

Part I: Lexicalized Tree Adjoining Grammars (LTAG)

2 Extending CFG for natural languages

Roughly, two kinds of grammar formalisms for natural languages can be distinguished:

1. Formalisms that have much more expressive power than needed for natural languages. Restrictions come then from specific applications and in these specific applications polynomial parsing is possible. But in general these formalisms are too powerful to be computationally tractable. Examples: transformational grammar, LFG, HPSG.
2. Formalisms with a limited expressive power that guarantees that any grammar in this formalism, no matter for which specific application, is computationally tractable, i.e., allows parsing in an acceptable (usually polynomial) time. Examples: GPSG, Constrained categorial grammar, TAG.

We are concerned with the second type of grammar formalisms. For this approach, a central issue is to determine (with respect to its complexity and generative capacity) the class of languages/grammar formalisms natural languages belong to. Still an open question.

Weak generative capacity (WGC) of a formalism: the set of string languages it generates. *Strong generative capacity (SGC)* of a formalism: the set of tree languages it generates.

2.1 Natural languages are not context-free

CFGs allow to describe a large range of natural language data (\rightarrow GPSG). In the 80's many people were interested in the question whether CFGs are powerful enough to describe all natural language constructions. This is not the case:

CFGs cannot describe cross-serial dependencies in Dutch Bresnan et al. (1982) and Swiss German Shieber (1985) in an adequate way.

The Dutch data:

- (3) ... dat Jan de kinderen zag zwemmen
... that Jan the children saw swim



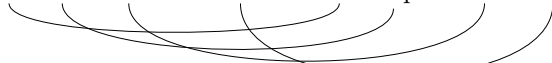
‘... that Jan saw the children swim’

- (4) ... dat Jan Piet de kinderen zag helpen zwemmen
... that Jan Piet the children saw help swim



‘... that Jan saw Piet help the children swim’

- (5) ... dat Jan Piet Marie de kinderen zag helpen laten zwemmen
... that Jan Piet Marie the children saw help make swim



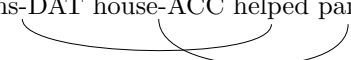
‘... that Jan saw Piet help Marie make the children swim’

In principle an unbounded number of crossed dependencies is possible. However, except for the first and last verb any permutation of the NPs and the verbs is grammatical as well (even though with a completely different dependency structure since the dependencies are always cross-serial). Therefore, the string language of Dutch cross-serial dependencies amounts roughly to $\{n^k v^k \mid k > 0\}$ which is a context-free language.

Bresnan et al. (1982) argue that the SGC of CFGs is too limited for the Dutch examples. Weakness of the argument: an argument about syntactic structure makes always certain theoretical stipulations. Although it is very probable, it does not absolutely prove that, even using different syntactic theories, there is no context-free analysis for the Dutch examples. It only shows that the syntactic structures Bresnan et al. think the appropriate ones cannot be obtained with a CFG.

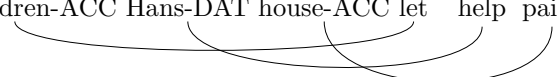
Shieber's argument about Swiss German cross-serial dependencies is more convincing since it relies only on the string language, i.e., it concerns the WGC of CFGs. The Swiss German data:

(6) ... das mer em Hans es huus hälfed aastrüiche
 ... that we Hans-DAT house-ACC helped paint



'... that we helped Hans paint the house'

(7) ... das mer d'chind em Hans es huus lönd hälfe aastrüiche
 ... that we 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.

Proposition 5 *The language L of Swiss German is not context-free (Shieber 1985).*

Argumentation goes as follows: Assume that L is context-free. Then the intersection of a regular language with the image of L under a homomorphism must be context-free as well. Find a homomorphism and a regular language such that the result is a non context-free language. Contradiction.

Further possible sentence:

(8) ... das mer d'chind em Hans es huus haend wele laa hälfe aastrüiche
 ... that we the children-ACC Hans-DAT house-ACC have wanted let help paint
 '... that we have wanted to let the children help Hans paint the house'

Swiss German allows constructions of the form $(Jan säit)$ ('Jan says') $das mer (d'chind)^i (em Hans)^j es huus haend wele (laa)^i (hälfe)^j aastrüiche$. In these constructions the number of accusative NPs $d'chind$ must equal the number of verbs (here laa) selecting for an accusative and the number of dative NPs $em Hans$ must equal the number of verbs (here $hälfe$) selecting for a dative object. Furthermore, the order must be the same in the sense that if all accusative NPs precede all dative NPs, then all verbs selecting an accusative must precede all verbs selecting a dative.

Homomorphism f :

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

Intersect $f(L)$ with the regular language $wa^*b^*xc^*d^*y$. The result is $\{wa^ib^jc^id^jy \mid i, j \geq 0\}$, a language that is not context-free.¹ Consequently, the original language L , Swiss German, is not context-free either.

Exercise 1 $L_2 := \{a^n b^n \mid n \geq 0\}$

1. Give a CFG for L_2 with nested dependencies, i.e., such that for each word $a_1 \dots a_n b_1 \dots b_n$ (the subscripts mark the occurrences of the a s and b s respectively) a_i and b_{n+1-i} are added by the same production for all $1 \leq i \leq n$.
2. Show that for L_2 there is no CFG displaying cross-serial dependencies, i.e., no CFG such that for each word $a_1 \dots a_n b_1 \dots b_n$, a_i and b_i are added by the same production for all $1 \leq i \leq n$. (Hint: Assume that such a CFG exists. Then show that in this case a CFG for the copy language $\{ww \mid w \in \{a, b\}^*\}$ exists. Contradiction since this language is not context-free.)

Exercise 2 Similar to Shieber's argument, one can apply first a homomorphism f , then intersect with some regular language, and then apply another homomorphism g in order to reduce the language of Swiss German to the copy language $\{ww \mid w \in \{a, b\}^*\}$. Find the corresponding homomorphisms and the regular language.

2.2 Strong lexicalization of CFGs leads to TAG

Schabes (1990); Joshi and Schabes (1997)

In the following we consider only languages that do not contain the empty word.

Definition 6 (Lexicalized Grammar) A grammar is lexicalized if it consists of

- a finite set of elementary objects of finite size each associated with a non-empty lexical item (called its anchor),
- and an operation/operations for composing these structures that do neither copy nor erase nor restructure unbounded components of their arguments.

The objects might be for instance phrase structure rules as in CFG or trees as in TAG or tree descriptions ("quasi trees") as in D-Tree Substitution Grammar (Rambow et al. 2001).

An elementary structure might contain more than one lexical item. We call the set of its lexical items then a *multicomponent anchor*.

Lexicalized grammars are *finitely ambiguous*, i.e., no sentence of finite length can be analyzed in an infinite number of ways. Consequently the recognition problem for lexicalized grammars is decidable.

Definition 7 (Lexicalization) A formalism F can be strongly (weakly) lexicalized by a formalism F' if for any finitely ambiguous grammar G in F there is a lexicalized grammar G' in F' such that G and G' are strongly (weakly) equivalent.

CFG can be weakly lexicalized by CFG since for each CFG a weakly equivalent lexicalized CFG can be found, namely the one in Greibach Normal Form (GNF) (see Hopcroft and Ullman 1979).²

A context-free derivation step can be considered as a tree substitution: a nonterminal leaf is replaced with a tree of height 1 (one mother node and n daughters). Extending the height of the trees permitted leads to Tree Substitution Grammars:

¹To see that, intersect with regular language $a^*b^*c^*d^*$, leads to $\{a^ib^jc^id^j \mid i, j \geq 0\}$. This language can be shown to be not context-free using the pumping lemma for context-free languages. (The pumping lemma says that from a certain word length n on two parts in a word can be pumped such that the length of the two parts plus the part in between is below or equal to n .)

²A CFG is in Greibach Normal Form if each production is of the form $A \rightarrow ax$ with $A \in N$, $a \in T$, $x \in (N \cup T)^*$.

Definition 8 (Tree Substitution Grammar) A Tree Substitution Grammar (TSG) consists of a quadruple $\langle T, N, I, S \rangle$ such that

- T and N are disjoint alphabets, the terminals and nonterminals
- I is a finite set of initial trees, and
- $S \in N$ is the start symbol.

A finite labeled tree is an initial tree if all internal nodes are labeled by non-terminal symbols and all leaves are labeled by terminal or non-terminal symbols.

In the following all trees we consider are finite labeled trees with labels from $N \cup T$. A tree is *derived* if it has been obtained from an initial tree by replacing some of the non-terminal leaves with initial trees having the same non-terminal as root label. Such operations are called *substitution*.

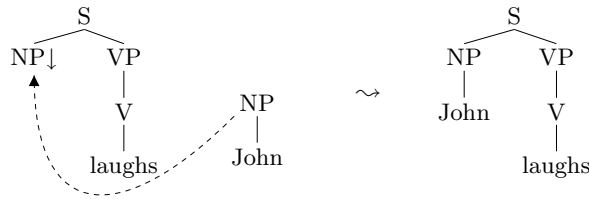


Figure 5: Sample substitution

A tree is *completed* if all leaves are labeled by terminals.³ The leaves with non-terminal labels are called substitution nodes and are marked with a downwards arrow. The tree language $T(G)$ of a TSG G is the set of all completed derived trees that have the root label S . The string language of G is then the set of strings yielded by the trees in the tree language.

TSGs are weakly equivalent to CFGs and each CFG is a TSG.

Proposition 6 CFG cannot be strongly lexicalized by TSG.

Proof: Consider the CFG G with productions $S \rightarrow S S$, $S \rightarrow a$. Assume that there is a strongly equivalent lexicalized TSG G' . Then each tree in the tree language is derived from some initial tree t with a leaf labeled with a such that the path between this leaf and the root has a constant length n . Below this leaf nothing can be added, i.e., each tree derived from t still has a path of length n . Let n_{max} be the maximal path length between root and leaf with label a in the initial trees of G' . Then there is no derived tree in the tree language of G' such that all paths have a length $> n_{max}$. But such trees exist in the tree language of G . Contradiction.

□

Then, trivially, CFGs cannot strongly lexicalize CFGs either.

Problem: TSGs do not permit the distance between two nodes in the same initial tree to increase.

This can be overcome if one allows to replace not only leaves but also internal with new elementary trees. In order to do so, one has to mark the leaf in the new elementary tree where the subtree will be placed that is below the internal node where the operation is performed.

A finite labeled tree is an *auxiliary* tree if all internal nodes are labeled with non-terminals, all leaves with terminals or non-terminals and if there is exactly one leaf marked as *foot node* (with an asterisk) that has the same non-terminal label as the root node of the tree.

³For a formal definition of finite labeled trees, initial and auxiliary trees and the operations substitution and adjunction see Kallmeyer (1999).

An *adjunction* of a new auxiliary tree β at a node n in a tree γ consists of replacing the subtree γ' with root n by β and then replacing the root of γ' by the foot node of β (i.e., putting the excised tree below β).

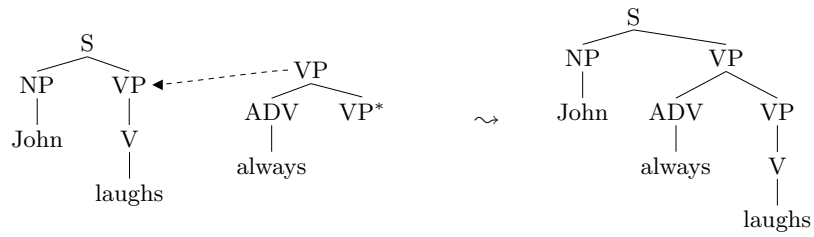


Figure 6: Sample adjunction

The above-mentioned CFG can be lexicalized using a tree grammar with substitution and adjunction (see Fig. 3).

Exercise 3 Consider the following CFG:

$$\begin{array}{ll}
 S \rightarrow NP VP & NP \rightarrow John \\
 VP \rightarrow ADV VP & ADV \rightarrow always \\
 VP \rightarrow V & V \rightarrow laughs
 \end{array}$$

Find a TSG that strongly lexicalizes this grammar.
 Why is this lexicalization not satisfying?