Language modeling with tree-adjoining grammars

Day 1

Kata Balogh & Simon Petitjean

(*Heinrich-Heine-Universität Düsseldorf*)

ESSLLI 2023

University of Ljubljana, 31 July – 4 August, 2023

erc TreeGraSP

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Funded by DFG Deutsche Forschungsgemeinschaft German Research Foundation

What this course is about

Language modeling with Tree-Adjoining Grammars

- language modeling → trying to implement syntactic theories
  - implement^1: general concepts → mathematical objects
  - implement^2: paper & pencil → electronic resource
- Why implementation? ⇒ day 3 part 2

> *As is frequently pointed out but cannot be overemphasized, an important goal of formalization in linguistics is to enable subsequent researchers to see the defects of an analysis as clearly as its merits; only then can progress be made efficiently.*
>
> [Dowty 1979:322]

What this course is not about

Details of …

- formal language theory [Hopcroft, Motwani & Ullmann 2006]
  (ESSLLI 2019 course: https://user.phil.hhu.de/balogh/esslli-2019-course/)
- parsing with mildly context-sensitive formalisms
  (LCFRS, 2-MCFG, 2-ACG) [Kallmeyer 2010]

…However, this is highly relevant for motivating TAG!

- complexity of a language
  - ⇒ determined by the weakest formal grammar that generates it
- expressive power of the formalism
  - ⇒ TAG: The formalism is part of the theory, so let's try to make it both convenient and minimally expressive!

Schedule

Schedule

- **Mon:** motivation & basic (L)TAG
- **Tue:** linguistic applications and using (L)TAG: syntax
- **Wed:**
  - linguistic applications and using (L)TAG: semantics
  - introduction to grammar engineering and XMG
- **Thu:** grammar implementation with XMG
- **Fri:** parsing TAG

Lecturers

- lecturers:
  - Kata Balogh (`balogh@hhu.de`)
  - Simon Petitjean (`petitjean@phil.hhu.de`)
- course page:
  - `https://tinyurl.com/26mm4bum`

Why working with TAG? (in a nutshell)

- formal complexity of natural languages → gain insights into
  - ⇒ the general structure of natural language
  - ⇒ the general human language capacity
  - ⇒ the adequacy of grammar formalisms
  - ⇒ lower bound of the computational complexity of NLP tasks

Hypothesis of the adequacy of expressive power

TAG exactly provides the expressive power needed to treat NL.

Expressive power in terms of a specific generative capacity:

- weak generative capacity → to generate **string languages**
- strong generative capacity → to generate **tree languages**
- derivational generative capacity

Grammar Formalisms

Aim: find an adequate formal system for natural language analysis

- mathematically concise representation of a grammar theory
- a formal system for linguistic analyses

A **formal grammar** (N, T, S, R) is

- Type 0 or unrestricted (phrase structure) grammar iff every production is of the form $\alpha \rightarrow \beta$ with $\alpha \in (N \cup T)^* \setminus T^*$ and $\beta \in (N \cup T)^*$;
  generates a recursively enumerable language (RE).
- Type 1 or context-sensitive grammar iff every production is of the form $\gamma A \delta \rightarrow \gamma \beta \delta$ with $\gamma, \delta, \beta \in (N \cup T)^*$, $A \in N$ and $\beta \neq \epsilon$;
  generates a context-sensitive language (CS).
- Type 2 or context-free grammar iff every production is of the form $A \rightarrow \beta$ with $A \in N$ and $\beta \in (N \cup T)^* \setminus \{\epsilon\}$;
  generates a context-free language (CF).
- Type 3 or right-linear grammar iff every production is of the form $A \rightarrow \beta B$ or $A \rightarrow \beta$ with $A, B \in N$ and $\beta \in T^* \setminus \{\epsilon\}$;
  generates a regular language (REG).

Why working with TAG? Formal reasons

How much expressive power do we need to treat NL?

(FSA = finite state automaton, PDA = push-down automaton, EPDA = embedded push-down automaton, LBA = linear bounded automaton, TG = transformational grammar, TM = Turing Machine)

type 0: recursively enumerable

HPSG, TG, TM — $a^{f(n)}$

type 1: context-sensitive

LFG, LBA — a^{2^n}, $a^n b^n c^n \ldots$, W^k

mildly context-sensitive

TAG, EPDA — $a^n b^m c^n d^m$, WW

type 2: context-free

CFG, PDA — $a^n b^m c^m d^n$, WW^R

type 3: regular

FSA — $a^n b^m c^k d^l$

Chomsky(-Schützenberger) hierarchy [Chomsky-Schuetzenberger 1963]

NL is **mildly** context-sensitive

[Joshi 1985]

- $\supset$ CFL
- cross-serial dep.
- semi-linear
- in PTIME

Chomsky-hierarchy: overview

Languages as problems:
"Can we decide for every word whether it belongs to L?"

type	grammar	rules	word problem
RE	phrase structure	$\alpha \longrightarrow \beta$	undecidable
CS	context-sensitive	$\gamma A \delta \longrightarrow \gamma \beta \delta$	exponential
CF	context-free	$A \longrightarrow \beta$	cubic
REG	right-linear	$A \longrightarrow aB \vert b$	linear

For Type 1-3 languages a rule $S \rightarrow \epsilon$ is allowed if S does not occur in any rule's right-hand side.

Mild context sensitivity

- for natural languages context-free grammars are just not ‘enough’
  - **expressivity challenge**: cannot describe all NL phenomena
    - cross-serial dependencies ($a^n b^m c^n d^m$); Schwyzerdütsch
    - duplication (yy); Bambara (spoken in Mali)
    - multiple agreement ($a^n b^n c^n$); Bantu languages
  - **low descriptive power**: problems with certain linguistic phenomena
    e.g. subcategorization, number agreement, case marking
  - **only weak-lexicalization** possible
- natural languages are almost context-free

mildly context sensitive languages

$$\text{RL} \subset \text{CFL} \subset \textbf{MCSL} \subset \text{CSL} \subset \text{RE}$$

[Joshi, 1985]

- for natural languages we need grammars, that are somewhat richer than context-free grammars, but more restricted than context-sensitive grammars

Mildl context sensitivity

- Joshi (1985): characterize the amount of context-sensitivity needed for NL
- mildly context sensitive formalisms are such that they
  - generate at least all CFs
  - can describe a limited amount of cross-serial dependencies
    (there is an $n \geq 2$ up to which the formalism can generate all string languages $\{w^n | w \in \Sigma^*\}$)
  - are polynomially parsable
  - their string languages are of constant growth
    (the length of the words generated by the grammar grows in a linear way)

Limits of CFG: expressivity challenge

- German: **nested dependency** (subordinate clauses)

(1) Jan sagte daß er die Kinder dem Hans das Haus streichen helfen ließ.
John said that he the children the Hans the house paint help let.
'John said that he let the children to help Hans to paint the house.'

- Schwyzerdütsch: **cross-serial dependency**

(2) Jan säit das mer d'chind em Hans es huus lönd hälfe aastriiche.
John said that we children.acc the Hans.dat the house.acc let help paint.
'John said that we let the children to help Hans to paint the house.'

(3) *mer d'chind de Hans es huus lönd hälfe aastriiche.
we children.acc the Hans.acc the house.acc let help paint.

Limits of CFG: expressivity challenge

Proof by Shieber

[Shieber 1985: 334-337]

- series of NPs followed by series of Vs
- raising verb can occur in between
  Jan säit das mer d'chind em Hans es huus lönd hälfe aastriiche.

(4) ... mer d'chind em Hans es huus haend wele laa hälfe aastriiche.
... we children.acc the Hans.dat the house.acc have wanted let help paint.
'... that we have wanted to let the children to help Hans to paint the house.'

- Jan säit das mer NP* es huus haend wele VP* aastriiche
- homomorphism f:

f(d'chind) = a	f(em Hans) = b	f(laa) = c	f(hälfe) = d
f(Jan säit das mer) = w	f(es huus haend wele) = x	f(aastriiche) = y	$f(s) = z$ otherwise

- $f(\text{Schwyzerdütsch}) \cap wa^*b^*xc^*d^*y = wa^mb^nxc^md^ny$
  - CFLs are closed under intersection with regular languages: $L1_{CF} \cap L2_{REG} = L3_{CF}$
  - $wa^*b^*xc^*d^*y$ is regular
  - by Pumping Lemma: $wa^mb^nxc^md^ny$ is not context-free
- ⇒ Schwyzerdütsch is not context-free

Limits of CFG: low descriptive power

Take a simple CFG

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$

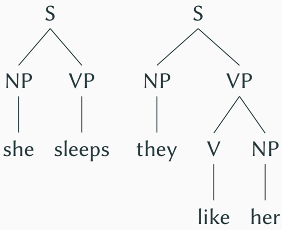
P = { S → NP VP, VP → V NP | V, V → *likes* | *like* | *sleeps*, NP → *she* | *her* | *they* }

Example derivations:

S → NP VP → she VP → she V → she sleeps

S → NP VP → they VP → they V NP → they like NP → they like her

Example derivation history:



Limits of CFG: low descriptive power

- subcategorization / argument selection

  (1) She sleeps. / She likes her. / *She likes.

  S⇒ NP VP⇒ Joe VP ⇒ Joe V ⇒ Joe sleeps

  S⇒ NP VP⇒ Joe VP ⇒ Joe V ⇒ Joe likes

- number agreement

  (2) They like her. / *They likes her.

- case marking

  (3) She likes her. / *She likes they.

- encode necessary information in the non-terminals?

Limits of CFG: low descriptive power

- extend for number agreement, argument selection (transitive vs. non-transitive) and case marking

  $S \rightarrow NP_{3sg/nom}\ VP_{3sg/itr}$, $S \rightarrow NP_{3pl/nom}\ VP_{3pl/itr}$,
  $S \rightarrow NP_{3sg/nom}\ VP_{3sg/tr}$, $S \rightarrow NP_{3pl/nom}\ VP_{3pl/tr}$,
  $VP_{3sg/tr} \rightarrow V_{3sg/tr}\ NP_{3sg/acc}$, $VP_{3pl/tr} \rightarrow V_{3pl/tr}\ NP_{3sg/acc}$,
  $VP_{3sg/itr} \rightarrow V_{3sg/itr}$, $VP_{3pl/itr} \rightarrow V_{3pl/itr}$,
  $NP_{3sg/nom} \rightarrow$ she, $NP_{3sg/acc} \rightarrow$ her, $NP_{3pl/nom} \rightarrow$ policemen,
  $V_{3sg/itr} \rightarrow$ sleeps, $V_{3pl/itr} \rightarrow$ sleep, $V_{3sg/tr} \rightarrow$ likes, $V_{3pl/tr} \rightarrow$ like
- every possible combination of arguments selection (e.g. transitive/non-transitive), number agreement and case marking must have a separate non-terminal and a separate re-write rule
- grammar writing is quite error prone (and boring)
- linguistic generalizations are difficult to express, e.g.
  - subject and verb must have the same number
  - the object of a transitive verb must be in accusative case
- solution: feature structures, unification, underspecification (see later)

Limits of CFG: lexicalization

Lexicalized grammar

A lexicalized grammar consists of:

(i) a finite set of structures each associated with a lexical item (anchor),

(ii) operation(s) for composing these structures.

Lexicalization

A formalism F can be lexicalized by another formalism F',
if for any finitely ambiguous grammar G in F there is a grammar G' in F',
such that (i) G' is a lexicalized grammar; and
(ii) G and G' generate the same set.

weak vs. strong lexicalization

- weak lexicalization: preserve the string language
- strong lexicalization: preserve the tree structure

Limits of CFG: lexicalization

- **Linguistically interesting:**
  - syntactic properties of lexical items can be accounted for more directly
  - each lexical item comes with the possibility of certain partial syntactic constructions
- **Formally interesting:**
  - a finite lexicalized grammar provides finitely many analyses for each string (finitely ambiguous)
- **Computationally interesting:**
  - the search space during parsing can be delimited (grammar filtering)
  - use of corpora in NLP

Lexicalization of CFG's

- lexicalize CFGs:
  - recursive ($X \Rightarrow^* X$) and elementary ($X \rightarrow X$) rules are disallowed
  - each rule must consist at least one terminal on the RHS
- lexicalized CFG $\rightsquigarrow$ e.g. Greibach normal-form: $A \rightarrow aX$ or $A \rightarrow a$
  ($a \in V_T$; $A \in V_N$; $X \in (V_N)^*$) [Greibach, 1965]

Question:

Can CFGs be strongly lexicalized (= the set of trees are preserved)?

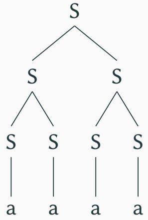
Answer:

No. Only weak lexicalization possible (= same string language).

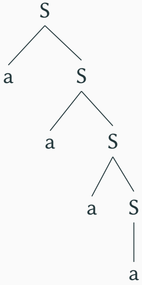
Lexicalization of CFG’s

- example:
    - a CFG G: $S \to SS$, $S \to a$
    - lexicalize $G \Rightarrow G'$: $S \to aS$, $S \to a$
- same string language, but not the same tree set
- only weak lexicalization possible

by G:



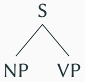
by G':



G cannot be strongly lexicalized with some finite CFG, e.g., G'.

From CFG to TAG: Tree Substitution Grammar (TSG)

- a CFG rule corresponds to a tree
  - LHS as the root node / RHS as the daughter nodes
  - e.g., S → NP VP



- tree rewriting
- **substitution:** replace a non-terminal leaf with a tree

- grammar on trees + substitution → **Tree Substitution Grammar**

  A TSG is a quadruple $TSG = \langle \Sigma, N, S, I \rangle$, where

  Σ is a set of terminal symbols;

  N is a set of non-terminal symbols;

  $S \in N$ is a distinguished non-terminal symbol;

  I is a finite set of initial trees.

From CFG to TAG: Tree Substitution Grammar

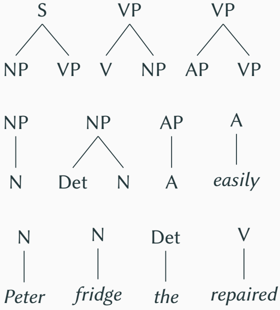
$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow V\ NP \mid AP\ VP \\
NP &\rightarrow N \mid Det\ N \\
AP &\rightarrow A \\
N &\rightarrow Peter \mid fridge \\
Det &\rightarrow the \\
A &\rightarrow easily \\
V &\rightarrow repaired
\end{aligned}
$$

$\}$

$\approx$

$G_{\text{TSG}} = \langle N, T, S, I \rangle$
$I = \{$

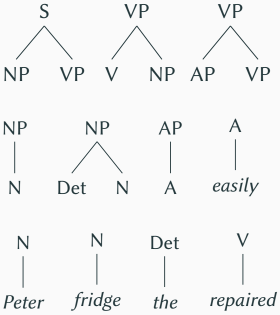


$\}$

From CFG to TAG: Tree Substitution Grammar

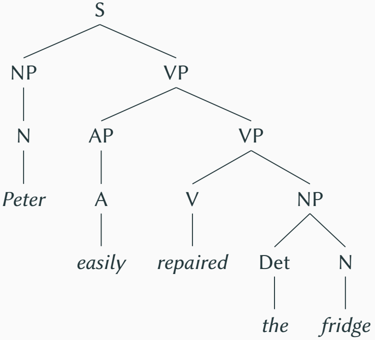
$G_{TSG} = \langle N, T, S, I \rangle$

$I = \{$



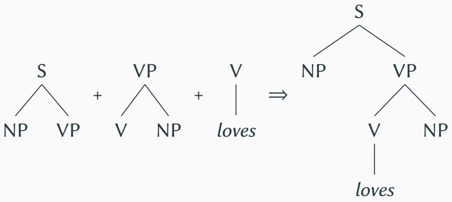
$\}$

Example derivation:



From CFG to TAG: Tree Substitution Grammar

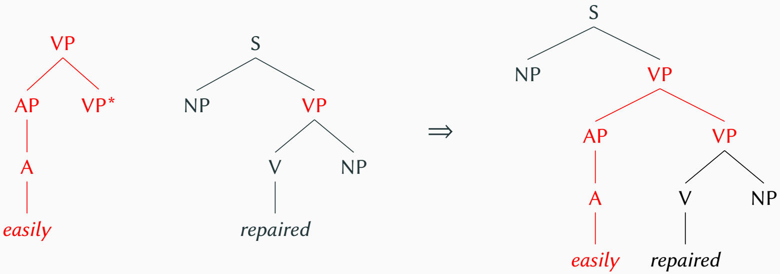
- TSG is weakly equivalent to CFG (same string language).
- Still no strong lexicalization of CFG ⇒ not possible to find a strongly equivalent (same tree language) lexicalized TSG for each CFG.
- TSG is not powerful enough to describe cross-serial dependencies.
- TSGs can capture more generalizations than CFGs.
- TSG offers extended domain of locality.



- Some applications of TSG:
  - in data-oriented parsing (DOP) (Bod 1995),
  - Lexicalized TSGs can be extracted from treebanks and used for probabilistic parsing (Post & Gildea 2009).

TSG + Adjunction

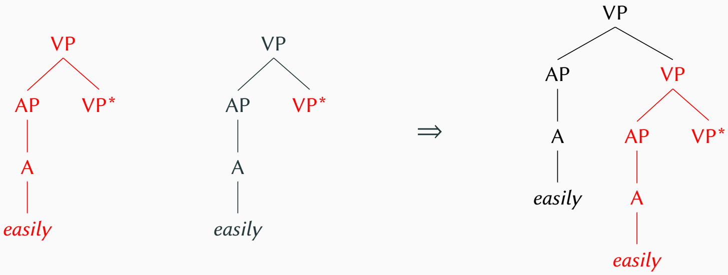
- lexicalization of CFG in a linguistically meaningful way
- TSG: still no strong lexicalization of CFG, no cross-serial dependencies etc.
- add **adjunction:**
    - replace a **non-terminal node** with an “auxiliary” tree
    - put the subtree of the replaced node under the **footnode** (*)



TSG + Adjunction

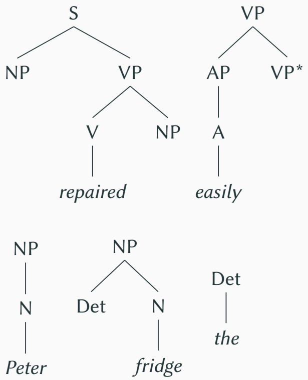
⇒ Adjunction at footnodes causes spurious ambiguities in derivations.

⇒ Therefore, this is usually forbidden.

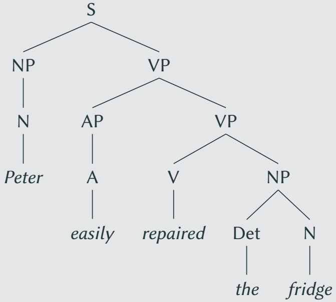


From CFG to TAG: Example with adjunction

- tree rewriting
- **Substitution:** replace a non-terminal **leaf** with a tree
- **Adjunction:** replace a non-terminal **node** with an "auxiliary" tree



Example derivation:



From CFG to TAG: Restrictions on adjunction (I)

Restrictions on the shape of auxiliary trees:

- The root node and the footnode must carry the same non-terminal.

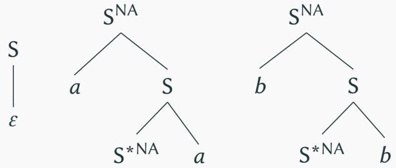
Specific adjunction constraints on target nodes:

- obligatory adjunction (OA): true/false
- null adjunction (NA): no adjoinable auxiliary tree
- selective adjunction (SA): a nonempty set of adjoinable auxiliary trees

Adjunction constraints are essential in generating non-context-free languages (e.g., the copy language $\{ww | w \in \{a, b\}^*\}$)!

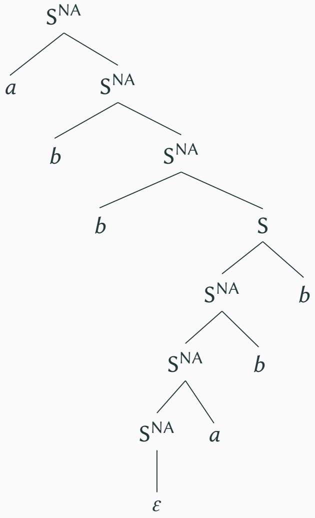
From CFG to TAG: Restrictions on adjunction

Example grammar for the copy language $\{ww | w \in \{a, b\}^*\}$:



$\Rightarrow$ TAG = TSG + adjunction + adjunction constraints

Example: derivation of *abbabb*



Tree-Adjoining Grammar

Tree Adjoining Grammar (TAG)

A Tree Adjoining Grammar is a tuple $G = \langle N, T, I, A, O, C \rangle$:

T and N are disjoint alphabets of terminals (T) and non-terminals (N),

I is a finite set of **intial trees**, and

A is a finite set of **auxiliary trees**.

$O : \{v \mid v \text{ is a node in a tree in } I \cup A\} \longrightarrow \{1, 0\}$ is a function, and

$C : \{v \mid v \text{ is a node in a tree in } I \cup A\} \longrightarrow \mathcal{P}(A)$ is a function.

The trees in $I \cup A$ are called **elementary trees**.

Let v be a node in $I \cup A$:

- **obligatory adjunction (OA)**: $O(v) = 1$
- **null adjunction (NA)**: $O(v) = 0$ and $C(v) = \emptyset$
- **selective adjunction (SA)**: $O(v) = 0$ and $C(v) \neq \emptyset$ and $C(v) \neq A$

Tree-Adjoining Grammar

TAG is **mildly context-sensitive** (MCS; Joshi 1985)

- generates the context-free languages
- generates cross-serial dependencies (i.e. *ww*)
- constant growth (or semi linear, no a^{2^n})
- polynomial time parsing ($\mathcal{O}(n^6)$)

[Schabes 1990, Joshi & Schabes 1997, Kallmeyer: 2010]

⇒ expressivity challenge ✓

TAG can **strongly lexicalize** finitely ambiguous CFG.

[Schabes 1990, Joshi & Schabes 1997]

(formally, computationally and linguistically interesting (see slide 17))
⇒ lexicalization ✓

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination, …
  - **Generalizations:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface

    ⇒ descriptive power ✓
  - intuitive implementation
- computationally adequate:
  - explicit/formalized
  - decidable, maybe even tractable
  - bidirectional
- psycholinguistically adequate:
  - strictly incremental derivations
  - correct predictions wrt. processing complexity

References

Bod, Rens. 2009. From exemplar to grammar: A probabilistic analogy-based model of language learning. *Cognitive Science* 33(5). 752–793.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2. 113–124.

Chomsky, Noam. 1957. *Syntactic structures.* Den Haag: Mouton.

Chomsky, Noam and Marcel-Paul Schützenberger. 1963. The algebraic theory of context-free languages. In Braffort, P. and D. Hirschberg (eds). *Computer programming and formal systems* (Studies in Logic and the Foundations of Mathematics 35). Elsevier. 118–161.

Dowty, David R. 1979. *Word meaning and Montague Grammar.* Reprinted 1991 by Kluwer Academic Publishers. Dordrecht: D. Reidel Publishing Company.

Greibach, Sheila A. 1965. A New Normal-Form Theorem for Context-Free Phrase Structure Grammars. *Journal of the ACM.* 12(1). 42–52.

Hopcroft, John E., Rajeev Motwani and Jeffrey D. Ullmann. 2006. *Introduction to Automata Theory, Languages, and Computation.* 3rd Edition. Addison-Wesley.

References

Joshi, Aravind K. 1985. Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions. In Dowty, D., L. Karttunen and A. Zwicky (eds). *Natural language parsing*. Cambridge University Press. 206–250.

Joshi, Aravind K. and Yves Schabes. 1997. Tree-Adjoining Grammars. In Rozenberg, G. and A. Salomaa (eds). *Handbook of formal languages* Vol. 3. Berlin, New York: Springer. 69–124.

Kallmeyer, Laura. 2010. *Parsing beyond Context-Free Grammars.* Berlin: Springer.

Post, Matt and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference*. Short Papers. Suntec: Singapore. ACL. 45–48.

Schabes, Yves. 1990. *Mathematical and computational aspects of lexicalized grammars.* PhD thesis: University of Pennsylvania.

Shieber, Stuart. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8. 333–343.