Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Networked Systems Group (NSG)

HS 2016

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

Discrete Event Systems

Solution to Exercise Sheet 4

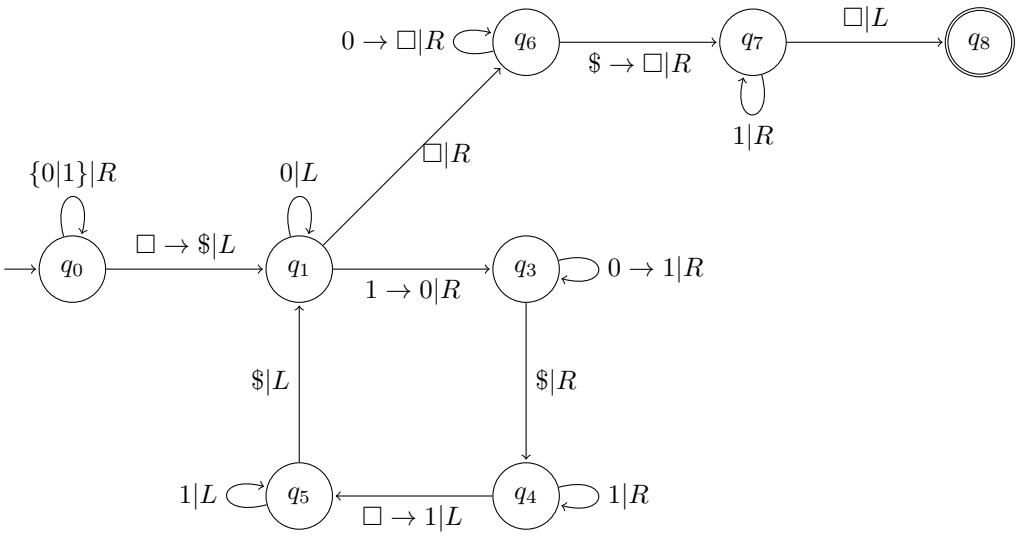
1 Designing Turing Machines

The proposed Turing machine decrements the value of a until $a = 0$. In each step, it adds a '1' to the output:

1. Move the TM head to the right of a and place a \$ sign. We will use this marker to return to the LSB of a.
2. Look at the LSB of a. If it is '1', we change it to 0 (transition between q_1 and q_3) and move to the right. Then, we continue moving to the right until we hit a $\square$, which is changed to a '1' (transition q_4 to q_5). Finally, we move back to the LSB of a.
3. If the LSB of a is '0', we search for the first '1' in a from the right (loop on q_1 and transition from q_1 to q_3).

3.1 If we find a '1', we change it to '0'. While moving back to the \$ symbol, we change all '0' to '1' (self-loop on q_3). Then, we proceed as in point 2 after passing the \$ symbol.

3.2 If we don't find a '1' in a at all (transition q_1 to q_6), we start the cleanup procedure: Remove all 0 on the right of the \$ symbol, and finally remove the \$ symbol itself and move to the right of u.



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 $\mathrm{sgn}(\texttt{inc}) = \mathrm{sgn}(\texttt{end})$, where $\mathrm{sgn}(\cdot)$ 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).