COMPSCI/SFWRENG 2FA3

Discrete Mathematics with Applications II

Winter 2020

Assignment 2

Dr. William M. Farmer

McMaster University

Revised: January 25, 2020

Assignment 2 consists of two problems. You must write your solutions to the problems using LaTeX.

Please submit Assignment 2 as two files, `Assignment_2_`*`YourMacID`*`.tex` and `Assignment_2_`*`YourMacID`*`.pdf`, to the Assignment 2 folder on Avenue under Assessments/Assignments. *`YourMacID`* must be your personal MacID (written without capitalization). The `Assignment_2_`*`YourMacID`*`.tex` file is a copy of the LaTeX source file for this assignment (`Assignment_2.tex` found on Avenue under Contents/Assignments) with your solution entered after each problem. The `Assignment_2_`*`YourMacID`*`.pdf` is the PDF output produced by executing

```
pdflatex Assignment_2_YourMacID
```

This assignment is due **Sunday, February 2, 2020 before midnight.** You are allow to submit the assignment multiple times, but only the last submission will be marked. **Late submissions and files that are not named exactly as specified above will not be accepted!** It is suggested that you submit your preliminary `Assignment_2_`*`YourMacID`*`.tex` and `Assignment_2_`*`YourMacID`*`.pdf` files well before the deadline so that your mark is not zero if, e.g., your computer fails at 11:50 PM on February 2.

Although you are allowed to receive help from the instructional staff and other students, your submission must be your own work. Copying will be treated as academic dishonesty! If any of the ideas used in your submission were obtained from other students or sources outside of the lectures and tutorials, you must acknowledge where or from whom these ideas were obtained.

Problems

1. **[10 points]** Let C be a set of *coefficients* closed under addition and multiplication such as the integers $\mathbb{Z}$, the rational numbers $\mathbb{Q}$, or the real numbers $\mathbb{R}$. A *polynomial over* C is a mathematical expression that is constructed from an *indeterminant* x and members of C by applying addition $(+)$ and multiplication $(*)$ operators. For example, $x*((2*x)+3)$ is a polynomial over $\mathbb{Z}$. Let P be the set of polynomials over some C. The *value of a polynomial* $p \in P$ *at* $c \in C$ is the result of replacing the indeterminant x with c. For example, the value of $x*((2*x)+3)$ at 5 is $5*((2*5)+3) = 65$.

   Let Poly be the inductive type defined by the following constructors:

   $\mathsf{X} : \mathsf{Poly}.$

   $\mathsf{Coeff} : \mathbb{Q} \to \mathsf{Poly}.$

   $\mathsf{Sum} : \mathsf{Poly} \times \mathsf{Poly} \to \mathsf{Poly}.$

   $\mathsf{Prod} : \mathsf{Poly} \times \mathsf{Poly} \to \mathsf{Poly}.$

   The members of Poly represent the polynomials over $\mathbb{Q}$. Define

   $\mathsf{val} : \mathsf{Poly} \times \mathbb{Q} \to \mathbb{Q}$

   by structural recursion using pattern matching so that, for all $p \in \mathsf{Poly}$ and $q \in \mathbb{Q}$, $\mathsf{val}(p, q)$ is the value of p at q.

Jatin Chowdhary — Chowdhaj — February 7^{th}, 2020

Solution:

1. The 1^{st} case to handle is the zero degree polynomial. However, since exponents are not defined, we will just call this *single numbers*. For example: f(x) = 5, f(x) = 4.

val(_ N) = _ + N

Note: The underscore (_) is a wildcard and can represent anything. But in this case it will be restricted to a Poly or a Number. And 'N' will be restricted to a Number.

2. The 2^{nd} case to handle is first degree polynomials without coefficients. For example: f(x) = x

val(X, Q) = Q

3. The 3^{rd} case to handle is coefficients. For example: f(x) = 5x

val(Coeff(N), Q)) = Q × N

4. The 3^{rd} case to handle is the sum of two polynomials.

Note: P1 and P2 are polynomials.

val((sum(P1, P2)), Q) = val(P1, Q) + val(P2, Q)

5. The 4^{th} case to handle is the product of two polynomials.

Note: P1 and P2 are polynomials.

val((prod(P1, P2)), Q) = val(P1, Q) $\times$ val(P2, Q)

2. **[10 points]** Let Bit be the inductive set defined by the following constructors:

   Zero : Bit.

   One : Bit.

   The members of Bit represent 0 and 1.

   Let BinNum be the inductive set defined by the following constructors:

   Nil : BinNum.

   Join : BinNum $\times$ Bit $\rightarrow$ BinNum.

   The members of BinNum not equal to Nil represent binary numerals; Nil represents an empty numeral. For example,

   $$\mathsf{Join}(\mathsf{Join}(\mathsf{Join}(\mathsf{Nil}, \mathsf{One}), \mathsf{Zero}), \mathsf{One}),$$

   represents the binary number 101.

   The function

   $$\mathsf{len} : \mathsf{BinNum} \rightarrow \mathbb{N}$$

   maps a member of BinNum to its length. len is defined by the following equations using recursion and pattern matching:

   $\mathsf{len}(\mathsf{Nil}) = 0.$

   $\mathsf{len}(\mathsf{Join}(u, b)) = \mathsf{len}(u) + 1.$

   The function

   $$\mathsf{val} : \mathsf{BinNum} \rightarrow \mathbb{N}$$

   maps a member of BinNum to the value of the binary numeral it represents. For example,

   $$\mathsf{val}(\mathsf{Join}(\mathsf{Join}(\mathsf{Join}(\mathsf{Nil}, \mathsf{One}), \mathsf{Zero}), \mathsf{One})) = (101)_2 = 5.$$

   val is defined by the following equations using recursion and pattern matching:

$$\mathsf{val}(\mathsf{Nil}) = 0.$$
$$\mathsf{val}(\mathsf{Join}(u, \mathsf{Zero})) = 2 * \mathsf{val}(u).$$
$$\mathsf{val}(\mathsf{Join}(u, \mathsf{One})) = (2 * \mathsf{val}(u)) + 1.$$

The function

$$\mathsf{add} : \mathsf{BinNum} \times \mathsf{BinNum} \to \mathsf{BinNum}$$

is intended to implement addition on members of BinNum. It is defined by the following equations using recursion and pattern matching:

$$\mathsf{add}(u, \mathsf{Nil}) = u.$$
$$\mathsf{add}(\mathsf{Nil}, u) = u.$$
$$\mathsf{add}(\mathsf{Join}(u, \mathsf{Zero}), \mathsf{Join}(v, \mathsf{Zero})) = \mathsf{Join}(\mathsf{add}(u, v), \mathsf{Zero}).$$
$$\mathsf{add}(\mathsf{Join}(u, \mathsf{One}), \mathsf{Join}(v, \mathsf{Zero})) = \mathsf{Join}(\mathsf{add}(u, v), \mathsf{One}).$$
$$\mathsf{add}(\mathsf{Join}(u, \mathsf{Zero}), \mathsf{Join}(v, \mathsf{One})) = \mathsf{Join}(\mathsf{add}(u, v), \mathsf{One}).$$
$$\mathsf{add}(\mathsf{Join}(u, \mathsf{One}), \mathsf{Join}(v, \mathsf{One})) = \mathsf{Join}(\mathsf{add}(\mathsf{add}(u, v), \mathsf{Join}(\mathsf{Nil}, \mathsf{One})), \mathsf{Zero}).$$

Notice that the algorithm behind the definition is essentially the same algorithm that children learn to add numbers represented as decimal numerals. The last equation is a bit complicated because it involves a carry of 1.

Lemma 1. For all $u, v \in \mathsf{BinNum}$,

$$\mathsf{len}(\mathsf{add}(u, v)) \leq \mathsf{len}(u) + \mathsf{len}(v).$$

Theorem 1. For all $u, v \in \mathsf{BinNum}$,

$$\mathsf{val}(\mathsf{add}(u, v)) = \mathsf{val}(u) + \mathsf{val}(v).$$

Theorem 1 states that add correctly implements addition on the members of BinNum.

Prove Theorem 1 assuming Lemma 1. (You are not required to prove Lemma 1.) Hint: Use strong induction with $P(n) \equiv \mathsf{val}(\mathsf{add}(u, v)) = \mathsf{val}(u) + \mathsf{val}(v)$ for all $u, v \in \mathsf{BinNum}$ such that $n = \mathsf{len}(u) + \mathsf{len}(v)$.

Put your name, MacID, and date here.

Put your solution here.