planetmath.org

Math for the people, by the people.

bounded minimization

Canonical name	BoundedMinimization
Date of creation	2013-03-22 19:05:18
Last modified on	2013-03-22 19:05:18
Owner	CWoo (3771)
Last modified by	CWoo (3771)
Numerical id	11
Author	CWoo (3771)
Entry type	Definition
Classification	msc 03D20
Synonym	bounded sum
Related topic	UnboundedMinimization
Related topic	RecursiveFunction
Related topic	BoundedMaximization
Defines	bounded summation
Defines	bounded product

One useful way of generating more primitive recursive functions from existing ones is through what is known as *bounded summation* and *bounded product*. Given a primitive recursive function $f : \mathbb{N}^{m+1} \to \mathbb{N}$, define two functions $f_s, f_p : \mathbb{N}^{m+1} \to \mathbb{N}$ as follows: for $\boldsymbol{x} \in \mathbb{N}^m$ and $y \in \mathbb{N}$:

$$f_s(\boldsymbol{x}, y) := \sum_{i=0}^{y} f(\boldsymbol{x}, i)$$

$$f_p(\boldsymbol{x}, y) := \prod_{i=0}^{y} f(\boldsymbol{x}, i)$$

These are easily seen to be primitive recursive, because they are defined by primitive recursion. For example,

$$f_s(\boldsymbol{x}, 0) = f(\boldsymbol{x}, 0), \quad \text{and} \quad f_s(\boldsymbol{x}, n+1) = g(\boldsymbol{x}, n, f_s(\boldsymbol{x}, n)),$$

where $g(\boldsymbol{x}, n, y) = \mathrm{add}(f(\boldsymbol{x}, n), y)$, which is primitive recursive by functional composition.

Definition. We call f_s and f_p functions obtained from f by *bounded sum* and *bounded product* respectively.

Using bounded summation and bounded product, another useful class of primitive recursive functions can be generated:

Definition. Let $f : \mathbb{N}^{m+1} \to \mathbb{N}$ be a function. For each $y \in \mathbb{N}$, set

$$A_f(\boldsymbol{x}, y) := \{z \in \mathbb{N} \mid z \leq y \text{ and } f(\boldsymbol{x}, z) = 0\}.$$

Define

$$f_{bmin}(\boldsymbol{x}, y) := \begin{cases} \min A_f(\boldsymbol{x}, y) & \text{if } A_f(\boldsymbol{x}, y) \neq \varnothing, \\ s(y) & \text{otherwise.} \end{cases}$$

f_{bmin} is called the function obtained from f by *bounded minimization*, and is usually denoted

$$\mu z \leq y (f(\boldsymbol{x}, z) = 0).$$

Proposition 1. *If $f : \mathbb{N}^{m+1} \to \mathbb{N}$ is primitive recursive, so is f_{bmin}.*

Proof. Define $g := \mathrm{sgn} \circ f$. Then

$$g(\boldsymbol{x}, y) := \begin{cases} 0 & \text{if } f(\boldsymbol{x}, y) = 0, \\ 1 & \text{otherwise.} \end{cases}$$

As f is primitive recursive, so is g, since the sign function sgn is primitive recursive (see http://planetmath.org/ExamplesOfPrimitiveRecursiveFunctionsthis entry).

Next, the function g_p obtained from g by bounded product has the following properties:

- if $g_p(\boldsymbol{x}, y) = 1$, then $g_p(\boldsymbol{x}, z) = 1$ for all $z < y$,
- if $g_p(\boldsymbol{x}, y) = 0$, then $g_p(\boldsymbol{x}, z) = 0$ for all $z \geq y$.

Finally, the function $(g_p)_s$ obtained from g_p by bounded sum has the property that, when applied to $(\boldsymbol{x}, y)$, counts the number of $z \leq y$ such that $g_p(\boldsymbol{x}, z) = 1$. Based on the property of g_p, this count is then exactly the least $z \leq y$ such that $g_p(\boldsymbol{x}, z) = 1$. This means that $(g_p)_s = f_{bmin}$ for all $(\boldsymbol{x}, y) \in \mathbb{N}^{m+1}$. Since g_p is primitive recursive, so is $(g_p)_s$, or that f_{bmin} is primitive recursive. $\square$

In fact, if f is a (total) recursive function, so is f_{bmin}, because all of the derived functions in the proof above preserve primitive recursiveness as well as totalness.

Remarks.

- In the definition of bounded minimization, if we take the y out, then we arrive at the notion of *unbounded minimization*, or just *minimization*. The proposition above shows that the set $\mathcal{PR}$ of primitive recursive functions is closed under bounded minimization. However, $\mathcal{PR}$ is not closed under minimization. The closure of $\mathcal{PR}$ under minimization is the set $\mathcal{R}$ of recursive functions (total or not).
- It is not hard to show that $\mathcal{ER}$, the set of all elementary recursive functions, is closed under bounded minimization.