Einführung in die Computerlinguistik

Natural Languages are not Context-Free

Laura Kallmeyer
Heinrich-Heine-Universität Düsseldorf

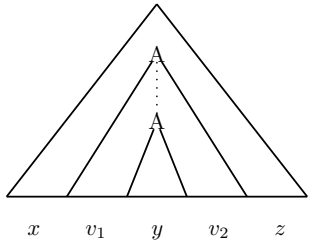
Wintersemester 2011/2012

Overview

1. Formal Properties of CFLs
2. CFG and Natural Languages
3. Cross-serial Dependencies
4. Swiss German is not Context-Free

Formal Properties of CFLs (1)

Observation: In CFG derivation trees of a certain minimal path length there is necessarily a path containing two different nodes with the same non-terminal. Then the part of the derivation tree in between these two nodes can be iterated. This means that the strings yielded by this part are pumped.



Formal Properties of CFLs (2)

Proposition 1 (Pumping lemma for context-free languages)
Let L be a context-free language. Then there is a constant c such that for all $w \in L$ with $|w| \geq c$: $w = xv_1yv_2z$ with

- $|v_1v_2| \geq 1$,
- $|v_1yv_2| \leq c$, *and*
- *for all $i \geq 0$: $xv_1^iyv_2^iz \in L$.*

Formal Properties of CFLs (3)

Two important closure properties of CFLs:

Proposition 2 *Context-free languages are closed under homomorphisms, i.e., for alphabets T_1, T_2 and for every context-free language $L_1 \subset T_1^*$ and every homomorphism $h : T_1^* \to T_2^*$, $h(L_1) = \{h(w) \mid w \in L_1\}$ is a context-free language.*

Proposition 3 *Context-free languages are closed under intersection with regular languages, i.e., for every context-free language L and every regular language L_r, $L \cap L_r$ is a context-free language.*

CFG and Natural Languages

- For a long time there has been a debate about whether CFGs are sufficiently powerful to describe natural languages. Several approaches have used CFGs, oftentimes enriched with some additional mechanism of transformation [Chomsky, 1956] or with features [Gazdar et al., 1985] for natural languages.
- In the 80's Stuart Shieber was able to prove in [Shieber, 1985] that there are natural languages that cannot be generated by a CFG. Before that, [Bresnan et al., 1982] made already a similar argument but their proof is based on the tree structures obtained with CFGs while Shieber argues on the basis of weak generative capacity, i.e., of the string languages.
- The phenomena considered in both papers are *cross-serial dependencies*.

Cross-serial Dependencies (1)

Cross-serial dependencies in Dutch [Bresnan et al., 1982]:

(1) ... dat Jan de kinderen zag zwemmen
... that Jan the children saw swim
'... that Jan saw the children swim'

The colours mark the dependencies between the two verbs and the two NPs: *the children* is an argument of *swim* while *Jan* is an argument of *saw*. The dependency links are in a crossing configuration.

Cross-serial Dependencies (2)

This phenomenon can be iterated:

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

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

Cross-serial Dependencies (3)

- 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 dependencies are not visible on the strings and the string language of Dutch cross-serial dependencies amounts roughly to $\{n^k v^k \mid k > 0\}$ which is a context-free language.

This is different for Swiss German because Swiss German has case marking.

Cross-serial Dependencies (4)

Cross-serial dependencies in Swiss German [Shieber, 1985]:

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

(5) ... 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'

Swiss German

- uses case marking
- and displays cross-serial dependencies.

Swiss German is not Context-Free (1)

Proposition 4 *The language L of Swiss German is not context-free [Shieber, 1985].*

The argumentation of the proof goes as follows:

- We 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.
- We find a particular homomorphism and a regular language such that the result obtained in this way is a non context-free language.
- This is a contradiction to our assumption and, consequently, the assumption does not hold.

Swiss German is not Context-Free (2)

Consider sentences of the following form:

(6) ... das mer d'chind em Hans es huus haend
... that we the children-ACC Hans-DAT house-ACC have

wele laa hälfe aastriiche
wanted let help paint
'... that we have wanted to let the children help Hans paint the house'

The NP verb pairs *d'chind laa* and *em Hans hälfe* both can be iterated.

Swiss German is not Context-Free (3)

With an additional embedding under *Jan säit* ('Jan says') we obtain constructions of the form

(7) Jan säit das mer (d'chind)^{i_1} (em Hans)^{j_1} (d'chind)^{i_2} (em Hans)^{j_2} ... es huus haend wele (laa)^{i_1} (hälfe)^{j_1} (laa)^{i_2} (hälfe)^{j_2} ... aastriiche

where

- the number of accusative NPs *d'chind* must equal the number of verbs (here *laa*) selecting for an accusative,
- the number of dative NPs *em Hans* must equal the number of verbs (here *hälfe*) selecting for a dative object, and
- the order of NPs and verbs 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.

Swiss German is not Context-Free (4)

The following homomorphism f separates the iterated noun phrases and verbs in these examples from the surrounding material:

$$\begin{aligned} f(\text{“d'chind”}) &= a \\ f(\text{“em Hans”}) &= b \\ f(\text{“laa”}) &= c \\ f(\text{“hälfe”}) &= d \\ f(\text{“Jan säit das mer”}) &= w \\ f(\text{“es huus haend wele”}) &= x \\ f(\text{“aastriiche”}) &= y \\ f(s) &= z \quad \text{otherwise} \end{aligned}$$

Swiss German is not Context-Free (5)

The images of the constructions we are interested in under f are of the form wv_1xv_2y where v_1 contains as and bs and v_2 contains cs and ds and if the kth element in v_1 is an a (a b resp.), then the $k$the element in v_2 is a c (a d resp.). All other sentences have a z somewhere in their image under f.

To make sure we concentrate only on the constructions of the described form and only on constructions where the accusative NPs precede the dative NPs, we intersect $f(L)$ with the regular language $wa^*b^*xc^*d^*y$.

$$L' = f(L) \cap wa^*b^*xc^*d^*y = \{wa^ib^jxc^id^jy \mid i, j \geq 0\}$$

Swiss German is not Context-Free (6)

If L is context-free, then L' must be context-free as well.

- Then the image of L' under a homomorphism f' with $f'(w) = f'(x) = f'(y) = \varepsilon$, $f'(a) = a$, $f'(b) = b$, $f'(c) = c$, $f'(d) = d$ is also context-free. This image is

$$f'(L') = L'' = \{a^ib^jc^id^j \mid i, j \geq 0\}$$

- Consequently, L'' satisfies the pumping lemma for context-free languages. Inspecting the word $a^kb^kc^kd^k$ where k is the constant from the pumping lemma, this can be shown to lead to a contradiction.

Consequently, L'' is not context-free, and neither are L' and L.

References

[Bresnan et al., 1982] Bresnan, J., Kaplan, R. M., Peters, S., and Zaenen, A. (1982). Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635. Reprinted in [**?**].

[Chomsky, 1956] Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124.

[Gazdar et al., 1985] Gazdar, G., Klein, E., Pullman, G., and Sag, I. (1985). *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, Massachusetts.

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