1 Some Primitive Recursive Functions (30)

Background
We have shown in class that the function $n \mapsto p_n$, where $p_n$ is the $n$th prime, is primitive recursive. Here is slight extension of this result.

Make sure to lean heavily on the results from lecture, writing down an explicit p.r. definition is far too tedious for this.

Task

A. Show that the function $\text{val}(x, n)$ that returns the least $k \geq 0$ such that $p_n^k$ divides $x$ is primitive recursive.

B. Show that prime decomposition is primitive recursive in the sense that the following function is primitive recursive: $\text{pd}(x) = \langle n_1, e_1, n_2, e_2, \ldots, n_r, e_r \rangle$ where

$$x = p_{n_1}^{e_1} p_{n_2}^{e_2} \cdots p_{n_r}^{e_r}$$

and $n_1 < n_2 < \ldots < n_r$, $e_i > 0$.

2 Length and Sequence Numbers (30)

Background
A polyadic function $\langle . \rangle : \mathbb{N}^* \to \mathbb{N}$ is called a coding function if there are “inverse” functions $\text{len} : \mathbb{N} \to \mathbb{N}$ and $\text{dec} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that

$$\text{len}(\langle a_1, a_2, \ldots, a_n \rangle) = n,$$
$$\text{dec}(\langle a_1, a_2, \ldots, a_n \rangle, i) = a_i, \quad 1 \leq i \leq n$$

for all sequences $a_1, a_2, \ldots, a_n$. Note that we are using 1-indexing. Thus $\text{len}$ determines the length of the sequence and $\text{dec}$ decodes it back into its elements. Moreover, these functions are supposed to be easily computable, but let’s ignore this for the time being. Consider an arbitrary pairing function $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with unpairing functions $\pi_1$ and $\pi_2$ such that $\pi_i(\pi(x_1, x_2)) = x_i$. 
Here is a way to define a coding function based on $\pi$ that explicitly encodes the length of the sequence, which greatly facilitates decoding. First, extend $\pi$ to any number $n \geq 3$ of arguments by setting

$$\pi(x_1, x_2, \ldots, x_n) = \pi(\pi(x_1, \ldots, x_{n-1}), x_n)$$

Thus, we interpret $\pi$ as a binary operation that associates to the left. Then define $\langle . \rangle : \mathbb{N}^* \rightarrow \mathbb{N}$ via

$$\langle \text{nil} \rangle = 0$$
$$\langle a_1, \ldots, a_n \rangle = \pi(a_1, a_2, \ldots, a_n, n - 1) + 1$$

Note that $\text{len}$ is trivial to compute here.

**Task**

A. Show that the sequence coding function $\langle . \rangle$ is injective.

B. Explain how to compute the decoding functions $\text{len}$ and $\text{dec}$.

C. Suppose $\pi$ is bijective. Is it necessarily the case that $\langle . \rangle$ is bijective?

D. What would happen if we defined the expanded function $\pi$ by associating to the right, as in

$$\pi(x_1, x_2, \ldots, x_n) = \pi(x_1, \pi(x_2, \ldots, x_{n-1}))$$

### 3 Primitive Recursive Word Functions (40)

**Background**

We defined primitive recursive functions on the naturals. A similar definition would also work for words over some alphabet $\Sigma$. We write $\varepsilon$ for the empty word and $\Sigma^*$ for the set of all words over $\Sigma$. Consider the clone of word functions generated by the basic functions

- **Constant empty word** $E : (\Sigma^*)^0 \rightarrow \Sigma^*$, $E() = \varepsilon$
- **Append functions** $S_a : \Sigma^* \rightarrow \Sigma^*$, $S(x) = xa$

and closed under **primitive recursion** over words: suppose we have a function $g : (\Sigma^*)^n \rightarrow \Sigma^*$ and a family of functions $h_a : (\Sigma^*)^{n+2} \rightarrow \Sigma^*$, where $a \in \Sigma$. We can then define a new function $f : (\Sigma^*)^{n+1} \rightarrow \Sigma^*$ by

$$f(\varepsilon, y) = g(y)$$
$$f(xa, y) = h_a(x, f(x, y), y) \quad a \in \Sigma$$

We will call the members of this clone the word primitive recursive (w.p.r.) functions.
Task

1. Show that the reversal operation \( \text{rev}(x) = x_n x_{n-1} \ldots x_1 \) is w.p.r.

2. Show that the prepend operations \( \text{pre}_a(x) = a x \) are w.p.r.

3. Show that the concatenation operation \( \text{cat}(x, y) = x y \) is w.p.r.

4. Show that every primitive recursive function is also a word primitive recursive function. By this we mean that for every p.r. function \( f : \mathbb{N}^k \to \mathbb{N} \) there is a w.p.r. function \( F : (\Sigma^*)^k \to \Sigma^* \) so that \( f(x) = D(F(C(x))) \) where \( C \) and \( D \) are simple coding and decoding functions (between numbers and words).

Comment

The opposite direction also holds, but it’s a bit more tedious to prove (yet more coding). Extra Credit if you want to do this.