CIRCUIT COMPLEXITY

Circuit complexity is discussed in section 9.3 of the textbook.

Above we have discussed complexity measures that limit the resources used by an algorithm (a Turing machine). The algorithmic approach has allowed us to obtain significant results and characterizations. However, it has the following drawbacks or limitations:

- The algorithmic method "makes sense" only when dealing with infinite sets. Any finite set can be decided by finite table-lookup, and would be considered trivial in this framework.

- (Related to above:) The algorithmic approach measures only asymptotic complexity, that is, what happens when inputs become arbitrarily large. Due to finite speed-up and tape compression results, it seems necessary to omit constant factors when measuring time or space.

- There are reasons to believe that the standard techniques[1] available in the algorithmic method are not sufficient to solve certain important questions like P $=^?$ NP. (This is discussed further in section 9.2 in the textbook.)

We can deal better with complexity issues of finite sets if, instead of measuring resources used by an algorithm, we measure the *size of the algorithm* deciding a finite set.

With an infinite set B we can then associate a function describing the rate of growth of the sizes of algorithms needed to decide membership in B for inputs of length n.

An above type of complexity measure is called *descriptional complexity* or *non-uniform complexity.* We consider here one particular non-uniform complexity measure: **Boolean circuit complexity**.

[1] That is, simulation to prove inclusion of complexity classes and diagonalization to prove strict inclusion.

For the definitions of the notions: Boolean circuit, circuit family, size complexity and depth complexity of a circuit family, see section 9.3.

The standard examples of circuits usually compute some function on Boolean values (parity etc.). Below we illustrate how circuits can be used for more general computational tasks.

Example. We recall the definition of the graph reachability problem encoded as a language:

$$\text{PATH} = \{< G, s, t > \quad | \quad G \text{ is a directed graph that has a directed path from node } s \text{ to node } t\}$$

We show that PATH has polynomial circuit size complexity.

We assume that the input graph has nodes $1, \ldots, n$ and we want to determine whether there is a path from 1 to n (this does not lose generality since, if necessary, we can rename the nodes).

The graph G is represented as an $n \times n$ binary matrix giving the adjacency matrix of G.

- For each $m = n^2$ we construct a Boolean circuit C_m that accepts (outputs the value 1) exactly those strings in $\{0,1\}^m$ that represent an adjacency matrix of a graph for which there is a path from node 1 to node n.
- For values m that are not a square, we choose C_m to be a circuit that always outputs zero.

In the following, we consider only the case $m = n^2$. The gates (and input elements) of C_m are:

$g_{i,j,k}$, $1 \leq i, j \leq n$, $0 \leq k \leq n$,

$h_{i,j,k}$, $1 \leq i, j, k \leq n$.

The intuitive meaning of the gates is as follows:

1. $g_{i,j,k}$ gets the value true if and only if

   (1) there is a path from i to j not using any intermediate node larger than k.

2. $h_{i,j,k}$ gets the value true if and only if

   (2) there is a path from i to j using k as an intermediate node, and using no node larger than k.

The input elements, constants and variables are $g_{i,j,0}$, $1, \leq i, j \leq n$:

- $g_{i,i,0}$, $1 \leq i \leq n$, is a constant 1,
- $g_{i,j,0}$, $i \neq j$, is a variable that gets the value of the (i, j) entry of the adjacency matrix.

The above is in accordance with property (1).

- The function symbol of $h_{i,j,k}$ is $\wedge$ (AND) and its inputs are $g_{i,k,k-1}$ and $g_{k,j,k-1}$.
- The function symbol of $g_{i,j,k}$, $1 \leq k \leq n$, is $\vee$ (OR) and its inputs are $g_{i,j,k-1}$ and $h_{i,j,k}$.

By induction on k (the third subindex of the gate names) we see that the above definitions guarantee that all gates have values as specified in (1) and (2). (To be explained in class.)

The output gate is $g_{1,n,n}$ and it has value 1 if and only if the graph has a path from node 1 to node n. Also, we observe that the size of C_m (with $m = n^2$) is $2n^3 + n^2$.

Note: C_m, $m \geq 1$, is a so called *uniform circuit family:* the circuit C_m can be constructed using logarithmic space when given 1^m as input.

- It can be shown that any language decided by a polynomial size uniform circuit family is in P. On the other hand, general polynomial size circuit families can define languages that are even undecidable.

- Conversely, any language in P can be decided by a uniform polynomial size circuit family. This follows from Theorem 9.30 by additionally observing that the circuit family used for the proof of the theorem can be constructed in logarithmic space (on input 1^m the m'th circuit C_m is constructed in log space).

Theorem. The class P consists of exactly languages that have a uniform circuit family of polynomial size.

If one shows that an NP-complete problem can be decided by a uniform polynomial size circuit family, this implies P $\neq$ NP. In fact, NP-complete problems are not known to have even general polynomial size circuits (not necessarily uniform). Note that it is easy to give an example of an undecidable language with polynomial size circuits (naturally the circuits cannot be uniform).

On the other hand, it is very hard to prove lower bounds for the size of circuits deciding any concrete NP-complete problem. The best known lower bounds are linear $c \cdot n$ with small constants c.