ASSIGNMENT 4: RECURSIONS THAT GO BEYOND PRIMITIVE RECURSION

MICHAEL BEESON

Ackermann's function $A(m,n)$ is defined for nonnegative integers m and n by the recursion equations

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

1. Prove that this function is well-defined, i.e., the computation implied by these equations always terminates.

2. What is the largest value of Ackermann's function that you are able to compute with a reasonable amount of effort? You are allowed to use a computer program if you wish, but it is not required. It is not important to spend much time and energy on this, because the only point of the exercise is to get a feel for the extremely rapid growth of this function.

3. The *iteration* of addition is multiplication, in the sense that adding a to itself b times is $a \times b$. The iteration of multiplication is exponentiation a^b, which we can also write $a \uparrow b$. The iteration of exponentiation is sometimes indicated by a tower of exponents, but it can be written as $a \uparrow\uparrow b$. Iterating this function we get $a \uparrow\uparrow\uparrow b$, and so on; Knuth introduced the notation $a \uparrow^n b$. Look at the function ξ defined on page 272 of the textbook, and show that the functions mentioned earlier in the problem arise from ξ by fixing the first argument of ξ to be 0 for addition, 1 for multiplication, 2 for exponentiation, etc. That is, ξ is essentially the same as Knuth's iterated uparrow function. (ξ is the original Ackermann's function; the function A above is a modified version due to Rosza Péter.)

4. Compare the definition to the function ξ defined on p. 272 of the textbook. Express the function defined there in terms of Ackermann's function. If you need a hint, look at the table of values of A in the Wikipedia article about Ackermann's function.

5. Show that any primitive recursive function $f(x)$ grows no faster than $2 \uparrow^n x$ for some n (here n depends on f but not on x).

6. Use the result of problem 5 to prove that ξ is not primitive recursive, and the result of problem 4 to prove that the function A of problem 1 is not primitive recursive.

7. A *functional* is a function F that can take (in some argument places) a function from numbers to numbers, and possibly can take numbers in other argument places. For example, $F(f) = f(5)$ defines a functional. If we consider functionals defined by the same

equations as for primitive recursive functions, but allowing function arguments, we get the "primitive recursive functionals of type 1". Show that Ackermann's function is included. (Hint, see problem 3.) (We include in this class also functions that only take numbers for arguments.)

8. Show that the class of primitive recursive functionals of type 1 still does not include all the computable functions from natural numbers to natural numbers.