Assignment 2

Theories of Computation: Assignment 2

due on Fri 14 March, 12:00

Suzie is a colleague of yours, studying the first year of their Computer Science degree at Birmingham. Because of this, Suzie learned about digital logic during their first semester.

Suzie writes a context-free grammar to generate logical expressions over the alphabet $\Sigma = \{\texttt{0},\ \texttt{1},\ .,\ +,\ \neg,\ (,\ )\}$. They represent this CFG in Backus-Naur Form as follows:

$$\Rightarrow A \quad ::= \quad \texttt{0} \mid \texttt{1} \mid (A + A) \mid (A \mathbin{.} A) \mid \neg A$$

Note that, for clairity, Suzie's grammar uses $\neg A$ to mean the negation of the logical expression A, instead of $\overline{A}$ as taught in *Computer Systems & Professional Practice*.

Given any expression E generated by the grammar, Suzie defines the *depth* of E as the *number of triangles in the longest branch* of its derivation tree *minus 1*. For example, the depth of $\texttt{1}$ is 0, the depth of $\neg\texttt{0}$ is 1, and the depth of $(\texttt{0} + (\texttt{0} + \texttt{0}))$ is 2.

Exercise 1. *Draw a derivation tree for the expression* $((\texttt{0}.\texttt{1}) + \neg(\texttt{0}.\texttt{0}))$ *and, by doing so, deduce the depth of the expression.* ***[3 points]***

Suzie writes an algorithm for computing the depth n of an expression generated by their grammar. When given an expression of depth $n \leq 10$, it returns an answer in $8n^3 + 8$ milliseconds; for trees of higher depth, it returns an answer in $2n + n^2 + 18$ milliseconds.

Exercise 2. *Show that Suzie's depth-checking algorithm is in the complexity class* $O(n^2)$. ***[3 points]***

Suzie realises that there is a relationship between the depth of an expression and the total number of Boolean symbols ($\texttt{0}$'s and $\texttt{1}$'s) in that expression.

Exercise 3. *Prove, using structural induction, that the total number of Boolean symbols in a given expression* E *generated by Suzie's CFG is at most* 2^n*, where* n *is the depth of* E. ***[4 points]***

When you submit, look for the word 'Submitted!' – otherwise, you may have not submitted properly.