Formal Methods
Fall 2017

S05

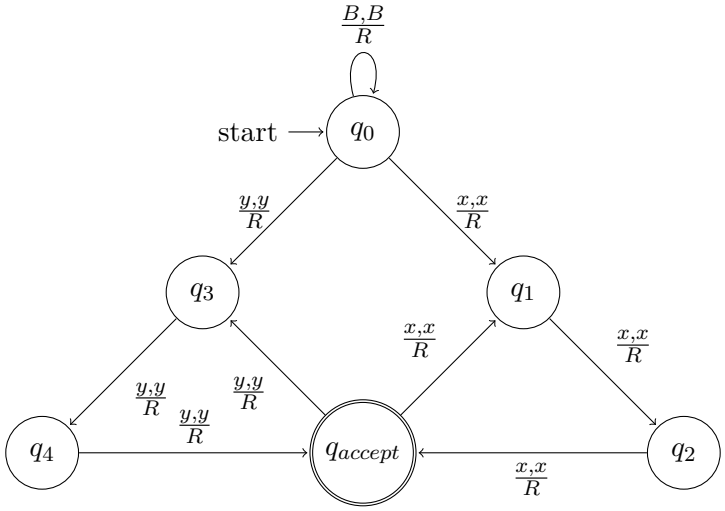
Professor : Ultes-Nitsche Ulrich
Assistant : Christophe Stammet

Submitted by Sylvain Julmy

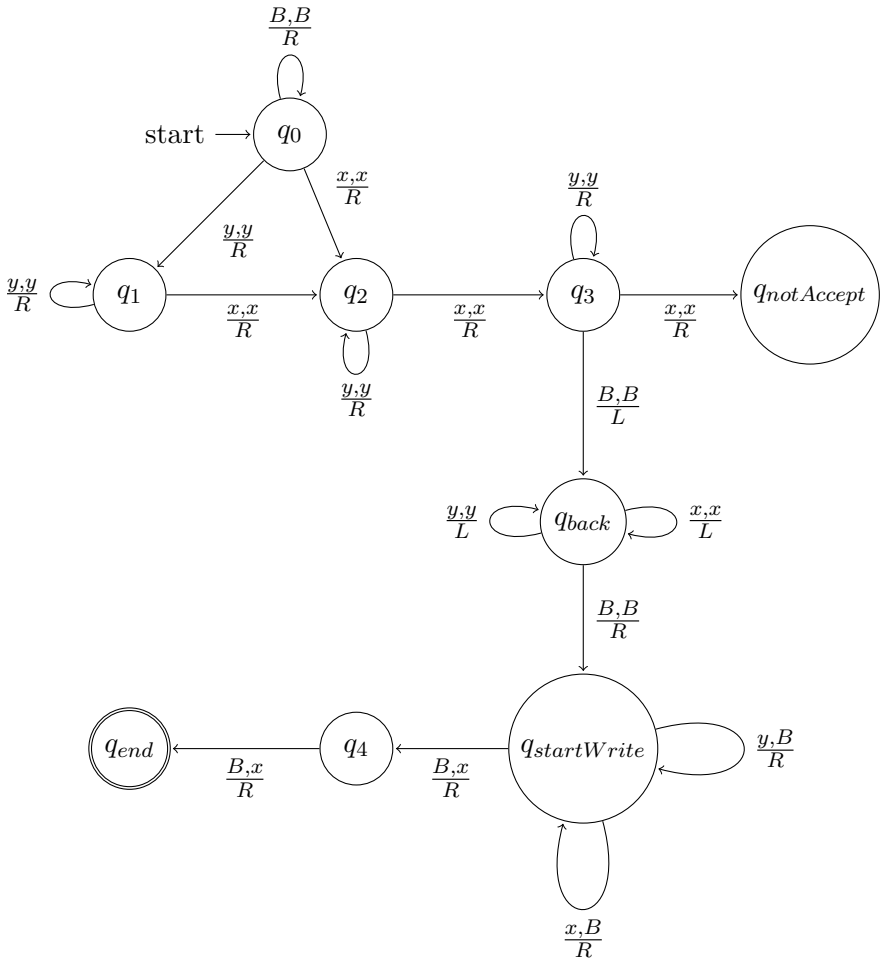
Exercise 1

(1)

We assume that the language is on the right of the Turing Machine tape.



(2)



Exercise 2

In order to prove the non-computability of the Busy Beaver function, we can use the following steps :

- Generate every Turing machine with n states, there is a finite number of these.
- For each machine, we run it and count out the number of 1 on the tape when the machine halt.

The problem is that we don't know which of the machine goes forever or not, and that is the halting problem solve in previous exercises sheet.
The other way is to use the statement from the exercice. If $a > b$, $BB(a)$ would always be greater than $BB(b)$. We imagine we have a Busy Beaver TM with n states, independently of the output of $BB(n)$, the Busy Beaver TM with $n+1$ states could always write at least 1 more 1 on the tape by simply adding a state to $TM_{BB}(n)$ which simply write an 1 and stop in an accepting state. So the $BB(n)$ function is strictly increasing.

Exercise 3

(1)

Given a Turing Machine M that decide the recursive language L, we can create the Turing Machine that decide $\overline{L}$: On input x, run M on x. Then, x is accepted if M rejects x and x is rejected if M accepts x. So we have construct the Turing Machine for $\overline{L}$, which stop in an accepting state where $w \in \overline{L}$ and stop in an non-accepting state where $w \notin \overline{L}$.

(2)

Recursive language are closed under the union.
Given two Turing Machine M_1 and M_2 which are accepting L_1 and respectivly L_2, we construct the Turing machine that decide $L_1 \cup L_2$: on input x, run M_1 and M_2, add accept x only if and only if M_1 or M_2 accept x.
Recursive language are closed under the intersection.
Given two Turing Machine M_1 and M_2 which are accepting L_1 and respectivly L_2, we construct the Turing machine that decide $L_1 \cap L_2$: on input x, run M_1 and M_2, add accept x only if and only if M_1 and M_2 accept x.