constituent parsing  
(alternative theories seem to be on the rise; “Parsey McParseface”)

## Machine translation:

- all major (non-neural) translation models in use are grammar-based  
(and their expressive power is often ill-understood)
- combination of parser and translation model challenging  
(although that is typically just a regular domain restriction)
- evaluation of theoretically well-behaved models (in practice)

# Summary

## Parsing:

- tree automata = CFG with subcategorization  
(which are the state-of-the-art models for many languages)
- wealth of open problems for non-constituent parsing  
(alternative theories seem to be on the rise; “Parsey McParseface”)

## Machine translation:

- all major (non-neural) translation models in use are grammar-based  
(and their expressive power is often ill-understood)
- combination of parser and translation model challenging  
(although that is typically just a regular domain restriction)
- evaluation of theoretically well-behaved models (in practice)

Thank you for the attention.