University of Toronto, Department of Computer Science
CSC 485H/2501H: Computational linguistics, Fall 2020

Assignment 3

Due date: 23:59 on Thursday, December 10, 2020.
Late assignments will not be accepted without a valid medical certificate or other documentation of an emergency.
For CSC485 students, this assignment is worth 33% of your final grade.
For CSC2501 students, this assignment is worth 25% of your final grade.

- Type the written parts of your submission 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 Discourse forum for the class. 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.
- The starter code directory for this assignment is accessible on Teaching Labs machines at the path `/u/csc485h/fall/pub/a3/`. In this handout, code files we refer to are located in that directory.
- The first line of each file that you submit must be a comment (with `%`) providing your name, student ID number, and UTORid.

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:
Matri: NPRP
fruits: Npl
reward: Vpl
rewards: 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:
Matri: NP
fruits: N
reward: V
rewards: 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:

Matri rewards the dog with fruits.

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.[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 through ssh or NX[2], the TRALE system can be run with this command:

```
$ /u/csc485h/fall/pub/trale/trale -fsg
```

Do not copy the TRALE or SICStus Prolog systems to your directory; run these *in situ* in your shell. You may alias the relevant commands for convenience if you like.[3] You may find that, in order to use the graphical user interface that comes with TRALE, you must increase the default virtual memory limit that comes with accounts on `teach.cs` in `bash` (the default shell on `teach.cs`), you do this with the command:


```
$ ulimit -v 16777216
```


If you are running a firewall, you may also need to modify its settings so that the graphical user interface can get 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.

[1] Hint: the mnemonics *sg* and *pl* stand for *singular* and *plural*, the numbers for singular and plural.

[2] On Linux, you will need to use the `-X` flag with your `ssh` command when connecting to the `teach.cs` server. On Windows or MacOS, using the NX client to remotely access `teach.cs` is recommended, as you will need to interact with TRALE's GUI. Instructions can be found at https://www.teach.cs.toronto.edu/using_cdf/remote_access_server.html

[3] In `bash`, add `alias trale='/u/csc485h/fall/pub/trale/trale -fsg'` to your `.bashrc`, then log out. On subsequent logins, you should then be able to run TRALE just by running `trale` from the shell.

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. For example, consider the following sentences:

(i) The student wished to listen.

(ii) The student continued to listen.

(iii) The student promised the instructor to listen.

(iv) The student invited the instructor to listen.

In (i), *the student* is the Experiencer of *wished*, as we would presume because it is the subject of *wished*; but interestingly, in the same example *the student* is also the Agent of *listen*, even though it does not seem to be the subject of *listen*.

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

- *Wish* has two thematic roles assignable: Experiencer and Theme; *promise* and *invite* have three: Agent, Theme and Recipient; *listen* has one: Agent; and *continue* has two: Agent and 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 as exemplified above.
- 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 each of the two verbs in the sentence, state which constituent receives that role.[4]

Fill your answers in the appropriate lines in `twoa.txt` and submit this file.

(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 *wish*, *continue*, *promise*, and *invite*, as well as for *listen*. 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 some starter code in `twob.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 are already there.

Submit this grammar as your altered `twob.pl` file.

(c.) (1 mark) Parse sentence (ii) above (*The student continued to listen*) in TRALE using your grammar, and submit the resulting parse tree. There should be only one tree generated.

Submit this output as the file `twoc.grafej`.

[4]Hint: the treatment of the NP *the student* is different in the last two sentences.

Implementation and submission details

Each of your submitted files must have the name and format as specified below.

- Again, the first line of each file that you submit must be a comment (with `%`) providing your name, student ID number, and UTORid.
- 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 forms (*e.g.*, *take*) and their past tense forms (*e.g.*, *took*) in order to receive full marks. This means a perfect grammar should be able to handle recursive structures like *the student continued to wish to promise the instructor to invite the student to listen*. You are encouraged to experiment with other forms, such as *taking*, *taken*, *takes*, but those will not be marked.
- Each `v_sem` should have features corresponding to its thematic roles (agent, theme, recipient, experiencer) with the **exact same names**. For example, *wish* has two thematic roles: Experiencer and Theme, so its `v_sem` must have the features `experiencer` and `theme`.

1 Test sentences

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

```
% Your name, student ID number, and UTORid 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, student ID number, and UTORid as a comment on the first line. Do not add any lines to the output you generate except commented lines (again, using %).

2 What to submit

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

- `reports.pdf`, containing:
  - A written report describing the design of your grammars and lexica, and your approach (how you meet the requirements stated above). Don't forget also to discuss the limitations of your grammars.
  - 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.
  - 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.
- Your grammars' source code `.pl` files, the `.gralej` files that you generate, and any other files indicated for submission in this assignment.
- Your `sentences.pl` file.

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 A3 <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).

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

CSC 485H/2501H, Fall 2020: Assignment 3

Family name: _____ Given 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 _____