Assignment 5

due 6pm Thursday, October 7, 2021

Problem 1 (3 points)

Remember the definition of substitution for propositional formulas:

$$\begin{aligned} A[B/p] &= B \quad \text{if } A \text{ is the variable } p \\ A[B/p] &= A \quad \text{if } A \text{ is any other atomic formula} \\ (\neg A)[B/p] &= \neg(A[B/p]) \\ (A_0 \star A_1)[B/p] &= A_0[B/p] \star A_1[B/p] \quad \text{for any binary connective } \star \end{aligned}$$

Remember also that a *truth assignment* τ is a function that assigns a truth value $\top$ or $\bot$ to each variable. If τ is such a truth assignment, p is a variable, and b is a truth value, $\tau[p \mapsto b]$ is the function defined by

$$(\tau[p \mapsto b])(q) = \begin{cases} b & \text{if } q \text{ is } p \\ \tau(q) & \text{otherwise} \end{cases}$$

In other words, $\tau[p \mapsto b]$ is the (possible) modification of τ that maps p to b and leaves the rest of the assignment the same.

Prove that for any A, B, p, and τ,

$$[\![A[B/p]]\!]_\tau = [\![A]\!]_{\tau[p \mapsto [\![B]\!]_\tau]}.$$

In other words, we can evaluate $A[B/p]$ at τ by evaluating B at τ, and then evaluating A with p replaced by that result.

Problem 2 (2 points)

Use the previous problem to show that for any A, B, and p, if A is valid, then $A[B/p]$ is valid.

Problem 3 (3 points)

Consider the formula $p_1 \leftrightarrow p_2 \leftrightarrow \cdots \leftrightarrow p_n$. Convince yourself that this formula is true if and only if an even number of variables are assigned the value *false*.

Prove that the smallest CNF formula that is equivalent to this must have at least 2^{n-1} clauses. (Hint: it suffices to show that any DNF formula equivalent to the negation, which says that an odd number of variables are false, has to have at least 2^{n-1} cubes. Start by showing that in any such DNF formula, each cube has to contain all the variables.)

Problem 4 (3 points)

Write a function in Lean that implements substitution for propositional formulas, and test it on one or two examples.

Problem 5 (4 points)

In class and in the textbook, we discuss a Lean function `PropForm.eval` that evaluates a propositional formula with respect to a truth assignment. Define a similar function, `CnfForm.eval`, that evaluates a formula in conjunctive normal form. (Do it directly: don't translate it to a propositional formula.)

Problem 6 (4 points)

In class and in the textbook, we define a Lean data type `NnfForm` for NNF formulas, and we define a function `PropForm.toNnfForm` that converts a propositional formula to one in NNF. Notice that the method we used to expand $\leftrightarrow$ can lead to an exponential increase in length.

Define a Lean data type `EnnfForm` for *extended* negation normal form formulas, which adds the $\leftrightarrow$ connective to ordinary NNF. Then define a function that translates any propositional formula to an extended NNF formula.