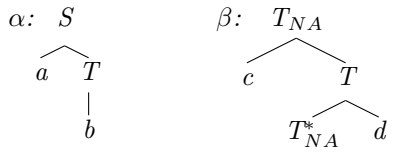
Tree Adjoining Grammars
Final exam

Laura Kallmeyer, Simon Petitjean

12.02.2016

Exercise 1 *Consider the following TAG with adjunction constraints:*



1. *Which language does this TAG generate?*
2. *Consider the CYK parsing algorithm from the course slides. We apply it to this TAG and an input string acbd.*

   It would start filling a chart (among others) with the following items:

	Item	*Rule*
1.	$[\alpha, 1_\top, 0, -, -, 1]$	*lex-scan (a)*
2.	$[\beta, 1_\top, 1, -, -, 2]$	*lex-scan (c)*
3.	$[\alpha, 22_\top, 2, -, -, 3]$	*lex-scan (b)*
4.	$[\beta, 22_\top, 3, -, -, 4]$	*lex-scan (d)*
5.	$[\beta, 21_\top, 2, 2, 3, 3]$	*foot-predict*
6.	$[\beta, 2_\bot, 2, 2, 3, 4]$	*move-binary (4,5)*
7.	$[\alpha, 2_\bot, 2, -, -, 3]$	*move-unary (3)*

   (a) What are the possible next items the parser can deduce from 6 and 7, given only the items so far in the chart?
   Explain which of the parsing rules can be applied and how they are applied.

   (b) Assume that later, the parser has found an item

$$[\beta, \varepsilon_\top, 1, 2, 3, 4]$$

   Given this item and the items from the table, which items can the parser now deduce from items 6 and 7?
   Explain which of the parsing rules can be applied and how they are applied.

Solution:

1. $\{ac^n bd^n \mid n \geq 0\}$. 2 Punkte
2. (a): only possibilities: $[\beta, 2_\top, 2, 2, 3, 4]$ with *null-adjoin* from 6 and $[\alpha, 2_\top, 2, -, -, 3]$ with *null-adjoin* from 7. In both cases, there is no matching auxiliary tree in the chart to perform an *adjoin* step.

   2 Punkte

   (b): Now we can apply *adjoin* with the new item and the matching item 6, which leads to $[\alpha, 2_\top, 1, -, -, 4]$.

   2 Punkte

Exercise 2

For the following, you can assume that we have already shown that $L_5 = \{a^n b^n c^n d^n e^n \mid n \geq 0\}$ *is not a TAL.*

1. *Show that* $L = \{(af)^n(fb)^n c^n d^n (gef)^n \mid n \geq 0\}$ *is not a TAL.*

2. *Show that* $L = \{w \in \{a,b,c,d,e,f,g\}^* \mid |w|_f = |w|_g > |w|_a = |w|_b = |w|_c = |w|_d = |w|_e\}$ *is not a TAL.*

Solution:

1. We assume that L is a TAL. In this case, the image of L under the homomophism f with $f(a) = a, f(b) = b, f(c) = c, f(d) = d, f(e) = e, f(f) = f(g) = \varepsilon$ is also a TAL. But this image is L_5, which is not a TAL. Contradiction, therefore our initial assumption is false.

   4 Punkte

2. We assume that L is a TAL. In this case, the image L' of L under the homomophism f as in 1. is also a TAL. $L' = \{w \in \{a,b,c,d,e\}^* \mid |w|_a = |w|_b = |w|_c = |w|_d = |w|_e\}$. Then L' intersected with the regular language denoted by $a^*b^*c^*d^*e^*$ is also a TAL. This intersection however is L_5. Contradiction, therefore our initial assumption is false.

   4 Punkte

Exercise 3

Show that $L = \{w_1 w_1 w_2 w_2 \mid w_1 \in \{a,b\}^*, w_2 \in \{c,d\}^*\}$ *is a TAL.*

Hint: You can give a TAG for this language or you can use the closure properties of TALs together with our knowledge that the copy language is a TAL.

Solution:

We know that the copy language $L_1 = \{w_1 w_1 \mid w_1 \in \{a,b\}^*\}$ is a TAL. Furthermore, its image $L_2 = \{w_2 w_2 \mid w_2 \in \{c,d\}^*\}$ under a homomorphism $a \to c, b \to d$ is also a TAL. Since TALs are closed under concatenation, $L = L_1 L_2$ is also a TAL.

4 Punkte

Exercise 4 *Consider the following example of scrambling in German:*

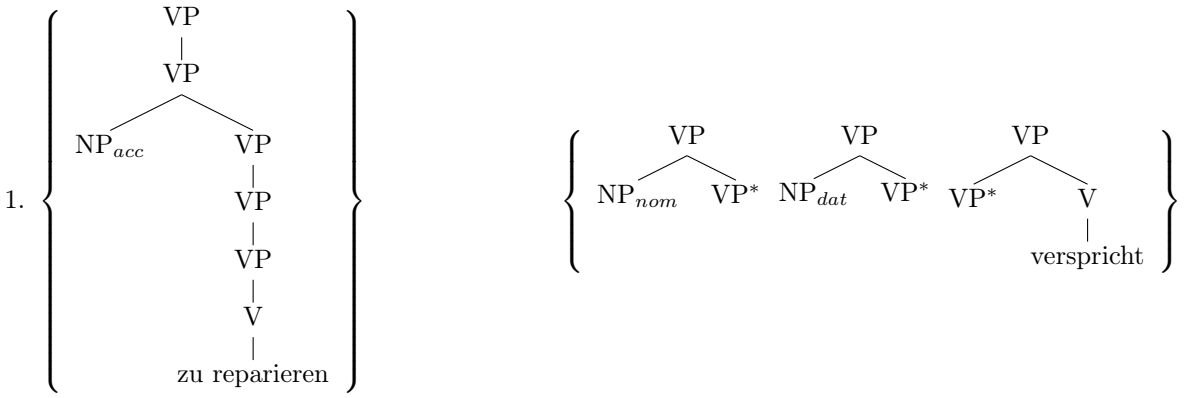
(1)
   a. *dass Hans dem Kunden den Wagen zu reparieren verspricht*
   b. *dass Hans den Wagen dem Kunden zu reparieren verspricht*
   c. *dass dem Kunden Hans den Wagen zu reparieren verspricht*
   d. *dass dem Kunden den Wagen Hans zu reparieren verspricht*
   e. *dass den Wagen Hans dem Kunden zu reparieren verspricht*
   f. *dass den Wagen dem Kunden Hans zu reparieren verspricht*

1. *Give a tree-local MCTAG analysis along the lines of slide 23 covering these sentences: Give the elementary tree sets needed for the two verbs (for simplification, "zu reparieren" is treated as a single lexical anchor).*
   Explain how, given your tree sets, the sets of the two verbs combine for (1-a) and for (1-e).

2. *Give a V-TAG analysis for (1) along the lines of slides 27,28 by giving, again, the elementary tree sets for the verbs including dominance links.*
   Integrity is not relevant for this example, you can therefore leave out the VP node with integrity marking and assume that the trees in the tree sets that are anchored by the two verbs have the following shape:

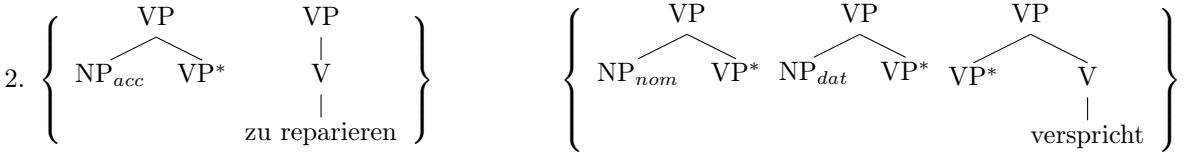
Ater having given the tree sets with their dominance links, explain how the word orders in (1) can be derived with these sets.

Solution:

1. 

In all cases in (1), the *verspricht* set is added to the *reparieren* tree by adjoining all three trees to different VP nodes. For (1-a), the two NP trees adjoin to the root (NP nom) and its daughter (NP dat) while for (1-e) the NP nom tree adjoins to the sister of the NP acc while the NP dat tree adjoins to the daughter of that sister. The *verspricht* tree can always adjoin to the VP node immediately dominating the V node.

6 Punkte

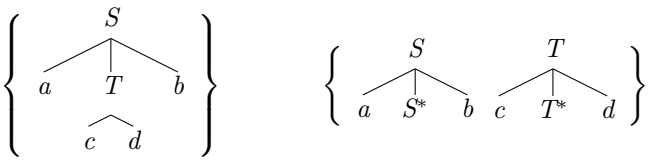
2. 

Dominance links: each VP foot node of an NP auxiliary tree dominates the root node of the lexicalized verbal tree in the same set.

In all cases in (1), the *verspricht* tree adjoins to the VP node of *reparieren*. Then the argument NP trees adjoin in the order from right, each time to the root of the new larger derived tree just obtained.

6 Punkte

Exercise 5 *Consider an MCTAG with terminals $\{a, b, c, d\}$, nonterminals $\{S, T\}$, no adjunction constraints (null adjunction at foot nodes is assumed) and the following elementary tree sets:*



Which language does this MCTAG generate if

1. *it is taken to be a tree-local MCTAG?*
2. *it is taken to be a set-local MCTAG?*

Solution:

1. $\{acdb, aaccddbb\}$ 1 Punkt

2. $\{a^n c^n d^n b^n \mid n \geq 1\}$ 2 Punkte

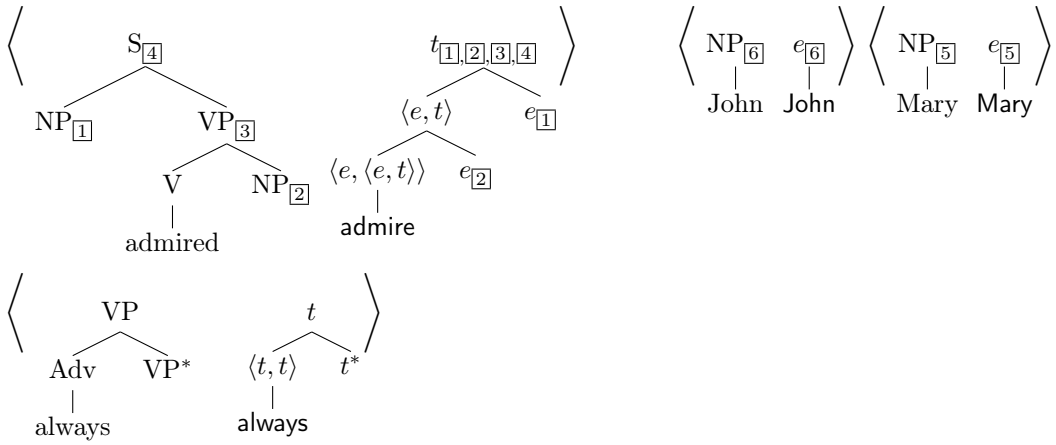
Exercise 6 *Consider the following sentence and its semantics (provided* John *and* Mary *are treated as constants):*

(2) *a.* *John always admired Mary.*
b. always(admire(Mary)(John))

Show how STAG derives this semantics by giving

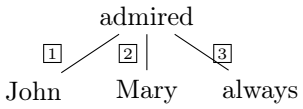
1. *the elementary tree pairs including their links used for (2) and*
2. *the derivation tree.*

Solution:

1. 

4 Punkte

2. Derivation tree:



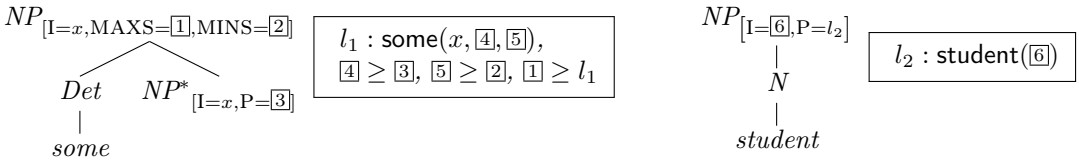
2 Punkte

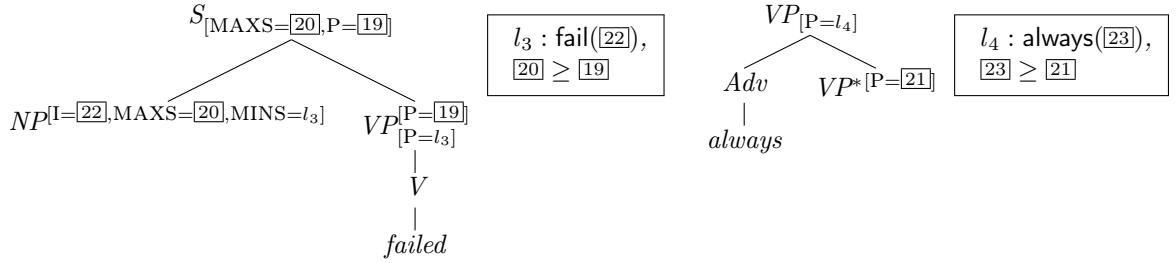
Exercise 7 *Consider the following sentence and the two scope orders it has:*

(3) *a.* *some student always failed*
b. some > always, always > some

(Sorry for this example, I was not very creative here ...)

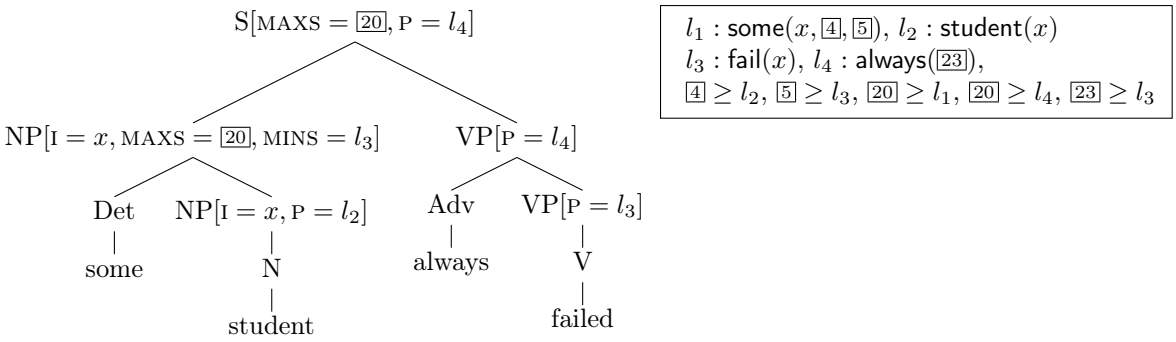
The pairs of elmentary tree and semantic representation used for this sentence in the approach of unification-based LTAG semantics are as follows:





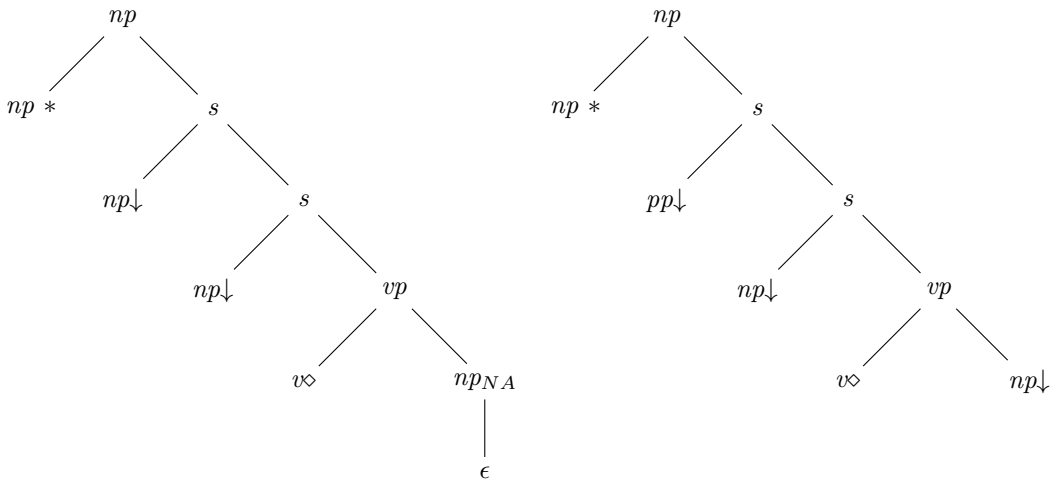
Give the derived tree after top-bottom unification together with the underspecified semantic representation for (3).

Solution:



6 Punkte

Exercise 8 *The goal of this exercise is to generate the two following trees for relative clauses:*



$\beta N1nx0Vnx1$ $\beta Npxnx0Vnx1$

- *The decomposition and code for $\beta N1nx0Vnx1$ in given below*
- *You need to provide the code for $\beta Npxnx0Vnx1$ in a new class* `betaNpxnx0Vnx1`

- *You can (and should) add new classes for missing fragments*

```
class Subject
export ?VP ?S ?SubjNP ?SubjMark
declare ?S ?VP ?SubjNP ?SubjMark
{
  <syn> {  node ?S [cat=s] {
                node ?SubjNP (mark=subst) [cat=np]
                node ?VP [cat=vp]
}}}

class VerbProjection
export ?VP ?V
declare ?VP ?V
{
  <syn> {  node ?VP [cat=vp] {
                node ?V (mark=anchor) [cat=v]
}}}

class Object
export ?VP ?V ?ObjNP ?ObjMark
declare ?VP ?V ?ObjNP ?ObjMark
{
  <syn> {  node ?VP [cat=vp] {
                node ?V [cat=v]
                node ?ObjNP (mark=ObjMark) [cat=np]
}}}

class ExtractedNP
export ?S ?Sr
declare ?Sr ?NP ?S
{
  <syn> {node ?Sr [cat=s] {
                node ?NP (mark=subst) [cat=np]
                node ?S [cat=s]
}}}

class NPFoot
export ?Sr
declare ?NP ?NPFoot ?Sr
{
  <syn>{  node ?NP [cat=np]{
                node ?NPFoot (mark=foot)[cat=np]
                node ?Sr [cat=s]
}}}

class betaN1nx0Vnx1
import Subject[] VerbProjection[] Object[] ExtractedNP[] NPFoot[]
declare ?XP
{
  ?ObjMark=nadj;
  <syn> {  node ?XP (mark=flex)[phon = e];
           ?ObjNP -> ?XP
}}
```