17. Families of Boolean Circuits

Goal: · Introduce circuits as a model of computation.
· Computers are built from digital circuits, so Turing machines should be built from Boolean circuits.

Motivation: (1) Study parallel computation from a complexity point of view (Section 18).
(2) Researchers believe circuits are the right model to attack the P vs. NP question (Section 19).
(Remember: We understood that we have to analyze the computations.)

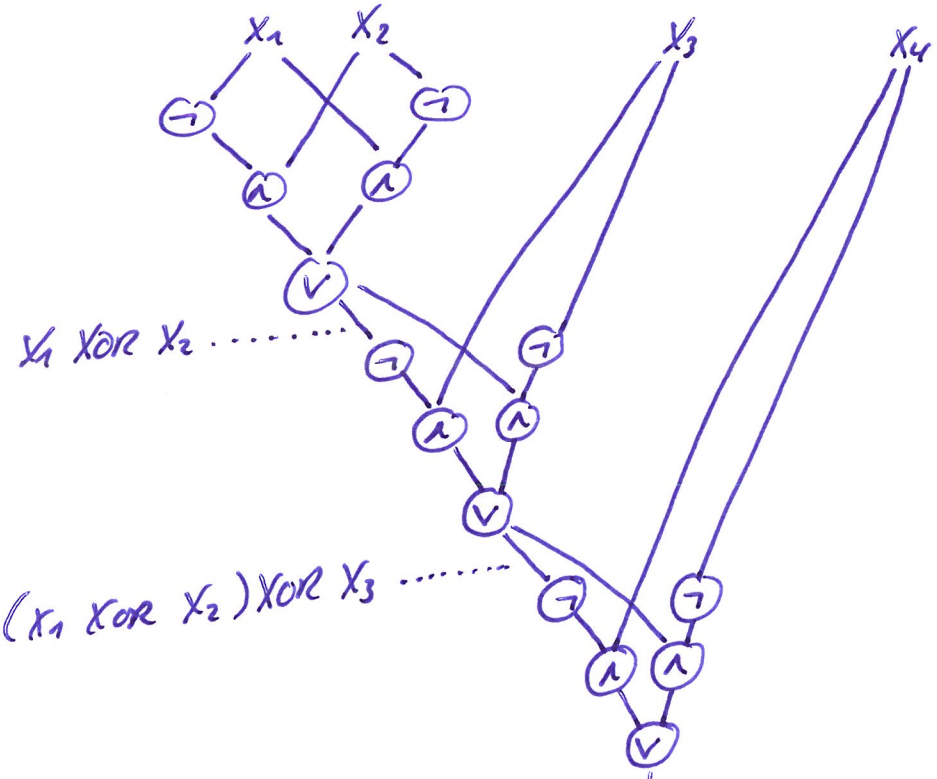
Definition:
- Boolean circuits are defined like in Section 11.
- The variables set to 0/1 we called the inputs.
- The last variable is called the output.
- We use functions to describe the input/output behavior of Boolean circuits:
  ↳ To a circuit C with n input variables we associate the function
  $$f_C : \{0,1\}^n \longrightarrow \{0,1\}$$
  where if C outputs $b \in \{0,1\}$ on input $a_1, \dots, a_n$, we set $f_C(a_1, \dots, a_n) := b$.
  ↳ We say C computes function f_C

(One may also consider circuits with multiple output gates computing functions $f : \{0,1\}^k \longrightarrow \{0,1\}^\ell$.)

Example:

The n-input parity-function $parity_n : \{0,1\}^n \to \{0,1\}$ outputs 1, if an odd number of 1s appears in the input variables.

A circuit computing $parity_4$ is the following:



$out = ((x_1 \text{ XOR } x_2) \text{ XOR } x_3) \text{ XOR } x_4.$

- Our goal is to use circuits to test membership in languages (that have been encoded into $\{0,1\}$).
- Circuits can only handle strings of fixed length.
- Instead of using a single circuit to test language membership use a family of circuits, one for each input length.

Definition:

- A circuit family is an infinite sequence of circuits $(C_i)_{i \in \mathbb{N}}$ where C_n has n input variables.
- The family decides a language L over $\{0,1\}$ if for every string $w \in \{0,1\}^*$: $w \in L$ iff $C_{|w|}(w) = 1$.

- The size of a circuit is the number of gates it contains.
  The size complexity of a circuit family $(C_i)_{i \in \mathbb{N}}$
  is the function
  $$f: \mathbb{N} \longrightarrow \mathbb{N}$$
  $$n \longmapsto \text{size of } C_n.$$
- The depth of a circuit is the maximal length (number of wires)
  from an input variable to the output gate.
  The depth complexity of the family is defined like for the size.

Comment:
- The size complexity is sometimes also called processor complexity
  as each gate is considered to be an individual processor.
- For the same reason, the depth complexity is also called
  parallel time complexity.

Definition:
- A circuit family $(C_i)_{i \in \mathbb{N}}$ is logspace uniform
  if there is a logspace algorithm that,
  on input 1^n computes C_n.

The uniformity requirement is reasonable:
↳ Knowing that a small circuit exists for certain elements in a language
  does not help if the circuit is hard to find.
↳ Indeed, without uniformity, there are circuit families
  that compute undecidable problems (see Section 19).

Example:
- Let $PARITY := \{ w \in \{0,1\}^* \mid \#1 \text{ in } w \text{ is odd} \}$.
  Let the inputs to the circuit be $x_1, \dots, x_n$.

-3-

Solution 1: Construct gates

$g_1 := x_1 \qquad g_{i+1} := x_{i+1} \text{ XOR } g_i,$

where $a \text{ XOR } b := (a \wedge \neg b) \vee (\neg a \wedge b)$.

This solution has $O(n)$ size and depth complexity.

Solution 2: • Construct a binary tree of XOR-gates.

• Has $O(n)$ size and $O(\log n)$ depth complexity.

• The improvement is significant: Exponentially less parallel time.

17.1 Boolean Matrix Multiplication

• We may use circuits to compute functions that output strings.

• Consider the Boolean matrix multiplication function:

Input: $2n^2$ variables
representing two $n \times n$ matrices
$A = \{a_{ik}\} \quad B = \{b_{ik}\}$.

Output: n^2 values representing the $n \times n$ matrix
$C = \{c_{ik}\}$ with $c_{ik} = \bigvee_{j=1}^{n} (a_{ij} \wedge b_{jk})$.

Circuit construction:

↳ Compute $a_{ij} \wedge b_{jk}$ in parallel for all choices of i,j,k.
Needs n^3 gates.

↳ Compute for each i,k: $\bigvee_{j=1}^{n} (a_{ij} \wedge b_{jk})$.
Use a binary tree with • $n-1$ $\vee$-gates
• $\log n$ depth.

Complexity: Size: $O(n^3) = O((\sqrt{n})^3) = O(n^{3/2})$.

Depth: $O(\log n)$.

Logspace uniformity:

The family is logspace uniform.

In particular, $\bigvee_{j=1}^{n}(a_{ij} \wedge b_{jk})$ only differ in the indices.

The indices can be stored in binary.

17.2 Reflexive Transitive Closure

Def: Describe a logspace uniform family of polylogarithmic depth and polynomial size circuits to compute the reflexive transitive closure R^* of a given binary relation R on n-elements.

Recall that R^* consists of all pairs (u,v) so that there is an R-path from u to v of length ≥ 0.

Suppose R is given by an $n \times n$ adjacency matrix. (This is our input.)

We note that

$$R^* = \bigvee_{i \in \mathbb{N}} R^i = \bigvee_{i=0}^{n-1} R^i \quad (\text{w.l.} \; R^0 = I).$$

$\bigvee$: bitwise $\vee$ of the matrices.

Note that $(R^i)_{uv} = 1$ iff there is an R-path of length exactly i from u to v.

Hence we can limit ourselves to powers of R up to $n-1$.

If there is an R-path from u to v of length $\geq n$, there is a shorter one by cutting out loops.

Circuit construction:

- We can represent the computation of R^i with a binary tree of size i and depth $\log i$.
- Each node in the tree computes the product of the two matrices below it.
- With the previous result, each node can be computed by a circuit of $O(n^{3/2})$ size and $O(\log n)$ depth.

  Hence, the circuit for R^{n-1} has size

$$(n-1)\cdot O(n^{3/2}) = O(n^{\frac{7}{2}})\cdot O(n^{3/2})$$
$$= O(n^{\frac{7}{2}+\frac{3}{2}}) = O(n^2).$$

  The depth is $O(\log n)\cdot O(\log n) = O((\log n)^2)$.

- We make circuits for each R^i, which adds another factor of $O(\sqrt{n})$ to the size and another layer of $O(\log n)$ to the depth.

Altogether,

the size complexity is $O(n^{5/2})$,

the depth complexity is $O((\log n)^2)$.