0.1 Introduction

0.1.1 Non-uniform models

The difference between **uniform** and **non-uniform** models of computation is that for **uniform** models, we will have a single program which can handle inputs of any length. Whereas, for non-uniform models, we have a different program for each input length $n \in \mathbb{N}$

0.1.2 CNFs

A CNF formula is a logical formula which fits the form:

$$\bigwedge_i \bigvee_j l_{ij}$$

Where each l_{ij} is a literal, i.e. either x or $\neg x$ for some variable x.
We define computation by CNF as follows:

Definition 1 *(Computation by CNF) Given a Boolean function, $f : \{0,1\}^n \to \{0,1\}$, we say that f is computed by a CNF, $A(x_1, \ldots, x_n)$ if, for any $\vec{b} \in \{0,1\}^n$, we have:*

$$A(b_1, \ldots, b_n) = f(b_1, \ldots, b_n)$$

I.e. for any assignment of A, f produces the same result.

Theorem 1 *We can show that for **every** Boolean function, $f : 0,1^n \to 0,1$ there is a CNF A_f which computes it.*

We show this as follows:

Proof 1 *Let $F := \{\vec{b} \in \{0,1\}^n : f(\vec{b}) = 0\}$ For $b \in \{0,1\}$ write:*

$$b \star x := \begin{cases} x & b = 0 \\ \neg x & b = 1 \end{cases}$$

and define:

$$A_f := \bigwedge_{\vec{b} \in F} \bigvee_{i=1}^{n} b_i \star x_i$$

Recall that we link Boolean functions, $f : 0,1^n \to 0,1$ with languages $\subseteq \{0,1\}^n$.

We say that a language $L \subseteq \{0,1\}^*$ is computed by a **family** $\{A_n\}_{n=1}^{\infty}$ of CNFs if, for each $n \in \mathbb{N}$, A_n is a CNF computing $L \cap \{0,1\}^n$.

With this we can define our first *non-uniform* class:

Definition 2 *The size of a CNF A, $|A|$ is the number of literals occurrences it contains.*

$\mathbf{CNF}^{poly}$ *is the class of languages computed by polynomially sized families of CNFs*

Undecidability

Proposition 1 $\mathbf{CNF}^{poly}$ *contains undecidable languages*

Proof 2 *Let $H \subseteq \mathbb{N}$ be the set of natural numbers coding for halting Turing Machines (excluding inputs). Define the following sequence of CNFs:*

$$A_n := \begin{cases} x_1 \vee \neg x_1 & n \in H \\ x_1 \wedge \neg x_1 & n \notin H \end{cases}$$

A_n evaluates to 1 on an input $\in \{0,1\}^n$ iff the n^{th} TM halts.

The induced language $\subseteq \{0,1\}^*$ is therefore clearly undecidable by reduction to H.

This is the case as if $H \subseteq \mathbb{N}$, $H \subseteq *$. This shows that on the set of all Languages, and therefore all Boolean functions in $\mathbf{CNF}^{\text{poly}}$, there exist undecidable languages.

0.1.3 Boolean Circuits: Basics

A Boolean circuit is similar to a formula, but we can reuse *sub-formulas*. They are represented as a *well-founded* diagram built from Boolean gates.

Boolean formulae which we have seen earlier can be though of as a special class of circuits whose underlying graphs are trees.

Definition 3 *(Boolean Circuit) An n-input Boolean circuit C is a* ***directed acylic graph*** *with:*

- *n sources, $x_1, \ldots, x_n$*

  These are visualised as vertices with no incoming edges (leaves in a tree)

- *one sink s*

  A vertex with no outgoing edges (root in a tree)

- *A labelling from nonsource vertices to the set of gates, by default, $\{\neg, \wedge, \vee\}$*

- *Vertices labelled $\neg$ are said to have* ***fan-in*** *1, i.e. 1 incoming edges*

- *Vertices labelled $\vee$ or $\wedge$ have* ***fan-in*** *2*

The size of C is written $|C|$, and is the number of vertices.

Computation of Boolean Circuits Given a dag C and an assignment $b : \{x_1, \ldots, x_n\} \to \{0, 1\}$ we can define a value $b(v)$ at a node v by induction over the structure of C:

- $b(x_i)$, is already defined as these are the input nodes
- any nonsource node labelled $\neg$ with an incoming edge from node u can be said to have a value $b(v) := 1 - b(u)$
- any nonsource node labelled with $\vee$, with incoming edges from u_0 and u_1 can be said to have a value $b(v) := \max(b(u_0), b(u_1)$
- Nonsource nodes labelled $\wedge$ with inputs from nodes u_0, u_1 has a value $b(v) := \min(b(u_0), b(u_1))$

With the above definitions, we can say that the value of $b(C)$ is the same as $b(s)$.

We can show that this definition of outputs allows us to construct a polytime algorithm for evaluating circuits.

Proposition 2 *(We can evaluate a boolean circuit C in polynomial time) The set of pairs (C, b) where*

- *C is an n input circuit and*
- *$b : 0, 1^n \to 0, 1$ s.t. $b(C) = 1$*

is in **P**

Depth of a Circuit The depth of a circuit is C is the length of the longest path, i.e. the maximum number of edges from a source node to s, in its underlying graph.

The depth of a node is the longest path from a source node to that node.

The inductive definition of circuit depth is the same as for the structural induction seen earlier.

Proposition 3 *If $L \subseteq \{0, 1\}^n$ is computed by a circuit C of depth d, then it is also computed by a formula of size $< 2^d$.*

Proof 3 *For a node v of C, define the formula $F(v)$ by induction on the depth of v.*

- *$F(x_i)$ is just the formula x_i, this has size 1 ($1 < 2 = 2^1$)*
- *If v of depth e is labelled with $\neg$ with an incoming edge from node u (of depth $< e$)m then $F(v) := \neg F(u)$ (of size $< 2^{e-1} + 1, \therefore < 2^e$)*

- *If v of depth e is labelled $\star \in \{\wedge, \vee\}$ with incoming edges from u_0, u_1 (both of depth $< e$), then $F(v) := (F(u_0) \star F(u_1))$ which has size $< 2^{e-1} + 2^{e-1} + 1$ which is $< 2^e$*

We have now constructed $F(s)$ which is a formula computing L in size $< 2^d$

We say that a language $L \subseteq \{0,1\}^*$ is computed by a circuit family $\{C_n\}_{n=1}^{\infty}$ if, for each $n \in \mathbb{N}$, C_n is an n-input circuit computing $L \cap \{0,1\}^n$

For $f : \mathbb{N} \to \mathbb{N}$ we define:

- $\mathbf{SIZE}(f(n))$ as the set of languages $L \subseteq \{0,1\}^*$ computed by circuits of size $\leq f(n)$
- $\mathbf{DEPTH}(f(n))$ as the set of languages $L \subseteq \{0,1\}^*$ computed by circuits of depth $\leq f(n)$

The class $\mathbf{P}\backslash poly$ is the class of languages computed by polynomial-size circuit families, i.e.:

$$\mathbf{P}\backslash poly := \bigcup_{c=1}^{\infty} \mathbf{SIZE}(n^c)$$