Name		Matrikel								SKZ			

Klausur 2

Berechenbarkeit und Komplexität

17. Januar 2013

Part 1 *QuadraticEquation*

A function $f : \mathbb{N}^2 \to \{0, 1\}$ is defined by

$$f(p, q) := \begin{cases} 1 & \text{if } \exists x \in \mathbb{R} : \ x^2 + px + q = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Recall that

$$x^2 + px + q = 0 \iff x = -p/2 \pm \sqrt{(p/2)^2 - q}.$$

1	yes	

Is f a LOOP computable function?

The key is to observe that

$$f(p, q) = 1 \ :\Leftrightarrow \ p^2 \geq 4q$$

2		no

Is $\{1^q01^p \mid p \in \mathbb{N} \wedge q \in \mathbb{N} \wedge f(p, q) = 1\}$ a regular language?

No, Pumping Lemma. Suppose a finite automaton with N states recognizes L. Let $w := 1^{N+4}01^{N+4}$. Note that $w \in L$. By the Pumping Lemma, there exists a natural number $m > 0$ such that all words of the form $1^{N+4+km}0^{N+4}$ are in L too. But that would mean that $(N + 4)^2 \geq N + 4 + km$ for all k, which contradicts $4q \leq p^2$.

Part 2 *RecFun6*

Consider the functions $f : \mathbb{N} \to \mathbb{N}$, $f(n) := \lfloor \sqrt{n} \rfloor$ and $g : \mathbb{N} \to \mathbb{N}$ where

$$g(n) \quad := \quad \begin{cases} \sqrt{n} & \text{if } \sqrt{n} \in \mathbb{N}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

(Note that $\lfloor x \rfloor$ ist the largest integer which does not exceed x.)

3	yes	

Is f LOOP computable?

Since $f(n) \leq n$ we can compute $f(n)$ by a bounded search.

4	yes	

Is f WHILE computable?

Of course. (1) It is WHILE computable because it is LOOP computable. (2) It is WHILE computable because $f(n)$ can be computed by a search in a while loop.

5		no

Is g LOOP computable?

g is not total, while every every LOOP computable function is total.

6	yes	

Is g Turing computable?

g is WHILE computable and therefore Turing computable.

Part 3 *RecursiveEnumerable6RAM*

We say that a RAM R accepts *a word $w \in \{1,2\}^*$ if R starts with the letters of w on its input tape and stops with 1 written on its output tape. $L(R)$ is the set of all words accepted by R.*

Let R be a RAM and let $f : \mathbb{N} \to \mathbb{N}$ be a Turing-computable total function. Suppose that R has the following property: When R accepts a word of length n, it does so in no more than $f(n)$ steps.

7	yes	

Is $L(R)$ necessarily recursive?

We simulate R by a Turing machine. First we compute $f(n)$. Then we start R with input w und execute $f(n)$ steps. If w has been accepted then $w \in L(R)$, otherwise $w \notin L(R)$. Therefore, $L(R)$ and $\overline{L(R)}$ are both recursively enumerable.

8		no

Is $L(R)$ necessarily finite?

9		no

Let L' be a recursively enumerable language. Can it be concluded that $L(R) \cap L'$ is recursive?

Consider the case $L(R) = \Sigma^*$ where R accepts any word. Thus, if the intersection $L(M) \cap L$ were recursive, it would mean that every recursively enumerable language is recursive. This is clearly not the case.

Part 4 *RecursiveEnumerable8*

Let a Turing machine M compute a partial function $f : \{0\}^* \to_P \{0\}^*$ on sequences of 0 and let $f' : \mathbb{N} \to_P \mathbb{N}$ be the function that maps the length of the input of M to the length of the output of M.

10	yes	

Is f' necessarily while-computable?

11		no

Is f' necessarily primitive recursive?

12	yes	

Is f' necessarily μ-recursive?

Part 5 *TermRewriting1*

13		no

Given some term rewriting system and two terms t_1 and t_2. Is it decidable if $t_1 \to^* t_2$?

In the exercises it was shown that Turing machines may be simulated by a term rewriting system.

14	yes	

Given some term rewriting system and two terms t_1 and t_2. Is it semi-decidable if $t_1 \to^* t_2$?

Try all rewritings up to a certain number of rewrite steps k; loop on k.

Part 6 *Decidable4*

Let $\langle f \rangle$ be the Gödel number *encoding of a μ-recursive function f as a bit-string and let $\langle n \rangle$ be the binary encoding of a natural number n as a bit string.*

Below f denotes a μ-recursive function $\mathbb{N} \to_P \mathbb{N}$ and n denotes a natural number.

15		no

Is the problem "f is defined on input n" decidable? (Problem instance is $(\langle f \rangle, \langle n \rangle)$.)

16		no

Is the problem "f is primitive recursive" decidable? (Problem instance is $\langle f \rangle$.)

17	yes	

Assume f is primitive recursive. Is the problem "f is defined on input n" decidable? (Problem instance is $(\langle f \rangle, \langle n \rangle)$.)

Part 7 Complexity2012

Answer the following questions.

18	yes		Does $(f(n)+7)^2 = O(f(n)^2+7)$ hold for all $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$?
19	yes		Does $2^{f(n)+7} = O(2^{f(n)})$ hold for all $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$?
20		no	Does $2^{f(n)+n} = O(2^{f(n)})$ hold for all $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$?
21		no	Does $log(n)^2 = O(log(n^2))$ hold?

Part 8 WHILEadditions

Consider the following LOOP program with input x_1 and output x_0.

```
x0 = 1
loop x1 do
  loop x0 do
    x0 = x0 + 1
  end
end
x0 = x0 + 1
```

22 | 1 Point

What ist the function $f : \mathbb{N} \to \mathbb{N}$ computed by the program? Please fill in; you do not need to justify your answer.
$f(n) =$

The inner loop doubles $x1$. The program computes $n \mapsto 2^n + 1$: If $m(1)$ initially contains the input n, then $m(0)$ holds $2^n + 1$ upon termination.

23 | yes |

Let $T(n)$ be the number of additions performed by the program for input $x_1 = n$. Is $T(n)$ primitive recursive?

Yes, because $T = f$ which is LOOP computable: All additions in the program increase x_0, and the program contains no subtractions. Therefore, the number $T(n)$ of additions performed is equal to the output $f(n)$. Note that the `x0=1` is, in fact, `x0=x0+1`.

Part 9 DivideAndConquer

Let $T(n)$ be the number of functions calls to `h` *resulting from evaluating $g(n, 1)$.*

```
function g(n, x) {
    if n==1
        return h(x)
    else
        n2 = floor(n/2)  //floor(x) = biggest integer not exceeding x
        sum = 0
        for k=1 to 3
            sum = sum + g(n2, x + n2 + k)
        return sum
```

You do not need to justify your answers.

24	yes		Is $T(4) = 9$?
25	1 Point		Determine $T(n)$ asymptotically for large n. Use Θ-notation.

$\Theta(n^{\log_2(3)}) = \Theta(3^{\log_2(n)})$