Computational Semantics and Pragmatics 2012

Raquel Fernández – ILLC, University of Amsterdam

Homework #2 Due: 19/11/2012, 10:00AM

The starting point for this set of exercises is the Haskell code in the companion files `COSPhw2_FOL.hs` and `COSPhw2_NL.hs` (based on code introduced in the book *Computational Semantics with Functional Programming* (*CSFP*)). The file `COSPhw2_FOL.hs` contains an implementation of the syntax and semantics of First Order Logic. The file `COSPhw2_NL.hs` contains an implementation of a simple natural language grammar and corresponding semantic representations.

All the exercises below ask you to extend these basic implementations. Each exercise focuses on a particular aspect (e.g. the addition of ditransitive verbs or of pre-nominal adjectives), but note that to make each of these aspects work you may have to extend the implementations with additional components that may not be explicitely mentioned in the exercises. In general, you probably want to enrich the vocabulary covered by the grammar (e.g. with more nouns, verbs, or determiners) so that you can construct a more varied set of sentences.

Submit two Haskell files with your extended versions of `COSPhw2_FOL.hs` and `COSPhw2_NL.hs`. For every exercise, include brief explanations (indicating which bit of code corresponds to which exercise) and the result of relevant sample queries as comments within the same Haskell file.

1. The definition of the logical form for the definite article *the* in `COSPhw2_NL.hs` follows the theory of definite descriptions proposed by Russell (see pages 131–132 of *CSFP*). Implement a FOL model in which the logical form of the sentences *Alice admires the princess*, *Alice admires every princess*, and *Alice admires some princess* are all true. Include the result of queries with `lfSent` and `eval` as comments within the Haskell file.

2. Extend the basic implementation with the quantifier *most* and with the quantified pronouns *someone* and *nothing*. Include the result of queries with `lfSent` and `eval` as comments within the Haskell file.

3. Extend your implementation with a ditransitive verb such as *offer*. Note that ditransitive verbs with a direct and an indirect object can be used in two types of constructions: a double NP construction such as *Alice offers Bob a drink* and an NP-PP construction such as *Alice offers a drink to Bob*. Make sure that the two types of syntactic constructions are licenced by your implementation and that both of them yield the same logical form. Include the result of queries with `lfSent` and `eval` as comments within the Haskell file.

4. Extend your implementation with adjectives such as *happy* or *blond* in pre-nominal positions, as in *a happy princess laughs*. Account for the fact that an indeterminate number of adjectives are possible (e.g. *some silly blond princess laughs*). The semantics of adjectives can be rather complex. You may want to limit yourself to intersective adjectives. Include the result of queries with `lfSent` and `eval` as comments within the Haskell file.

5. Extend your implementation with relational prepositions such as *on* or *under* to account for prepositional phrases that modify a noun as in the sentence *every boy on the roof cheered*. Include the result of queries with `lfSent` and `eval` as comments within the Haskell file.