Homework #4b

For Friday, February 18

⋆ 1. The set of propositional formulas in prefix form is defined inductively, as follows (the underlying set consists of strings of variables and logical symbols):

- $\bot$ is a prefix formula
- any variable p_i is a prefix formula
- if φ is a prefix formula, so is $\neg\varphi$
- if φ and ψ are prefix formulas, so is $\wedge\varphi\psi$
- if φ and ψ are prefix formulas, so is $\vee\varphi\psi$
- if φ and ψ are prefix formulas, so is $\rightarrow\varphi\psi$
- if φ and ψ are prefix formulas, so is $\leftrightarrow\varphi\psi$

Intuitively, this is just another notation for propositional formulas in which the connectives come *in front* of the arguments, instead of *in between* them. For example, one writes $\wedge p_1 p_2$ instead of $(p_1 \wedge p_2)$. Notice, however, that in this representation no parentheses are used.

a. Convert $\wedge \rightarrow p_1 \neg p_2 \vee p_3 p_4$ to a regular propositional formula.

b. Convert $((p_1 \vee p_2) \rightarrow ((\neg p_3) \vee p_4))$ to a prefix formula.

c. Define a function recursively that maps prefix propositional formulas to regular ones (you can assume that the set of prefix formulas is freely generated).

d. Define a function recursively that maps regular propositional formulas to prefix ones.

∘ 2. Prove unique readability for prefix formulas, i.e. that the set of prefix formulas is freely generated. This amounts to showing that there is only one way to "parse" a given formula.

∘ 3. In the programming language of your choice, define a data structure to represent propositional formulas as trees. (That is, a propositional formula is either a variable, or an operation with pointers to its arguments). Write a parser for propositional formulas, that is, a program that takes a string as input and turns it into a parse tree. The routine should print "ok" if successful, or "error" if the string is not a formula.

Now write routines that convert a formula to prefix form; that take an assignment of truth values to the variables as input and determine whether or not the resulting formula is true; and that determine whether there is *any* assignment that makes the formula true.

4. Do problems 1 and 2 on page 14 of van Dalen.

⋆ 5. Do problem 3 on page 14. In other words, show that for every θ, ψ, and φ, if φ is a subformula of ψ and ψ is a subformula of θ, then φ is a subformula of θ. (Hint: use induction on θ, and state the relevant property of θ clearly.)

6. Do problem 4 on page 14. In other words, show that if φ is a subformula of ψ and $\theta_0, \theta_1, \ldots, \theta_k$ is a formation sequence for ψ, then for some $i \leq k$, $\varphi = \theta_i$. Be precise: use the definitions of $PROP$, formation sequences, and subformulas presented in class.

7. Do problem 5–8 on page 14–15.

8. Do problem 9 on page 15. Note that the "number of connectives" in φ counts all the occurences of $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, and $\perp$; this can be defined more formally by recursion on φ. (Hint: use induction on φ to show that the number of subformulas of φ is always at most twice the number of connectives plus one.)

9. Show that if φ is any propositional formula, there is a formation sequence for φ that involves only subformulas of φ.

∘ 10. Do problem 11 on page 15.