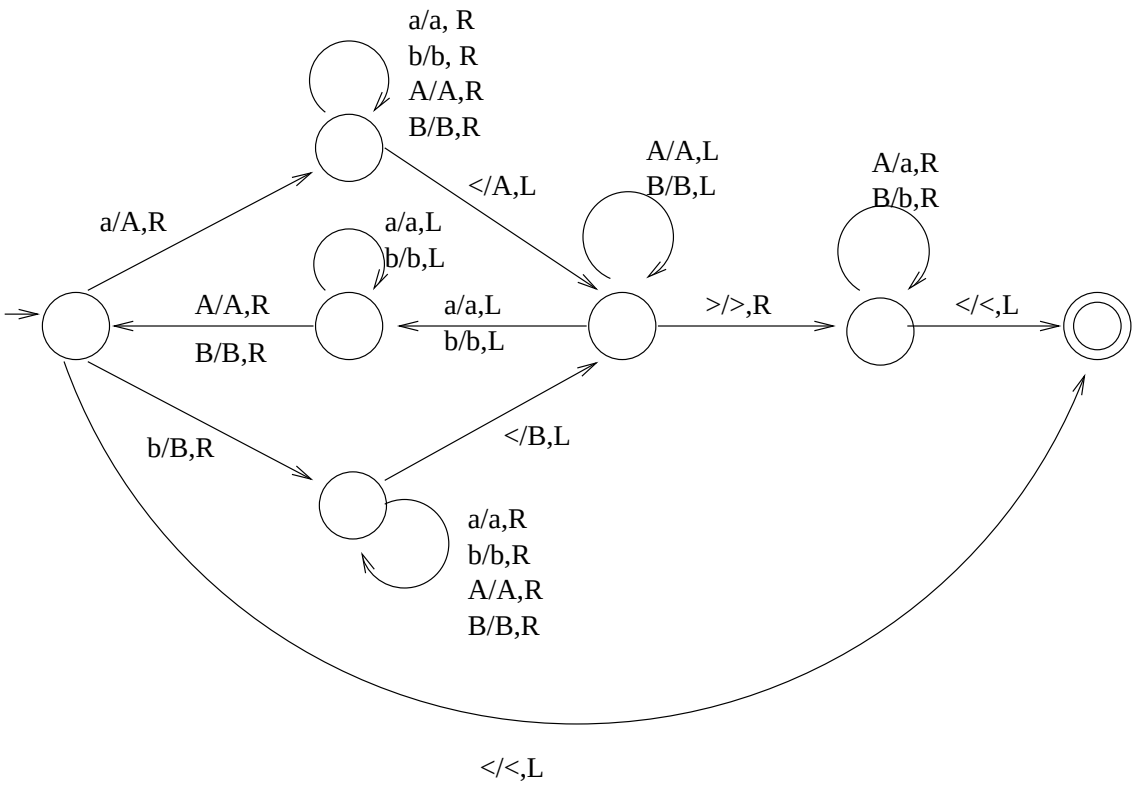
Exercise session 8

1. Simulate the behaviour of a given Turing machine as a game. (Your exercise teacher will have some transition diagrams of different size of Turing machines.) If there are enough students in the group, you can make a competition between the machines in recognizing a language. Each student corresponds a state, performs a required writing operation on the tape, moves the reading head and gives the control to the correct successor state. The accepting or rejecting final state reports the result. (One point for each participant of the game!)

2. What does the following Turing machine do? Simulate its behaviour with different input strings of as and bs!



3. Construct a standard Turing machine, which subtracts one from the input binary string. I.e. an integer n is given as a binary string x, in which the most

significant bits are left and least significant right (e.g. 8=1000). If $n > 0$, the machine replaces x by the binary representation of integer $n-1$. If $n = 0$, the tape remains the same, and the machine moves to the rejecting final state.

4. Construct a standard Turing machine, which lists all words in language 1^n, $n = 0, 1, ...$ The machine begins with emty tape, and generates unary numbers 1, 11, 111, 1111, ... (Notice! Your machine will never halt.)

5. Construct a Turing machine, which generates binary representations of all natural numbers 0,1,00,01,10,11,000,001,... You can represent the binary numbers as the least significant bit on left.

6. Construct a Turing machine, which reads the input string, until it finds two consecutice as. The input alphabet is $\{a, b\}$.

7. Construct a Turing machine, which divides the input number represented as binary number by two, if it is even. With odd numbers the machine transfers to the rejecting final state. The binary numbers are represented

   a) the most significant bit on right
   b) the most significant bit on left

8. Construct a standard Turing machine, which recognizes the language $\{ww^R | w \in \{a, b\}^*\}$.

9. Construct a standard Turing machine, which transform the string w into string ww^R ($w \in \{a, b\}^*$).

10. Construct a standard Turing machine, which recognizes the language $\{w \in \{a, b\}^* | w$ contains equal number of a's and b's$\}$.

11. Construct a 2-tape Turing machine, which gets input string $w \in \{a, b\}^*$ on its 1. tape, and writes string w^R ($=w$ in reversed order) onto 2. tape.
    (Hint: Yoi can simulate ϵ-transitions with Turing machines by transitions $a/a, S$ or $b/b, R$, in which S is a new direction "stay".)

12. Let's consider the following nondeterministic Turing machine:

    $M = (\{q_0, q_1, q_2, q_f\}, \{0, 1\}, \{0, 1\}, \delta, q_0, q_f, q_{no})$,

    whose transition diagram is defined as

$\delta(q_0, 0) = \{q_0, 1, R), (q_1, 1, R)\}$
$\delta(q_1, 1) = \{q_2, 0, L)\}$
$\delta(q_2, 1) = \{q_0, 1, R)\}$
$\delta(q_1, <) = \{q_f, <, R)\}$

What does the machine do? Hint: simulate its behaviour with different binary strings. (You can use JFLAP, if you want.)

13. Construct a nondeterministic Turing machine, which recognizes the language $\{ww|w \in \{a, b\}^*\}$.

14. Consider the nondeterministic Turing machine TEST_COMPOSITE (look at the English material given to you), which recognizes composite numbers. Could you make a prime tester machine by changing the accepting and rejecting states of the machine? Justify your answer!

More challenging:

15. Construct a standard Turing machine, which begins with empty tape and generates as many 1s as possible on its tape before halting.(The machine must halt finally!) The machine can be composed of at most 3 states and the accepting final state.

16. Describe (unformally) a nondeterministic Turing machine, which recognizes the following language: The words of the language are of form $w_1\#w_2\#...\#w_n$ for any n such that for all i $w_i \in \{a, b\}^*$ and for some j w_j is the binary representation of integer j. N.B.! Utilize the nondeterminism as much as possible, i.e. prefer much branched but short paths. (The machine guesses the correct path nondeterministically.)

17. Describe (unformally) a nondeterministic Turing machine, which solves the *Hamilton circle problem*: given a directed graph, decide if there is a path which goes through all vertices exactly once before returning back to the starting vertex. (Hint: you can use a multiple tape Turing machine for representing the graph as an adjecent list or matrix. You can suppose that alphabets $a, ..., z$ are enough for naming the vertices.)