University of Toronto, Department of Computer Science
**CSC 485F—Computational Linguistics, Summer 2020**

# Assignment 2

---

**Due date:** 23:59, Monday 15 June 2020.
*Late assignments will not be accepted without a valid medical certificate or other documentation of an emergency.*
*This assignment is worth 45% (CSC 485) of your final grade.*

- Fill out both sides of the assignment cover sheet, and staple together all answer sheets (in order) with the cover sheet (sparse side up) on the front. (Don't turn in a copy of this handout.)
- Please type your reports in no less than 12pt font; diagrams and tree structures may be drawn with software or neatly by hand.
- What you turn in must be your own work. You may not work with anyone else on any of the problems in this assignment. If you need assistance, contact the instructor or TA for the assignment.
- Any clarifications to the problems will be posted on the course bulletin board. You will be responsible for taking into account in your solutions any information that is posted there, or discussed in class, so you should check the page regularly between now and the due date.

# 1. Using features in grammars [10 marks]

Grammar 1 handles NPs of various different types. Grammar 2 is much simpler and easier to read, but doesn't appropriately constrain the NPs generated.

**Grammar 1**
**Rules:**
S → NPsg VPsg
S → NPpl VPpl
VPsg → Vsg NP
VPpl → Vpl NP
VPsg → Vsg NP PP
VPpl → Vpl NP PP
PP → P NP
NPsg → NPRP
NPsg → Det Nsg
NPpl → Det Npl
NPpl → Npl
NP → NPsg
NP → NPpl

**Lexicon:**
*Fred*: NPRP
*biscuits*: Npl
*feed*: Vpl
*feeds*: Vsg
*the*: Det
*dog*: Nsg
*puppies*: Npl
*with*: P

**Grammar 2**
**Rules:**
S → NP VP
VP → V NP
VP → V NP PP
PP → P NP
NP → N
NP → Det N

**Lexicon:**
*Fred*: NP
*biscuits*: N
*feed*: V
*feeds*: V
*the*: Det
*dog*: N
*puppies*: N
*with*: P

**A.** (2 marks) Code up Grammar 1 in TRALE (using types for non-terminals), and show the parse tree for the following sentence according to that grammar:

> Fred feeds the dog with biscuits.

Submit the grammar as the file `onea.pl` and the parse tree as `onea.gralej`. Your grammar should declare one type for each non-terminal listed in this grammar.

**B.** (7 marks) Code up Grammar 2 in TRALE, and augment it with features so as to restrict its language to that of Grammar 1. *You must use features. Do not add extra non-terminals.*

Submit this grammar as the file `oneb.pl`. Again, your grammar should declare one type for each non-terminal in this grammar, but you will need additional types to represent the values of the features that you introduce.

**C.** (1 mark) Show the parse tree for the sentence in part A above, this time using your augmented Grammar 2.

Submit this output as the file `onec.gralej`.

After connecting to the `teach.cs` server with X11 forwarding enabled:

```
ssh -X <your-utorid>@teach.cs.toronto.edu
```

the TRALE system can be run with this command:

```
/h/u2/csc485h/summer/pub/trale/trale -fsg
```

(which you are welcome to alias). Do *not* copy the TRALE or SICStus Prolog systems to your directory; run these *in situ* in your shell. You may find that, in order to use the graphical user interface that comes with TRALE, you must remove the very small default virtual memory limit that comes with accounts on `teach.cs`. In `tcsh`, you do this with the command:

```
limit vmemoryuse unlimited
```

If you have your own firewall, you may also need to modify its settings, so that the graphical user interface can tunnel through it. By default, TRALE attempts a range of ports beginning at 5001, and the port number can be specified through the environment variable INTERFACE_PORT.

Use File > Export > Gralej to render trees for your assignment submission.

**Hint:** The mnemonics *sg* and *pl* stand for *singular* and *plural*, the numbers for singular and plural.

## 2. Verb complements and gap features [20 marks]

Gap features enable us to assign the right grammatical functions to the arguments of a verb, which in turn guarantees that they will be assigned the right thematic roles. For example, in the grammar for the passive construction that we saw in class, we can associate the NP subject with the object position through a "gap" feature, so that, in the semantics, the subject NP can be interpreted as the Theme of the verb. Gap features aren't the only way to handle these cases, however. In this assignment, you will search for an alternative to handle a different example.

There are many more situations in which NPs are interpreted as if they occurred in positions in which they do not overtly occur than the passive. Consider the following sentences:

i. The student tended to sleep.

ii. The student appeared to sleep.

iii. The student promised the teacher to sleep.

iv. The student requested the teacher to sleep.

For example, in (i), *the student* is the Agent of *tended*, as we would presume because it is the subject of *tended*; but interestingly in (i), *the student* is also the Experiencer of *sleep*, even though it does not appear to be the subject of *sleep*.

In answering the three questions on the next page, assume the following:

- *Tend* has two thematic roles assignable: Agent and Theme. *Promise* and *request* have three: Agent, Theme and Beneficiary. *Sleep* has one: Experiencer; and *appear* has one: Theme.
- Every subject and object must be linked to at least one thematic role. (This is the mapping of thematic roles to grammatical functions that we saw in class.)
- Every grammatical function (e.g., subject, object) can be linked with at most one thematic role from the same verb. This does not mean that the NPs that bear those grammatical functions can only bear one thematic role, because NPs can bear more than one grammatical function.
- A complement clause (i.e., an embedded verb phrase or sentence) can be assigned the Theme thematic role from its verb. In other words, NPs are not the only constituents that can be assigned thematic roles.

**A.** (5 marks) For each of the sentences (ii)-(iv) above, for each thematic role assignable by either verb in the sentence, state which constituent receives that role. Submit your answer as the file `twoa.txt`.

**Hint:** The treatment of the NP *the teacher* is different in the last two sentences.

**B.** (14 marks) Devise a phrase structure grammar augmented with features for the above sentences. Be sure to give the necessary lexical entries for the four verbs, *tend* , *appear*, *promise*, and *request*, as well as *sleep*. Submit this grammar as the file `twob.pl`.

Use only features that are necessary to ensure the appropriate interpretation of NPs with respect to semantic roles.

To make the class's grammars a bit more uniform, we have given you a head start with the file `teach.cs:/h/u2/csc485h/summer/pub/twosub.pl`. You will have to add types and features to this. Do not remove types, and do not alter the subtyping or feature declarations that already appear there.

**C.** (1 mark) Parse sentence (ii) above (*The student appeared to sleep*) in TRALE using your grammar, and submit the resulting parse tree. There should be only one tree generated. This output should be submitted as the file `twoc.gralej`.

# Implementation details

In order for the grader to be able to semi-automatically test your work, each file must have the exact name and format specified in the following subsections.

**Some overall specifications for your files:**

- The first line of each file must be a comment (with `%`) with your name, login ID, and student ID.
- Each grammar rule, lexical entry, or sentence must be separated by one or more blank lines in its appropriate file.
- You should organize and use comments to remark upon how your grammar and lexicon function, just as you would programming language code, to make it easily understandable.
- Unlike the standard convention for specifying context-free grammars, the TRALE system requires lexical entries to provide the grammatical description on the right-hand side and the word on the left-hand side, e.g., `john ---> np.` Do not capitalize non-terminals or proper names like *John*, as capitalized tokens are interpreted as variables by Prolog. TRALE also does not allow the use of the pipe (|) for disjunction on either side. Use a separate lexical entry for each word that you wish to assign a feature structure to.

- In your grammar source code files, the grammar rules should all come first, followed by the lexical entries, in alphabetical order of the words.
- You need to handle all verbs in both their infinitive form (*take*) and their past tense form (*took*) in order to receive full marks. This means a perfect grammar should be able to handle recursive structures like *the student appeared to tend to promise the teacher to request the student to sleep*. You are encouraged to experiment with other forms, e.g. *taking, taken, takes*, but those will not be marked.
- Each `v_sem` should have features corresponding to its thematic roles (agent, theme, beneficiary, experiencer).

# 1 Test sentences

**Name of the file**: `sentences.pl`
**An example**:

```
% Your name, login ID and student ID go here.
test_sent([nadia,won,an,elephant]).
test_sent([i,could,have,demanded,a,rutabaga]).
test_sent([autopoiesis,always,reminded,her,of],fails).
```

**Note:** Use this file to provide extra sentences not mentioned in this assignment handout that you may have tried during development. The sentences must be delimited by commas, and *no* words should be capitalized. If a test sentence is supposed to fail, give `test_sent` an extra argument that tells us this.

### Output parse trees

All your output parse trees should be directed to files with the extension `.gralej`. All of these files should have your name, ID, and student number as the first line. Do not add any lines to the output you generate except commented lines (again with %).

# 2 What to submit

You must electronically submit your grammars, parse trees and test sentences. Specifically:

- A written report describing the design of your grammars and lexica, and your approach (how you met the requirements stated above). Don't forget also to discuss the limitations of your grammars.

  This PDF must also include a typed copy of the Student Conduct declaration as on the last page of this assignment handout. Type the declaration as it is and sign it by typing your name.
- A written report on your testing strategy—in particular, why you chose the sentences that you used for testing the grammar and lexicon. This should be no more than one page.

- The `.gralej` files that you generate.
- Your grammar source code.

**Note:** You do *not* need to submit other code that you write, e.g., code for running your test sentences, calling the pretty-printer, etc.

Submit all required files using the `submit` command on `teach.cs`:

```
% submit -c <course> -a A2 <filename-1>...<filename-n>
```

where `<course>` is either `csc485h` or `csc2501h`, and `<filename-1>` to `<filename-n>` are the *n* files you are submitting. Make sure every file you turn in contains a comment at the top that gives your name, your login ID on `teach.cs`, and your student ID number.

## 3 Grading scheme

We will test your grammars on the examples in this handout as well as on some held-out test sentences (i.e., sentences that you haven't seen).

Both grammars in a part B will be marked as to style and commenting (1/7), simplicity of your feature geometry and type system (2/7), in addition to correctness (4/7). The grammar in 1A will be marked as to style (1 mark) and simplicity (1 mark).

# CSC 485, Summer 2020: Assignment 2

Family name: _____ First name: _____

Student #: _____ Date: _____

I declare that this assignment, both my paper and electronic submissions, is my own work, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters and the Code of Student Conduct.

**Signature:** _____