Boolean Circuits

Lucie Kundratová

14.10.2020

Boolean function

A Boolean function with n inputs and m outputs is a function

$$f: \{0,1\}^n \rightarrow \{0,1\}^m$$

Boolean circuit

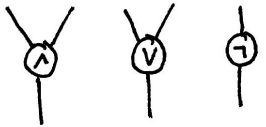
- directed acyclic graph
- inputs = nodes of indegree 0

  labeled with variable x_i or with a constant 0 or 1
- gates = nodes of indegree $k > 0$

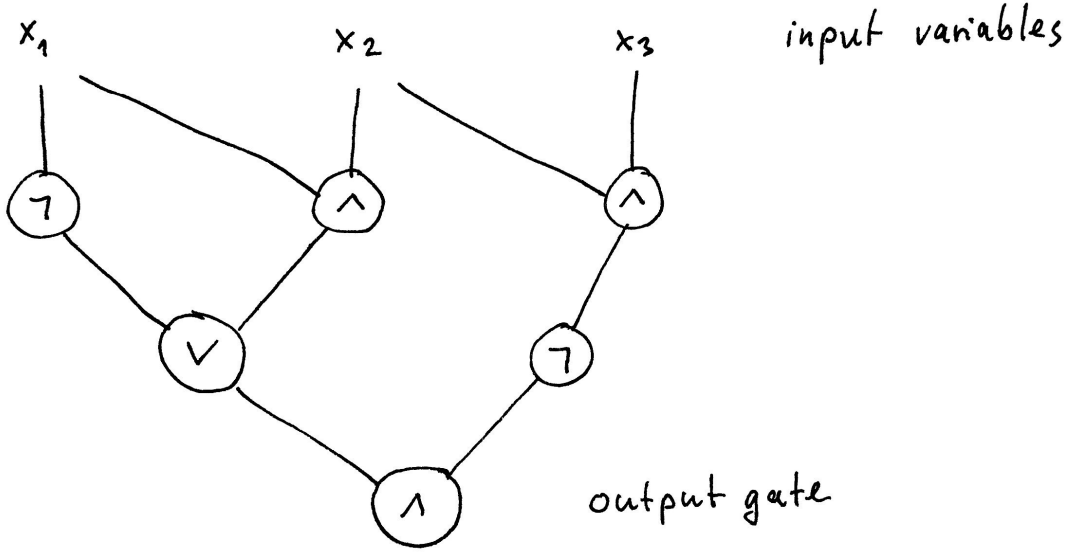
  labeled with a Boolean function on k inputs

  ↓

  AND, OR, NOT



- output node
- nodes connected by wires
- boolean formula = boolean circuit in which every node has the out-degree at most 1



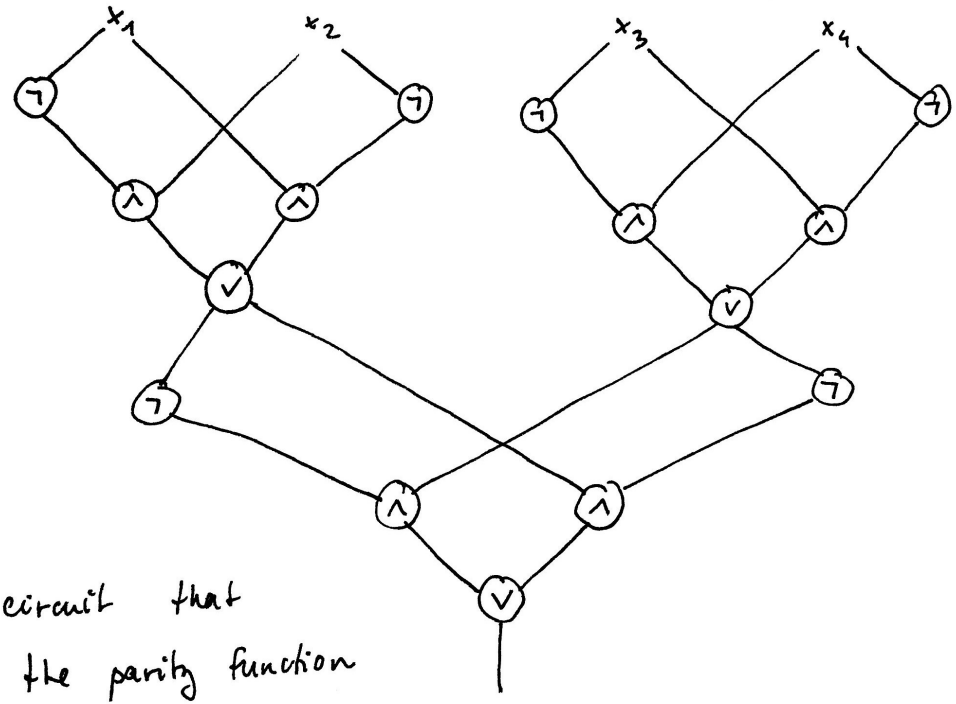
example of a Boolean circuit

- use functions to describe the input/output behavior of Boolean circuits
- Boolean circuit C with n input variables

  function $f_C: \{0,1\}^n \rightarrow \{0,1\}$
- $f_C(a_1, \ldots, a_n) = b$ ... C outputs b when its inputs $x_1, \ldots, x_n$ are set to $a_1, \ldots, a_n$
- C computes the function f_C
- multiple output gates → range $\{0,1\}^k$

example: Parity function

$parity_n : \{0,1\}^n \to \{0,1\}$ outputs 1 $\Leftrightarrow$ odd number of 1s



A Boolean circuit that computes the parity function on four variables.

Circuit family

A circuit family C

- infinite list of circuits $(C_0, C_1, C_2, \dots)$
- C_n has n input variables

Language

- subset of $\{0,1\}^*$
  - ↳ set of all finite binary strings

C **decides** a language A over $\{0,1\}$
if for every string w

$$\boxed{w \in A \iff C_n(w) = 1}$$

where n is the length of w.

Size & depth

The size of a circuit

- number of gates it contains

Two circuits are equivalent

- same input variables and output the same value on every input assignment

A circuit is size minimal

- if no smaller circuit is equivalent to it

A circuit family for a language is minimal

- if every C_i on the list is a minimal circuit

Size complexity of a circuit family $(C_0, C_1, C_2, \ldots)$

- the function $f: \mathbb{N} \to \mathbb{N}$
- $f(n)$ is the size of C_n

The depth of a circuit

- the length (number of wires) of the longest path from an input variable to the output gate

Depth minimal circuits

Depth complexity

Circuit size complexity

of a language is the size complexity of a minimal circuit family for that language

Circuit depth complexity - similarly

Example

Circuits that compute parity function on n variables with $O(n)$ gates.

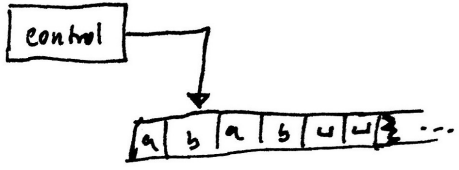
- binary tree with XOR functions = 2-parity function

A - language of strings - odd number of 1s.

$\Rightarrow$ A has circuit complexity $O(n)$

Turing Machines

- first proposed by Alan Turing in 1936
- infinite tape and unlimited memory
- tapehead - read
  - write symbols
  - move around the tape



Schematic of Turing machine

- initially - the tape contains only the input string
  - blank everywhere else

- it can store information by - writing it
  - reading it after moving the head

- continues computing → decides to produce an output by entering designated states
  - accept
  - reject

- if it doesn't enter an accepting/rejecting state, never halts

Formal definition of Turing machine

A Turing machine is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states
2. Σ is the input alphabet not containing the blank symbol $\sqcup$
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function
   - tells us how the machine gets from one step to the next
   - certain state q, head over a tape square with a and $\delta(q, a) = (r, b, L)$, the machine writes the symbol b replacing the a and goes to state r, moves to left
5. $q_0 \in Q$ is the starting state
6. $q_{accept} \in Q$ is the accept state
7. $q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$

How does TM computes?

- receives input $w = w_1 w_2 \cdots w_n \in \Sigma^*$ on the leftmost n squares
- the head starts on the leftmost square
- proceeds according to the rules
- if TM tries to move its head to the left off the tape, the head stays in the same place
- continues until it enters the accept or reject states.

Configuration of the Turing machine

= settings of current state, current tape contents, current head location

Types of TM

A deterministic Turing machine

- finite control = finite collection of states
- finite collection of tapes
- the exact action taken is governed by the current state, the symbols read and the next move function

A nondeterministic Turing machine

- the next move function is multivalued
- several computation on a given input, several output values
- accepts its input $\Leftrightarrow$ at least one of outputs signals acceptance

Time complexity

TM accepts language A if TM accepts exactly those strings in A

time bound T

$t: N \to N$

TM runs in time $t(n)$ if for every input string w of length n every computation of TM on w halts within $t(n)$ steps

Complexity classes:

$$TIME(t(n)) = \{A : \text{some deterministic TM accepts } A \text{ in time } O(t(n))\}$$

$$NTIME(t(n)) = \{A : \text{some nondeterministic TM accepts } A \text{ in time } O(t(n))\}$$

$$P = \bigcup_k TIME(n^k)$$

$$NP = \bigcup_k NTIME(n^k)$$

Theorem

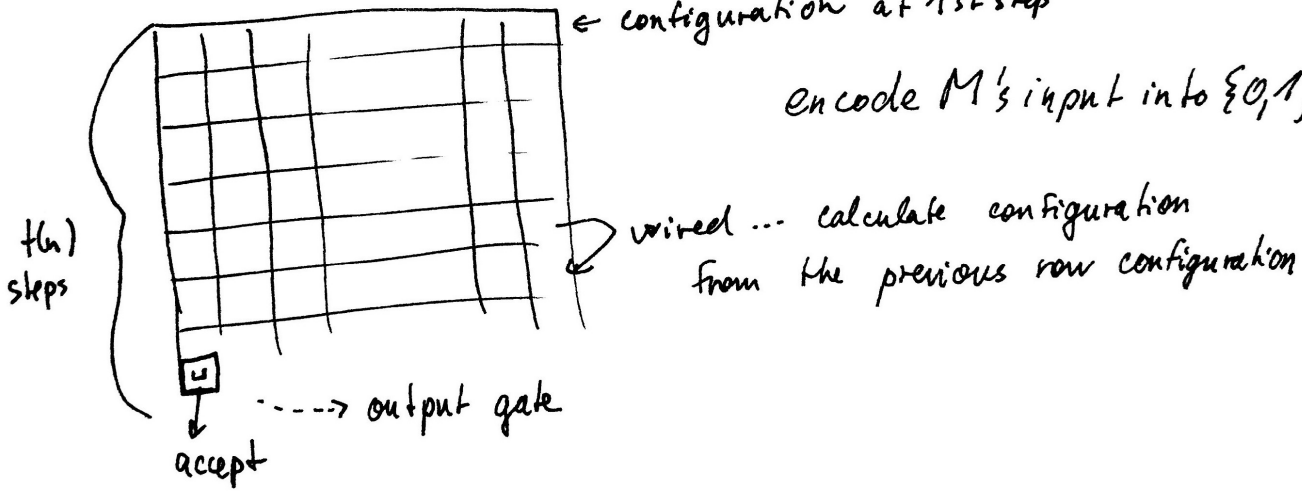
Let $t: \mathbb{N} \to \mathbb{N}$ be a function, where $t(n) \geq n$.

If $A \in \text{TIME}(t(n))$, then A has circuit complexity $O(t^2(n))$.

Proof idea

Let M be TM which decides A in time $t(n)$.

- $\forall n$ we construct a circuit C_n that simulates M on inputs of length n



Proof

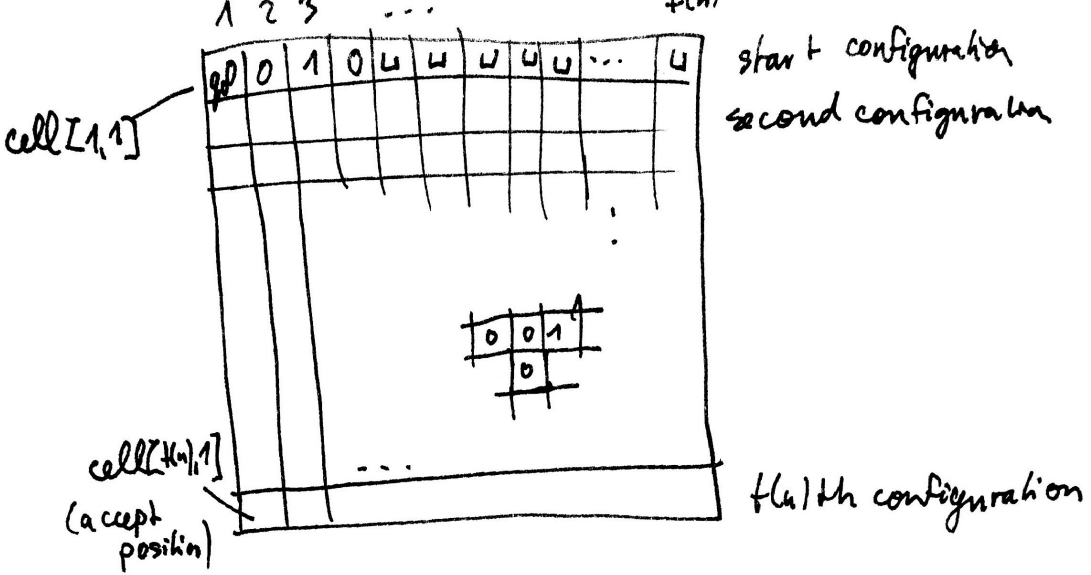
- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ decide A in time $t(n)$ and let w be an input of length n to M.
- Define a tableau for M on w to be $t(n) \times t(n)$ table.

  rows = configurations

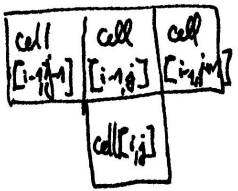
  top row = start configuration

  ith row = configuration at the i-th step of computation
- Represent both the state and the tape symbol under the tape head by a single composite character ... $1\boxed{q\,0}11$
- Each entry of the tableau: tape symbol $\in \Gamma$ or combined $\in Q \times \Gamma$
- The entry at the i-th row and j-th column is $cell[i,j]$
- Assume: • M accepts only when its head is on the leftmost tape cell and that cell contains $\sqcup$
  • Once M has halted it stays in the same configuration

→ looking at the leftmost cell in the final row → whether M has accepted



The content of each cell is determined by certain cells in the preceding row (by transition function)



Construction of C_n

- several gates for each cell in the tableau - compute its value from the three cells that affect it

Lights (illustrative purposes)

k = number of elements in $\Gamma \cup (\Gamma \times Q)$

k lights for each cell → total of $kt^2(n)$ lights

light $[i,j,s]$, $1 \le i, j \le t(n)$ and $s \in \Gamma \cup (\Gamma \times Q)$

The condition of the lights in a cell indicates the contents of that cell.

light $[i,j,s]$ is on → cell $[i,j]$ contains the symbol s

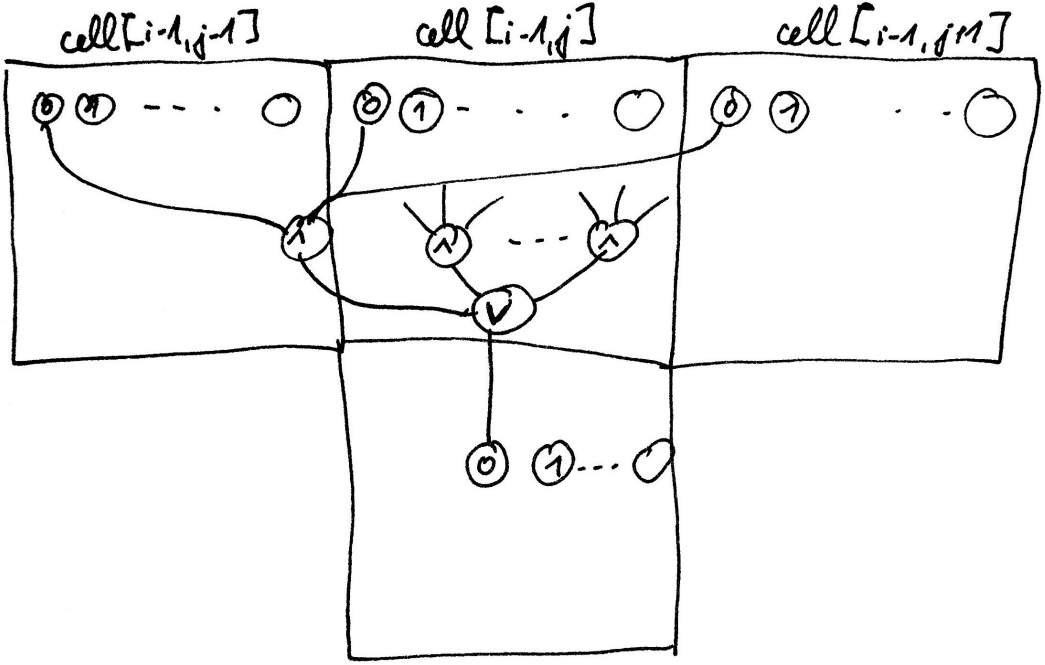
(one light on per cell)

pick One light - light $[i,j,s]$

- which of settings of the three cells cause cell $[i,j]$ to contains (b, σ)
- suppose the cells contain $(a,b,c) \xrightarrow{\tau}$ cell $[i,j]$ contains

  We wire the circuit so, that if the three with (a,b,c) are on, so is light $[i,j,s]$

  • AND gate whose output is connected to light $[i,j,s]$
- several different settings (a_1,b_1,c_1)... → OR



Circuitry for one light

- repeat for each light
- exceptions at the boundaries (2 preceding cells)
- first row
  - cells contain the start configuration
  - their lights wired to the input variables
  - light $[1, 1, \boxed{q_0 1}]$ is connected to input w_1 ...

    $[1, 1, \boxed{q_0 0}]$ is connected through a NOT gate to input w_1

    ⋮

    ↖ input string w determines these values
  - light $[1, n+1, \sqcup], \ldots,$ light $[1, t(n), \sqcup]$ are on
  - the other lights in the first row are off

We have constructed a circuit that simulates M through its $t(n)^{th}$ step

- assign one of the gates to be the output gate - attached to light $[t(n), 1, \boxed{q_{accept} 0}]$

□

Stronger bound

If language A is in TIME $(t(n))$ then A has circuit complexity $O(t(n) \log t(n))$.

Let M accept A in time $t(n)$. Convert M to 1-tape machine. This increases the time to $\boxed{O(t^2(n))}$. Then by the last theorem we have $O(t^4(n))$.

- modify the original machine so that it is oblivious
  = the head motions are only dependent upon the length of the input but not the input itself

  ↑ $\boxed{O(T(n) \log T(n))}$ time simulation using two tapes

- Once the motions of the heads are known the construction may be repeated - only necessary to build circuitry for the cells which contain the head (the others do not change symbol)

- every language in P has polynomial circuit complexity
- however there are nonrecursive languages with low circuit complexity

Definitions

Let $f: \mathbb{N} \to \mathbb{N}$

$f(n)$-advice sequence is a sequence of binary strings $A = (a_1, a_2, \ldots)$
where $|a_n| \leq f(n)$

For language $B \subseteq \{0,1,\#\}^*$ let $\boxed{B @ A = \{x \mid x\#a_{|x|} \in B\}}$

Let $\boxed{P/f(n) = \{B @ A : B \in P \text{ and } A \text{ is an } f(n)\text{-advice sequence}\}}$

Let $\boxed{P/poly = \bigcup_k P/n^k}$ ... nonuniform P

Theorem

C is in P/poly $\Leftrightarrow$ C has polynomial circuit complexity

$\Rightarrow$ If C is in P/poly then $C = B \circledcirc A$ for some B in P and polynomial advice sequence A. Then B has polynomial circuit complexity. Presetting the advice strings into the appropriate inputs of the circuits for B obtains the polynomial size circuits for C.

$\Leftarrow$ We encode the polynomial size circuits for C as an advice sequence.

□

Corollary

Assume that for some $\mathcal{Z} \in NP$ the function $C(\mathcal{Z}_n)$ is not bounded by any polynomial in n.

Then $P \neq NP$

Fundamental problem

Is there a language $\mathcal{Z} \in NP$ with superpolynomial circuitsize complexity?

Theorem (Karp and Lipton 1982)

Assume that every NP language can be computed by a family of polynomial size circuits ($NP \subseteq P/poly$).

Then the polynomial time hierarchy PH collapses to its second level

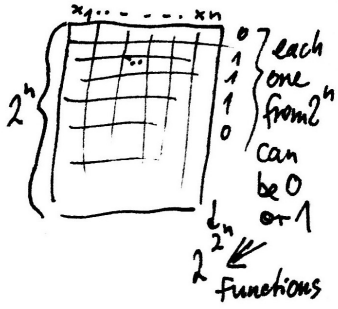
$$PH = \Sigma_2^P = \Pi_2^P$$

Theorem (Shannon 1949, Muller 1956)

For every n there are Boolean functions with n inputs and one output having the circuit complexity $\Omega(2^{n-\log n})$.

Proof

- 2^{2^n} Boolean functions with n unknowns and one output



- at most $(n+m)^{O(m)}$ circuits of size $\leq m$

  [for every gate we can choose from which of the other gates or inputs ($n+m$) wires will come]

  ↳ which is less than 2^{2^n} for $m \leq (2^n/c \cdot n)$ and c sufficiently large constant

□

Let $(n+m)^{O(m)} \leq (nm)^{km}$ for some constant k be the number of circuits

For $m = \frac{2^n}{cn}$ we have

$$\log\left(nm^{km}\right) = km \log(nm) = k\frac{2^n}{cn}\log\left(n\frac{2^n}{cn}\right)$$

$$= k\frac{2^n}{cn}(n - \log c)$$

$$= \frac{k}{c}2^n - \frac{\log c}{c}k < 2^n$$ for sufficiently large constant c

Thank you
for your
attention

Sources

R. Boppana and M. Sipser, The complexity of finite functions

M. Sipser, Introduction to the Theory of Computation

J. Krajíček, Bounded Arithmetic, Propositional Logic and Complexity Theory, CUP, 1995

A Boolean connective

is a Boolean function with one output

A basis

is a finite set of connectives

A Boolean circuit with input variables $x_1, \dots, x_n$
output variables $y_1, \dots, y_m$
basis of connectives $\Omega = \{g_1, \dots, g_s\}$

is a labeled acyclic directed graph
whose out-degree 0 nodes are labeled by y_j's
in-degree 0 nodes are labeled by x_i's or by constants from Ω
in-degree $l \geq 1$ nodes are labeled by functions from Ω of arity l