Parsing Beyond Context-Free Grammars: Formal Properties of Tree Adjoining Languages

Laura Kallmeyer
Heinrich-Heine-Universität Düsseldorf

Winter 2021/22

Overview

1. **Closure properties**
2. **Pumping lemma**

[Kal10]

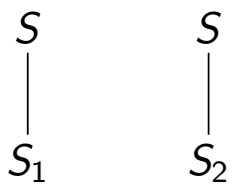
Closure properties (1)

One of the reasons why the TAG formalism is appealing from a formal point of view is the fact that it has nice closure properties [VSJ85, VS87].

Proposition 1

TALs are closed under union.

This can be easily shown as follows: Assume the two sets of non-terminals for the two TALs to be disjoint, S_1 and S_2 being their respective start symbols. Then build a large TAG putting the initial and auxiliary trees from the two grammars together.
Furthermore, we have to add a new start symbol S and the following two further initial trees:



Closure properties (2)

Proposition 2

TALs are closed under concatenation.

In order to show this, assume again the sets of non-terminals to be disjoint and S_1 and S_2 to be the respective start symbols. Then

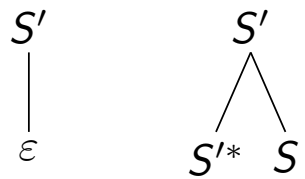
- build the unions of the initial and auxiliary trees,
- introduce a new start symbol S and add one further initial tree:

Closure properties (3)

Proposition 3

TALs are closed under Kleene closure.

The idea of the proof is as follows: Let S be the start symbol of the original TAG. We then add a new start symbol S' and add one initial and one auxiliary tree as follows:



Closure properties (4)

Proposition 4

TALs are closed under substitution.

In order to obtain the TAG that yields the language after substitution, we replace all terminals by start symbols of the corresponding TAGs.

As a corollary one obtains:

Proposition 5

TALs are closed under arbitrary homomorphisms.

Closure properties (5)

Proposition 6

TALs are closed under intersection with regular languages.

The proof in [VS87] uses extended push-down automata (EPDA), the automata that recognize TALs. We will introduce EPDAs later. Vijay-Shanker combines such an automaton with the finite state automaton for a regular language in order to construct a new EPDA that recognizes the intersection.

Pumping lemma (1)

- In CFLs, from a certain string length on, two parts of the string can be iterated ("pumped").
- The proof idea is the following: Context-free derivation trees from a certain maximal path length on have the property that a non-terminal occurs twice on this path. Then the part between the two occurrences can be iterated. This means that the strings to the left and right of this part are pumped.

The same kind of iteration is possible in TAG derivation trees since TAG derivation trees are context-free. This leads to a pumping lemma for TALs [VS87].

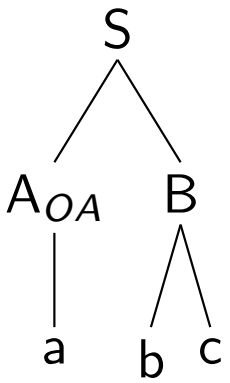
Pumping lemma (2)

What does "TAG derivation trees are context-free" mean?
Idea of constructing a CFG that describes the derivation trees of a TAG (without edge labels):

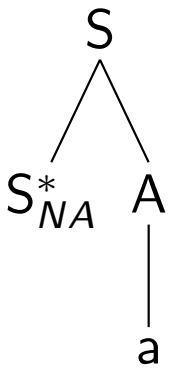
- Whenever we can add trees derived from elementary trees $\gamma_1, \ldots \gamma_k$ to an elementary γ at the nodes $v_1 \ldots v_k$ (ordered in some way), we assume a production $\gamma \rightarrow \gamma_1 \ldots \gamma_k$.
- Whenever an elementary trees γ does not have substution nodes or OA nodes, we add $\gamma \rightarrow \varepsilon$
- The set of terminals in the CFG is $\emptyset$, the nonterminals are the elementary trees and an additional S.
- For every initial tree α with the TAG start label as root label, we add $S \rightarrow \alpha$.

Pumping lemma (3)

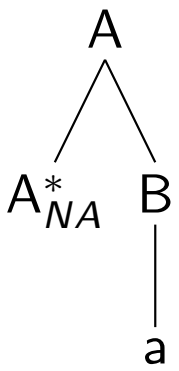
α



β_1



β_2



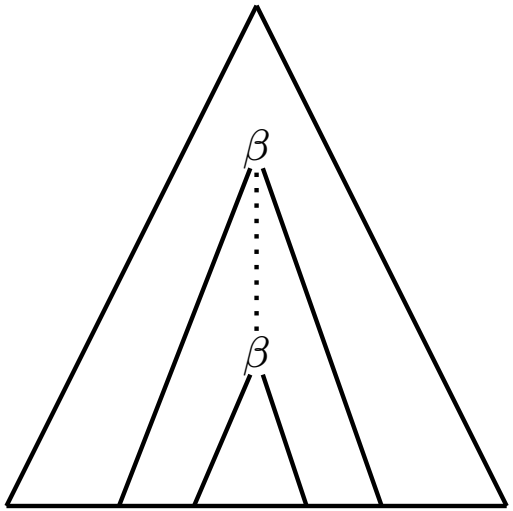
CFG productions:

$S \to \alpha \qquad \alpha \to \beta_1\ \beta_2 \mid \beta_2 \qquad \beta_1 \to \beta_1\ \beta_2 \mid \beta_1 \mid \beta_2 \mid \varepsilon \qquad \beta_2 \to \beta_2 \mid \varepsilon$

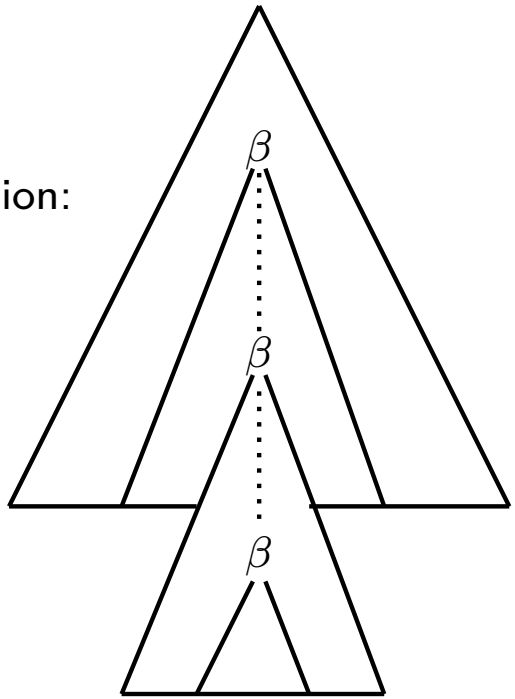
Pumping lemma (4)

Back to the pumping lemma:

From a certain word length on (i.e., also a certain derivation tree height), we have the following pattern:



Iteration:



Pumping lemma (5)

In other words,

- A derived auxiliary tree β' can be repeatedly adjoined into itself.
- Into the lowest β' (low in the sense of the derivation tree) another auxiliary tree β'' derived from β is adjoined.

Pumping lemma (6)

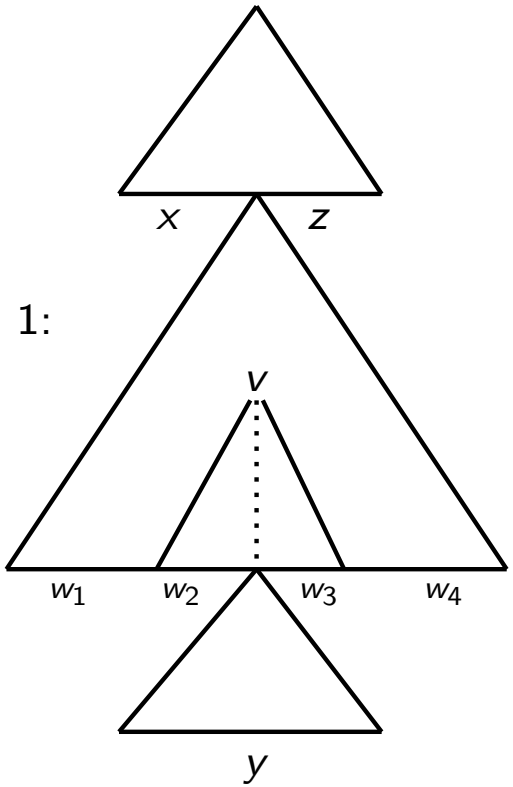
What does that mean for the derived tree?

Let v be the node in β' to which β' can be adjoined and to which the final β'' is adjoined as well. There are three cases for the corresponding derived trees before adjoining the final β'':

1. v is on the spine (i.e., on the path from the root to the foot node),
2. v is on the left of the spine, or
3. v is on the right of the spine.

Pumping lemma (7)

Case 1:

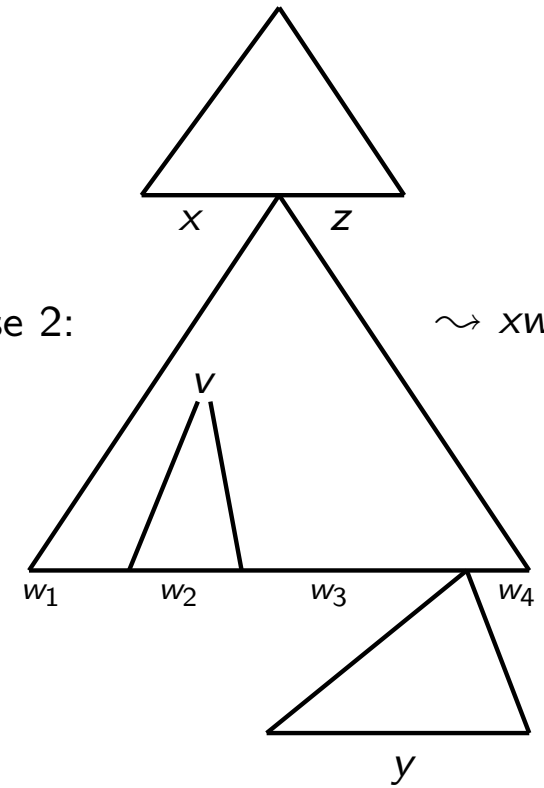


$\rightsquigarrow xw_1^n v_1 w_2^n y w_3^n v_2 w_4^n z$

Note that v is a node while v_1 and v_2 are strings here.

Pumping lemma (8)

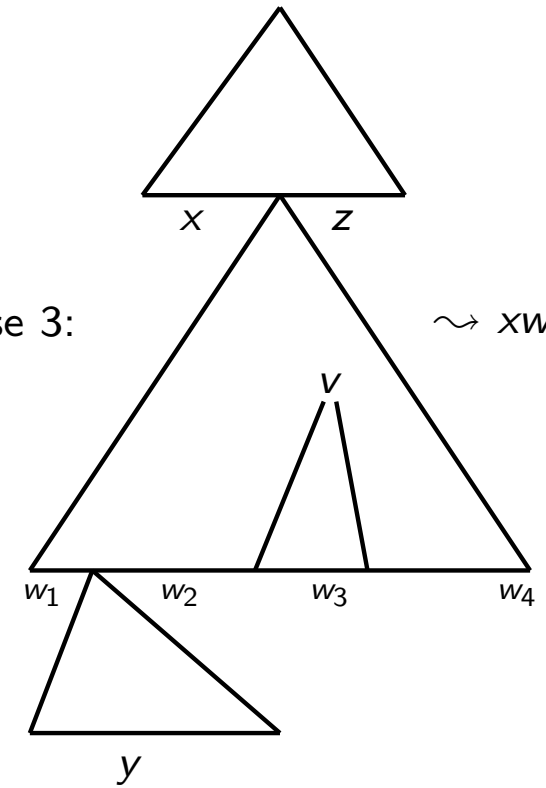
Case 2:



$\rightsquigarrow xw_1^{n+1}v_1w_2v_2w_3(w_2w_4w_3)^nyw_4z$

Pumping lemma (9)

Case 3:



$\rightsquigarrow xw_1y(w_2w_1w_3)^nw_2v_1w_3v_2w_4^{n+1}z$

Pumping lemma (10)

Proposition 7 (Pumping Lemma for TAL)

If L is a TAL, then there is a constant c such that if $w \in L$ and $|w| \geq c$, then there are x, y, z, v_1, v_2, w_1, w_2, w_3, $w_4 \in T^*$ such that

- $|v_1 v_2 w_1 w_2 w_3 w_4| \leq c$, $|w_1 w_2 w_3 w_4| \geq 1$, *and*
- *one of the following three cases holds:*
  1. $w = x w_1 v_1 w_2 y w_3 v_2 w_4 z$ *and* $x w_1^n v_1 w_2^n y w_3^n v_2 w_4^n z$ *is in the string language for all* $n \geq 0$, *or*
  2. $w = x w_1 v_1 w_2 v_2 w_3 y w_4 z$ *and* $x w_1^{n+1} v_1 w_2 v_2 w_3 (w_2 w_4 w_3)^n y w_4 z$ *is in the string language for all* $n \geq 0$, *or*
  3. $w = x w_1 y w_2 v_1 w_3 v_2 w_4 z$ *and* $x w_1 y (w_2 w_1 w_3)^n w_2 v_1 w_3 v_2 w_4^{n+1} z$ *is in the string language for all* $n \geq 0$.

Pumping lemma (11)

As a corollary, the following weaker pumping lemma holds:

Proposition 8 (Weak Pumping Lemma for TAL)

If L is a TAL, then there is a constant c such that if $w \in L$ and $|w| \geq c$, then there are $x, y, z, v_1, v_2, w_1, w_2, w_3, w_4 \in T^*$ such that

- $|v_1 v_2 w_1 w_2 w_3 w_4| \leq c$,
- $|w_1 w_2 w_3 w_4| \geq 1$,
- $w = x v_1 y v_2 z$, *and*
- $x w_1^n v_1 w_2^n y w_3^n v_2 w_4^n z \in L(G)$ *for all* $n \geq 0$.

In this weaker version, the w_1, w_2, w_3, w_4 need not be substrings of the original word w.
This is an existential pumping lemma ("there are words that contain substrings that can be iterated"), in contrast to the one for CFLs where we have iterable substrings in every word w with $|w| \geq c$.

Pumping lemma (12)

A pumping lemma can be used to show that certain languages are not in the class of the string languages satisfying the pumping proposition.

Proposition 9

The double copy language $L := \{www \mid w \in \{a, b\}^*\}$ *is not a TAL.*

Pumping lemma (13)

Proof: Assume that L is a TAL.
Then $L' := L \cap a^*b^*a^*b^*a^*b^* = \{a^n b^m a^n b^m a^n b^m \mid n, m \geq 0\}$ is a TAL as well. Assume that L' satisfies the weak pumping lemma with a constant c.
Consider the word $w = a^{c+1}b^{c+1}a^{c+1}b^{c+1}a^{c+1}b^{c+1}$.
None of the w_i, $1 \leq i \leq 4$ from the pumping lemma can contain both as and bs. Furthermore, at least three of them must contain the same letters and be inserted into the three different a^{c+1} respectively or into the three different b^{c+1}. This is a contradiction since then either $|v_1| \geq c + 1$ or $|v_2| \geq c + 1$.

Pumping lemma (14)

Another example of a language that can be shown not to be a TAL, using the pumping lemma, is $L_5 := \{a^n b^n c^n d^n e^n \mid n \geq 0\}$.

Note that $L_4 := \{a^n b^n c^n d^n \mid n \geq 0\}$ is a TAL. Therefore, TAG is sometimes said to be able to "count up to 4". CFG can count up to 2.

References I

[Kal10] Laura Kallmeyer.
Parsing Beyond Context-Free Grammars.
Cognitive Technologies. Springer, Heidelberg, 2010.

[VS87] K. Vijay-Shanker.
A Study of Tree Adjoining Grammars.
PhD thesis, University of Pennsylvania, 1987.

[VSJ85] K. Vijay-Shanker and Aravind K. Joshi.
Some computational properties of Tree Adjoining Grammars.
In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 82–93, 1985.