

# 11-711: Algorithms for NLP

## Homework Assignment #4: Parsing

Out: November 3, 2009  
Due: November 17, 2009

Consider the following simple context-free grammar and lexicon from Homework 3, which we will continue to use in this assignment:

Grammar:

- (1) S  $\rightarrow$  NP VP
- (2) NP  $\rightarrow$  det n
- (3) NP  $\rightarrow$  n
- (4) NP  $\rightarrow$  NP PP
- (6) VP  $\rightarrow$  v NP
- (7) VP  $\rightarrow$  v NP PP
- (8) PP  $\rightarrow$  p NP

Lexicon:

- I: n
- flowers: n, v
- garden: n, v
- hose: n, v
- in: p
- the: det
- watered: v
- with: p

### Problem 1

1. The Earley parsing algorithm presented in class cannot handle multiple part-of-speech assignments for input words. Describe how you would extend the algorithm so that it can.
2. Run your modified Earley algorithm on the input “I watered the flowers in the garden with the hose \$” using the grammar and POS assignments given at the beginning of this assignment. (Augment the grammar with a new rule  $S' \rightarrow S \$$ .) Your algorithm should only perform recognition, not parsing.

You may do this by hand, but we recommend writing code to execute the algorithm. In order to make this task easier, we are providing a basic Python framework very similar to that from Homework 3. If you choose to use our framework, first acquaint yourself with the Rule and Arc classes in `parsing.py` (unchanged from Homework 3). Then read through `parse-early.py` paying specific attention to the TODO items that indicate what sections you need to complete.

- (a) If you use our framework, it will output an html file that displays each set in the proper format. Open the output in a web browser and attach a print-out.

- (b) If you execute the algorithm by hand, show the final content of each item set  $S_i$  that is constructed by the algorithm, where each item in a  $S_i$  has the form  $[A \rightarrow B... \cdot C...X, j]$ .
3. Draw all possible parses for the sentence by tracing the backpointers.

## Problem 2

Use the LR table construction algorithm presented in class to build the SLR parsing table for the grammar given at the beginning of this assignment. (Augment it with a new rule  $S' \rightarrow S$ .)

1. Draw the FSA that results from the construction of the item sets  $S_i$ , plus the resulting action and goto tables.
2. Does your parsing table contain multiple entries (conflicting actions)? Briefly explain why or why not.

## Problem 3

1. Use the parsing table you constructed in Problem 2 to parse the sentence “I watered the flowers in the garden” with the GLR parsing algorithm. (Yes, you need to execute the algorithm by hand, but we shortened the sentence so it should not be too painful.) Assume a POS tagger has already converted the input into the form “n v det n prep det n \$”, which is the input to the GLR parser. Show the content of the GSS and the list of parse nodes at the end of processing each of the input words. Do not perform any ambiguity packing.
2. Can local ambiguity packing be applied at any point in the parsing of the above input? If so, indicate which paths in the GSS are collapsed together and which parse nodes get packed. Show the content of the GSS and the list of parse nodes at the end of processing each of the input words when ambiguity packing is performed.

## Problem 4

Consider the following context-free grammar fragment that has been augmented with feature structures:

- (1)  $VP \rightarrow v NP$   
 $((x1 \text{ subcat}) = *subj-obj)$   
 $((x0 \text{ obj}) = x2)$   
 $(x0 = x1)$
- (2)  $VP \rightarrow v NP PP$   
 $((x1 \text{ subcat}) = *subj-obj-comp)$   
 $((x0 \text{ obj}) = x2)$   
 $((x0 \text{ comp}) = x3)$   
 $(x0 = x1)$

Assume the following feature structures have already been created for the v, NP, and PP parse nodes:

v: { (root \*left) (subcat \*subj-obj-comp) }

NP: { (root \*flowers) (det \*) (agr \*3p) }

PP: { (root \*garden) (prep \*in) }

The parser is now applying the unification constraints in Rule (2) above and is creating the left-hand-side VP constituent.

1. Convert the feature structures into DAG representations, execute the unification constraints in the rule, and show the resulting feature structures of all parse nodes after performing the unifications as specified in the rule. If the unification fails at any point, clearly indicate the constraint being violated.
2. Now assume we are applying Rule (1) rather than Rule (2), using the original feature structures of the v and NP nodes. Again, convert the feature structures into DAG representations, execute the unification constraints in the rule, and show the resulting feature structures of all parse nodes after performing the unifications as specified in the rule. If the unification fails at any point, clearly indicate the constraint being violated.
3. Explain briefly how the unification constraints in the rules above can be used to disambiguate sentences such as “I left the flowers in the garden”.