Problem Set 6
Due: Wednesday, May 15th by 11:00pm

Instructions

Solutions submission. You must submit your solution via Gradescope. In particular:

- Submit a single PDF file containing your solutions to Tasks 1–6 (and optionally 7). Follow the prompt on Gradescope to link tasks to your pages.

Task 1 – Barking Up the Strong Tree [20 pts]
The function \( f(m) \) is defined for \( m \in \mathbb{N} \) recursively as follows:
\[
\begin{align*}
    f(0) &= 1 \\
    f(1) &= 2 \\
    f(m) &= f(m - 1) + 2 \cdot f(m - 2) & \text{if } m \geq 2
\end{align*}
\]

Use strong induction to prove that

\[ \forall n \in \mathbb{N} \ (f(n) = 2^n) \]

Write an English proof, following the template given in lecture.

Task 2 – Live Strong and Prosper [20 pts]
The function \( g(m) \) is defined for integers \( m \geq 1 \) recursively as follows:
\[
\begin{align*}
    g(1) &= 0 \\
    g(2l) &= 1 + g(l) & \text{where } l \geq 1 \text{ is an integer} \\
    g(2l + 1) &= 1 + g(l) & \text{where } l \geq 1 \text{ is an integer}
\end{align*}
\]

The first line gives the definition of \( g \) for \( m = 1 \), the second line gives the definition for even \( m \), and the third line gives the definition for odd \( m \geq 3 \). Since these three cases are mutually exclusive and exhaustive, they define \( g \) completely.

Use strong induction on \( n \) to show that \( 2^{g(n)} \leq n \) for all integers \( n \geq 1 \).

Write an English proof, following the template given in lecture.
Recall the definition of lists of numbers from lecture:

**Basis Step:** \textit{nil} \in \textbf{List}

**Recursive Step:** for any \(a \in \mathbb{Z}\), if \(L \in \textbf{List}\), then \(a :\!\!\!: L \in \textbf{List}\).

For example, the list \([1, 2, 3]\) would be created recursively from the empty list as \(1 :\!\!\!: (2 :\!\!\!: (3 :\!\!\!: \text{nil}))\). We will consider "::" to associate to the right, so \(1 :\!\!\!: 2 :\!\!\!: 3 :\!\!\!: \text{nil}\) means the same thing.

The next two problems use three recursively-defined functions. The first is \textit{len}, which calculates the length of the list. It is defined recursively as follows:

\[
\begin{align*}
\text{len}(\text{nil}) & := 0 \\
\text{len}(a :\!\!\!: L) & := 1 + \text{len}(L) \quad \forall a \in \mathbb{Z}, \forall L \in \textbf{List}
\end{align*}
\]

The second function, \textit{concat}, which concatenates two lists into a single list, is defined by:

\[
\begin{align*}
\text{concat}(\text{nil}, R) & := R \quad \forall R \in \textbf{List} \\
\text{concat}(a :\!\!\!: L, R) & := a :\!\!\!: \text{concat}(L, R) \quad \forall a \in \mathbb{Z}, \forall L, R \in \textbf{List}
\end{align*}
\]

For example, we get \(\text{concat}(1 :\!\!\!: 2 :\!\!\!: \text{nil}, 3 :\!\!\!: \text{nil}) = 1 :\!\!\!: 2 :\!\!\!: 3 :\!\!\!: \text{nil}\) from these definitions.

The third function, \textit{positives}, which returns only the positive numbers in the list, is defined by:

\[
\begin{align*}
\text{positives}(\text{nil}) & := \text{nil} \\
\text{positives}(a :\!\!\!: L) & := \text{positives}(L) \quad \text{if } a \leq 0 \quad \forall a \in \mathbb{Z}, \forall L \in \textbf{List} \\
\text{positives}(a :\!\!\!: L) & := a :\!\!\!: \text{positives}(L) \quad \text{if } a > 0 \quad \forall a \in \mathbb{Z}, \forall L \in \textbf{List}
\end{align*}
\]

For example, from these definitions, we get \(\text{positives}(-1 :\!\!\!: 2 :\!\!\!: -3 :\!\!\!: \text{nil}) = 2 :\!\!\!: \text{nil}\).

**Task 3 – List Me By a Mile** [20 pts]

\textbf{a)} Write a chain of equalities, citing the appropriate definitions, showing that

\[
\text{concat}(1 :\!\!\!: 2 :\!\!\!: \text{nil}, 3 :\!\!\!: \text{nil}) = 1 :\!\!\!: 2 :\!\!\!: 3 :\!\!\!: \text{nil}
\]

\textbf{b)} Write a chain of equalities, citing the appropriate definitions, showing that

\[
\text{positives}(1 :\!\!\!: -2 :\!\!\!: 3 :\!\!\!: \text{nil}) = 1 :\!\!\!: 3 :\!\!\!: \text{nil}
\]

\textbf{c)} Use structural induction to prove that

\[
\forall L \in \textbf{List} \; (\text{len}(\text{positives}(L)) \leq \text{len}(L))
\]

**Task 4 – List Me, List Me, Now You Gotta Kiss Me** [18 pts]

Let \(R, S \in \textbf{List}\). Use structural induction on \(L\) to prove that

\[
\forall L \in \textbf{List} \; (\text{concat}(\text{concat}(L, R), S) = \text{concat}(L, \text{concat}(R, S))
\]

i.e., that \textit{concat} is associative.
Task 5 – A Few of My Favorite Strings [12 pts]

For each of the following, write a recursive definition of the set of strings satisfying the given properties. Briefly justify that your solution is correct.

a) Binary strings that start with 1 and have odd length.

b) Binary strings where every 1 is immediately followed by a 00.

c) Binary strings with an odd number of 1s.

Task 6 – Donny, You’re Out of Your Element [18 pts]

Let $A$ and $B$ be the following sets:

\[
A := \{ n \in \mathbb{Z} : 3 \mid n \} \\
B := \{ n \in \mathbb{Z} : (3 \mid n) \lor (6 \mid n) \}
\]

Write an English proof that $A = B$.

Do not use the Meta Theorem template. Instead, prove the necessary biconditional by proving each implication independently.

Task 7 – Extra Credit: Stone By the Company He Keeps [0 pts]

Consider an infinite sequence of positions $1, 2, 3, \ldots$ and suppose we have a stone at position 1 and another stone at position 2. In each step, we choose one of the stones and move it according to the following rule: Say we decide to move the stone at position $i$; if the other stone is not at any of the positions $i + 1, i + 2, \ldots, 2i$, then it goes to $2i$, otherwise it goes to $2i + 1$.

For example, in the first step, if we move the stone at position 1, it will go to 3 and if we move the stone at position 2 it will go to 4. Note: no matter how we move the stones, they will never be at the same position.

Use induction to prove that, for any given positive integer $n$, it is possible to move one of the stones to position $n$. For example, if $n = 7$ first we move the stone at position 1 to 3. Then, we move the stone at position 2 to 5. Finally, we move the stone at position 3 to 7.

---

1 No punny shtuff — Nihilist