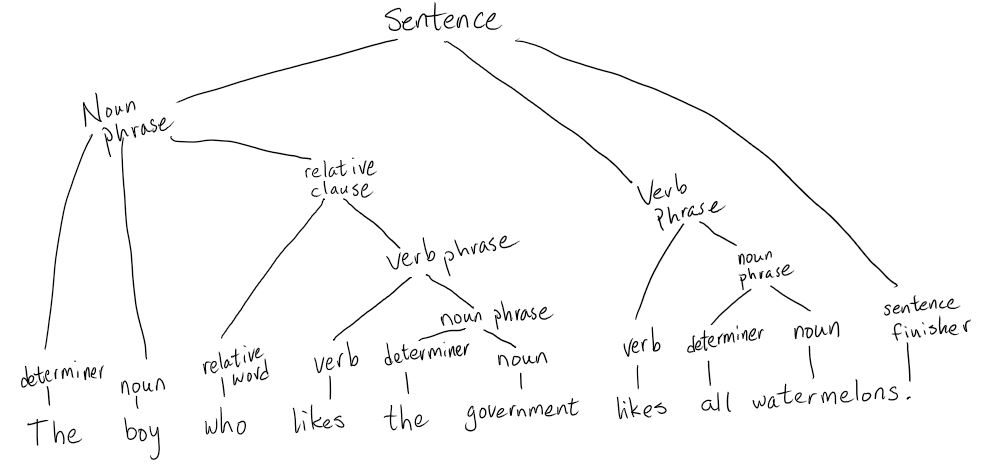
CSC 173

Prolog Week 3-4 Write Up

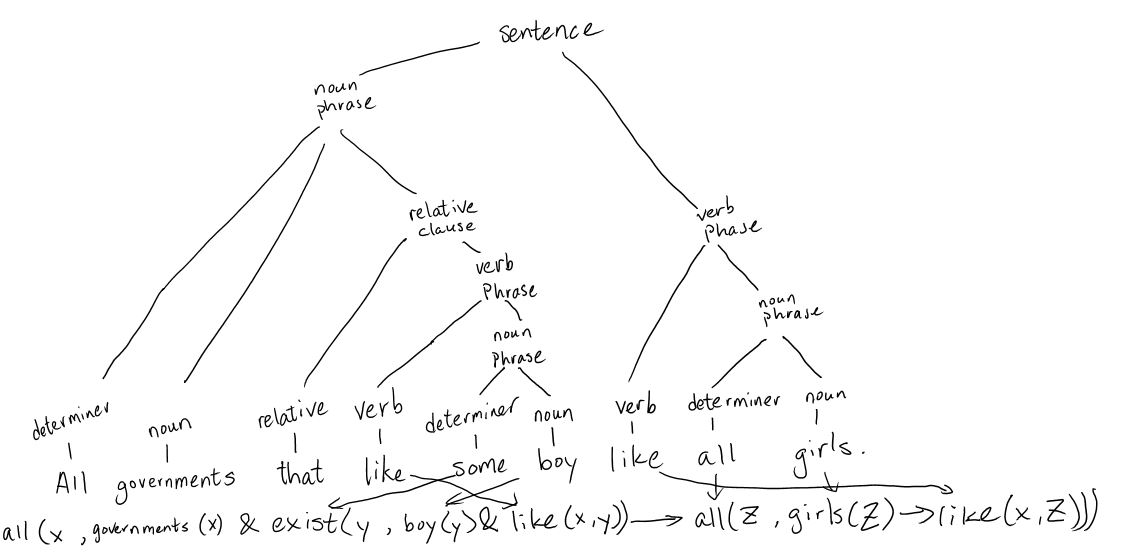
Yufei (Edward) Du

11/6/2015

* Introduction
  + This program will read in a line of English sentence from keyboard input, parse it into a parse tree, and then translate the sentence into logical expression (if possible).
  + There are 3 main parts in the program:
    - Input Reader (from Chapter 5 of the textbook)
    - Parser (Used the algorithm in the textbook and CB’s tutorial page)
    - Translator (Used the algorithm in CB’s tutorial page)
* Detailed Explanation of Each Part
  + Input Reader
    - The code for this part is provided directly by the textbook. I did not pay much attention to it. I will just summarize this part as “the textbook’s magic”.
  + Parser
    - The parser has 3 arguments: input as list, variable for output, and expected result (“[] by default”). It then parses the list to a parse tree by parsing each word according to CFGs emulating English grammar and facts as a “dictionary” that contains attributes of words.
    - The CFG:
      * Sentence -> Noun Phrase + Verb Phrase
      * Noun Phrase -> Determiner + Noun
      * Noun Phrase -> Noun
      * Noun Phrase -> Determiner + Noun + Relative Clause
      * Verb Phrase -> Verb
      * Verb Phrase -> Verb + Noun Phrase
      * Verb Phrase -> BeVerb + Adjective
      * Verb Phrase -> BeVerb + Noun Phrase
      * Relative Clause -> Relative(that,who,etc) + Verb Phrase
      * Relative Clause -> Relative(that,whom,etc) + Noun Phrase + Verb
    - Detailed grammar information, such as the vowel and singularity of a noun, are checked inside each rule. For example, in the rule of Noun, the program will get the information of the word’s vowel and singularity from the fact of this word and make sure the determiner matches the singularity and vowel. The following information are stored in facts
      * Noun: singularity, is vowel or not
      * Determiner: singularity, works with vowel or not
      * Verb: singularity, does it need a noun phrase after it or not
      * BeVerb(is,are): singularity
      * Adjective: nothing special
      * Relative Pronoun: Subjective or Objective
      * Sentence Finisher (punctuation marks): nothing special.
    - An example of the CFG:



* + - * Thus, the output is: sentence(noun\_phrase(determiner(the), noun(boy), rel\_clause(relative(who), verb\_phrase(verb(likes), noun\_phrase(determiner(the), noun(government))))), verb\_phrase(verb(likes), noun\_phrase(determiner(all), noun(watermelons))))
  + Translator
    - The translator translates the parse tree to a logical expression. It has 2 arguments: variable for output, and the parse tree input. It uses a CFG to read the parse tree and translate each node in the tree using rules. The inputs and outputs of each rules need to be connected to correctly print the output, so the argument of each rule has outputs and inputs from other rules (P1, P2, P3, etc.).
    - The CFG:
      * Sentence -> Noun Phrase + Verb Phrase
      * Noun Phrase -> Determiner + Noun
      * Noun Phrase -> Determiner + Noun + Relative Clause
      * Relative Clause -> Relative (that, who, etc.) + Verb Phrase
      * Relative Clause -> Relative (that, whom, etc.) + Noun Phrase + Verb
      * Verb Phrase -> Verb
      * Verb Phrase -> Verb + Noun Phrase
      * Verb Phrase -> BeVerb + Adjective
    - Example:



* + - * Output: all(x101, governments(x101)&exist(x102, boy(x102)&like(x101, x102))->all(x103, girls(x103)->like(x101, x103)))
* Run instruction
  + Read in input: read\_in(X). Then use keyboard input.
  + Read and parse: parse(X). Then use keyboard input.
  + Read, parse, and translate: translate(X). Then use keyboard input.
  + Just parse: sentence(X,input,[]).
  + Just translate: ss(X,input).