Homework 1

Classical computation

Task 1. Suppose that we have given a binary input (input alphabet is {0,1}). Construct a Turing machine that checks whether the numbers of 0's and 1's in the input are the same. If number of 0's is equal to number of 1's, then machine should write 1 in first symbol of input, otherwise - first symbol on the tape should be 0.

Example:

0011001101 - here machine should write 1 in first symbol after the work.

0111111 - here machine should write 0 in first symbol after the work.

Solution.

We assume that the external alphabet A is {0,1}. The alphabet S ={_,0,1,∗} consists of the symbols of the external alphabet, the empty symbol _, and auxiliary mark *. Machine starts work in state q0 and ends work in state qf. We have the following set of states: {q0,qf,r0,r1,l0,l1,l2,lf,q3,q4}. In the beginning head of the machine is reading the first input symbol.

In each iteration we try to remove from the word one 0 (by moving to the right) and one 1 (by moving to the left). Work starts with first iteration:

If the word is empty, accept by writing 1. If first symbol is 0, replace with *. Machine starts to move to the right.

(q0,0)→(r1,∗,+1)

(q0,1)→(r0,1,+1)

(q0,_)→(qf,1,−1)

Moving to the right:

(r0,0)→(r1,∗,+1)

(r0,1)→(r0,1,+1)

(r0,*)→(r1,∗,+1)

(r0,_)→(l0,_,-1)

(r1,0)→(r1,0,+1)

(r1,1)→(r1,1,+1)

(r1,*)→(r1,∗,+1)

(r1,_)→(l1,_,-1)

After reaching the right end of the word, state l0 means there was no symbol 0, state l1 means there was removed symbol 0.

Moving to the left:

(l0,0)→(lf,1,-1) - should not occur since state l0 means there were no symbols 0.

(l0,1)→(lf,1,-1)

(l0,*)→(l0,*,-1)

(l0,_)→(q3,_,+1)

(l1,0)→(l1,0,-1)

(l1,1)→(l2,*,-1)

(l1,*)→(l1,*,-1)

(l1,_)→(q4,_,+1)

(l2,0)→(l2,0,-1)

(l2,1)→(l2,1,-1)

(l2,*)→(l2,*,-1)

(l2,_)→(r0,_,+1)

(lf,0)→(lf,0,-1)

(lf,1)→(lf,1,-1)

(lf,*)→(lf,*,-1)

(lf,_)→(q4,_,+1)

After reaching the left end of the word:

- state l0 means there was no symbol 0 and no symbol 1, so input should be accepted (state q3 to finalize procedure),
- state l1 means there was removed symbol 0 but not removed symbol 1, so input should be rejected (state q4 to finalize procedure),
- state l2 means that symbol 0 and symbol 1 were removed, so proceed with next iteration starting at state r0,
- state lf means that there was removed symbol 1 but not removed symbol 0, so input should be rejected (state q4 to finalize procedure).

Finishing:

(q3,0)→(qf,1,−1)

(q3,1)→(qf,1,−1)

(q3,*)→(qf,1,−1)

(q4,0)→(qf,0,−1)

(q4,1)→(qf,0,−1)

(q4,*)→(qf,0,−1)

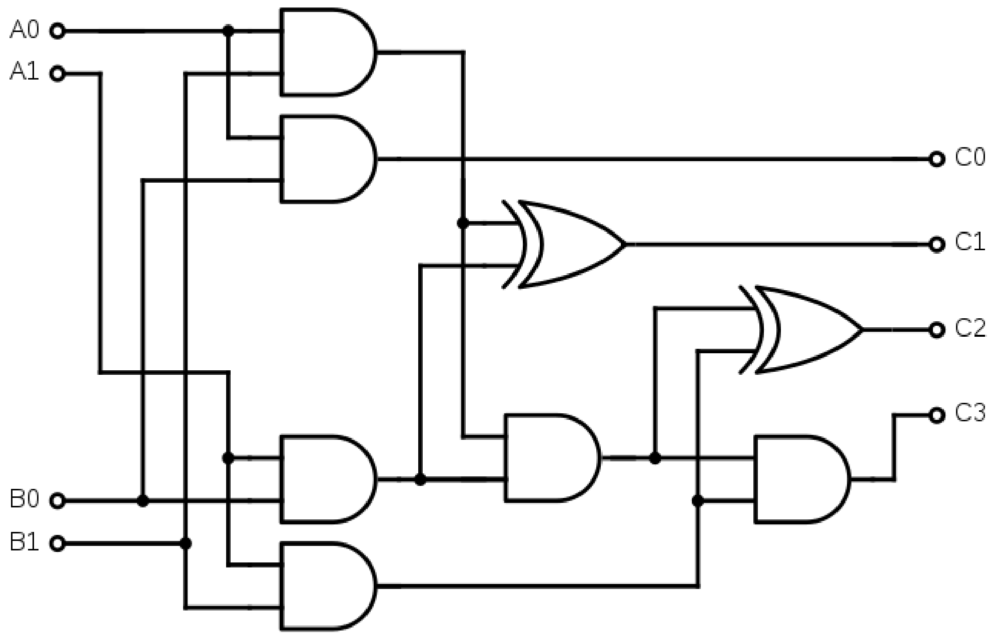
Task 2. Construct a Boolean circuit that outputs the multiplication of 2 2-bit numbers. Input is represented as 4 bits, first 2 bits represent the first number, last 2 bits represent the second number. Output will be of 4 bits. Examples:

Input 0110 means 01*10=1*2, output should be 2=0010

Input 1111 means 11*11=3*3, output should be 9=1001

Solution.

Here we use AND and XOR operations. C0 is least significant (right-most) bit of the output, C3 – most significant (left-most) bit of the output.



Task 3. Reduce the following Boolean function to the 3-SAT formula (remember the 3-CNF form of 3-SAT):

$$(x_1 \wedge x_2) \vee \neg(x_3 \wedge \neg x_4)$$

Solution.

$y_1 = x_1 \wedge x_2 \Rightarrow (x_1 \vee x_2 \vee \neg y_1) \wedge (\neg x_1 \vee x_2 \vee \neg y_1) \wedge (x_1 \vee \neg x_2 \vee \neg y_1) \wedge (\neg x_1 \vee \neg x_2 \vee y_1)$

$y_2 = \neg x_4 \Rightarrow (x_4 \vee y_2) \wedge (\neg x_4 \vee \neg y_2)$

$y_3 = x_3 \wedge y_2 \Rightarrow (x_3 \vee y_2 \vee \neg y_3) \wedge (\neg x_3 \vee y_2 \vee \neg y_3) \wedge (x_3 \vee \neg y_2 \vee \neg y_3) \wedge (\neg x_3 \vee \neg y_2 \vee y_3)$

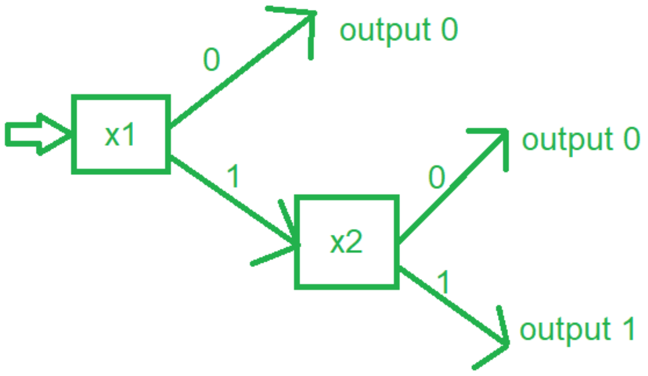
$z = y_1 \vee y_3 \Rightarrow (y_1 \vee y_3 \vee \neg z) \wedge (\neg y_1 \vee y_3 \vee z) \wedge (y_1 \vee \neg y_3 \vee z) \wedge (\neg y_1 \vee \neg y_3 \vee z)$

With all together we get the following formula:

$$(x_1 \vee x_2 \vee \neg y_1) \wedge (\neg x_1 \vee x_2 \vee \neg y_1) \wedge (x_1 \vee \neg x_2 \vee \neg y_1) \wedge (\neg x_1 \vee \neg x_2 \vee y_1) \wedge (x_4 \vee y_2) \wedge (\neg x_4 \vee \neg y_2) \wedge (x_3 \vee y_2 \vee \neg y_3) \wedge (\neg x_3 \vee y_2 \vee \neg y_3) \wedge (x_3 \vee \neg y_2 \vee \neg y_3) \wedge (\neg x_3 \vee \neg y_2 \vee y_3) \wedge (y_1 \vee y_3 \vee \neg z) \wedge (\neg y_1 \vee y_3 \vee z) \wedge (y_1 \vee \neg y_3 \vee z) \wedge (\neg y_1 \vee \neg y_3 \vee z) \wedge (z)$$

Task 4. We are given query type of algorithm. Suppose that our task is to compute the given Boolean function output by querying values of the variables. We are allowed to do fixed number of queries.

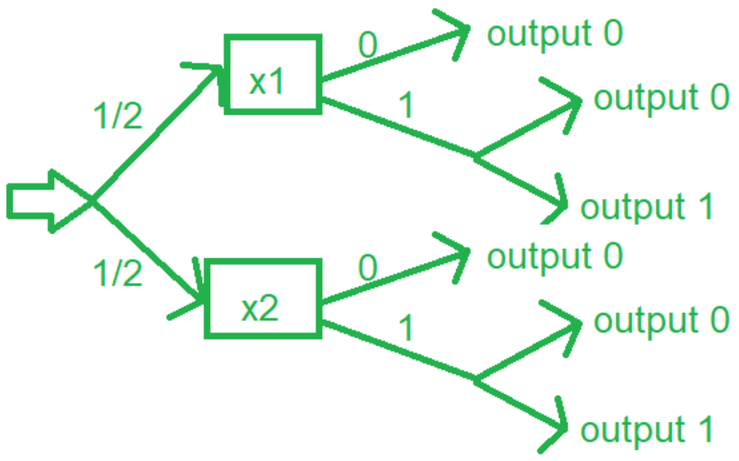
Example: $x_1 \wedge x_2$. We have to compute $x_1 \wedge x_2$ by querying variables. In case of deterministic algorithm, suppose that we can query 2 variables at most. Then we can design the following algorithm:



Here we first query x1. If x1=0, then algorithm will output 0. If x1=1, then algorithm will query the value of x2. If x2=0, then output 0, if x2=1, then output 1.

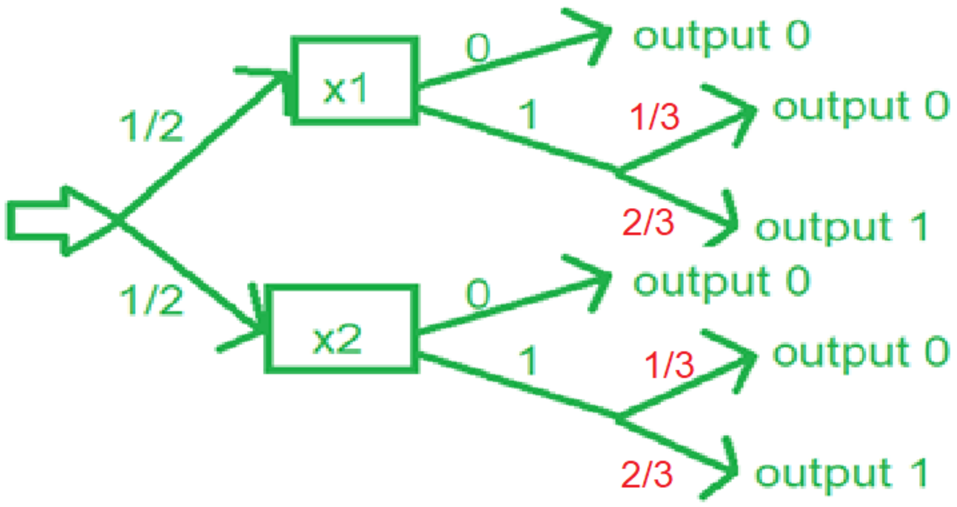
Design a probabilistic query algorithm to solve $x_1 \wedge x_2$ with one query, such that probability of correct answer is more than 1/2.

Example of probability 1/2:



Here we pick with probability 1/2 x1 or x2, ask its value. If value is 0, we output 0, but if value is 1, we output 0 or 1 with equal probabilities. So, if value of variable is 1, we have 50% of error, because we do not know the value of the second variable.

Solution.



To complete the task we add probabilistic outcomes to the last decision. We pick one variable with 50% probability. Then, if value is equal to 0, value of AND function is 0. If value of the variable is 1, then we say with 1/3 probability that AND function is 0 and with 2/3 probability that AND function is 1.

Analysis. If at least one variable is 0, then With probability at least 1/2 we pick that variable, and say that value of AND is 0. With remaining 1/2 probability we pick variable with value 1. In such case with probability 1/3 we output AND value as 0, so total probability is 1/2+1/2*1/3=4/6=2/3 – this is the probability to correctly say that AND function value should be 0.

If both variables are 1, then in each branch we output 1 with probability 2/3.

As we see, for any input our probabilistic algorithm gives correct output with probability 2/3, by querying one variable.