ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2013 Prof. R. Wattenhofer / K.-T. Foerster, T. Langner, J. Seidel

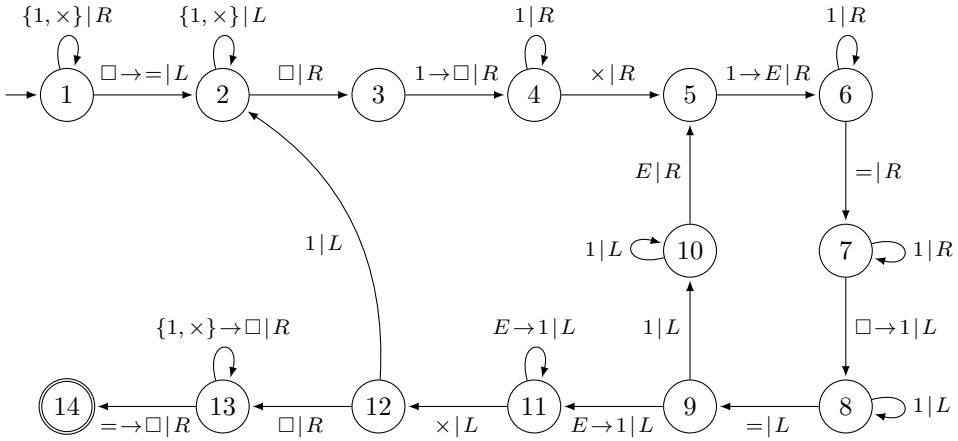
Discrete Event Systems

Solution to Exercise Sheet 5

1 Turing again

a) (i) First we write a '=' (as a seperator) to the right of the last 1 of b (state 1). Then we remove the leftmost 1 of a (states 2–4) and write for every 1 of b a 1 to the right of '=' (states 5–10) by replacing every already used 1 with 'E'. When there are only 'E's to the left of '=' (until '×'), we replace them with '1's (states 11–12) and start again. This is repeated until there are no '1's to the left of '×'. When this is the case, we have calculated the correct number to the right of '=' and we only need to remove all superfluous symbols (states 13–14).

(ii) A TM operated by the following DFA computes the desired output:



b) Obviously, every program that can run on a TM of type M_2 can also be run on a TM of type M_1, just ignore that additional tape exists. The other directions is a bit harder: The (limited) TM M_1 implements a mapping of a cell index x in M_2 to a cell index $f(x)$ in M_1 as follows:

$$f(x) = \begin{cases} 2x & x \geq 0 \\ 2|x| - 1 & x < 0 \end{cases} .$$

To implement this, the TM M_1 works in two modi A and B, defined as follows:

Modus A:

$$\begin{aligned} M_2 \to R \quad &\Rightarrow \quad M_1 \to 2R \\ M_2 \to L \quad &\Rightarrow \quad M_1 \to 2L \end{aligned}$$

Modus B:

$$\begin{aligned} M_2 \to R \quad &\Rightarrow \quad M_1 \to 2L \\ M_2 \to L \quad &\Rightarrow \quad M_1 \to 2R \end{aligned}$$

M_1 starts in modus A. If M_1 reaches the left end of the tape and cannot execute going to the left twice ($2L$), then M_1 goes one step to the right and switches to the other modus (in some sense, this is a similar construction as mapping the set of integers to the set of natural numbers).

2 An Unsolvable Problem

a) It is surprisingly easy to prove that your boss is demanding too much. Assume a function `halt(P: Program): boolean` which takes a program `P` as a parameter and returns a boolean value denoting whether `P` terminates or not.

Now consider the following program `X` which calls the `halt()` function with itself as an argument just to do the contrary:

```
function X() {
  if (halt(X))
    while(true);
  else
   return;
}
```

Obviously, if `halt(X)` is true `X` will loop forever, and vice versa.

b) If the simulation stops we can definitively decide that the program does not contain an endless loop. However, while the simulation is still running, we do not know whether it will finish in the next two seconds or run forever. Put differently: There is no upper bound on the execution time of the simulation after which we can be sure that the program contains an endless loop.

c) As we have seen, it is not possible to predict whether a general program terminates or not. However, under certain constraints we can solve the halting problem all the same. For example, consider a restricted language with only one form of a loop (no recursion etc.):

```
for (init; end; inc) {...}
```

where `init`, `end` and `inc` are constants in $\mathbb{Z}$. The loop starts with the value `init` and adds `inc` to `init` in every round until this sum exceeds `end` if `end` > 0 or until it falls below `end` if `end` < 0. Obviously, there is a simple way to decide whether a program written in this language terminates: For every loop, we check whether sgn(`inc`) = sgn(`end`), where sgn$(\cdot)$ is the algebraic sign. If not, the program contains an endless loop (unless `init` itself already fulfills the termination criterion which is also easy to verify).

3 Dolce Vita in Rome

We define the following random variables.

X = number of ice creams bought in total
X_i = indicator variable for buying ice cream at shop i

That means X_i is 1 if Hector and Rachel buy ice cream at shop i and 0 otherwise. Since the probability that the i-th shop is the best so far equals $\frac{1}{i}$ and the expectation of an indicator variable is simply the probability of it being 1, we have

$$\mathbf{E}[X_i] = \frac{1}{i} \ .$$

Furthermore, we can express X as $\sum_{i=1}^{n} X_i$ and by using linearity of expectation, we obtain:

$$\mathbf{E}[X] = \mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} \frac{1}{i} = H_n \ .$$

Here H_n is the n-th *Harmonic Number*. H_n grows about as fast as the natural logarithm of n. The reason for this is that the sum of the first n harmonic numbers can be approximated by

$$\int_1^n \frac{1}{x} \mathrm{d}x = \ln(n) \ .$$

More precisely, we have $H_n = \ln(n) + \mathcal{O}(1)$ and thus the two students roughly consume a logarithmic number of ice creams (in the total number of shops n).