**Saarland University**
**Department 6.2 - Computer Science**
**Prof. Dr. W. J. Paul**
**Dr. Mikhail Kovalev**

# Theoretical Computer Science - WS13/14

**Exercise Sheet 4 (due: 10.03.14, 45 points)**

---

### Exercise 1: (Fibonacci numbers) (7 points)

Give the formal definition of a primitive recursive function $f \in \mathbb{N}_0 \to \mathbb{N}_0$, where

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n+1) &= f(n) + f(n-1). \end{aligned}$$

In your definitions use PR bijections $b^2 \in \mathbb{N}_0^2 \to \mathbb{N}_0$ and $b_i^2 \in \mathbb{N}_0 \to \mathbb{N}_0$.

### Exercise 2: (Turing machine) (3 points)

A configuration $k$ of a 1-tape Turing machine $M = (A, Z, \delta, z_0, E)$ can be written as

$$k = uzv,$$

where $u, v \in A^*$ are non-empty portions of the tape to the left and to the right from the head of the machine and $z \in Z$ is the current state. Give the semantics of the transition $k \vdash k'$ (i.e., define the next configuration $k'$) for the case when $u = v = \epsilon$.

### Exercise 3: (Turing machine) (4 points)

Define a 1-tape Turing machine which subtracts 1 from the binary number written on the tape (machine 'tape = tape-1'). You can assume the tape to be in the format

$$B \dots BuvB \dots B,$$

where $u \in \{1\}$ and $v \in \{0,1\}^* \cup \{B\}$. You can also assume the head of the machine in state $z_0$ to point to $u$. Make your computation regular, i.e., after performing subtraction return the head to the first non-blank symbol on the tape.

### Exercise 4: (Turing machine) (4 points)

Define a regular 2-tape Turing machine which appends the non-blank content of the first tape to the non-blank content of the second tape (machine 'tape 2 = tape 2 $\circ$ tape 1'). A computation of a Turing machine is regular if (i) non-blank portions of the tapes are contiguous, (ii) initially all heads point to first non-blank symbols of the tapes and (iii) if the computation terminates, all heads again point to first non-blank symbols of the tapes. A machine is regular, if all computations are regular.

### Exercise 5: (programming Turing machines) (4 points)

Let a regular $k$-tape Turing machine $F$ compute the (possibly partial) function $f : \mathbb{N}_0^{r+1} \to \mathbb{N}_0$. Construct a Turing machine $M$, which computes the unbounded $\mu$-operator over $f$, i.e., the function $\mu f \in \mathbb{N}_0^r \to \mathbb{N}_0$, s.t.,

$$\mu f(x) = \begin{cases} min\{n \mid f(n,x) = 0 \wedge \forall m \leq n : f(m,0) \neq \Omega\} & \text{if such } n \text{ exists} \\ \Omega & \text{otherwise.} \end{cases}$$

**Saarland University**
**Department 6.2 - Computer Science**
**Prof. Dr. W. J. Paul**
**Dr. Mikhail Kovalev**

# Theoretical Computer Science - WS13/14

**Exercise Sheet 4 (due: 10.03.14, 45 points)**

---

You can assume initially tape 1 to contain initially the arguments $x_1, \ldots, x_r$:

$$bin(x_1)\#bin(x_2)\#\ldots\#bin(x_r)$$

and all the other tapes to be empty. In the end state, if the machine terminates, the value of the function $\mu f(x)$ should be written on tape 1 and all other tapes should be empty.

In you construction you can use the following primitive 'subroutine' Turing machines:

- `tape i := tape j;` - copying of tape $j$ to tape $i$,
- `tape i` $:=$ `tape i` $\circ$ `tape j;` - concatenating tape $i$ and tape $j$,
- `erasetape i;`,
- `tape i := tape i + 1;` - adding one to the binary number written on tape $i$. If the tape is empty, this simply writes 1 on the tape,
- `tape i := tape i - 1;` - subtracting 1 from the binary number written on tape $i$. This assumes a non-empty tape and no leading zeroes,
- `tape i = 0;` - the machine checks whether tape $i$ contains number 0; if yes - moves to state $true$, otherwise moves to state $false$; you can write conditional statements of the form

```
if tape i = 0 then
  M1;
else
  M2;
```

You can also use simple while-machine, i.e. write while-loops of the form

```
while tape i = 0 do M1;
```

and

```
while tape i != 0 do M1;
```

Concatenation of Turing machines `M1` and `M2` is denoted by `M1;M2`. Running the computation of a $k$-tape Turing machine `M1` on tapes $i_1, \ldots i_k$ is denoted by `M1(i1, ..., i_k)`.

**Saarland University**
**Department 6.2 - Computer Science**
**Prof. Dr. W. J. Paul**
**Dr. Mikhail Kovalev**

# Theoretical Computer Science - WS13/14

**Exercise Sheet 4 (due: 10.03.14, 45 points)**

---

### Exercise 6: (Turing machine) (4 points)

Let $M_1 = (A, Z_1, \delta_1, z_0, E_1)$ be a simple `tape = 0 ?` Turing machine defined in the lecture:

$$\begin{aligned}
\delta_1(z_0, 1) &= (false, 1, N) \\
\delta_1(z_0, 0) &= (z_1, 0, R) \\
\delta_1(z_1, B) &= (true, B, L) \\
\delta_1(z_1, X) &= (false, X, L) \qquad \text{for } X \neq B.
\end{aligned}$$

Let $M_2 = (A, Z_1, \delta_2, z_0', E_1)$ be another regular 1-tape machine, s.t., $Z_0 \cup Z_1 = \emptyset$. Define the machine $M_3 = (A, Z_3, \delta_3, z_0, \{z_e\})$ which performs a simple while-loop:

```
while tape = 0 do M1;
```

Assume $z_e \notin Z_1 \cup Z_2$.

### Exercise 7: (simulation of a multi-tape Turing machine) (4 points)

Give an (informal) explanation, why we can simulate a $k$-tape Turing machine by a 1-tape Turing machine in time bounded by $O(n^2)$.

### Exercise 8: (simulation of a RAM machine) ($3 \times 5$ points)

A RAM machine is an abstract computational model which is often referred as a generalized computer with infinite memory and register content. A RAM machine consists of the following components:

- a program $\pi$ - consists of a numbered list of instructions which is considered to be fixed during execution of the machine. The number of the instruction is called its label,
- a program counter $b \in \mathbb{N}$ - contains the label of the instruction to be executed next by the machine,
- register/memory content $c : \mathbb{N}_0 \to \mathbb{N}_0$. We do not distinguish between registers and memory cells, except for the special register $c(0)$ which we use as a destination register for our instruction,
- an input tape $w \in A^*$. This contains a user input to the machine and is considered to be fixed during the execution. The computer reads the input from left to right and can read every symbol only once (i.e., the head moves strictly from left to right),
- an output tape $x \in A^*$. The machine prints its output from right to left and can not read the already printed symbols (i.e., the head moves strictly from right to left).

The configuration of the machine is expressed as a tuple $(b, c, w, q, x)$, where

**Saarland University**
**Department 6.2 - Computer Science**
**Prof. Dr. W. J. Paul**
**Dr. Mikhail Kovalev**

# Theoretical Computer Science - WS13/14

**Exercise Sheet 4 (due: 10.03.14, 45 points)**

| Instruction | Effect |
|---|---|
| `load i` | $\mathtt{c(0)} := \mathtt{c(i)}; \mathtt{b} := \mathtt{b}+1$ |
| `store i` | $\mathtt{c(i)} := \mathtt{c(0)}; \mathtt{b} := \mathtt{b}+1$ |
| `add i` | $\mathtt{c(0)} := \mathtt{c(0)} + \mathtt{c(i)}; \mathtt{b} := \mathtt{b}+1$ |
| `sub i` | $\mathtt{c(0)} := \begin{cases} \mathtt{c(0)} - \mathtt{c(i)} & \mathtt{c(0)} \geq \mathtt{c(i)} \\ \mathtt{0} & \mathtt{otherwise} \end{cases}; \mathtt{b} := \mathtt{b}+1$ |
| `read` | $\mathtt{c(i)} := \mathtt{w[q]}; \mathtt{q} := \mathtt{q}+1; \mathtt{b} := \mathtt{b}+1$ |
| `print a` | $\mathtt{x} := \mathtt{ax}; \mathtt{b} := \mathtt{b}+1$ |
| `goto i` | $\mathtt{b} := \mathtt{i}$ |
| `if c(0) = i goto l` | $\mathtt{b} := \begin{cases} \mathtt{l} & \mathtt{c(0)} = \mathtt{i} \\ \mathtt{b}+1 & \mathtt{otherwise} \end{cases}$ |
| `ind load i` | $\mathtt{c(0)} := \mathtt{c(c(i))}; \mathtt{b} := \mathtt{b}+1$ |
| `ind store i` | $\mathtt{c(c(i))} := \mathtt{c(0)}; \mathtt{b} := \mathtt{b}+1$ |
| `c load i` | $\mathtt{c(0)} := \mathtt{0}; \mathtt{b} := \mathtt{b}+1$ |
| `end` | end of instructions |

Table 1: Instructions supported by the abstract RAM machine

- $b \in \mathbb{N}$ is a program counter,
- $c \in \mathbb{N}_0 \to \mathbb{N}_0$ is the current memory and register content,
- $w \in A^*$ is the content of the input tape,
- $q \in \{1, \ldots |w|\}$ is the position of the head on the input tape,
- $x \in A^*$ is the current state of the output tape.

Initially we start with the program counter equal 1, the content of all memory cells being 0, the position of the head of the input tape being 1, and the output tape being empty.

The set of supported instructions and their effect is given in Table . `ind load` and `ind store` stands for 'indirect load' and 'indirect store' operations and `c load` stands for 'constant load'.

Our goal is to simulate the abstract RAM machine by a 4-tape Turing machine $M$, which is comprised of $p+1$ 4-tape Turing machines $M_0, M_1, \ldots, M_p$ (where $p$ is the number of instructions in the program of the RAM machine). Machine $M_0$ does the initialization of the Turing machine $M$ and establishes the simulation relation (we define it below). Machines $M_1, \ldots, M_p$ execute 1 instruction each. They start in a state where the simulation relation holds and guarantee that the relation is maintained afterwards.

The simulation relation between the current state of the Turing machine $M$ and the configuration $(b, c, w, q, x)$ of the RAM machine is defined in the following way:

**Saarland University**
**Department 6.2 - Computer Science**
**Prof. Dr. W. J. Paul**
**Dr. Mikhail Kovalev**

# Theoretical Computer Science - WS13/14

**Exercise Sheet 4 (due: 10.03.14, 45 points)**

---

- the current state of $M$ is the initial state of machine $M_b$,
- track 1 of $M$ contains the printed output $x$,
- track 2 contains the current content of non-empty memory cells (except for register $c(0)$ - its content is always stored on the track). Let $\{i_1, \ldots, i_k\}$ be the set of all indices such as $c(i_j) \neq 0$. Then the content of track 2 is

$$\mathtt{B}\cdots\mathtt{B}\#\mathtt{0}\#\mathtt{bin(c(0))}\#\mathtt{bin(i_1)}\#\mathtt{bin(c(i_1))}\#\cdots\mathtt{bin(i_k)}\#\mathtt{bin(c(i_k))}\#\mathtt{B}\cdots\mathtt{B}$$

- track 3 contains the unread portion of the input, i.e. the string $\mathtt{w_q}, \ldots, \mathtt{w_{|w|}}$,
- track 4 is empty. Machines $M_j$ can use it as a scratch buffer when simulating instruction execution.

Machine $M$ starts in a state where track 1 contains the initial input to the RAM machine and all other tracks are empty. Machine $M_0$ performs the following actions:

- copies the content of track 1 to track 3 (and moves the head of track 3 to the leftmost symbol),
- erases track 1,
- writes #0#bin(c(0))# on track 2 (and moves the head of track 2 to the leftmost symbol),
- goes to the initial state of machine $M_1$.

The actions performed by machine $M_j$ depend on the instruction it executes. Describe the actions performed by machine $M_j$ for the following cases:

1. machine $M_j$ executes instruction `add i`,
2. machine $M_j$ executes instruction `if c(0) = i goto l`,
3. machine $M_j$ executes instruction `print a`,
4. machine $M_j$ executes instruction `load a`,
5. machine $M_j$ executes instruction `ind store a`.