OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

INFORMATIK UND MATHEMATIK

# Natural Language Processing

**Thema:** Exercise sheet 4: CFG parsing and compositional semantics

**Von:** Timo Baumann, timo.baumann@oth-regensburg.de

**Datum:** 18.04.2024

**Abgabe bis:** 24.04.2024 24:00

---

## In-class exercises

### C.1 Montague semantics

Work on the following task in small groups (3–4 students; **NOT** your submission team members!) for 30 minutes. Take care (everyone!) to understand and be able to follow along what your fellow students are discussing and doing.

Given the parse and grammar on the example slides (page 11), create the derivation of `Every man loves a woman.` Which one of the two resulting formulae that are given on the slide is the outcome?

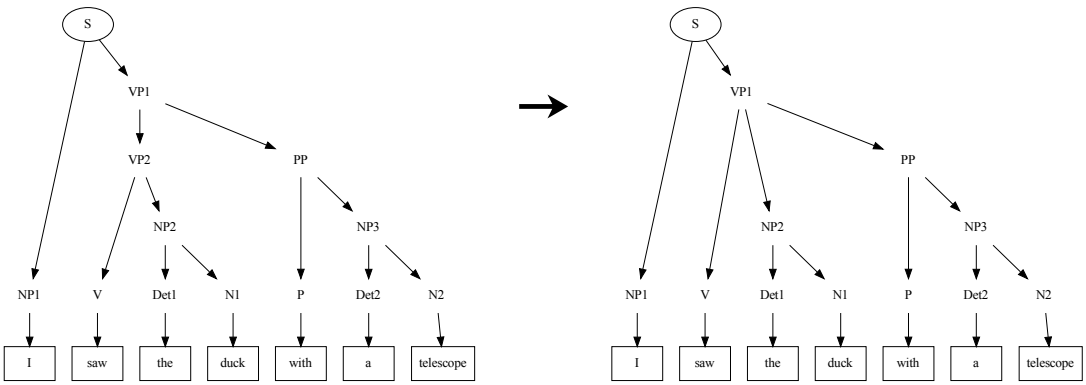
## H 4 Take-home exercises

Please hand in the exercise solutions via the Moodle course, one submission per submission team of 2-3 students.

### H 4.1 CKY for arbitrary context-free grammars

For the time being, the CKY algorithm only worked for grammars in Chomsky normal form (CNF). The goal of this exercise is to enable your parser to also work with grammars that are not in CNF. You do this by (a) transforming the grammar so that it is (almost) in CNF, i.e. rules are unary or they are binary and produce only non-terminals, (b) extending the algorith so that unary rules also work for non-terminals (e.g. NP → N), (c) re-transform the the parse tree to accord to the original grammar.

1. Extend the parsing algorithm such that it also accepts rules which produce unary non-terminals. Apply these rules after all binary rules have been checked. You may have to check unary rules multiple times (to account for chains of unary rules).
2. Add the function `is_relaxedCNF()` (or similarly named) to `Grammar` which checks if a grammar is in accordance to the requirements of the extended parsing algorithm.
3. extend the `Grammar` class so that rules that are not in normal form are replaced such that the resulting grammar is in the relaxed CNF as described above. Extend the class `Symbol` (or derive a new class from Symbol) so that you can recognize rules which you have changed during normalization. You do not need to consider ϵ-productions in your implementation (they are a bit harder to de-normalize).
4. extend the `ParseTree`/`ParseNode` classes so that they detect (and remove) extra nodes that are the result of your special rules introduced during normalization. For example, a rule VP → V NP PP will be displayed as shown on the right.




Document all functionality by providing appropriate tests with corresponding non-normalized grammars. In the case that you cannot solve all tasks, write down what does not work and speculate about why it doesn't work.

## H 4.2 Semantic Parsing

Sketch out how you can integrate semantic analysis using the lambda-calculus into your parser:

1. How can the additionally required information for words and productions be stored in grammar rules? (Both as text in the grammar file and in the object representations at runtime).
2. Write a grammar (adhering to your description provided above) which can parses at least one of the examples in the lecture or exercises and which then provides a meaning for that sentence.
3. Which representation in Python can you use to store lambda expressions and what mechanism can you use to convert the text-based representation in the grammar file to that representation?
4. How do you need to adapt the parser to support semantic parsing?

In this exercise, it is not necessary to actually implement semantic parsing; it is enough to describe in sufficient detail and sufficiently *concretely*, how you would tackle the task.

### Bonus points

An implementation of the semantic parsing algorithm that corresponds to the approach that you sketched out above (including a description of the grammar, knowledge base and some example parses) is worth up to 6 bonus points in the exam (which has a total of 100 points; 5 bonus points likely correspond to a half grade-improvement).

**Important note:** bonus points are **personal**, i.e., you may work in your regular submission team but in this case you have to provide information on how the bonus points are to be split among the team members (e.g., $1/3$ for each of three team members). You may also choose to work on this on your own, in order to be able to be elligible for all 6 bonus points. Given these caveats, you may of course not share your approach and/or results with other students.