Gruppe	Hemmecke (10:15)		Hemmecke (11:00)									Popov			
Name		Matrikel									SKZ				

Klausur 2

Berechenbarkeit und Komplexität

13. Januar 2017

Part 1 RecFun2016

Let $f_1, f_2 : \mathbb{N} \to_P \mathbb{N}$ be two partial functions that are defined as follows:

$$f_1(x) = \begin{cases} x+1 & \text{if } x \text{ is odd,} \\ \text{undefined} & \text{otherwise} \end{cases} \qquad f_2(x) = \begin{cases} x^2 & \text{if } x \text{ is even,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Let $g(x) = f_1(f_2(x))$ and $h(x) = f_1(4x+1) + f_2(4x)$.

1		no	*Is f_1 primitive recursive?*
2	yes		*Is f_2 μ-recursive?*
3	yes		*Is g μ-recursive?*

$g : \mathbb{N} \to_P \mathbb{N}$ is a function that is nowhere defined, so the representation $g(x) = (\mu t)(x)$ (where $t(y,x) = s(p_2^2(y,x))$ is clearly primitive recursive) proves that g is μ-recursive.
In general, the composition of μ-recursive functions is μ-recursive.

4	yes		*Is h primitive recursive?*

Even though f_1 and f_2 are not primitive recursive (since they are not total functions), their combination (as defined here) is. f_1 is only called with an odd number as argument and f_2 is only called with an even number. So $h(x) = 4x + 2 + 16x^2$ and that is clearly primitive recursive.

5		no	*Can every total function of type $\mathbb{N} \to \mathbb{N}$ be computed by a LOOP program?*

From the Ackermann function ack one can easily construct a total function of $\alpha(x) = \text{ack}(y,z)$ where y and z are such that $0 \le z < 2^n$, $x = 2^n y + z$ for $n = \left\lceil \frac{\log_2(x+1)}{2} \right\rceil$.The function $\alpha(x)$ is not primitive recursive, because it basically **is** the Ackermann function. The y and z are the "upper" and "lower" half of the binary representation of x.

Part 2 Grammar2016

Consider the grammar $G = (N, \Sigma, P, S)$ where $N = \{S\}$, $\Sigma = \{a,b\}$, $P = \{S \to aBbA, aB \to abA, bA \to baB, A \to aa, B \to bb\}$.

6		no	*Is $abababab \in L(G)$?*

A word in $L(G)$ always ends in aa or bb.

7		no	*Is the grammar G right linear?*
8	yes		*Is there a linear bounded automaton M such that $L(M) = L(G)$?*

G is a context-sensitive grammar.

9		no	*Does for every grammar $G' = (N', \Sigma', P', S')$ with $\Sigma' = \{0,1\}$ exist a Turing machine M over the alphabet Σ' such that $L(M) = \overline{L(G')}$?*

Let L' be a recursively enumerable language that is not recursive. Then there exists a grammar G' sucht that $L' = L(G')$. However, $\overline{L'}$ is not recursively enumerable, so there does not exist a Turing machine M with $L(M) = \overline{L(G')} = \overline{L'}$.

Part 3 *Decidable2016*

Consider the following problems. In each problem below, the input of the problem is the code $\langle M \rangle$ of a Turing machine M with input alphabet $\{0, 1\}$.
Problem A: *Does $L(M)$ contain the word 011000001111?*
Problem W: *Does $L(M)$ contain more than 2017 words?*
Problem C: *Is $L(M)$ a context-sensitive language?*
Problem Z: *Does M always halt when 0 is under the head?*

10	yes	

Is A semi-decidable?

Simulate M on the input word 011000001111. If M halts in an accepting state, then $011000001111 \in L(M)$. For semi-decidability that is enough.

11	yes	

Is W semi-decidable?

Simulate M in such a way that every possible word is checked. If that simulation finds 2018 words as accepted by M, the simulator can stop and answer YES.

12		no

Is C decidable?

Rice Theorem.

13	yes	

Is Z decidable?

The code $\langle M \rangle$ of a Turing machine $M = (Q, \Gamma, \sqcup, \Sigma, \delta, q_1, F)$ is a finite string and encodes (among other things) the transition function δ. A Turing machine that decides Z has to check whether $(q, 0)$ is undefined for all $q \in Q$. Since domain$(\delta) \subseteq Q \times \Gamma$ is a finite set, this is a check can be decided in finitely many steps from the encoding $\langle M \rangle$ without the need of simulating M.

14	yes	

Let $P, P' \subseteq \{0, 1\}^*$ and let M be a Turing machine that for every $w \in P$ computes a $w' \in P'$ and for every $w \notin P$ computes a word $w' \notin P'$. Assume P is not decidable. Can it be concluded that P' is not decidable?

We have $P(w) \iff P'(f(w))$ where f is the "computable function" (that is required in Definition 42) computed by M. Thus $P \leq P'$. Apply Theorem 32 (lecture notes).

Part 4 *Complexity2016*

Let $f(n) = 3^n(2^n + n^{2017})$, $g(n) = 6^{n+1} + n \cdot 3^n$, and $h(n) = 2^{3n} \log_2 n$.

15	yes		*Is it true that $f(n) = \Theta(g(n))$?*
16	yes		*Is it true that $log_2(h(n)) = O(f(n))$?*
17	yes		*Is it true that $g(n) = O(h(n))$?*
18		no	*Is it true that $\frac{1}{n} = O\left(\frac{2017}{n^2}\right)$?*

Part 5 *LoopWhile2016*

Let P be a LOOP program that computes a primitive recursive function $f : \mathbb{N} \to \mathbb{N}$ with time complexity $T(n) \in \Theta(n^{2017})$ where $x_1 = n$ is the input of the program P and x_0 its output. Note that the time complexity $T(n)$ of a LOOP program is given by the number of executed statements during the run of the program with input n. Furthermore, let W be the following WHILE program that computes a (partial) function $g : \mathbb{N} \to_P \mathbb{N}$.

while x_1 ***do while*** x_1 ***do*** P ***end;*** $x_1 := x_0 - 1$ ***end;***

19		no

Can it be concluded that g is LOOP computable?

If $f(n) = 2$ for every $n \in \mathbb{N}$, then W only terminates for input $x_1 = 0$, i. e., W does not compute a total function. In this case g is not LOOP computable, i. e., we have a counterexample.

20	yes	

Is the problem "$n \in \mathrm{range}(g)$" semi-decidable?
(Formally: Let $b : \mathbb{N} \to \{0,1\}^*$ be the (Turing-computable) function that takes a natural number n as input and returns the binary representation of n. Is the set $R = \{ b(n) \in \{0,1\}^* \mid n \in \mathrm{range}(g) \}$ semi-decidable?)

We construct a Turing machine D that takes a word w as input and simulates the WHILE program W (in "parallel" for every natural number n). If z_n is the output of the computation of W on n, it computes $b(z_n)$ and compares the result with w. If it is equal, then D stops with answer YES, otherwise it continues the search.

21	yes	

*Can W be rewritten into another WHILE program W' that computes the same function g such that W' uses only one **while** loop?*

Transform W into Kleene normalform.

22	yes	

Let W_P be the above WHILE program instantiated with a given program P. Does there exist a LOOP program P with time complexity $\Theta(n^{2017})$ such that $L = \{ 0^e \mid \exists n \in \mathbb{N} : W_P$ computes for input n output $e \}$ is regular?

Let P' be any LOOP program that runs with complexity $\Theta(n^{2017})$. Choose as P the program $\boxed{P'; x_0 := 0}$. Then $g(n) = 0$ for all $n \in \mathbb{N}$ and thus $L = \{\varepsilon\}$ is regular.

23		no

Let Q be the following LOOP program.

loop x_1 ***do loop*** x_1 ***do*** $x_0 := x_0 + 1$ ***end end****;*
$x_1 := x_0 + 1$*;*
loop x_1 ***do loop*** x_1 ***do*** $x_0 := x_0 + 1$ ***end end***

Is the complexity of program Q (depending on input $x_1 := n$) in $O(n^2)$?

After the first loop we have $x_0 = n^2$. In the third line the output is computed as $x_0 = (n^2 + 1)^2$. So there must have been $O(n^4)$ executions of the statement $x_0 := x_0 + 1$.

Part 6 *OpenComputability2016*

Let $T(n)$ be the number of multiplications executed during the run of the following program while evaluating $g(n, 1)$.

```
function g(n, x)
    if n==0 then
        return x
    else
        if odd(n) then
            return g(n-1, x+x)
        else
            k = floor(n/2)
            return g(k, x) * g(k, x+1)
```

24 *1 Point* *Compute $T(9)$.*

$T(9) =$

$$
\begin{aligned}
g(9,1) &= g(8,2) \\
&= g(4,2) * g(4,3) \\
&= g(2,2) * g(2,3) * g(2,3) * g(2,4) \\
&= g(1,2) * g(1,3) * g(1,3) * g(1,4) * g(1,3) * g(1,4) * g(1,4) * g(1,5) \\
&= g(0,4) * g(0,6) * g(0,6) * g(0,8) * g(0,6) * g(0,8) * g(0,8) * g(0,10) \\
&= 4 * 6 * 6 * 8 * 6 * 8 * 8 * 10
\end{aligned}
$$

So, $T(9) = 7$.

25 *1 Point* *Determine $T(n)$ asymptotically for large n. Use Θ-notation.*

$T(n) =$

A recursion formula for T is $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + 1$, i. e., according to the notation in Theorem 49 (Master theorem) we have $a = 2$, $b = 2$, and $f(n) = 1 \in O(n^{(\log_2 2)-\varepsilon})$ for $\varepsilon = \frac{1}{2}$. Thus $T(n) = \Theta(n^{\log_2 2}) = \Theta(n)$.