Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Networked Systems Group (NSG)

HS 2015

Prof. L. Vanbever & A. El-Hassany,
based on Prof. R. Wattenhoffer's material

Discrete Event Systems

Exercise Sheet 5

1 Designing Turing Machines

Alice is very happy because she was accepted for an internship at Tintel, one of the world's leading processor manufacturers. Unfortunately, she has only attended the famous DES lecture during her studies at ETH and knows nothing about electronic circuits. Therefore, she wants to solve her first assignment using a Turing Machine—please assist her:

Alice is asked to implement a *binary to unary converter*. This converter takes a number a in binary (alphabet $\{0, 1\}$) and converts it to a unary number u (alphabet $\{1\}$). Initially, the TM head points to the MSB of a. At the end, the head should point to the right-most digit of u.

Provide a plain text description of your TM as well as a finite state machine controlling the tape head. Use the following notation for transitions:

'$\alpha \to \beta \mid \gamma$'	read α from the tape at the current position, then write a β and finally move left if $\gamma = L$ or move right if $\gamma = R$.
'$\alpha \mid \gamma$'	abbreviation for transitions of the form $\alpha \to \alpha \mid \gamma$ (these transitions do not modify the content of the tape).

Hint: The number n in unary representation consists of n ones. Also, you might want to extend the alphabet Γ to put temporary symbols on the tape.

2 Turing Again [Exam]

a) Construct a TM M that multiplies two positive integers $a, b \geq 1$ (both encoded in unary, i.e., n is encoded with n ones). At the beginning, the tape contains the number a in unary, followed by '$\times$', followed by the number b in unary. The head is positioned over the leftmost '1' of a. After the calculation, the tape should only contain the unary encoded result c (with $c = a \cdot b$), and the head should be positioned over the leftmost '1' of c.

Hint: Empty cells contain the symbol $\square$. You may extend the alphabet if you want.

(i) Describe how your TM M functions in three to four sentences.

(ii) Give a DFA (try to use a small number of states) that operates your TM M. Use the following notation for your transitions ($\gamma = L/R/N$ indicates a direction in which the head should move, i.e. left, right or none):

$\alpha \mid \gamma$	Read α at the current position and then move the head in the direction of γ.
$\alpha \to \beta \mid \gamma$	Read α at the current position, replace it with β, and then move the head in the direction of γ.
$\alpha \to \square \mid \gamma$	Read α at the current position, delete it, and then move the head in the direction of γ.

b) Let M_1 be a TM with the following property: The tape is not infinite in both directions, but just in one direction, i.e., the tape has cells in the range of $[0, \infty)$. Let M_2 be a TM with a normal tape, i.e., infinite in both directions ($(-\infty, \infty)$). Can a TM of type M_1 compute everything that a TM of type M_2 can compute and vice versa? If yes, show how to simulate M_1 on M_2 and the other way around. If not, give a function that can only be computed on one of the two and show why this is the case.

3 An Unsolvable Problem

It's the first day of your internship at the software firm Bug Inc., and your boss calls you to his office in order to explain your task for the next three months. He says that many clients complain that the programs of Bug Inc. often contain faulty loops that never terminate. In order to prevent such errors in future, you are asked to implement a program that may check whether a given program will halt on all possible inputs or not.

Having succeeded the DES course with flying colors, you know that this is not possible. You happily show the proof by contraction to your boss. Namely, you show him that by assuming a procedure `halt(P:Program):boolean` that takes a program `P` and decides whether `P` halts on all possible inputs or not. Then, you construct a program `X` that terminates if `halt(X)` is false and loops endlessly if `halt(X)` is true, which yields the desired contradiction.

a) Your boss still disagrees and proposes the following method: `halt(Y)` simply simulates the execution of program `Y`. If the program terminates it returns true, and if it loops it returns false. Where is the problem of this approach?

b) Your boss is finally convinced but argues that your proof is a very special case that hardly reflects reality. Are there assumptions under which it is always possible to check whether a program halts or not?