Computational Syntax Exam, 30 May 2016

9:00-12:00 at Viktoriagatan 30, Gothenburg.

Masters in Language Technology, University of Gothenburg.

Aids. You are allowed to have a *cheat sheet*, which is an A4 sheet of paper with hand-written text and pictures, on both sides if you want. You must submit your cheat sheet together with your exam answers.

Grades: max 60p, VG 48p, G 24p.

Teacher: Aarne Ranta, aarne@chalmers.se, tel. 031 772 10 82

Write the answer to each question on a separate sheet, and number the sheets.

Question 1: context-free grammars (12p)

Build a context-free grammar (a.k.a. BNF grammar) that deals with the following constructions:

- sentence built from a noun phrase and a verb phrase, e.g. “we sleep”, “John sees us”
- noun phrase built from a pronoun or a proper name e.g. “we”, “John”
- verb phrase built from an intransitive (i.e. one-place) verb such as “sleep” alone, or a transitive (two-place) verb such as “see” with a noun-phrase complement
- verb phrases modified by adverbs, e.g. “sleep here”
- pronouns “we” and “she”
- Proper names “John” and “Mary”
- intransitive verb “sleep”
- transitive verb “see”
- adverbs “here”, “today”

The grammar must capture English number and case agreement precisely, so that

we sleep here today
she sees us

are recognized but not for instance

us sleep
we sleeps
John sees we

Hint: you may need more categories than in a GF grammar with parameters (Question 2).

Question 2: grammars with parameters (12p)

Write a GF concrete syntax that defines exactly the same language as the context-free grammar of Question 1 and whose only abstract syntax functions are

PredVP : NP -> VP -> S
UseV : V -> VP
ComplV2 : V2 -> NP -> VP
AdvVP : VP -> Adv -> VP
UsePron : Pron -> NP
UsePN : PN -> NP
we, she : Pron
John, Mary : PN
sleep : V
see : V2
here, today : Adv

Hint: now you have to introduce parameters in the concrete syntax. The concrete syntax must specify (1) the parameter types (2) the linearization types of categories (3) the linearizations of the functions

Question 3: trees and probability (12p)

Show (1) the abstract syntax tree (AST), (2) the parse tree (concrete syntax tree, phrase structure tree), and (3) the dependency tree for the sentence

she sees us here today

as constructed in the grammar of Question 2. The dependency tree should use the following labels (but you don't need to show POS tags in the dependency tree):

PredVP	*nsubj*	*head*
ComplV2	*head*	*dobj*
AdvVP	*head*	*advmod*

Compute the probability of the abstract syntax tree, assuming an even (unbiased) probability distribution. Show how you calculate the probability from its components, not just the end result!

Question 4: discontinuous constituents (12p)

Let us simplify the abstract syntax of Question 2 so that we don't need parameters for agreement, but extend it so that we need to deal with different word orders when we write a concrete syntax for German. The linearizations of lexical items are shown in comments:

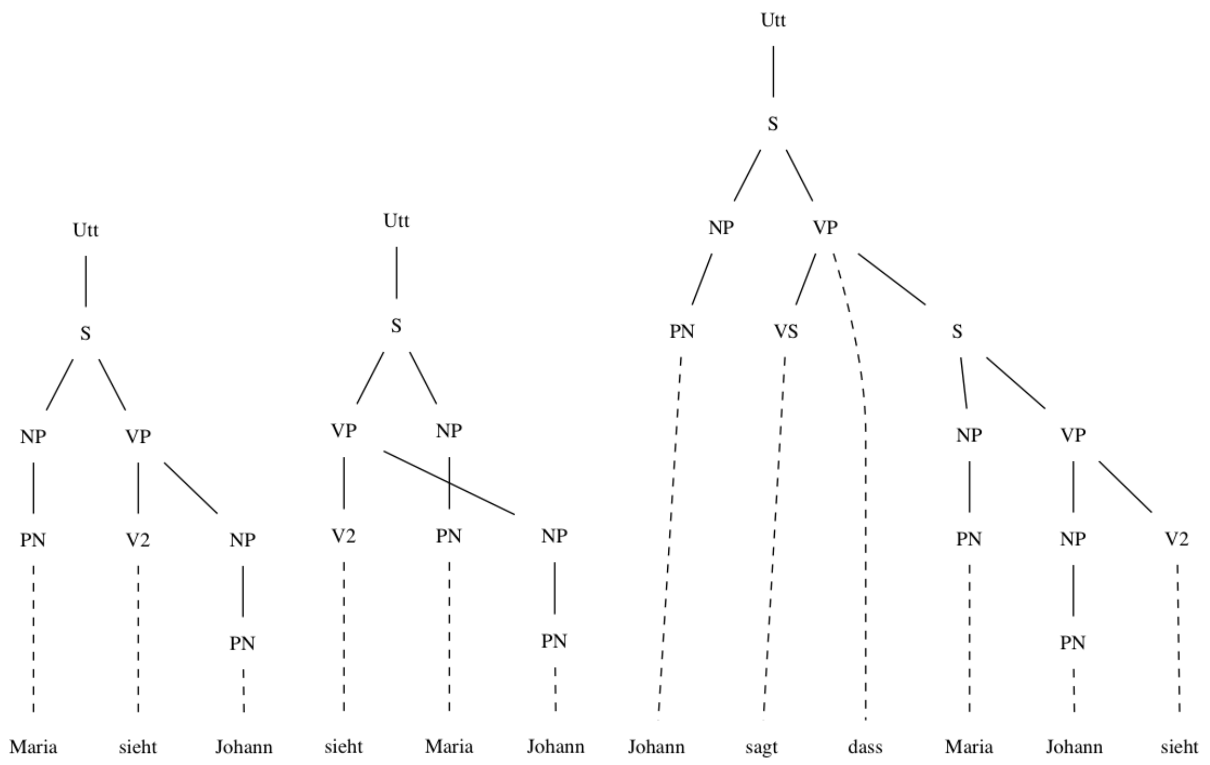
```
PredVP : NP -> VP -> S
UseV : V -> VP
ComplV2 : V2 -> NP -> VP
UsePN : PN -> NP
John, Mary : PN          -- Johann, Maria
sleep : V                -- schläft
see : V2                 -- sieht

AssertS: S -> Utt
QuestionS : S -> Utt
ComplVS : VS -> S -> VP
say : VS                 -- sagt
```

The linearization of S's and VP's is the crucial question. The problem is that German uses different word orders depending on the syntactic context of a sentence. In this question, we have to distinguish between three cases:

- **Main clause, SVO**: *Maria sieht Johann* ("Mary sees John", assertion)
- **Inverted clause, VSO**: *sieht Maria Johann* ("does Mary see John", question)
- **Subordinate clause, SOV**: *Johann sagt dass Maria Johann sieht* ("John says that Mary sees John", complement of VS; the word *dass* is added in the ComplVS rule)

If there is no object, these rules still apply by just omitting the O part.

Here is another illustration, showing parse trees and the corresponding abstract syntax trees:



```
AssertS
 (PredVP (UsePN Mary)
   (ComplV2 see (UsePN John)))
```

```
QuestionS
  (PredVP (UsePN Mary)
     (ComplV2 see (UsePN John)
```

```
AssertS (PredVP (UsePN John) (ComplVS say
       (PredVP (UsePN Mary)
             (ComplV2 see (UsePN John)))))
```

The task is to write a concrete syntax for German that correctly renders the word order in all these cases, without changing the abstract syntax.

Hint. The key to the solution is to use discontinuous constituents, that is, records that contain more strings than one.

Question 5: VP complement verbs (12p)

The GF Resource Grammar Library has the category VV for verbs that take verb phrase (VP) complements. Let us assume the abstract syntax of Question 2 and add the following rules:

```
ComplVV : VV -> VP -> VP      -- can sleep
can_VV : VV                   -- can (sleep)
want _VV : VV                 -- want (to sleep)
stop_VV : VV                  -- stop (sleeping)
NegPredVP : NP -> VP -> S     -- we don't sleep
```

As we can see, in English a VV can take its complement in at least three ways: bare infinitive, infinitive with the mark *to*, and the *ing* form. Looking at NegPredVP, we can also see that some VV's are **auxiliary verbs**, which means that their negation is formed directly (*I can't sleep*), whereas some are "ordinary verbs", which need an auxiliary "do" to form the negation (*I don't want to sleep*). None of these distinctions is universal across languages, so they should not enter the abstract syntax.

The task is now to revise the English concrete syntax of Question 2 so that it correctly handles VV verbs and negative predication. This means that you have to

- change the linearization types of V,V2, and VP
- change the linearizations of PredVP and ComplV2 accordingly
- define the linearization type of VV
- write the linearization rules for the five new functions listed above

You should make sure that your solution works for at least the following examples:

Mary wants to sleep here
we don't want to sleep here
we can see her
she can't see us
we stop sleeping
John doesn't stop sleeping

You may find it awkward to iterate ComplVV when the complement VP is built with an auxiliary:

we can't stop sleeping -- OK
we want to can sleep -- wrong

You don't need to solve this problem, but if you do have a solution, you can get a bonus point :-)