Quantum Information Theory (SS'06)

Problem Set No 4 (20 scores)
emission: 22.06.06; absorption: 06.07.06

▷ Aufgabe 1 (Turing Adder)

Let there be an alphabet Σ, consisting of finitely many distinct letters σ, a set of possible states S, consisting of finitely many states s, and a set of possible moves $M = \{\mathtt{L}, \mathtt{R}\}$.

An instruction is a tupel $s\sigma \Rightarrow \sigma' s' m$, read: if in state s, reading symbol σ, then write symbol σ', change to state s', and move $m \in M$. If s' ist the halt state s_H, the move is void.

Recall that, once the alphabet and the set of possible states are specified, any Turing machine is completely specified by a finite set of instructions.

As an example consider the case of a Turing machince which only needs 5 internal states $S = \{s_1, s_2, s_3, s_4, s_H\}$, and a two-letter alphabet $\Sigma = \{\mathtt{1}, \mathtt{0}\}$, where the only significant symbol is `1`, the symbol `0` signifying `BLANK`. This alphabet furnishes a unary representation of intergers, where a given integere N is written as a sequence of N consecutive symbols `1`, surrounded by `BLANK`. In our example, the machines' programme is specified by six instructions

$$\begin{aligned}
s_1\mathtt{0} &\Rightarrow \mathtt{0}s_2\mathtt{R}\\
s_2\mathtt{0} &\Rightarrow \mathtt{0}s_3\mathtt{R}\\
s_2\mathtt{1} &\Rightarrow \mathtt{1}s_2\mathtt{R}\\
s_3\mathtt{0} &\Rightarrow \mathtt{0}s_H-\\
s_3\mathtt{1} &\Rightarrow \mathtt{0}s_4\mathtt{L}\\
s_4\mathtt{0} &\Rightarrow \mathtt{1}s_2\mathtt{R}
\end{aligned} \tag{1}$$

Initially, the machine is in state s_1. The initial state of the tape is specified

$$\ldots\mathtt{0011011100}\ldots \tag{2}$$

with the initial position of the read-write head on the first blank on the left side of the two consecutive `1`.

What is the output of the Turing machine – i.e what does the machine perform?

▷ Aufgabe 2 (Halting Problem)

In the lecture you were confronted with the halting problem, stated "Will some given Turing machine T eventually halt for some given input x?". You also heard that Turing demonstrated, that there exists no universal algorithm which solves this problem. In this excercise, you will develop the argument and also learn of the significance of this "no-go" theorem.

(a) Have a look at the following code bit `GOLDBACH:` $N = 4$; $*$`if there are no prime numbers` p, q `such that` $N = p + q$ `then` s_H; `else` $N = N + 2$ `and goto` $*$. If we would have a universal algorithm which would decide whether any given algorithm ever halts or not what would we learn if we would run this Algorithm on `GOLDBACH`?

(b) For binary alphabet – how many different Turing machines with S internal states (plus halt state) are there?

(c) The Turing proof rests on the fact, that the Turing machines are countable. Can you indicate a code which assigns any given Turing T machine an integer n_T such that every integer describes a possible Turing machine?
Hint: Recall that every positive integer admits a unique represenatition $n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$, where p_i are distinct prime numbers, and $a_1, a_2, \ldots, a_k$ are non-negative integers.

(c) Define the "halting function"

$$h(n,x) := \begin{cases} 0 & \text{if TM } n \text{ does not halt upon input } x \\ 1 & \text{if TM } n \text{ halts upon input } x \end{cases} \tag{3}$$

Turing's point was, that this function is not computable. The argument is based on a proof by contradiction.

Turing considers the special case $h(x,x)$, that is the value of the halting function for Turing machine x which is fed its own Turing number as input. According to the definition of computability, $h(x,x)$ is computable iff there is an algorithm (a Turing machine) which evaluates $h(x,x)$. Call this Algorithm `HALT`. Now consider another algorithm, called `TURING`, in pseudo code

```
TURING
input x
output y
begin
   y=HALT(x)
   if y=0 then
      s_H
   else
      loop forever
   endif
end
```

With `HALT` a valid machine, also `TURING` is a valid machine. Thus it has a Turing number t, say.

Verify that the assumption "h(x,x) is computable" leads to the contradiction $h(t,t) = 1$ iff $h(t,t) = 0$. Do you reckognize the Russel-Whitehead antinomy?

▷ Aufgabe 3 (Complexity of Arithmetics)

Working with binary representation of integers, what is the complexity of arithmetic operations $+, -, \times, \div$?

▷ Aufgabe 4 (Euclid algorithm)

Recall that for two integer a, b, the greatest common divisor $gcd(a,b)$ is that number which is the largest of all numbers, each of which divides both a, b. Verify that the following version of the Euclid algorithm not only computes $\gcd(a,b)$, but also gives the numbers k and l of the representation $\gcd(a,b) = ka + lb$.

```
input a,b
output g=gcd(a,b),k,l                % gcd(a,b)=ka+lb
begin                                % here we go
   (a_0,a_1):=(a,b)                  %    some initialization
   (k_0,k_1):=(1,0)                  %
   (l_0,l_1):=(0,1)                  %
   while a_1\neq0 do                 %    the Euclid loop
      begin                          %    [is cycled ? times]
      q:=a_0 DIV a_1                 %       divide-and-remainder operation
      r:=a_0 MOD a_1                 %       [costs ?? steps]
      (a_0,a_1):=(a_1,r)             %
      (k_0,k_1):=(k_1,k_0-qk_1)      %       some re-shuffeling
      (l_0,l_1):=(l_1,l_0-ql_1)      %       [costs ??? steps]
      end                            %    that's it!
   g=a_0                             %    just for readability
   k=k_0                             %
   l=l_0                             %
end                                  % there we are!
```

What is the complexity of this algorithm?