CS748: Arithmetic Circuit Complexity.

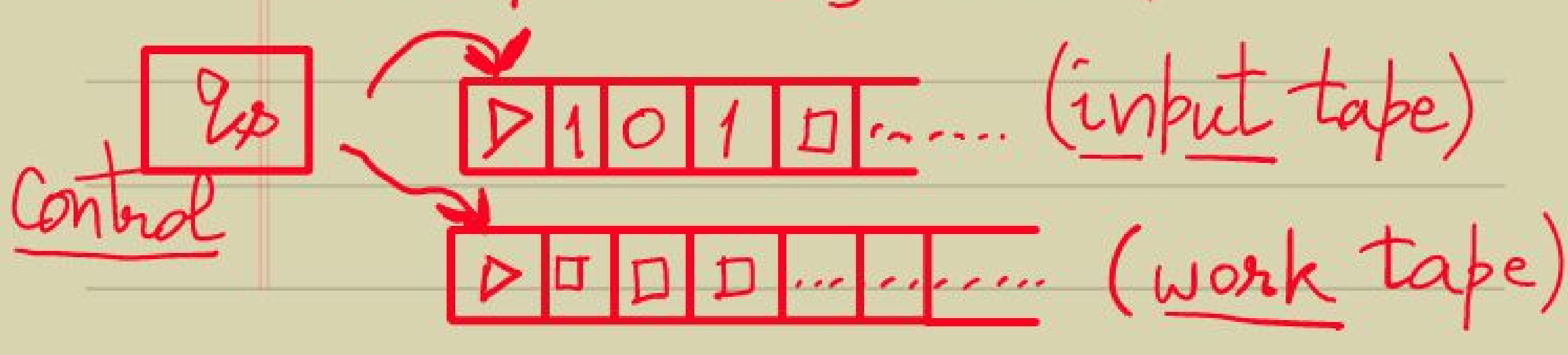
(Refer: Course webpage)

- Classically, computation is modelled using Turing machines.

- I.e. A computer program is seen as a machine $M = (\Gamma, Q, \delta)$ where,
  - Γ is the alphabet, say $\triangleright$ (start), $\square$ (blank), 0 & 1.
  - Q is the set of states (at least q_s & q_f).
  - δ is the transition function $\delta: Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{S, L, R\}^2$.

    head movement

- Example configuration:



- <u>Time</u> is the number of transition steps.

- <u>Space</u> is the number of work tape cells used.

- For input size n & a function $f : \mathbb{N} \to \mathbb{R}_{>0}$ we can talk about complexity classes $\mathrm{Dtime}(f(n))$ & $\mathrm{Space}(f(n))$ as the set of problems that are computable in time $O(f(n))$ & space $O(f(n))$ respectively.

- This leads us to a <u>zoo</u> of classes!

$$P := \bigcup_{c \in \mathbb{N}} \mathrm{Dtime}(n^c)$$

$$\mathrm{Pspace} := \bigcup_{c \in \mathbb{N}} \mathrm{Space}(n^c)$$

$$NP := \bigcup_{c \in \mathbb{N}} Ntime(n^c)$$

$$\mathbb{L} := Space(\lg n)$$

▷ $\mathbb{L} \subseteq P \subseteq NP \subseteq Pspace \subseteq EXP \subseteq EXPspace \subseteq EEXP.$

- There are also randomized versions:

$$ZPP \subseteq RP \subseteq BPP \subseteq PP \subseteq Pspace$$

- and oracle-based classes:

$$\Sigma_1 \subseteq \Sigma_2 \subseteq \Sigma_3 \subseteq \ldots\ldots \subseteq PH \subseteq Pspace.$$

$$\Sigma_1 = NP, \quad \Sigma_2 = NP^{\Sigma_1}, \quad \Sigma_3 = NP^{\Sigma_2}, \quad PH = \bigcup_{c \in \mathbb{N}} \Sigma_c$$

- This course will take a different route to build a zoo of computational classes!

- Instead of seeing computation as a sequence of very simple steps, we will view it as an algebraic expression.

- Definition: An arithmetic circuit C, over a field $\mathbb{F}$, is a rooted dag as follows. The leaves are the variables $x_1, \ldots, x_n$ (input) & the root outputs a polynomial $C(\bar{x})$.

  The internal vertices are gates that compute $*$ or $+$ in $\mathbb{F}[\bar{x}]$.

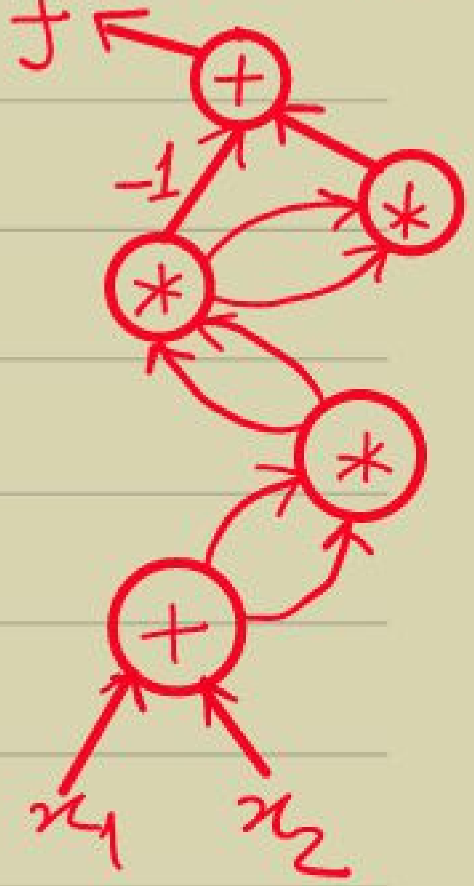
  The edges are called wires & they can have constant (in $\mathbb{F}$) labels to do scalar multiplication.

  The #wires (& the size of the constants?) comprise the size of the circuit C.

  A max-path from a leaf to the root determines the depth of C.

  $\deg(C)$ refers to the degree of the intermediate polynomials computed.

- Eg. The polynomial $f = (x_1+x_2)^8 - (x_1+x_2)^4$ has the following circuit representation:



- Note that the circuit for f is quite compact (though f has 14 monomials!)

- Repeated-squaring is used.
- Eg. $(X+1)^{2^n}$ has size $O(n)$.

- Definition: fanin (resp. fanout) of a circuit refers to the max indegree (resp. outdegree) of the gates/vertices.

  A circuit with fanout $=1$ is called a formula.

- Suppose $\mathcal{F} := \{f_i(x_1,\ldots,x_i) \mid i \in \mathbb{N}\}$ is a family of polynomials (call it problem).

  We will say that a family of

circuits $\mathcal{C} = \{C_i(x_1, \ldots, x_i) \mid i \in \mathbb{N}\}$ solves $\mathcal{F}$ if $\forall i$, $f_i = C_i$.

In this case, we can say that $\mathcal{F}$ can be solved in size bounded by size(C_n) & depth bounded by depth(C_n).

- Eg. depth corr. to time & size corr. to space.
- This gives us a new way to measure the complexity of polynomials (or problems) — arithmetic circuit complexity.

- Arithmetic complexity classes were first defined by Valiant (1979).

In particular, the arithmetic analogs of P & NP.

- Defn: $VP_{\mathbb{F}}$ consists of families of polynomials, say $\{f_n\}_n$, over $\mathbb{F}$, that can be solved by circuits of poly(n) size & poly(n) degree. (Why?)

- Eg. The family $\{x^{2^n}\}_n$ is <u>not</u> in $VP_{\mathbb{F}}$, for any $\mathbb{F}$.

  Though it is computable by poly(n)-size circuits, its degree is too high!

- An interesting polynomial (family) in VP is the determinant:

$$\det_n(\bar{x}) := \sum_{\pi \in Sym(n)} sgn(\pi) \cdot \prod_{i=1}^{n} x_{i,\pi(i)} .$$

Clearly $\det_n \in P$

- We will see later that $\det_n \in VP$.

  (We'll abuse the notation a bit: by the polynomial $\det_n$ we actually mean the family $\{\det_n\}_n$.)

- VP is the <u>algebraic</u> analog of P.

  (The degree restriction is put to avoid computing very large numbers like 2^{2^n}, when $\mathbb{F} = \mathbb{Q}$.)

- What is the analog of NP?

- <u>Defn</u>: A polynomial family $\{f_n\}_n$ is in $\text{VNP}_{\mathbb{F}}$ if : $f_n(\bar{x}) = \sum_{\bar{w} \in \{0,1\}^{t(n)}} g_{n+t(n)}(\bar{x}, \bar{w})$, ← verifier ← witness

where $\{g_n\}_n \in \text{VP}_{\mathbb{F}}$ & $t(n) = \text{poly}(n)$.

- One can think of $\bar{w}$ as a "witness" & so summing over all of them gives the arithmetic analog of an NP problem.

- A standard problem in VNP is:

Permanent $\quad per_n(\bar{x}) := \sum_{\pi \in Sym(n)} \prod_{i \in [n]} x_{i,\pi(i)}$.

▷ $per_n(\bar{x}) \in \text{VNP}$.

Pf: • Let g be the function that takes an $n \times n$ matrix (x_{ij}), a vector $\bar{b} \in \{0,1\}^n$ & computes

$$g_{n^2+n}(\bar{x}, \bar{b}) := \prod_{i \in [n]} \left((-1+2b_i) \cdot \sum_{j \in [n]} b_j x_{ij} \right).$$

• Ryser's formula states:

$$\mathrm{per}_n(\bar{x}) = \sum_{\bar{b}\in\{0,1\}^n} g_{n^2+n}(\bar{x},\bar{b}).$$

[Pf sketch: Rewrite RHS as

$$\sum_{T\subseteq[n]} (-1)^{n-|T|} \cdot \Big(\prod_{i\in[n]} \sum_{j\in T} x_{i,j}\Big).$$ ← product of row-sums

Note that the monomials involved are formed by picking a variable from each of the rows, eg. $x_{1,i_1}\cdot x_{2,i_2}\cdots x_{n,i_n}$.

Let $r := \#\{i_1, i_2, \ldots, i_n\}$. Then, the monomial can be associated to 2^{n-r} many other columns T: the sign contribution being

$$\sum_{S\subseteq[n-r]} (-1)^{|S|} = \sum_{0\le \ell\le n-r} (-1)^{\ell}\cdot\binom{n-r}{\ell} \cdot (1-1)^{n-r} = 0.$$

⇒ The surviving monomials have distinct $i_1, \ldots, i_n$ (thus $r=n$). □] □

– It is clear that:

▷ $VP \subseteq VNP$.

<u>Conjecture</u> 1: $VP \neq VNP$. (Valiant's hypothesis)

- Note that per_n of a boolean matrix, that represents a bipartite graph, is simply the number of perfect matchings.
  This is equivalent to the functional problem of <u>#SAT</u>.
  (Valiant's <u>#P-completeness</u> of per_n.)

- Thus, conjecture 1 is like the <u>#SAT ∉ FP</u> question.
  Also, called the <u>arithmetic analog</u> of the $P \stackrel{?}{=} NP$ question!

- <u>Exercise</u>: $VP = VNP \stackrel{(!)}{\Rightarrow} P/poly = NP/poly$ $= NC^3/poly$ (Bürgisser 2000)
  (Note- Constants may be large in char = 0.)

- Let us now place det_n in VP using <u>Newton identities</u>.