

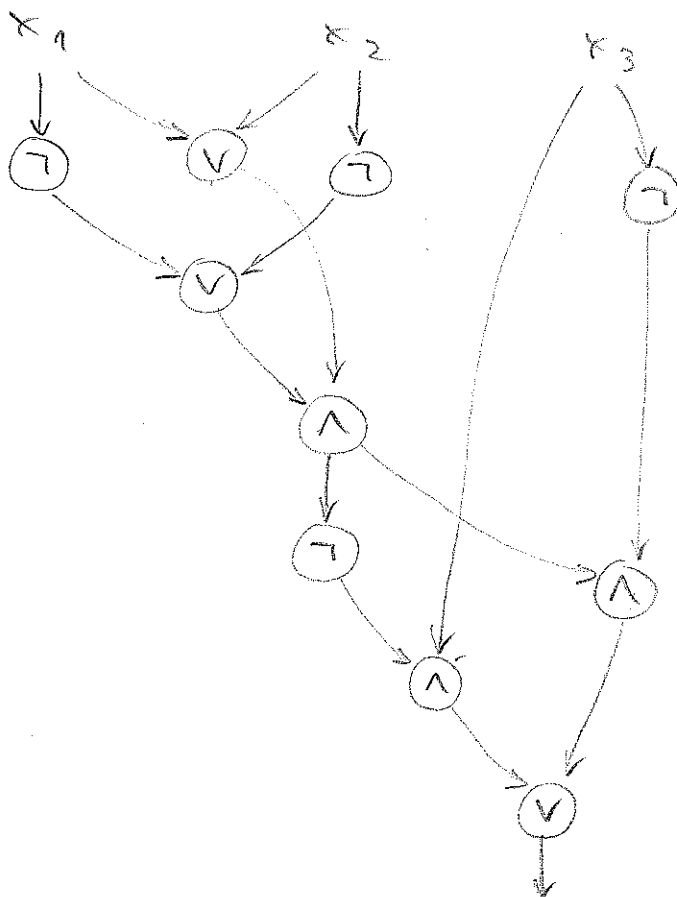
# Exercises on boolean circuits

E 11.1

Exercise 1: Consider the boolean expression

$$E = (x_3 \wedge \neg((x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2))) \vee (\neg x_3 \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2))$$

Construct a boolean circuit that computes the value of  $E$ , given inputs for  $x_1, x_2, x_3$



What is the size of this circuit? 6

( $\neg$  does not count)

What is the depth? 4

How does the size compare to the length of  $E$

Exercise: Reduction from Reachability to Circuit Value

E99.2

Reachability: given a directed graph  $G = (V, E)$   
with  $V = \{1, \dots, n\}$  and  $E \subseteq V \times V$ ,  
is there a path from node 1 to node  $n$  in  $G$ .

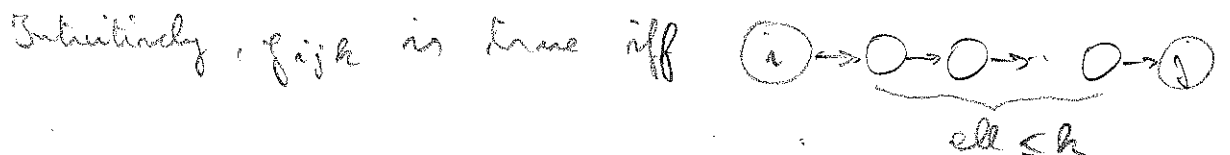
Circuit Value: given a boolean circuit  $C$  without input variables, is the output of  $C$  equal to  $T$ ?

We show how to reduce Reachability to Circuit Value, i.e. how to construct from a graph  $G$  a circuit  $R(G)$  such that:

1 is reachable from  $n$  in  $G$  iff  
the value of  $R(G)$  is  $T$

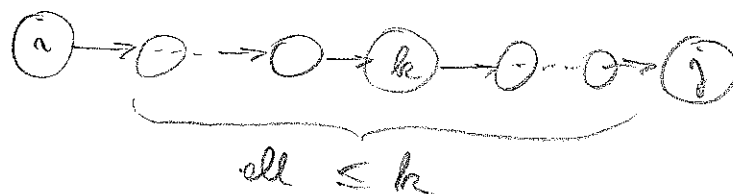
In  $R(G)$  we use gates of two forms:

1)  $g_{ijk}$ , with  $1 \leq i, j \leq n$  and  $0 \leq k \leq n$



i.e., there is a path from  $i$  to  $j$  not using any intermediate node bigger than  $k$

2)  $h_{ijk}$ , with  $1 \leq i, j, k \leq n$  iff



i.e., there is a path from  $i$  to  $j$  not using any intermediate node bigger than  $k$ , but using  $k$  as intermediate node.

We describe now the gates and how they are connected.

- for  $k=0$ , all  $g_{ij0}$  gates are constant gates

$$g_{ij0} = T \quad \text{iff} \quad \left. \begin{array}{l} - i = j, \text{ or} \\ - i \rightarrow j \text{ in } G \end{array} \right\} \begin{array}{l} \text{This is how } G \\ \text{is reflected in} \\ R(G) \end{array}$$

(note that there are no  $h_{ij0}$  gates)

- for  $k=1, 2, \dots, n$

- $h_{ijk}$  is an AND gate with predecessors

$$g_{i,k,k-1} \quad \text{and} \quad g_{k,j,k-1}$$

- $g_{ijk}$  is an OR gate with predecessors

$$g_{ij,k-1} \quad \text{and} \quad h_{ijk}$$

The output gate is  $g_{m,n}$

Note that  $R(G)$  can be computed from  $G$  in logarithmic space.

Note that the circuit  $R(G)$  is legitimate, since it contains no cycles: we can remove the gates  $1, 2, \dots, 2n^3 + n^2,$

- in non-decreasing order of the third index, and
- with  $h_{ijk}$  preceding  $g_{ijk}$

We have to show that the value of the output gate of  $R(G)$  is  $T$  iff there is a path from 1 to  $n$  in  $G$ .

We prove by induction on  $k$  that the values of the gates correspond to the informal meaning we gave them:

- for  $k=0$ : this holds
- if it is true up to  $k-1$ , the definitions of  $g_{ijk}$  and  $h_{ijk}$  guarantee that it is true also for  $k$ .

Exercise: A boolean function  $f$  is said to be monotone (E19.4)

if it has the following property: if one of the values changes from 0 to 1, then the value of  $f$  does not change from 1 to 0.

We show that  $f$  is monotone iff it can be expressed by a circuit with only AND and OR gates.

" $\Leftarrow$ " Consider a circuit  $C$  with only AND and OR gates expressing  $f$ .

We show by induction on the depth of a node  $N$ :

if the value of an input  $x$  changes from 0 to 1 then the value of  $N$  does not change from 1 to 0.

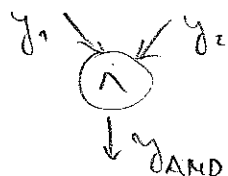
Base:  $\text{depth}(N) = 0$ , then  $N$  is either an input or a constant node.

- if it is a constant, its value does not change
- if it is an input different from  $x$ , - " -
- if it is input  $x$ , its value changes from 0 to 1 (and not from 1 to 0)

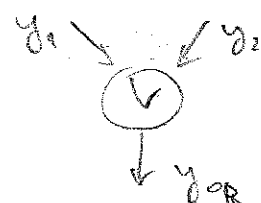
Induction: suppose that for all nodes at level  $b$ , the value does not change from 1 to 0.

Consider a node  $N$  at level  $b+1$ . We show that the value of  $N$  does not change from 1 to 0.

Case 1:  $N$  is an AND node



Case 2:  $N$  is an OR node



We need to consider various subcases, corresponding to the changes of  $y_1, y_2$  from 0 to 1.

old		new		$y_{AND}$		$y_{OR}$	
$y_1$	$y_2$	$y_1$	$y_2$	old	new	old	new
0	0	1	1	0	1	0	1
0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	1
0	1	1	1	0	0	1	1
1	0	1	1	0	0	1	1

We see that both for an AND node and for an OR node, the output value never changes from 1 to 0.

" $\Rightarrow$ " We show by induction on  $n$  that every monotone boolean function  $f(x_1, \dots, x_n)$  of  $n$  variables can be represented by a circuit with AND and OR gates only.

Base case: 0 arguments:  $f$  is constant, and hence monotone

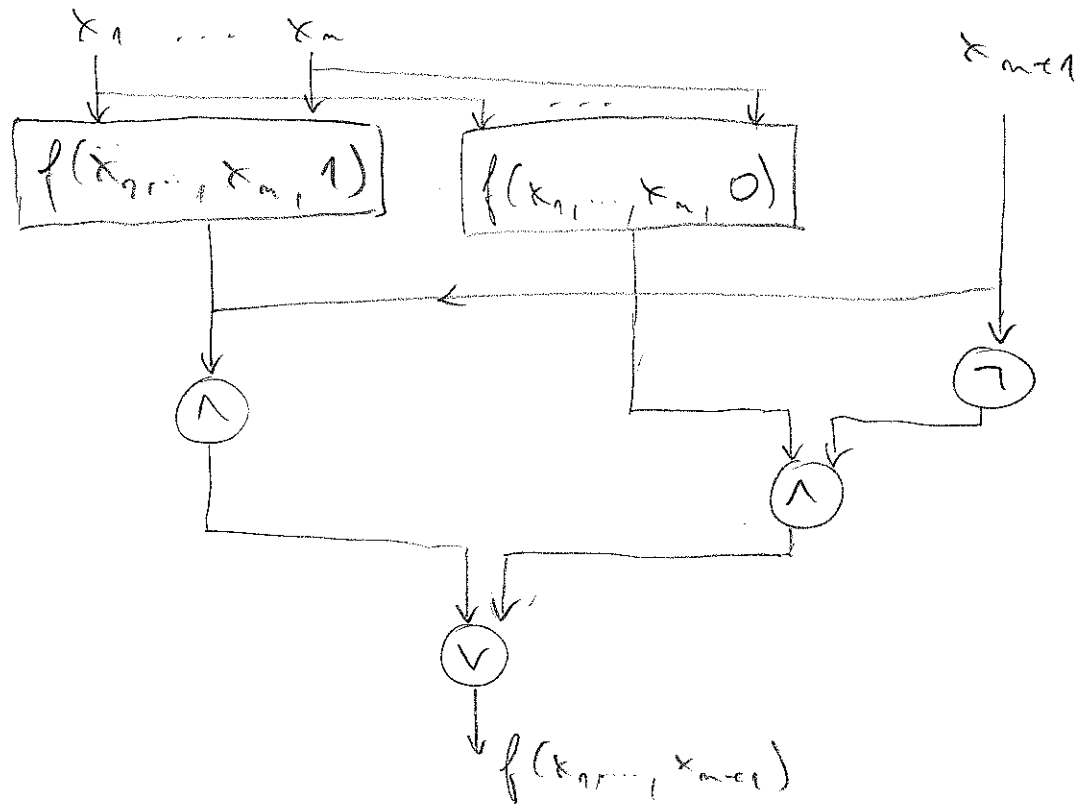
Inductive case: assume the claim holds for  $n$ .

We show it holds for a function  $f(x_1, \dots, x_{n+1})$ .

We exploit the fact that

$$f(x_1, \dots, x_n, x_{n+1}) = (x_{n+1} \wedge f(x_1, \dots, x_n, 1)) \vee (\bar{x}_{n+1} \wedge f(x_1, \dots, x_n, 0))$$

Hence, we can construct a circuit  $C_f$  computing  $f(x_1, \dots, x_n, x_{n+1})$  as follows: (E 11.6)

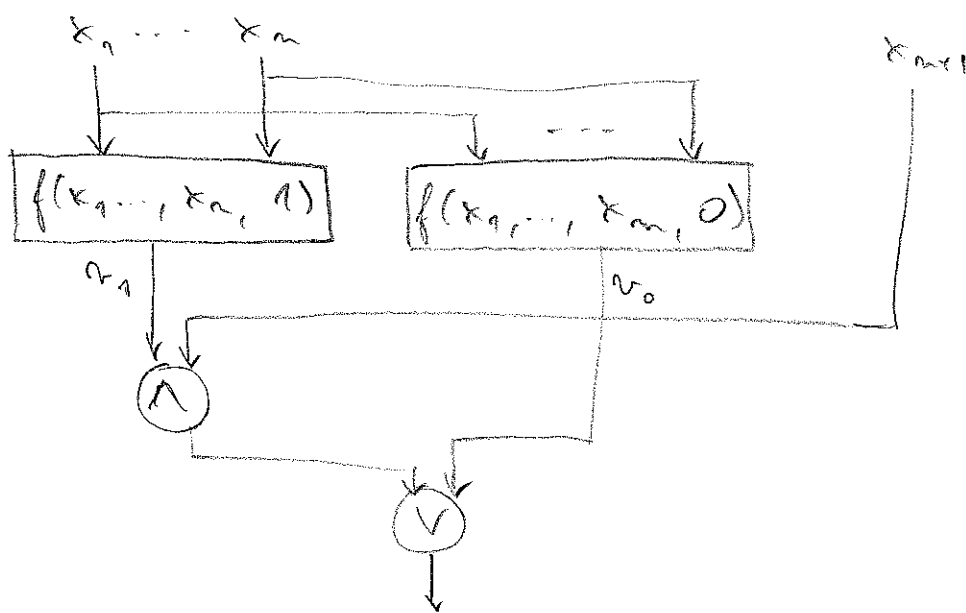


Observe that, since  $f(x_1, \dots, x_{n+1})$  is monotone, we have that also  $f(x_1, \dots, x_n, 0)$  and  $f(x_1, \dots, x_n, 1)$  are monotone.

Hence, since  $f(x_1, \dots, x_n, 0)$  and  $f(x_1, \dots, x_n, 1)$  are  $n$ -variable monotone functions, by inductive hypothesis they can be represented by circuits with AND and OR gates only.

Hence, it suffices to show that we can get rid of the only remaining NOT gate.

Consider the following circuit  $C_f'$  in which we have eliminated the NOT gate.



Let us consider the possible values of  $f$  in  $C_f$  and  $C'_f$ .  
 It depends on the values of  $x_n, v_0, v_1$

	$x_n$	$v_0$	$v_1$	$C_f$	$C'_f$
1)	0	0	0	0	0
2)	0	0	1	0	0
3)	0	1	0	1	1
4)	0	1	1	1	1
5)	1	0	0	0	1
6)	1	0	1	1	0
7)	1	1	0	0	1
8)	1	1	1	1	1

Note that  $C_f = C'_f$ ,  
 except for case (7).

However, since  $f(x_1, \dots, x_{n+1})$  is monotone, cases (3) and (7) cannot occur, since they would mean that  $f(x_1, \dots, x_n, x_{n+1})$  changes from  $v_0 = 1$  for  $x_{n+1} = 0$  to  $v_1 = 0$  for  $x_{n+1} = 1$ .

Hence,  $C'_f$  is the correct circuit consisting of AND and OR gates only and computing  $f(x_1, \dots, x_{n+1})$ .

Consider the problem FALSE-SAT:

Given a boolean expression  $E$  that is false when all its variables are made false, is there some other truth assignment that makes  $E$  false, besides all-false?

Decide whether the problem is in NP or coNP.

Describe its complement.

If the problem or its complement is NP-complete, prove it.

Proof:

The problem is NP-complete.

- In NP: given a boolean expression  $E$ , we need to check:

1) that  $E$  is false when all variables are assigned false

2) that there is some other truth-assignment making  $E$  false

(1) can be done in poly-time by a DTM

(2) can be done in poly-time by a NTM

guess a truth-assignment  $T$  different from all false, and answer yes if and  $T, F$  evaluates to false

- NP-hard: by a reduction from SAT

Let  $E$  be a boolean expression with variables  $x_1, \dots, x_n$ .

We construct an expression  $E'$  s.t.  $E \in \text{SAT}$  iff  $E' \in \text{FALSE-SAT}$

1) Test if  $E$  is true when all variables are false (polynomial)

If so,  $E \in \text{SAT}$ , and we convert it to a fixed expression that is in FALSE-SAT, e.g.  $x \wedge y$ .



2) Otherwise, let  $E'$  be  $\neg E \wedge (x_1 \vee x_2 \vee \dots \vee x_n)$ .

(E 11.9)

Clearly, the reduction is poly-time.

We have that  $E'$  is false when all of  $x_1, \dots, x_n$  are false.

Notice that in case (2),  $E$  is false when all variables are false.

Hence, if  $E \in \text{SAT}$ , then it is satisfied by a truth assignment  $T$  different from all-false.

Thus,  $\neg E$  is made false by  $T$ , and  $E' \in \text{FALSE-SAT}$ !

Conversely, if  $E' \in \text{FALSE-SAT}$ , then since  $x_1 \vee \dots \vee x_n$  is false only for the all-false truth assignment, there must be some other truth-assignment  $T$  that makes  $\neg E$  false. Then  $T$  makes  $E$  true, and  $E \in \text{SAT}$ .