MA 3219 – Computability Theory

Frank Stephan. Departments of Computer Science and Mathematics, National University of Singapore, 3 Science Drive 2, Singapore 117543, Republic of Singapore

Email fstephan@comp.nus.edu.sg
Webpage `http://www.comp.nus.edu.sg/~fstephan/computability.html`
Telephone office +65-6874-7354
Office room number S14#06-06
Office hours Thursday 15.00-17.00h

Assignment for 02.03.2005. Can be corrected on request, it is not obligatory to hand the homework in.

Church's Thesis and Programming Languages. This homework just gives a framework on computation on integers instead of natural numbers. The resulting notions will turn out to be similar. The homework consists of several parts.

Setting. Assume that a programming language "Basic Loop Language" (BLL) can manupulate integer numbers and has the following syntactical constructs.

- Everything is defined as functions looking in principal like the following:

```
function euclid(x,y)
var v,w
begin
  loopwhile v > w or v < w beginloop
    if v > w then v = v-w; else w = w-v; endif; endloop;
  return(v); end.
```

- A variable header defines variables and parameters followed by a body how to manipulate the contents of the variables ending with a command to return the computed value.
- Expressions consist of adding and substracting variables and numerical integer constants. Also single variables or numerical constants count as expressions.
- Conditions compare expressions with respect to their size, the comparisons $<$, $\leq$, $=$, $\neq$, $>$, $\geq$ are permitted. Furhtermore one can use Boolean combinations of conditions instead of single conditions.
- Statements are either assignments or conditional statements or loops or a sequence of statements between the keywords "begin" and "end".
- Assignments tell variables to replace their value by the value of an expression. For example, `x = v+w+2; y = y+17; z = 3;` are assignments.
- Conditional statements are if-then-else statements as above with their traditional meaning, the tested conditions consist of Boolean combinations of comparison of variables with other variables or constants. The last keyword of an if-then-else statement is "endif" and the keywords "then" and "else" separte the two sequences of statements which are done if the Boolean condition is true and false, respectively.

- There are two types of loops:
`loopwhile cond beginloop ... endloop`
executes the body of the loop as long as the condition "cond" is true.
`looptimes exp beginloop ... endloop`
executes the body exactly as often as the value of the expresssion "exp" at the time of entering the loop. If the involved variables change their value later, this does not effect the number of times the loop is executed. If the value is 0 or negative, the loop is not executed at all.

- Functions are written down one after the other. One function f can use another function g to compute values as long as g is written down before f is written down. No function can call itself.

1. Analyzing a program. What is the following BLL-Function computing?

```
function f(n)
var m,k;
begin
  m = 0; k = 0;
  loopwhile m < n beginloop
    m = m+8; k = k+1; endloop;
  loopwhile m > n beginloop
    m = m-8; k = k-1; endloop;
  return(k); end.
```

Determine f(-8), f(-4), f(0), f(4), f(8), f(12), f(16) and f(17). Give an easy mathematical definition of f.

2. Programming in BLL. Write functions *abs* and *cube*. Here $abs(x) = x$ if $x \geq 0$ and $abs(x) = -x$ if $x < 0$; $cube(x) = x * x * x$. Note that only additions and substractions are allowed but not multiplications. Furthermore, one has to take care of negative numbers.

3. The basic concepts. Define the following terms using BLL.

- computable functions from the integers to the integers;
- total computable (= recursive) functions from the integers to the integers;
- recursively enumerable subsets of the integers;
- recursive (= decidable) subsets of the integers.

Note that "recursive function" is a synonym for "total computable function" and "recursive set" for "decidable set". This synonyms also explain the origin of the term "recursively enumerable".

4. Primitive-recursive functions. The primitive-recursive functions can be defined by saying "f is computed by a BLL-function such that neither the program of the function nor of a called subfunction has the syntactic construct "XYZ". What is

"XYZ"? Furthermore, assume that g with two inputs is a BLL-computable function given by a program not using the syntactic construct "XYZ". Write a program for a function f which can call g besides the above given basic constructions but which does not use "XYZ". The function f satisfies the following conditions:

- if $x = 0$ then $f(x) = g(0, 0)$;
- if $x > 0$ then $f(x) = g(x, f(x - 1))$;
- if $x < 0$ then $f(x) = g(x, f(x + 1))$.

5. Robustness. If one would only permit expressions of the type $x + 1$, $x - 1$, x and 0 where x stands for an arbitrary variable name, would there be any substantial change in the computational power of the concepts defined in Exercise 3 above? Or would these concepts remain unchanges and just programs become longer? Note that you still would be permitted to define auxiliary functions first and use them later.