Introduction to Tree Adjoining Grammar

Natural Language Syntax with TAG

Wolfgang Maier and Timm Lichte
University of Düsseldorf

DGfS-CL Fall School 2011

1st week, 1st session
Aug 29, 2011

Overview: First week

1. Going beyond Context-Free Grammar (Monday)
2. Formal definition of TAG (Tuesday)
3. TAG and natural languages (Wednesday)
4. TAG parsing (Thursday)
5. Extensions of TAG (Friday)

Overview: Second week

1. Principles underlying the shape of elementary trees
2. XTAG-analyses of raising/control
3. XTAG-analyses of extraction
4. How to implement an LTAG
5. How to run and test an LTAG

Outline

1. Organizational issues
2. CFG and natural languages
   - Context-Free Grammars
   - Are Natural Languages Context-Free?
   - Mild Context-Sensitivity
3. Tree Substitution Grammar
   - Definition
   - Properties
4. Adjunction

Organizational issues

- Course web page:
  `http://www.sfb991.uni-duesseldorf.de/a02/dgfs-11`
- Requirements for obtaining 4 ETCS credits:
  - Participation in each class
  - Solving at least 75% of the exercises
  - Writing a short essay (4 pages) or solving an implementation task

Context-Free Grammars

Context-Free Grammar (CFG)

- Disjoint sets of terminals and non-terminals
- A non-terminal start symbol
- A set of rewriting rules stating how to replace a non-terminal by a sequence of non-terminal and terminal symbols.

Context-Free Grammars

Context-Free Grammar (CFG)

- Disjoint sets of terminals and non-terminals
- A non-terminal start symbol
- A set of rewriting rules stating how to replace a non-terminal by a sequence of non-terminal and terminal symbols.

Example

$\mathrm{S} \rightarrow \mathrm{a\ S\ b} \quad \mathrm{S} \rightarrow \mathrm{ab}$

Context-Free Grammar (CFG)

- Disjoint sets of terminals and non-terminals
- A non-terminal start symbol
- A set of rewriting rules stating how to replace a non-terminal by a sequence of non-terminal and terminal symbols.

Example

S $\rightarrow$ a S b   S $\rightarrow$ ab

Generates the string language $\{a^n b^n \mid n \geq 1\}$.

CFG language

Definition (CFG language)

Let $G = \langle N, T, P, S \rangle$ *be a CFG. The (string) language* $L(G)$ *of* G *is the set* $\{w \in T^* \mid S \stackrel{*}{\Rightarrow} w\}$ *where*

- *for* $w, w' \in (N \cup T)^*$: $w \Rightarrow w'$ *iff there is a* $A \to \alpha \in P$ *and there are* $v, u \in (N \cup T)^*$ *such that* $w = vAu$ *and* $w' = v\alpha u$.

Definition (CFG language)

Let $G = \langle N, T, P, S \rangle$ be a CFG. The (string) language $L(G)$ of G is the set $\{w \in T^* \mid S \stackrel{*}{\Rightarrow} w\}$ where

- *for $w, w' \in (N \cup T)^*$: $w \Rightarrow w'$ iff there is a $A \rightarrow \alpha \in P$ and there are $v, u \in (N \cup T)^*$ such that $w = vAu$ and $w' = v\alpha u$.*
- *$\stackrel{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$:*
  - *$w \stackrel{0}{\Rightarrow} w$ for all $w \in (N \cup T)^*$, and*
  - *for all $w, w' \in (N \cup T)^*$: $w \stackrel{n}{\Rightarrow} w'$ iff there is a v such that $w \Rightarrow v$ and $v \stackrel{n-1}{\Rightarrow} w'$.*
  - *for all $w, w' \in (N \cup T)^*$: $w \stackrel{*}{\Rightarrow} w'$ iff there is a $i \in \mathbb{N}$ such that $w \stackrel{i}{\Rightarrow} w'$.*

Definition (CFG language)

Let $G = \langle N, T, P, S\rangle$ be a CFG. The (string) language $L(G)$ of G is the set $\{w \in T^* \mid S \stackrel{*}{\Rightarrow} w\}$ where

- *for $w, w' \in (N \cup T)^*$: $w \Rightarrow w'$ iff there is a $A \rightarrow \alpha \in P$ and there are $v, u \in (N \cup T)^*$ such that $w = vAu$ and $w' = v\alpha u$.*
- *$\stackrel{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$:*
  - *$w \stackrel{0}{\Rightarrow} w$ for all $w \in (N \cup T)^*$, and*
  - *for all $w, w' \in (N \cup T)^*$: $w \stackrel{n}{\Rightarrow} w'$ iff there is a v such that $w \Rightarrow v$ and $v \stackrel{n-1}{\Rightarrow} w'$.*
  - *for all $w, w' \in (N \cup T)^*$: $w \stackrel{*}{\Rightarrow} w'$ iff there is a $i \in \mathbb{N}$ such that $w \stackrel{i}{\Rightarrow} w'$.*

A language is called context-free iff it is generated by a CFG.

Context-Free Languages (CFLs)

Context-Free Languages (CFLs)

- can be recognized in polynomial time ($\mathcal{O}(n^3)$);

Context-Free Languages (CFLs)

- can be recognized in polynomial time ($\mathcal{O}(n^3)$);
- are accepted by push-down automata;

Context-Free Languages

Context-Free Languages (CFLs)

- can be recognized in polynomial time ($\mathcal{O}(n^3)$);
- are accepted by push-down automata;
- have nice closure properties (e.g., under homomorphisms, intersection with regular languages . . .);

Context-Free Languages

Context-Free Languages (CFLs)

- can be recognized in polynomial time ($\mathcal{O}(n^3)$);
- are accepted by push-down automata;
- have nice closure properties (e.g., under homomorphisms, intersection with regular languages ...);
- satisfy a pumping lemma;

Context-Free Languages

Context-Free Languages (CFLs)

- can be recognized in polynomial time ($\mathcal{O}(n^3)$);
- are accepted by push-down automata;
- have nice closure properties (e.g., under homomorphisms, intersection with regular languages ...);
- satisfy a pumping lemma;
- can describe nested dependencies ($\{ww^R \mid w \in T^*\}$).

Context-Free Languages (CFLs)

- can be recognized in polynomial time ($\mathcal{O}(n^3)$);
- are accepted by push-down automata;
- have nice closure properties (e.g., under homomorphisms, intersection with regular languages ...);
- satisfy a pumping lemma;
- can describe nested dependencies ($\{ww^R \mid w \in T^*\}$).

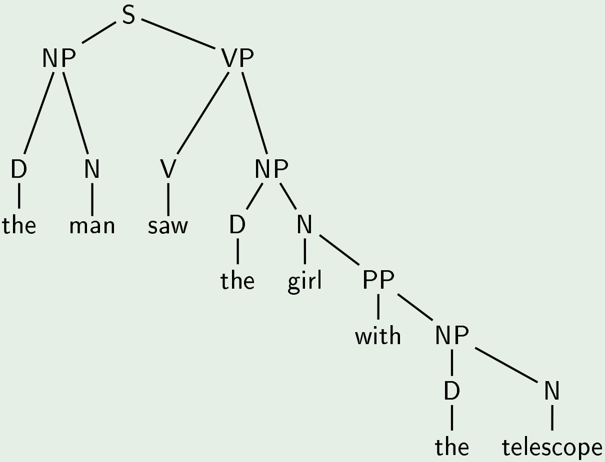
[Hopcroft and Ullman, 1979]

CFG Natural Language Example

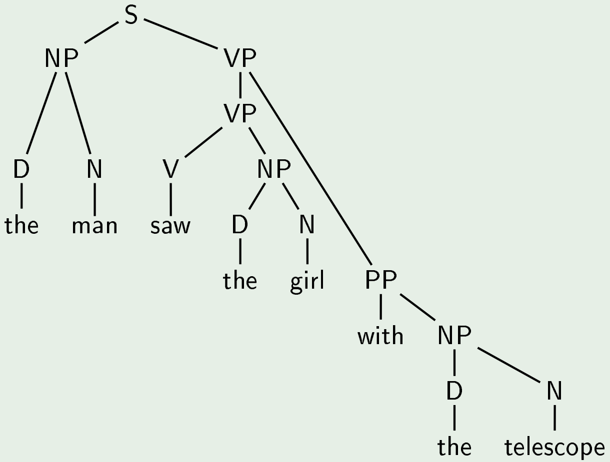
Sample CFG $G_{telescope}$

- Nonterminals: $\{S, NP, VP, PP, N, V, P, D\}$
- Terminals: {the, man, telescope, saw, girl, with, John}
- Productions:

$$
\begin{array}{lcl lcl}
S & \rightarrow & NP\ VP & NP & \rightarrow & D\ N \\
VP & \rightarrow & VP\ PP \mid V\ NP & N & \rightarrow & N\ PP \\
PP & \rightarrow & P\ NP & & & \\
N & \rightarrow & \text{man} \mid \text{girl} \mid \text{telescope} & D & \rightarrow & \text{the} \\
N & \rightarrow & \text{John} & P & \rightarrow & \text{with} \\
V & \rightarrow & \text{saw} & & &
\end{array}
$$


S
NP
VP
D
N
V
NP
the
man
saw
D
N
the
girl
PP
with
NP
D
N
the
telescope

Example derivation



Why CFG is not enough

... for modeling natural language:

1. only atomic non-terminals
2. only weak lexicalization
3. expressive power is too low

Why CFG is not enough (1) - Atomic non-terminals

$S \rightarrow NP\ VP$	$NP \rightarrow John$	$NP \rightarrow Mary$	
$VP \rightarrow V$	$VP \rightarrow V\ NP$	$V \rightarrow sleeps$	$V \rightarrow likes$

Possible derivation:

$S \Rightarrow NP\ VP \Rightarrow John\ VP \Rightarrow John\ V \Rightarrow John\ sleeps$

$S \stackrel{*}{\Rightarrow} John\ likes\ Mary$

$S \stackrel{*}{\Rightarrow} John\ sleeps\ Mary$

Why CFG is not enough (1) - Atomic non-terminals

$S \rightarrow NP\ VP$	$NP \rightarrow John$	$NP \rightarrow Mary$	
$VP \rightarrow V$	$VP \rightarrow V\ NP$	$V \rightarrow sleeps$	$V \rightarrow likes$

Possible derivation:

$S \Rightarrow NP\ VP \Rightarrow John\ VP \Rightarrow John\ V \Rightarrow John\ sleeps$

$S \stackrel{*}{\Rightarrow} John\ likes\ Mary$

$S \stackrel{*}{\Rightarrow} John\ sleeps\ Mary$

How to treat subcategorization frames, number agreement, and case marking?

(1) a. Kim depends on Sandy.
*Kim depends Sandy.
*Kim depends.
b. *The children depends on Sandy.
c. Kim depends on her/*she.

Why CFG is not enough (1)

How to treat subcategorization frames, number agreement, and case marking?

$\Longrightarrow$ encode the necessary information into the non-terminal symbols

$$
\begin{array}{ll}
NP_{3sg-nom} \rightarrow John & NP_{3sg-acc} \rightarrow Mary \\
V_{3sg-itr} \rightarrow sleeps & V_{3sg-tr} \rightarrow likes \\
S \rightarrow NP_{3sg-nom}\ VP_{3sg-itr} & S \rightarrow NP_{3sg-nom}\ VP_{3sg-tr} \\
VP_{3sg-itr} \rightarrow V_{3sg-itr} & VP_{3sg-tr} \rightarrow V_{3sg-tr}\ NP_{3sg-acc}
\end{array}
$$

$S \stackrel{*}{\Rightarrow} John\ likes\ Mary$

$S \stackrel{*}{\Rightarrow} John\ sleeps$

Why CFG is not enough (1)

How to treat subcategorization frames, number agreement, and case marking?

$\Longrightarrow$ encode the necessary information into the non-terminal symbols

$NP_{3sg-nom} \rightarrow John$	$NP_{3sg-acc} \rightarrow Mary$
$V_{3sg-itr} \rightarrow sleeps$	$V_{3sg-tr} \rightarrow likes$
$S \rightarrow NP_{3sg-nom}\ VP_{3sg-itr}$	$S \rightarrow NP_{3sg-nom}\ VP_{3sg-tr}$
$VP_{3sg-itr} \rightarrow V_{3sg-itr}$	$VP_{3sg-tr} \rightarrow V_{3sg-tr}\ NP_{3sg-acc}$

$S \overset{*}{\Rightarrow} John\ likes\ Mary$

$S \overset{*}{\Rightarrow} John\ sleeps$

Drawback: Every possible combination of subcategorization frame, number agreement, and case marking necessitates its own rule (let alone the number of non-terminal symbols).

Why CFG is not enough (1)

$\Longrightarrow$ grammar writing is tedious and error prone
$\Longrightarrow$ generalizations very hard to express

Remedy: feature structures instead of atomic non-terminal symbols, unification, underspecification

Why CFG is not enough (2) - Only weak lexicalization

Lexicalization

In a lexicalized grammar, each element of the grammar contains at least one lexical item (terminal symbol).

G_1: $S \rightarrow SS$, $S \rightarrow a$
G_2: $S \rightarrow aS$, $S \rightarrow a$

- **Computationally interesting:** the number of analyses for a sentence is finite (if the grammar is finite of course).
- **Linguistically interesting:** each lexical item allows for of certain syntactic constructions, which one would like to associate with it.

Why CFG is not enough (2)

Lexicalizing a CFG:

- Greibach normal form: $A \rightarrow aB_1 \dots B_k$ $(k \geq 0)$
- weak lexicalization: string language is preserved
- strong lexicalization: tree structure is preserved

Question

Can CFGs be lexicalized such that the set of trees remains the same (strong lexicalization)?

Answer

No. Only weak lexicalization (same string language).

Why CFG is not enough (2)

Lexicalizing a CFG:

- Greibach normal form: $A \to aB_1 ... B_k$ $(k \geq 0)$
- weak lexicalization: string language is preserved
- strong lexicalization: tree structure is preserved

Question

Can CFGs be lexicalized such that the set of trees remains the same (strong lexicalization)?

Answer

No. Only weak lexicalization (same string language).

Why CFG is not enough (3) - Expressivity

Question

Is CFG powerful enough to describe all natural language phenomea?

Why CFG is not enough (3) - Expressivity

Question

Is CFG powerful enough to describe all natural language phenomea?

Answer: No!

Some NL constructions cannot be adequately described with a CFG.

Why CFG is not enough (3) - Expressivity

Question

Is CFG powerful enough to describe all natural language phenomea?

Answer: No!

Some NL constructions cannot be adequately described with a CFG.

Cross-serial dependencies in Dutch

(2) ... dat Wim Jan Marie de kinderen zag helpen leren zwemmen
... that Wim Jan Marie the children saw help teach swim
'... that Wim saw Jan help Marie teach the children to swim'

Why CFG is not enough (3)

Swiss German

(3) … das mer em Hans es huus hälfed aastriiche
… that we Hans_{Dat} house_{Acc} helped paint
'… that we helped Hans paint the house'

Why CFG is not enough (3)

Swiss German

(3) ... das mer em Hans es huus hälfed aastriiche
... that we Hans$_{Dat}$ house$_{Acc}$ helped paint
'... that we helped Hans paint the house'

(4) ... das mer d'chind em Hans es huus lönd hälfe aastriiche
... that we the children$_{Acc}$ Hans$_{Dat}$ house$_{Acc}$ let help paint
'... that we let the children help Hans paint the house'

Why CFG is not enough (3)

Swiss German

(3) ... das mer em Hans es huus hälfed aastriiche
... that we Hans_{Dat} house_{Acc} helped paint
'... that we helped Hans paint the house'

(4) ... das mer d'chind em Hans es huus lönd hälfe aastriiche
... that we $\text{the children}_{Acc}$ Hans_{Dat} house_{Acc} let help paint
'... that we let the children help Hans paint the house'

- Swiss German uses case marking and displays cross-serial dependencies.
- [Shieber, 1985] shows that Swiss German is not context-free.

Why CFG is not enough (3)

A formalism that can generate cross-serial dependencies can also generate the copy language $\{ww \mid w \in \{a, b\}^*\}$.

Why CFG is not enough (3)

A formalism that can generate cross-serial dependencies can also generate the copy language $\{ww \mid w \in \{a, b\}^*\}$.

The copy language is not context-free.

Why CFG is not enough (3)

A formalism that can generate cross-serial dependencies can also generate the copy language $\{ww \mid w \in \{a, b\}^*\}$.

The copy language is not context-free.

$$\Downarrow$$

We need extensions of CFG in order to describe all NL phenomena!

CFG: Mild context-sensitivity (1)

Idea [Joshi, 1985]: characterize the amount of context-sensitivity necessary for natural languages.

Mildly context-sensitive formalisms

CFG: Mild context-sensitivity (1)

Idea [Joshi, 1985]: characterize the amount of context-sensitivity necessary for natural languages.

Mildly context-sensitive formalisms

1. generate (at least) all CFLs,

CFG: Mild context-sensitivity (1)

Idea [Joshi, 1985]: characterize the amount of context-sensitivity necessary for natural languages.

Mildly context-sensitive formalisms

1. generate (at least) all CFLs,
2. can describe a limited amount of cross-serial dependencies (there is a $n \geq 2$ up to which the formalism can generate all string languages $\{w^n \mid w \in T^*\}$),

Idea [Joshi, 1985]: characterize the amount of context-sensitivity necessary for natural languages.

Mildly context-sensitive formalisms

1. generate (at least) all CFLs,
2. can describe a limited amount of cross-serial dependencies (there is a $n \geq 2$ up to which the formalism can generate all string languages $\{w^n \mid w \in T^*\}$),
3. are polynomially parsable, and

CFG: Mild context-sensitivity (1)

Idea [Joshi, 1985]: characterize the amount of context-sensitivity necessary for natural languages.

Mildly context-sensitive formalisms

1. generate (at least) all CFLs,
2. can describe a limited amount of cross-serial dependencies (there is a $n \geq 2$ up to which the formalism can generate all string languages $\{w^n \mid w \in T^*\}$),
3. are polynomially parsable, and
4. their string languages are of constant growth. (the length of the words generated by the grammar grows in a linear way, e.g., $\{a^{2^n} \mid n \geq 0\}$ does not have that property)

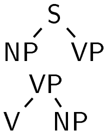
TSG: Definition (1)

- Elements of a CFG represent very small syntactic trees.

- Elements of a CFG represent very small syntactic trees.

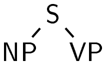
$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$



- Elements of a CFG represent very small syntactic trees.

S $\rightarrow$ NP VP



VP $\rightarrow$ V NP

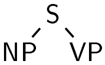
- From a linguistic point of view, we would rather have entire constructions as elementary building blocks.

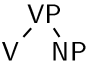
TSG: Definition (1)

- Elements of a CFG represent very small syntactic trees.

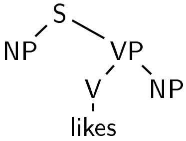
S → NP VP

VP → V NP





- From a linguistic point of view, we would rather have entire constructions as elementary building blocks.



TSG: Definition (2)

A Tree Substitution Grammar (TSG) is a set of finite labeled trees called syntactic trees which have

- internal nodes labeled with non-terminals, and
- leaves labeled either with terminals or non-terminals.

TSG: Definition (2)

A Tree Substitution Grammar (TSG) is a set of finite labeled trees called syntactic trees which have

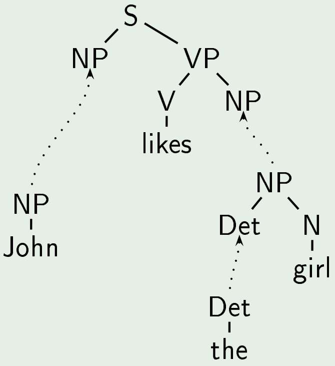
- internal nodes labeled with non-terminals, and
- leaves labeled either with terminals or non-terminals.

We build larger trees by substitution:

- Pick a non-terminal leaf (substitution node)
- Replace it with a tree the root node of which has the same label

TSG: Definition (3)

Substitution example



TSG: Definition (4)

Definition (Tree Substitution Grammar)

A Tree Substitution Grammar (TSG) is a tuple $G = \langle N, T, S, I \rangle$ where

- N, T *are disjoint alphabets of non-terminal and terminal symbols,*
- $S \in N$ *is the start symbol,*
- I *is a finite set of syntactic trees with labels from* N *and* T.

TSG: Definition (4)

Definition (Tree Substitution Grammar)

A Tree Substitution Grammar (TSG) is a tuple $G = \langle N, T, S, I \rangle$ *where*

- N, T *are disjoint alphabets of non-terminal and terminal symbols,*
- $S \in N$ *is the start symbol,*
- I *is a finite set of syntactic trees with labels from* N *and* T.

Every tree in I *is called an elementary tree.*

TSG: Definition (4)

Definition (Tree Substitution Grammar)

A Tree Substitution Grammar *(TSG) is a tuple* $G = \langle N, T, S, I \rangle$ *where*

- N, T *are disjoint alphabets of non-terminal and terminal symbols,*
- $S \in N$ *is the start symbol,*
- I *is a finite set of syntactic trees with labels from* N *and* T*.*

Every tree in I *is called an* elementary tree*.*
G *is called* lexicalized *if every tree in* I *has at least one leaf with a label from* T*.*

TSG: Definition (5)

TSG derivation step

- select a node with a non-terminal label A,
- pick a fresh instance of an elementary tree with root label A from the grammar,
- and substitute the node for the new tree.

Definition (TSG language)

Let $G = \langle N, T, S, I \rangle$ *be a TSG.*

1. *We call a tree* γ *that can be derived from an instance of an elementary tree* $\gamma_e \in I$ *a derived tree in* G.

Definition (TSG language)

Let $G = \langle N, T, S, I \rangle$ be a TSG.

1. *We call a tree γ that can be derived from an instance of an elementary tree $\gamma_e \in I$ a derived tree in G.*
2. *The tree language $L_T(G)$ of G is the set of all derived trees γ in G with root label S and only terminal leaf labels.*

TSG: Definition (6)

Definition (TSG language)

Let $G = \langle N, T, S, I \rangle$ *be a TSG.*

1. *We call a tree* γ *that can be derived from an instance of an elementary tree* $\gamma_e \in I$ *a* ***derived tree*** *in* G.
2. *The* ***tree language*** $L_T(G)$ *of* G *is the set of all derived trees* γ *in* G *with root label* S *and only terminal leaf labels.*
3. *For every tree* γ *with* $t_1, \dots, t_n$ *being the labels of the leaves in* γ *ordered from left to right, we define* $yield(\gamma) = t_1 \dots t_n$.

Definition (TSG language)

Let $G = \langle N, T, S, I \rangle$ be a TSG.

1. *We call a tree γ that can be derived from an instance of an elementary tree $\gamma_e \in I$ a derived tree in G.*
2. *The tree language $L_T(G)$ of G is the set of all derived trees γ in G with root label S and only terminal leaf labels.*
3. *For every tree γ with $t_1, \ldots, t_n$ being the labels of the leaves in γ ordered from left to right, we define $yield(\gamma) = t_1 \ldots t_n$.*
4. *The string language of G is $\{w \mid \text{there is a } \gamma \in L_T(G) \text{ such that } w = yield(\gamma)\}$.*

TSG: Properties (1)

In spite of the larger domains of locality, the following holds:

Proposition (Equivalence of CFG and TSG)

CFG and TSG are weakly equivalent. Furthermore, except for some relabeling of the nodes, they are even strongly equivalent.

TSG: Properties (2)

CFG $\Rightarrow$ TSG

Every CFG can be immediately written as a TSG with every production being understood as a tree with a single root and a daughter for every righthand side symbol

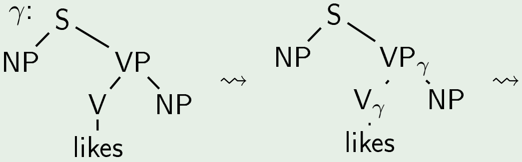
TSG: Properties (2)

CFG ⇒ TSG

Every CFG can be immediately written as a TSG with every production being understood as a tree with a single root and a daughter for every righthand side symbol

TSG ⇒ CFG

In order to construct an equivalent CFG for a given TSG, we have to encode the dependencies between nodes from the same tree within the non-terminal symbols.



$$\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP}_\gamma$$
$$\mathrm{VP}_\gamma \rightarrow \mathrm{V}_\gamma\ \mathrm{NP}$$
$$\mathrm{V}_\gamma \rightarrow \text{likes}$$

TSG: Properties (4)

- TSGs are almost strongly equivalent to CFGs
- Nevertheless they offer an extended domain of locality

$\Longrightarrow$ They capture more generalizations than CFGs!

TSG: Properties (4)

- TSGs are almost strongly equivalent to CFGs
- Nevertheless they offer an extended domain of locality

$\Longrightarrow$ They capture more generalizations than CFGs!

- TSGs are used in the context of data-oriented parsing (DOP) [Bod, 1995].
- Lexicalized TSGs can be extracted from treebanks and used for probabilistic parsing [Post and Gildea, 2009].
- [Cohn et al., 2009] also induce Probabilistic Tree Substitution Grammars from treebanks and use them successfully for parsing.

Adjunction (1)

Tree Adjoining Grammars (TAG)

[Joshi et al., 1975, Joshi and Schabes, 1997]:

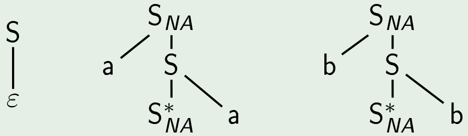
- Tree-rewriting grammar.
- Extension of CFG that allows to replace not only leaves but also internal nodes with new trees.
- Can generate the copy language.

Tree Adjoining Grammars (TAG)

[Joshi et al., 1975, Joshi and Schabes, 1997]:

- Tree-rewriting grammar.
- Extension of CFG that allows to replace not only leaves but also internal nodes with new trees.
- Can generate the copy language.

TAG for the copy language

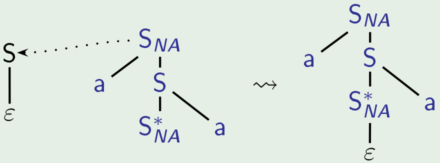


Adjunction (2)

TAG derivation of *abab*

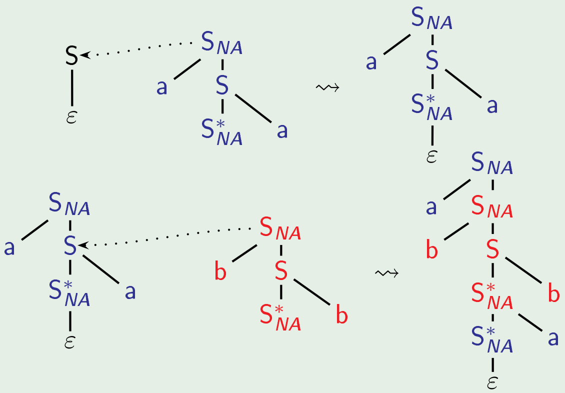
Adjunction (2)

TAG derivation of *abab*



Adjunction (2)

TAG derivation of *abab*



Bod, R. (1995).
Enriching Linguistics with Statistics: Performance Models of Natural Language.
Number 1995-14 in University of Amsterdam ILLC Dissertation Series. Academische Pers, Amsterdam, The Netherlands.

Cohn, T., Goldwater, S., and Blunsom, P. (2009).
Inducing compact but accurate tree-substitution grammars.
In Proceedings of HLT-NAACL 2009, pages 548–556, Boulder, Colorado.

Hopcroft, J. E. and Ullman, J. D. (1979).
Introduction to automata theory, languages and computation.
Addison Wesley.

Joshi, A. K. (1985).
Tree adjoining grammars: How much contextsensitivity is required ro provide reasonable structural descriptions?
In Dowty, D., Karttunen, L., and Zwicky, A., editors, Natural Language Parsing, pages 206–250. Cambridge University Press.

Joshi, A. K., Levy, L. S., and Takahashi, M. (1975).
Tree Adjunct Grammars.
Journal of Computer and System Science, 10:136–163.

Joshi, A. K. and Schabes, Y. (1997).
Tree-Adjoning Grammars.
In Rozenberg, G. and Salomaa, A., editors, Handbook of Formal Languages, pages 69–123. Springer, Berlin.

Post, M. and Gildea, D. (2009).
Bayesian learning of a tree substitution grammar.
In Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 45–48, Suntec, Singapore.

Shieber, S. M. (1985).
Evidence against the context-freeness of natural language.
Linguistics and Philosophy, 8:333–343.