Dennis Nikolay 2544912
Mark Schuegraf 2543591
Julia Schulz 2553760

Einführung in die Computerlinguistik
4. Übungsblatt

Aufgabe 4.1

1. Education is also a key focus with 2014 seeing the introduction of a third day to the event dedicated to schools and university groups alongside the already existing 2 day workshop programs.
   Regel: Possible agreement error. The noun 'program' seems to be countable, so consider using: programs.
   (a) Die Regel überprüft einen Satz auf Vorkommen gewisser falscher Ausdrücke und schlägt dann eine Korrektur vor.
   (b) Es wird nicht erkannt, dass sich die "2" auf die Anzahl der Veranstaltungstage und nicht etwa auf die Anzahl der Workshops bezieht. Deswegen wird falscher Alarm ausgelöst.
   (c) Ja in dem man Satz für Satz über Jahre hinweg für jede falsch ausgelöste Fehlermeldung einen Ausnahmefall hinzufügt.
2. After she was turned, she and Eric often had sex.
   Vorgeschlagene Änderung: Vertauschung von "often" und "had."
   Regel: The adverb 'often' is usually put between 'had' and 'sex'.
   (a) Die Regel überprüft die Reihenfolge typischerweise in Sequenz auftretender Wörter.
   (b) In der Vergangenheit funktioniert die Regel nicht, während im Präsens z.B. "We often have sex" ein korrekter Satz ist.
   (c) "Often" aus dem Token Pool entfernt.

Aufgabe 4.2

(a) G=<{PP, AP, Det, N, A, Prp, PN, Pro},
{NN, Det, A, Prp, GPRT},
{NP -> Det AP NN C | Det NN C,
C -> PP | epsilon,
AP -> A | A AP | GPRT AP,
PP -> Prp NP},
NP>

(b)

$G1 = \langle V, \Sigma, P, S \rangle$ mit

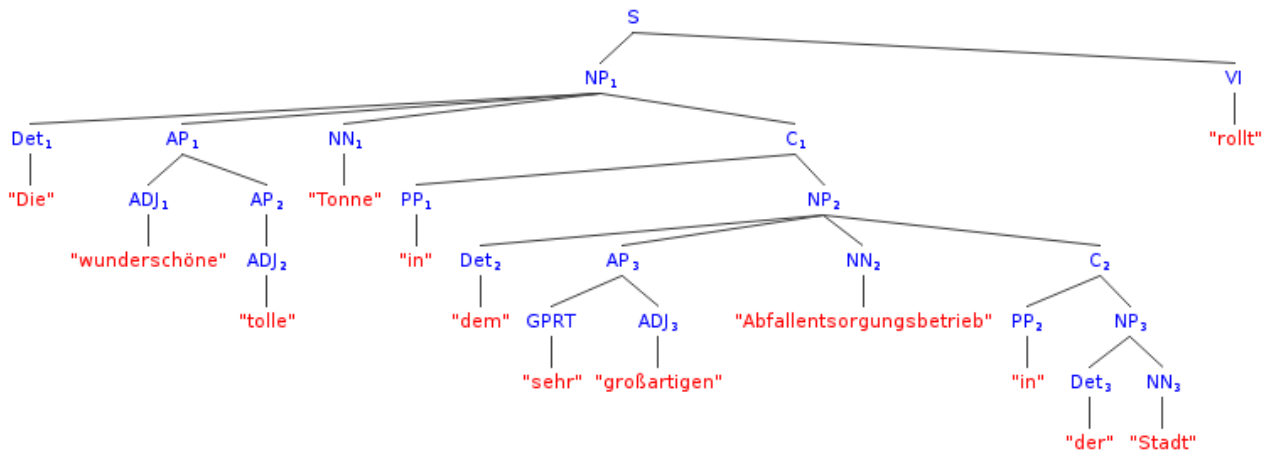
$V = \{S, SRel, NP, VI, VT, N, Det, RPro\} \cup \Sigma$

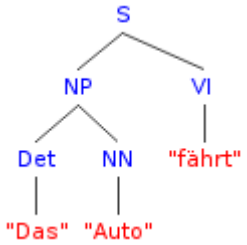
$\Sigma = \{schläft, arbeitet, studiert, wählte, Student, Fach, der, das, er\}$

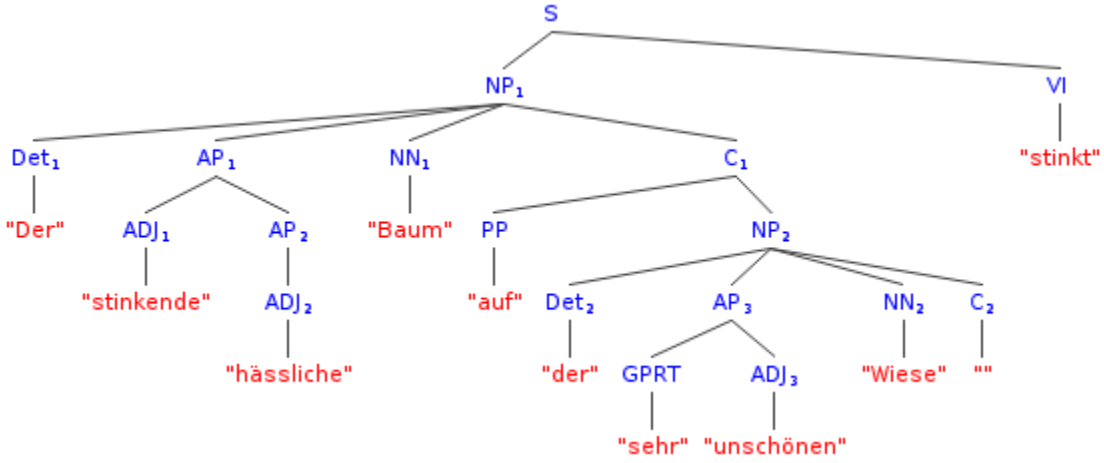
$P =$

$S \rightarrow NP\ VI$
$S \rightarrow NP\ VT\ NP$
$SRel \rightarrow RPro\ NP\ VT$
$SRel \rightarrow RPro\ VI$
$NP \rightarrow Det\ N$
$NP \rightarrow Det\ N\ SRel$
$NP \rightarrow Pro$
$NP \rightarrow Det\ AP\ NN\ C$

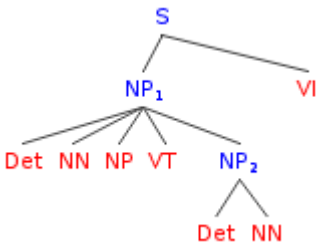
NP → DET NN C
C → PP
C→ ε
C →PP NP
AP → ADJ
AP → ADJ AP
AP → GPRT AP
PP → P NP
S = S







(c) Ja,denn man kann z.B. folgendes ableiten: (was nicht funktionieren sollte)



aber nicht folgenden Satz: (was funktionieren sollte)

„Das Auto fährt mit einem Affenzahn die Straße runter“

Aufgabe 4.3

(a) ART NN APPR NE VVFIN APPR ART NN, ART APPR ART ADJA NN VVFIN, VVFIN. ART ADJA ADJA NN VMPP APPR CARD KON CARD APPRART NN ART NE CARD.

NPs: Die Computerlinguistik [in Saarbrücken], einer Forschungstradition [,die auf die frühen Siebziger zurückgeht], Die hochgradig interdisziplinäre Arbeit, im Rahmen [des SFB 100]

im Rahmen und im Rahmen des SFB 100 erkennt unsere Grammatik nicht.

(b) ITJ, KOUS PPER ART VVFIN VVFIN, KOUS VVFIN PPER PPER KOUI ADV NN CARD NN VVFIN, KOUS PPER APPR PDS ADJA NN ADV APPR NN VVFIN VMPP.

NPs: heute Abend [zwei Kinokarten], diesem schlechten Wetter, im Trockenen

heute Abend zwei Kinokarten wird nicht erkannt, ebenso im Trockenen.

Aufgabe 4.4

Eingabe (Skript):

```
#!/bin/bash
listOfTags="ADJA ADJD ADV APPR APPRART APPO APZR ART CARD FM ITJ KOUI KOUS
KON KOKOM NN NE PDS PDAT PIS PIAT PIDAT PPER PPOSS PPOSAT PRELS PRELAT
PRF PWS PWAT PWAV PAV PTKZU PTKNEG PTKVZ PTKANT PTKA TRUNC VVFIN
VVIMP VVINF VVIZU VVPP VAFIN VAIMP VAINF VAPP VMFIN VMINF VMPP XY"

for tag in $listOfTags
do
        count=`grep $tag /home/dennis/Schreibtisch/tiger_pos.txt|wc -w;`
        if [ -z "$count" ]; then echo "$tag: 0"; else  echo "$tag: $count"; fi
done
```

Ausgabe:
ADJA: 108884
ADJD: 37040
ADV: 78718
APPR: 180828
APPRART: 29724
APPO: 512
APZR: 708
ART: 224250
CARD: 31880
FM: 2522
ITJ: 38
KOUI: 1888
KOUS: 14262
KON: 44092
KOKOM: 5308
NN: 368410
NE: 114948
PDS: 6804
PDAT: 5830
PIS: 9708
PIAT: 12490
PIDAT: 0
PPER: 27096
PPOSS: 28
PPOSAT: 13140
PRELS: 12462
PRELAT: 592
PRF: 11470
PWS: 1672
PWAT: 364
PWAV: 3482
PAV: 2
PTKZU: 8848
PTKNEG: 10586
PTKVZ: 10186
PTKANT: 162
PTKA: 1052
TRUNC: 2718
VVFIN: 71262
VVIMP: 310
VVINF: 26784
VVIZU: 3028
VVPP: 35460
VAFIN: 49178
VAIMP: 6
VAINF: 6338
VAPP: 2720

VMFIN: 17608
VMINF: 1010
VMPP: 16
XY: 2010

APPR kommt am häufigsten vor, PAV am seltensten

Aufgabe 4.5

Unser Vater ist ein Computer. Deswegen haben wir von klein an die simple Programmiersprache C0 gelernt. Es folgt ein Auszug der Grammatik dieser nun leider sehr selten gewordenen Sprache:

$\langle Di \rangle$	$\longrightarrow$	$0 \mid \ldots \mid 9$	digit
$\langle DiS \rangle$	$\longrightarrow$	$\langle Di \rangle \mid \langle Di \rangle \langle DiS \rangle$	digit sequence
$\langle Le \rangle$	$\longrightarrow$	$a \mid \ldots \mid z \mid A \mid \ldots \mid Z$	letter
$\langle DiLe \rangle$	$\longrightarrow$	$\langle Le \rangle \mid \langle Di \rangle$	alphanumeric symbol
$\langle DiLeS \rangle$	$\longrightarrow$	$\langle DiLe \rangle \mid \langle DiLe \rangle \langle DiLeS \rangle$	sequence of symbols
$\langle Na \rangle$	$\longrightarrow$	$\langle Le \rangle \mid \langle Le \rangle \langle DiLeS \rangle$	name
$\langle C \rangle$	$\longrightarrow$	$\langle DiS \rangle \mid \langle DiS \rangle u \mid null$	*int-*/*uint-*/null-pointer-constant
$\langle CC \rangle$	$\longrightarrow$	$'\langle DiLe \rangle'$	*char*-constant
$\langle BC \rangle$	$\longrightarrow$	$true \mid false$	*bool*-constant
$\langle id \rangle$	$\longrightarrow$	$\langle Na \rangle \mid \langle id \rangle.\langle Na \rangle \mid \langle id \rangle[\langle E \rangle] \mid \langle id \rangle *$	identifier
$\langle F \rangle$	$\longrightarrow$	$\langle id \rangle \mid -_1 \langle F \rangle \mid (\langle E \rangle) \mid \langle C \rangle \mid \langle id \rangle \&$	factor
$\langle T \rangle$	$\longrightarrow$	$\langle F \rangle \mid \langle T \rangle \cdot \langle F \rangle \mid \langle T \rangle / \langle F \rangle$	term
$\langle E \rangle$	$\longrightarrow$	$\langle T \rangle \mid \langle E \rangle + \langle T \rangle \mid \langle E \rangle -_2 \langle T \rangle$	expression
$\langle Atom \rangle$	$\longrightarrow$	$\langle E \rangle > \langle E \rangle \mid \langle E \rangle \geq \langle E \rangle \mid \langle E \rangle < \langle E \rangle \mid$	atom
		$\langle E \rangle \leq \langle E \rangle \mid \langle E \rangle == \langle E \rangle \mid \langle E \rangle \neq \langle E \rangle \mid \langle BC \rangle$	
$\langle BF \rangle$	$\longrightarrow$	$\langle id \rangle \mid \langle Atom \rangle \mid \sim \langle BF \rangle \mid (\langle BE \rangle)$	boolean factor
$\langle BT \rangle$	$\longrightarrow$	$\langle BF \rangle \mid \langle BT \rangle \wedge \langle BF \rangle$	boolean term
$\langle BE \rangle$	$\longrightarrow$	$\langle BT \rangle \mid \langle BE \rangle \vee \langle BT \rangle$	boolean expression
$\langle St \rangle$	$\longrightarrow$	$\langle id \rangle = \langle E \rangle \mid \langle id \rangle = \langle BE \rangle \mid \langle id \rangle = \langle CC \rangle \mid$	assignment statement
		$if\ \langle BE \rangle\ \{\langle StS \rangle\}\ else\ \{\langle StS \rangle\} \mid$	cond. statement
		$if\ \langle BE \rangle\ \{\langle StS \rangle\} \mid$	
		$while\ \langle BE \rangle\ \{\langle StS \rangle\} \mid$	loop
		$\langle id \rangle = \langle Na \rangle (\langle PaS \rangle) \mid$	function call
		$\langle id \rangle = new\ \langle Na \rangle *$	allocation of memory
$\langle rSt \rangle$	$\longrightarrow$	$return\ \langle E \rangle \mid return\ \langle BE \rangle \mid return\ \langle CC \rangle$	return-statement
$\langle PaS \rangle$	$\longrightarrow$	$\varepsilon \mid \langle ParS \rangle$	parameter sequence
$\langle ParS \rangle$	$\longrightarrow$	$\langle E \rangle \mid \langle E \rangle, \langle ParS \rangle \mid \langle BE \rangle \mid \langle BE \rangle, \langle ParS \rangle$	non empty sequence
$\langle StS \rangle$	$\longrightarrow$	$\langle St \rangle \mid \langle St \rangle; \langle StS \rangle$	statement sequence
$\langle program \rangle$	$\longrightarrow$	$\langle TyDS \rangle; \langle VaDS \rangle; \langle FuDS \rangle \mid$	$C0$-program
		$\langle VaDS \rangle; \langle FuDS \rangle \mid$	no type declaration
		$\langle TyDS \rangle; \langle FuDS \rangle \mid$	no variable declaration
		$\langle FuDS \rangle$	only function declaration
$\langle TyDS \rangle$	$\longrightarrow$	$\langle TyD \rangle \mid \langle TyD \rangle; \langle TyDS \rangle$	type declaration sequence
$\langle TyD \rangle$	$\longrightarrow$	$typedef\ \langle TE \rangle\ \langle Na \rangle$	type declaration
$\langle Ty \rangle$	$\longrightarrow$	$int \mid bool \mid char \mid unint \mid \langle Na \rangle$	type names
$\langle TE \rangle$	$\longrightarrow$	$\langle Ty \rangle [\langle DiS \rangle]$	type expression, array
		$\langle Ty \rangle * \mid struct \{\langle VaDS \rangle\}$	pointer, struct
$\langle VaDS \rangle$	$\longrightarrow$	$\langle VaD \rangle \mid \langle VaD \rangle; \langle VaDS \rangle$	variable declaration sequence
$\langle VaD \rangle$	$\longrightarrow$	$\langle Ty \rangle\ \langle Na \rangle$	variable declaration
$\langle FuDS \rangle$	$\longrightarrow$	$\langle FuD \rangle \mid \langle FuD \rangle; \langle FuDS \rangle$	function declaration sequence
$\langle FuD \rangle$	$\longrightarrow$	$\langle Ty \rangle\ \langle Na \rangle (\langle PaDS \rangle) \{\langle VaDS \rangle; \langle body \rangle\} \mid$	function declaration
		$\langle Ty \rangle\ \langle Na \rangle (\langle PaDS \rangle) \{\langle body \rangle\}$	no local variables
$\langle PaDS \rangle$	$\longrightarrow$	$\varepsilon \mid \langle ParDS \rangle$	parameter declaration sequence
$\langle ParDS \rangle$	$\longrightarrow$	$\langle VaD \rangle \mid \langle VaD \rangle, \langle ParDS \rangle$	non empty
$\langle body \rangle$	$\longrightarrow$	$\langle rSt \rangle \mid \langle StS \rangle; \langle rSt \rangle$	function body

Zum Beispiel lassen sich so rührende Sätze wie "bool besterVater = true" formulieren.

(Wort für Wort Übersetzung: Du bist der beste Vater.)

Ebenfalls möglich sind standartisierte Antworten, wie man sie als Kind öfter bekommt:

```
char* papaWannKommenWirEndlichAn() {
        char* antwort = malloc(sizeof(char)*4);
        antwort = "Bald";
        return antwort;
}
```

(Wort für Wort Übersetzung: Papa wann kommen wir endlich an? Bald.)

Zu guter Letzt können Bedürfnisse einfach ausgedrückt werden. "int magenInhalt = 0; bool hungrig = true;"

(Wort für Wort Übersetzung: Mein Magen knurrt. Ich hab hunger.)