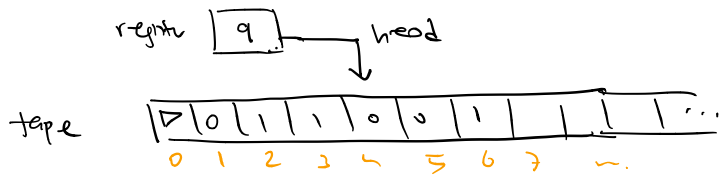
Turing machine (TM)



Head can read & write symbols.

Register can hold a single state.

Mathematical definition

A TM is a triple $M = (\Gamma, Q, \delta)$

a) Γ (alphabet) is a finite set of symbols

$$\Gamma = \{ \triangleright, \underbrace{\sqcup}_{\text{blank}} \} \sqcup \underbrace{\Sigma}_{\mathbb{B} = \{0,1\}}$$

b) Q (states) is finite set

$$\underbrace{q_s}_{\text{start state}}, \underbrace{q_h}_{\text{halt state}} \in Q$$

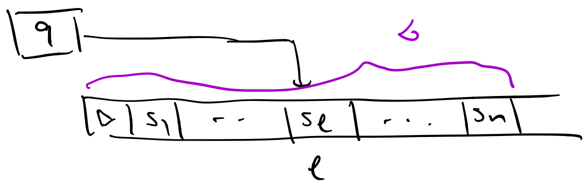
Sometimes there are two distinct halting states $\underbrace{q_1}_{\text{accepting}}$ & $\underbrace{q_0}_{\text{rejecting}}$.

c) Transition function

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{-1, 0, 1\}$$

How the machine works

We will write $\langle q, \sigma, \ell \rangle$ for the configuration of TM at a given step.



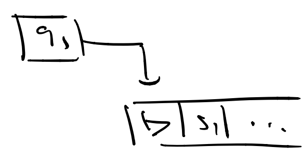
$q \in Q$, $\ell \in \mathbb{N}$, $\sigma \in \Gamma^*$ where

Γ^* is the set of strings in Γ i.e.

$$\Gamma^* = \bigsqcup_{k \in \mathbb{N}} \Gamma^k$$

1) Initial conf. $\langle q_s, \sigma, 0 \rangle$

$$\boxed{\sigma = \triangleright \underbrace{s_1 s_2 \cdots s_n}_{\text{input}}}$$



2) If $\langle q, \sigma, \ell \rangle$ is conf. at step t then

$\langle q', \sigma', \ell' \rangle$ at step $t+1$ is given by

$$\sigma' = \triangleright s_1 \cdots s_{\ell-1} s'_\ell s_{\ell+1} \cdots s_n$$

$$\ell' = \ell + \Delta \qquad \Delta \in \{\pm 1, 0\}$$

where q', s_ℓ', Δ are determined by

$$\boxed{\delta(q, s_\ell) = (q', s_\ell', \Delta)}.$$

Rem To avoid going out of the tape on the left

$$\delta(q, \triangleright) = (q', \triangleright, \Delta) \quad \Delta \neq -1.$$

3) Final conf. (if halts)

$$\langle q, \tilde{\sigma}, \ell \rangle \qquad q = q_h \quad (q_1 \text{ or } q_0)$$

$$\boxed{\tilde{\sigma} = \triangleright \underbrace{\tilde{s}_1 \ldots \tilde{s}_m}_{\text{output}}}$$

There are 3 possibilities of M

$$M(\sigma) = \begin{cases} 1 \ (\text{accepts}) & \text{if final conf. } q_1 \\ 0 \ (\text{rejects}) & \text{if } \quad " \quad q_0 \\ \text{loops} & \text{o/w} \end{cases}$$

halting conf.

A TM M s.t. $M(\sigma) \in \{0,1\} \ \forall \ \sigma \in \Gamma^*$ is called a decider.

Ex $M = (\Gamma, Q, \delta)$

$$\Gamma = \{ \triangleright, \text{␣} \} \sqcup \mathbb{B}$$

$$Q = \{ q_s, \underbrace{q_1, q_2, q_3}, q_h \}$$

1 $\delta(q_s, \triangleright) = (q_1, \triangleright, +1)$

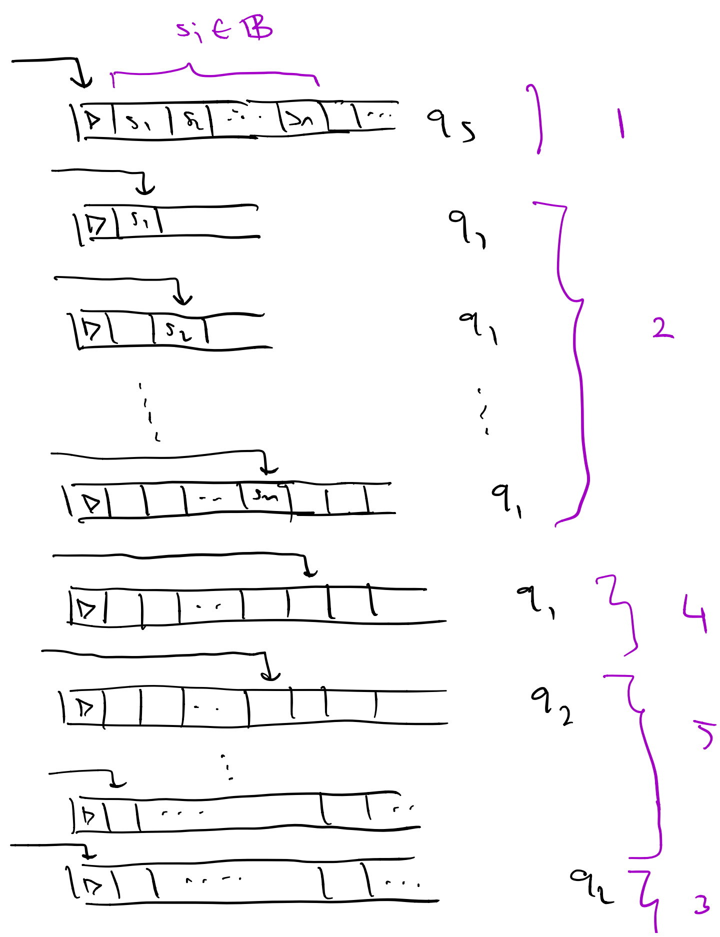
2 $\delta(q_1, s) = (q_1, \text{␣}, +1)$

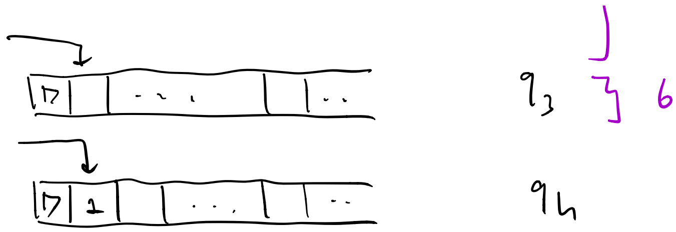
3 $\delta(q_2, \triangleright) = (q_3, \triangleright, +1)$

4 $\delta(q_1, \text{␣}) = (q_2, \text{␣}, -1)$

5 $\delta(q_2, \text{␣}) = (q_2, \text{␣}, -1)$

6 $\delta(q_3, \text{␣}) = (q_h, 1, 0)$



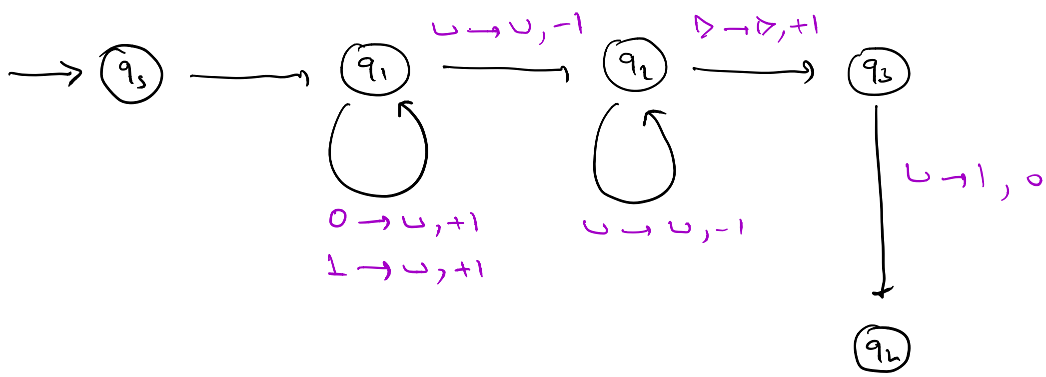


$$\varphi_M : \mathbb{B}^* \longrightarrow \mathbb{B}^* \qquad \varphi_M(\sigma) = 1 \quad \forall\, \sigma \in \mathbb{B}^*$$

$$s_1 \cdots s_n \longmapsto 1$$

constant function at 1.

State diagram of M



Computable functions

Every TM computes a function

$$\varphi_M : \Sigma^* \longrightarrow \Sigma^* \quad \Big\}$$ defined on σ for which M halts.

$$\varphi_M(s_1 \cdots s_n) = \tilde{s}_1 \cdots \tilde{s}_m$$

A function $f : \Sigma^* \to \Sigma^*$ is computable if $\exists$ M such that $f(\sigma) = \varphi_M(\sigma)$ $\forall$ $\sigma \in \Sigma^*$.

Church-Turing thesis

The class of functions computable by a TM corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm.

Decidability

1) A subset $L \subseteq \Sigma^*$ is called a language.

a) A language is Turing-recognizable if there exists a TM M s.t.

$$M(\sigma) = \begin{cases} 1 & \text{if } \sigma \in L \\ 0 \text{ or loops} & \text{if } \sigma \notin L \end{cases}$$

is not required to halt.

b) A language is decidable if $\exists$ TM M s.t.

i) M is a decider.

ii) $M(\sigma) = \begin{cases} 1 & \text{if } \sigma \in L \\ 0 & \text{if } \sigma \notin L \end{cases}$

We can associate a language to M

$$L(M) = \{ \sigma \in \Sigma^* \mid M(\sigma) = 1 \}$$

(L is decidable $\Longleftrightarrow$ $L = L(M)$ for some decider M)

2) A function $P : \Sigma^* \longrightarrow \mathbb{B}$ is called a predicate.

We can identify P with the language

$$L(P) = \{ \sigma \in \Sigma^* \mid P(\sigma) = 1 \}$$

We say P is decidable if $L(P)$ is decidable.

<u>Ex</u> $\Sigma = \{ 0 \}$

$L \subseteq \Sigma^*$ where $\sigma \in L$ if

$$\sigma = \underbrace{0 \cdots 0}_{2^n} \qquad n \geq 0$$

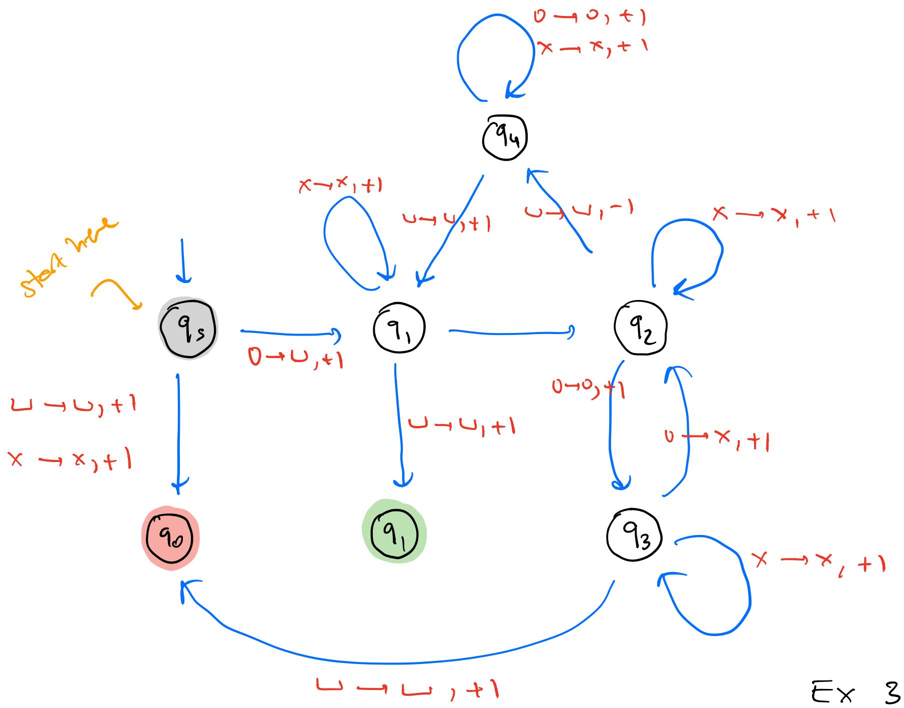
Then L is decidable.

Let us describe the TM:

$$Q = \{ q_s, q_1, q_0, q_1, q_2, q_3, q_4 \}$$

$$\Gamma = \{ \sqcup, x, 0 \} \quad (\text{omitting } \triangleright \text{ simplifies})$$

δ is described by the state diagram



Ex 3.4, Sipser.

Question: Are there L that are not decidable?

Representing a TM as a string

Let us do this using $\mathbb{B} = \{0, 1\}$

Suppose $M = (\Gamma, Q, \delta)$

We can represent

i) elements of Q

will be specified $\to$ $q\ \underbrace{b_i \cdots b_0}_{\in \mathbb{B}^*}$

ii) elements of Γ

will be specified

$s \in \Sigma \qquad a\, \underbrace{b_j \ldots b_0}_{\in \mathbb{B}^*}$

special symbols (including ± 1)

$\sqcup \qquad a\, 0^{j-1}\, 00$

$\underbrace{0 \cdots 0}_{j-1}$

$\triangleright \qquad a\, 0^{j-1}\, 01$

$-1 \qquad a\, 0^{j-1}\, 10$

$+1 \qquad a\, 0^{j-1}\, 11$

where j is large enough so that each element is represented by a disjoint string.

(similarly for i)

iii) representing δ : $\delta(\tilde{q}, \tilde{s}) = (\tilde{q}', \tilde{s}', \ell)$

(*) $\boxed{(}\, q\, b_i \ldots b_0 \,\boxed{,}\, a\, b_j \ldots b_0 \,\boxed{,}\, q\, b_i' \ldots b_0' \,\boxed{)}\, a\, b_j' \ldots b_0' \,\boxed{,}\, a\, 0^{j-1}10$ or $a\, 0^{j-1}11\, \boxed{)}$

So far we represent M using

$\underbrace{0, 1, a, q, (,), ,}$ ← comma

choose binary representation for these.

Let $\langle M \rangle$ denote the string in $\mathbb{B}^*$ obtained by concatenating (*) separated by commas in the lexicographic order

$$\underbrace{(-,-,-,-,-)}_{\text{binary}} \leq \underbrace{(-,-,-,-,-)}_{\text{binary}}, \dots$$

Lexicographic order: $0 \leq 0$, $0 \leq 1$, $1 \leq 1$.

$10 \leq 11$ ($1 \leq 1$, $0 \leq 1$), $10 \geq 01$ ($1 \geq 0$)

Rem Each $\mathbb{B}^*$ represents a natural number.

$$b_n \dots b_0 \longmapsto \sum_{i=0}^{n} b_i 2^i$$

Therefore we can associate a natural number to each M:

$$M \longmapsto \langle M \rangle \longmapsto n \in \mathbb{N}.$$

Universal TM

There exists a TM $\boxed{U}$ such that

$$U(\langle M \rangle \sqcup \sigma) = M(\sigma) \quad \forall \text{ TM } M.$$

Consider the language ($\subseteq \mathbb{B}^*$)

$L_{TM} = \{ \langle M \rangle \sqcup \sigma \mid M$ is a TM s.t. $\boxed{M(\sigma) = 1} \}$

Existence of U implies that L_{TM} is Turing-recognizable:

$\langle M \rangle \sqcup \sigma \in L_{TM} \iff M(\sigma) = 1$ i.e.

$U(\langle M \rangle \sqcup \sigma) = M(\sigma) = 1.$

<u>Halting</u> problem

Is L_{TM} decidable?

<u>Theorem</u>: L_{TM} is undecidable (not decidable).

<u>Proof</u> Suppose that H is a decider for L_{TM}:

$$H(\underbrace{\langle M \rangle \sqcup \sigma}) = \begin{cases} 1 & \text{if } M(\sigma) = 1 \\ 0 & \text{if } M(\sigma) \in \{0, \text{loop}\} \end{cases}$$

Use H to construct another TM

D: on input $\langle M \rangle$ run $H(\langle M \rangle \sqcup \langle M \rangle)$

$M(\langle M \rangle) \in \{0, \text{loop}\}$ - accept if H rejects ($\iff M$ does not accept $\langle M \rangle$)

$M(\langle M \rangle) = 1$ - reject if H accepts ($\iff M$ accepts $\langle M \rangle$)

$$D(\langle D \rangle) = \begin{cases} 1 & D(\langle D \rangle) = 0 \\ 0 & D(\langle D \rangle) = 1 \end{cases}$$

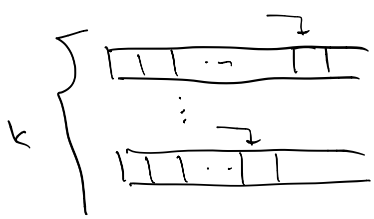
Diagonalization argument:

$\boxed{\text{Contradiction}}$

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	...	$\langle D \rangle$
M_1	0	1	1		
M_2	0	1	0		
M_3	0	0	0		
$\vdots$				$\ddots$	
D	1	0	1	...	$\boxed{?}$

Variants of TM

a) Multitape TM



$$\delta : Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{\pm 1, 0\}^k$$

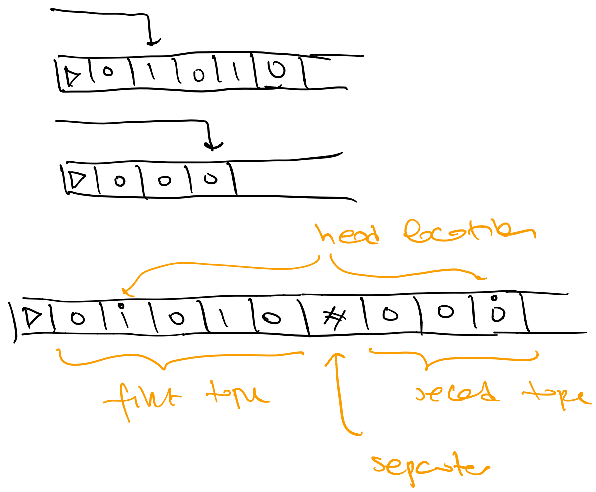
For every multitape TM there is a $\underbrace{\text{equivalent}}$ single tape TM.

they can simulate each other.

idea

M :

S :

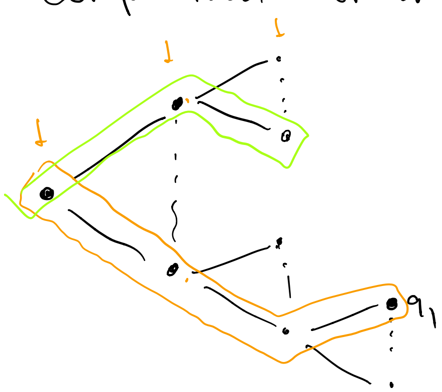


b) Nondeterministic TM N

$$\delta : Q \times \Gamma \longrightarrow \underbrace{P(Q \times \Gamma \times \{\pm 1, 0\})}_{\text{set of subsets}}$$

$$\delta(q,s) = \{(q_1, s_1, \ell_1), \ldots, (q_n, s_n, \ell_n)\}$$

Computation branches out like a tree



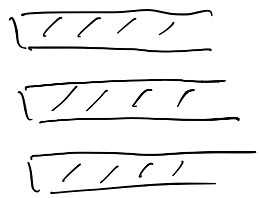
$N(\sigma) = 1$ if a path reaches q_1.

$N(\sigma) = 0$ if every path halts without reaching q_1.

Every non-deterministic TM has an equivalent deterministic TM.

idea

D :



input tape of N

N's tape at some branch

location of computation tree of N

"Do a breath search instead of a depth search."

Summary: All variants have the same computational power.

Circuits

Let $\mathcal{A}$ be a set of Boolean functions

$$\mathcal{A} = \{ f_j : \mathbb{B}^{r_j} \to \mathbb{B} \}.$$

A circuit C over $\mathcal{A}$ consists of

1) input variables $x_1, x_2, \ldots, x_n$

2) auxiliary variables $y_1, y_2, \ldots, y_m$

$$y_j = f_j(u_1, \ldots, u_{r_j})$$

where $f_j \in \mathcal{A}$ and

$$u_i = \begin{cases} \text{input } x_1, \ldots, x_n \\ \quad \text{or} \\ \text{auxiliary } y_k \text{ when } k < j. \end{cases}$$

y_m is the output of the circuit.

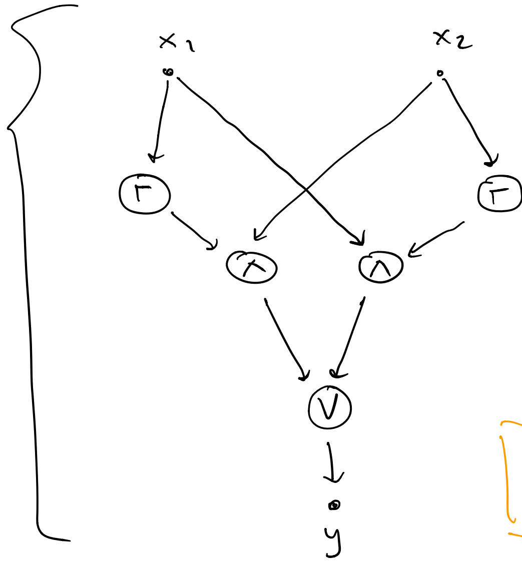
C computes $F : \mathbb{B}^n \to \boxed{\mathbb{B}}$ if

$$\underbrace{C(x_1, \ldots, x_n)}_{y_m} = F(x_1, \ldots, x_n) \quad \forall x_j \in \mathbb{B}.$$

no closed loops

C can be represented as a directed acyclic graph.

C :



$$\boxed{A = \{ \wedge, \vee, \neg \}}$$

$$C(x_1, x_2) = XOR(x_1, x_2) = \begin{cases} 1 & x_1 \neq x_2 \\ 0 & x_1 = x_2 \end{cases}$$

We will also represent circuits using "wires" & "gates".

a) NOT : $\mathbb{B} \longrightarrow \mathbb{B}$ $\quad x \longmapsto \neg x$

alternative notation

$0 \longmapsto 1$

$1 \longmapsto 0$

b) OR : $\mathbb{B}^2 \longrightarrow \mathbb{B}$ $\quad (x, y) \longmapsto x \vee y$

disjunction

00	$\longmapsto$	0
01		1
10		1
11		1

c) AND : $\mathbb{B}^2 \longrightarrow \mathbb{B}$ $\quad (x, y) \longmapsto x \wedge y$

conjunction

00	$\longmapsto$	0
01		0
10		0
11		1

Rem More generally we can study functions

$$f : \mathbb{B}^n \longrightarrow \mathbb{B}^m \quad (\text{logic gates})$$
$$(x_1, .., x_n) \longmapsto (y_1, .., y_m)$$

such a function amounts to

$$f_i : \mathbb{B}^n \longrightarrow \mathbb{B} \qquad i = 1, 2, .., m$$
$$(x_1, .., x_n) \longmapsto y_i$$

Disjunctive normal form (DNF)

Let $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ be a Boolean function.

a) If

$$f(x_1, .., x_n) = \begin{cases} 1 & x_i = a_i \in \mathbb{B} \ \forall i \\ 0 & o/w \end{cases}$$

then

$$f(x_1, .., x_n) = \mathrm{NOT}^{a_1}(x_1) \wedge \dots \wedge \mathrm{NOT}^{a_n}(x_n)$$

$$\boxed{\mathrm{NOT}^a(x) = \begin{cases} \neg x & a = 0 \\ x & a = 1 \end{cases} = \begin{cases} 1 & a = x \\ 0 & a \neq x \end{cases}}$$

b) For arbitrary $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ consider

$$S = \{ u = (u_1, .., u_n) \mid f(u) = 1 \}$$

We can write

$$f(x_1,\dots,x_n) = \bigvee_{u \in S} \underbrace{\chi_u(x_1,\dots,x_n)}_{\bigwedge \underbrace{x_i \text{ or } \neg x_i}_{\text{literal}}}$$

where

$$\underbrace{\chi_u(x_1,\dots,x_n)}_{\text{a function as in (a)}} = \begin{cases} 1 & \text{if } x_i = u_i \ \forall i \\ 0 & \text{o/w} \end{cases}$$

$$\chi_u(x_1,\dots,x_n) = \bigwedge_{i=1}^{n} \underbrace{\text{NOT}^{u_i}(x_i)}_{x_i \text{ or } \neg x_i} \qquad \blacksquare$$

<u>Ex</u> 1) NAND : $\mathbb{B}^2 \to \mathbb{B}$ $\quad (x_1,x_2) \mapsto \neg(x_1 \wedge x_2)$

$$S = \{ (0,0), (0,1), (1,0) \}$$

$\to$ $$\text{NAND}(x_1,x_2) = \underbrace{(\neg x_1 \wedge \neg x_2)}_{(0,0)} \vee \underbrace{(\neg x_1 \wedge x_2)}_{(0,1)} \vee \underbrace{(x_1 \wedge \neg x_2)}_{(1,0)}$$

2) XOR : $\mathbb{B}^2 \to \mathbb{B}$ $\quad (x_1,x_2) \mapsto x_1 \oplus x_2$ $\leftarrow$ mod 2 addition

$$S = \{ (0,1), (1,0) \}$$

$\to$ $$\text{XOR}(x_1,x_2) = \underbrace{(\neg x_1 \wedge x_2)}_{(0,1)} \vee \underbrace{(x_1 \wedge \neg x_2)}_{(1,0)}$$

De Morgan's identities

$$x \wedge y = \neg(\neg x \vee \neg y)$$
$$x \vee y = \neg(\neg x \wedge \neg y)$$

Conjunctive normal form (CNF)

Let $f : \mathbb{B}^n \to \mathbb{B}$ Boolean function.

Write $g = \neg f$ in DNF

$$g(x_1, \dots, x_n) = \bigvee_{u \in S} \chi_u(x_1, \dots, x_n)$$

Take $\neg$ of this eq. $(\neg g = \neg(\neg f) = f)$

$$f(x_1, \dots, x_n) = \neg\left(\bigvee_u \chi_u(x_1, \dots, x_n)\right)$$
$$= \neg\left(\neg \bigwedge_u \neg \chi_u(x_1, \dots, x_n)\right)$$
$$= \bigwedge_u \neg \chi_u(x_1, \dots, x_n)$$

$$\neg \chi_u(x_1, \dots, x_n) = \neg\left(\bigwedge_i \mathrm{NOT}^{a_i}(x_i)\right)$$
$$= \neg\left(\neg \bigvee_i \neg \mathrm{NOT}^{a_i}(x_i)\right)$$
$$= \bigvee_i \underbrace{\neg \mathrm{NOT}^{a_i}(x_i)}_{\text{literals i.e. } x_i \text{ or } \neg x_i}$$

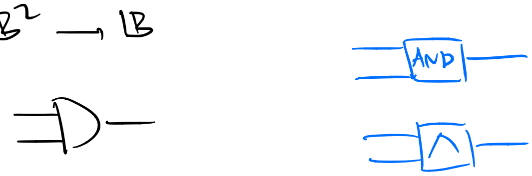
So $f = \bigwedge_u \bigvee_i (x_i \text{ or } \neg x_i)$. This is the CNF.

Representing circuits

A circuit computing f can be represented using "wires" and "gates" from a fixed set.

a) $\neg : \mathbb{B} \to \mathbb{B}$

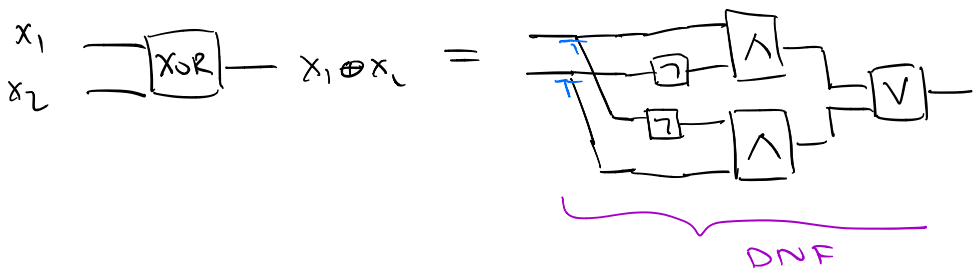
b) $\wedge : \mathbb{B}^2 \to \mathbb{B}$



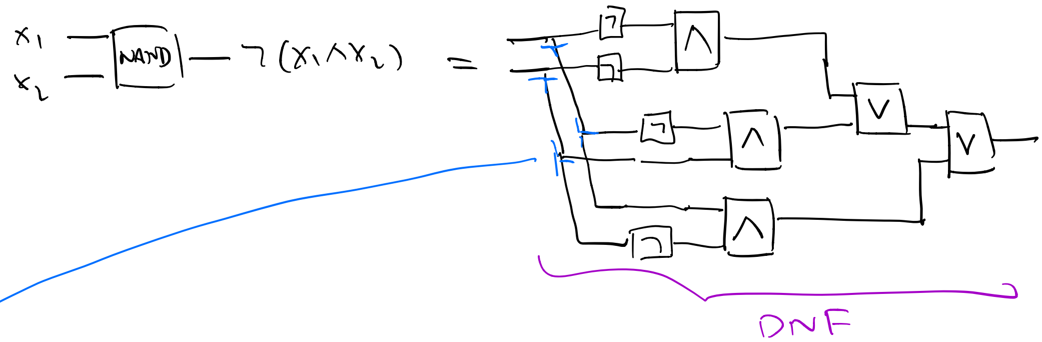
c) $\vee : \mathbb{B}^2 \to \mathbb{B}$

We can compose these to obtain more complicated circuits:

1)

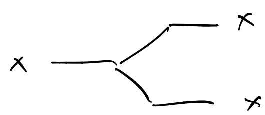


2)



In the circuit we are using a new gate

FANOUT : $\mathbb{B} \longrightarrow \mathbb{B}^2 \qquad x \longmapsto (x, x)$

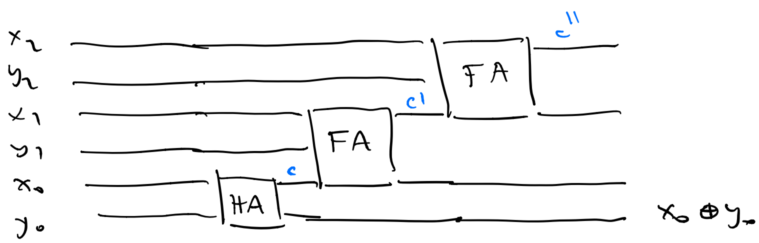


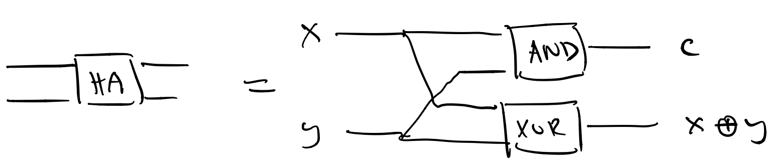
splitting a bit into two copies.

3) Adding $x = x_n \cdots x_0$ and $y = y_n \cdots y_0$

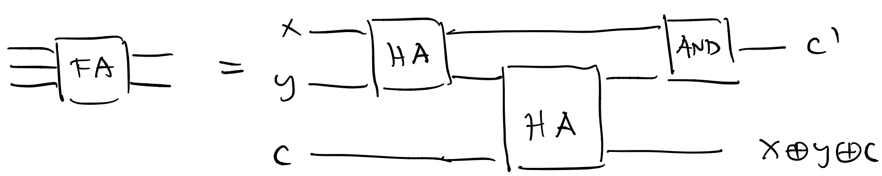
For simplicity $n = 2$:

$$\begin{array}{r} {}^{1}\,{}^{1}\,{}^{1} \\ 101 \\ 111 \\ \hline 1100 \end{array}$$





(carry bit)



Universal gates

$A = \{ f_j : \mathbb{B}^{n_j} \longrightarrow \mathbb{B} \}$ is called universal if any $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ $(n \geq 1)$ can be computed using a circuit over A.

Thm $A = \{ \vee, \wedge, \neg \}$ is universal.

Proof: Given $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ use DNF to write it as

$$f(x_1, \dots, x_n) = \bigvee_{u \in S} \chi_u(x_1, \dots, x_n)$$

and translate this into a circuit.

a) Consider $f : \boxed{\mathbb{B}} \longrightarrow \mathbb{B}$ $(n = 1)$

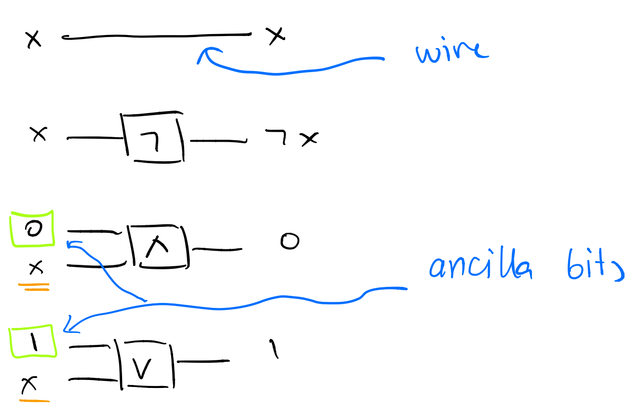
There are 4 different possibilities:

i) $\begin{matrix}0\\1\end{matrix} \longmapsto \begin{matrix}0\\1\end{matrix}$

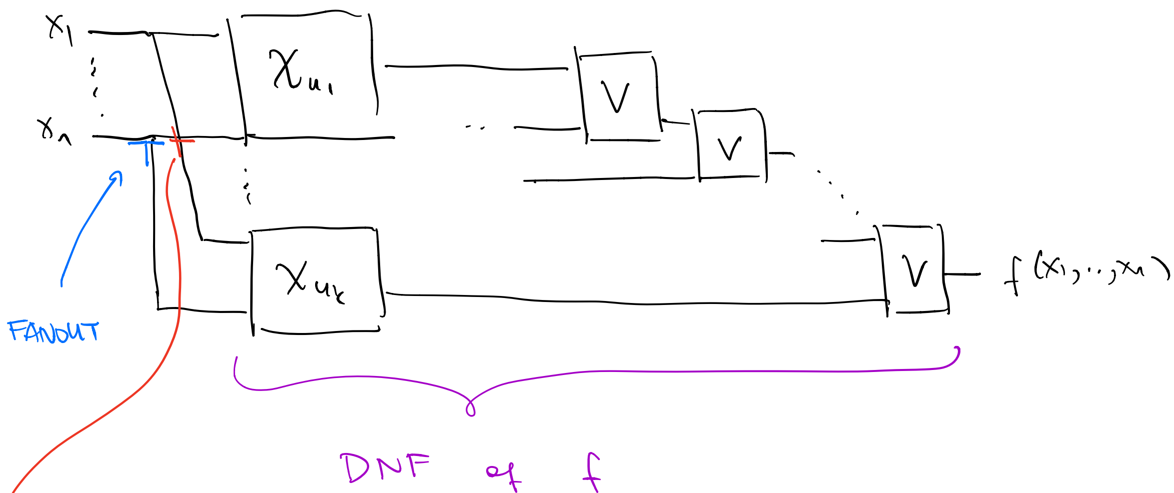
ii) $\begin{matrix}0\\1\end{matrix} \longmapsto \begin{matrix}1\\0\end{matrix}$

iii) $\begin{matrix}0\\1\end{matrix} \longmapsto \begin{matrix}0\\0\end{matrix}$

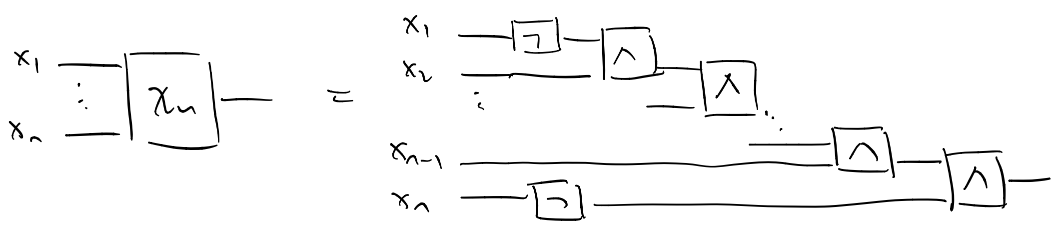
iv) $\begin{matrix}0\\1\end{matrix} \longmapsto \begin{matrix}1\\1\end{matrix}$



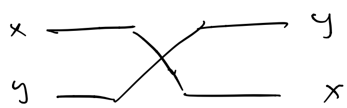
b) The circuit for $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ $(n \geq 2)$ would look like



e.g. $u = (0, 1, \dots, 1, 0)$



CROSSOVER :



$(\mathbb{B}^2 \longrightarrow \mathbb{B}^2)$

Therefore $\boxed{\{\neg, \wedge, \vee\}}$ is universal (provided that wires, ancilla bits, FANOUT's are available)

$\blacksquare$

Other universal gates

1) $\mathcal{A} = \{ \neg, \wedge \}$ since

$$x_1 \vee x_2 = \neg(\neg x_1 \wedge \neg x_2)$$

2) $\mathcal{A} = \{ \neg, \vee \}$ since

$$x_1 \wedge x_2 = \neg(\neg x_1 \vee \neg x_2)$$

3) $\mathcal{A} = \{ \wedge, \oplus \}$

4) $\mathcal{A} = \{ \text{NAND} \}$

5) $\mathcal{A} = \{ \text{NOR} \}$

$\text{NOR}(x,y) = \neg(x \vee y)$

Later on ...

We have seen that $\mathcal{A} = \{ \wedge, \vee, \neg \}$ is universal i.e. any

$(n=m)$

$$f : \mathbb{B}^n \to \mathbb{B}^m$$

can be expressed as a circuit over $\mathcal{A}$.

<u>Question</u>: Is there a finite set of quantum gates such that any

$$U : (\mathbb{C}^2)^{\otimes n} \to (\mathbb{C}^2)^{\otimes n} \in \mathcal{U}((\mathbb{C}^2)^{\otimes n})$$

can be approximated? Ans: Yes.

Turing machine vs. Circuit model

A predicate $f : \mathbb{B}^* \longrightarrow \mathbb{B}$ gives a sequence of Boolean functions

$$f_n : \mathbb{B}^n \longrightarrow \mathbb{B}$$

$$f_n(x_1, \dots, x_n) = f(x_1 \cdots x_n) \qquad n = 1, 2, \dots$$

Same as language, recall $L = f^{-1}(1)$.

We have seen that there are predicates that can not be computed (decided) by a TM, e.g. the halting function.

We also learned that any Boolean function computable by a circuit.

Therefore given $f : \mathbb{B}^* \longrightarrow \mathbb{B}$ there is a sequence of circuits

$$C_1, C_2, \dots, C_n, \dots$$

where C_n computes f_n and using $\{C_n\}_{n=1}^{\infty}$ we can compute any f.

Too powerful! Recall that some f (e.g. halting function) can not be computed (decided) by a TM.

Uniform circuit family

A family of circuits $\{C_n\}_{n=1}^{\infty}$ is called uniform if there exists a $\boxed{\text{TM } M \text{ (deterministic)}}$ such that

$$\varphi_M : \mathbb{B}^* \longrightarrow \mathbb{B}^* \quad (\text{function computed by } M)$$

$n \in \mathbb{N}$, $\langle n \rangle = \underbrace{b_k \cdots b_0}_{\text{input to tape of TM } M} \longmapsto \underbrace{\tilde{b}_k \cdots \tilde{b}_0}_{\text{output on the tape of TM } M}$

$n = \sum_{i=0}^{n} 2^i b_i$

$$\varphi_M(\langle n \rangle) = \langle C_n \rangle$$

Fact: The class of functions computable (decidable) by a TM is the same as the class of functions computable (decidable) by uniform circuit families.

Asymptotic notation

Let f and g be functions $\mathbb{N} \longrightarrow \mathbb{R}_{\geq 0}$.

a) We say $f(n)$ is in the class of functions $O(g(n))$ if there exists $c, \boxed{n_0} \in \mathbb{N}$ such that

$$f(n) \leq c\, g(n) \quad \forall\ n \geq \boxed{n_0}.$$

For short we say $f(n)$ is $O(g(n))$ (or we write $f(n) = O(g(n))$)

b) $f(n)$ is $\Omega(g(n))$ if there exists $c, n_0 \in \mathbb{N}$ s.t.

$$c\, g(n) \leq f(n) \quad \forall\ n \geq n_0$$

c) $f(n)$ is $\Theta(g(n))$ if $f(n)$ is both $O(g(n))$ and $\Omega(g(n))$.

Ex 1) If $f(n)$ is a polynomial of degree k then $f(n)$ is $O(n^{\ell})$ for any $\ell \geq k$.

2) $\log n$ is $O(n^k)$ for any $k > 0$.

3) n^k is $o(n^{\log n})$

<u>Rem</u> Other uses of O-notation

$f(n) = 2^{O(g(n))}$ means $\exists\, c, n_0 \in \mathbb{N}$ s.t.

$$f(n) \le 2^{c\,g(n)} \quad \forall\, n \geq n_0.$$

<u>Time</u> complexity

1) i) Let M be a TM that halts on all inputs (decider).
The time complexity of M is the function

$$t_M : \mathbb{N} \to \mathbb{N}$$

$t_M(n)$ = the maximum number of steps M performs on any input of length n.

We say M is a $t_M(n)$ time machine.

ii) Let N be a non-deterministic TM. (decider)

Time complexity of N

$$t_N : \mathbb{N} \to \mathbb{N}$$

$t_N(n)$: the max # steps N performs on any branch of its computation on any input of size n.

Rem : Note that time complexity is model dependent.

1) A $t(n)$ time multitape TM has an equivalent $O(t^2(n))$ time single-tape TM.

2) A $t(n)$ time non-deterministic TM has an equivalent $2^{O(t(n))}$ time deterministic single-tape TM.

2) Let $t: \mathbb{N} \to \mathbb{N}$ be a function.

i) The time complexity class $\text{Time}(t(n))$ is defined as follows

$\text{Time}(t(n)) = \{ L \mid L$ is a language decided by $\underbrace{O(t(n)) \text{ time}}$ $\boxed{\text{deterministic}}$ (single-tape) TM. $\}$

i.e. a TM s.t. $t_M(n)$ is in $O(t(n))$.

ii) $\text{NTime}(t(n)) = \{ L \mid L$ is a language decided by a $\underbrace{O(t(n)) \text{ time}}_{t_N(n) \text{ is } O(t(n))}$ $\boxed{\text{non-det}}$ TM. $\}$

Complexity classes P and NP

$$P = \underbrace{\bigcup_{k \in \mathbb{N}} \mathrm{Time}(n^k)}_{\text{languages decidable in polynomial time on a det. (single-tape) TM}} \quad \text{and} \quad NP = \underbrace{\bigcup_{k \in \mathbb{N}} \mathrm{NTime}(n^k)}_{\text{decidable in poly. time on a non-det TM.}}$$

polynomial

non-det. polynomial

Alternate description of NP

A verifier for L is a TM V s.t. (deterministic)

$$L = \{ \sigma \mid V \text{ accepts } \underbrace{\langle \sigma, w \rangle}_{\sigma \sqcup w} \text{ for some string } w \}$$

witness (certificate)

Time complexity of verifier is measured as a function of the length of σ.

Thm L has polynomial time verifier $\iff L \in NP$.

Sketch proof:

The sequence of non-det. choices made by an accepting computational branch can be seen as a witness, and vice versa. $\boxtimes$

One million dollar problem

$P = NP$ or $\underbrace{P \neq NP}_{\text{current belief}}$

Problem that are in P

1) Let G be a directed graph:

$G = (V, E)$

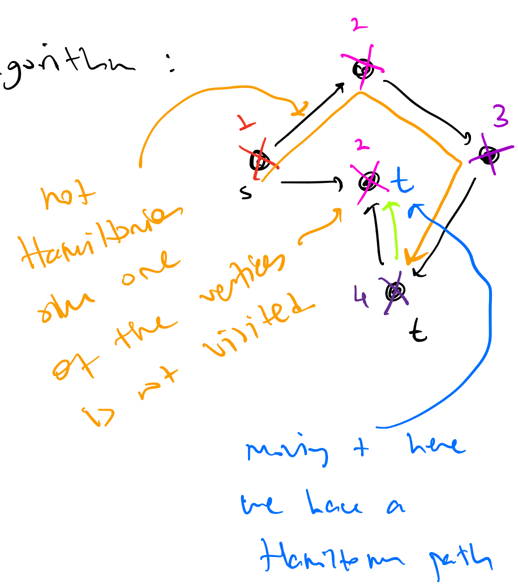
set of vertices (nodes) — set of edges

$\circ \longrightarrow \circ$ edges are directed

PATH = { $\langle G, s, t \rangle$ | G is a directed graph that has a directed path from s to t }

e.g. $\underset{s}{\bullet} \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \underset{t}{\bullet}$

algorithm:



1) cross out s

2) at each step crossout a node if

$\otimes \longrightarrow \circ$ ← crossout.

3) If t is crossed out accept otherwise reject.

Rem For graphs time complexity will be measured as a function of $|V|$ since $|\langle G \rangle|$ is assumed to be polynomial in $|V|$.

length of string

size of set V

2) RELPRIME = $\{ \langle x, y \rangle \mid x$ & y are relatively prime$\}$

algorithm: use Euclid's algorithm for finding the greatest divisor.

Problem that are in NP

1) A Hamiltonian path in a directed graph G is a directed path that goes through each node exactly once.

HAMPATH = $\{ \langle G, s, t \rangle \mid G$ is a directed graph with Hamiltonian path from s to $t\}$

It is believed that HAMPATH $\notin$ P.
but verifying a witness can be done in polynomial time, i.e. HAMPATH $\in$ NP.

2) FACTORING $= \{ \langle x, \ell \rangle \mid \ell < x$ &

$\exists\ k < \ell$ s.t. k divides $x \}$

verification is in polynomial time (long division).

" COMPOSITES $= \{ \langle x \rangle \mid x = pq$ for $p, q > 1 \}$ "

3) A Boolean formula is an expression ϕ involving Boolean variables $\{x_i\}$ and Boolean operations $\{ \vee, \wedge, \neg \}$

e.g. $\phi = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_3)$.

A Boolean formula is satisfiable if $\phi(a_1, \dots, a_n) = 1$ for some $(a_1, \dots, a_n) \in \mathbb{B}^n$.

SAT $= \{ \langle \phi \rangle \mid \phi$ is satisfiable Boolean formula $\}$

<u>Thm</u> [Cook - Levin]

$$\text{SAT} \in P \iff P = NP$$

Since current belief $P \neq NP \implies$ SAT $\notin P$.

Reducibility

A language L_1 is (polynomial time) reducible to another language L_2 if there exists a TM M operating in polynomial time such that

$$\sigma \in L_1 \iff \underbrace{\varphi_M(\sigma)}_{\text{output of } M} \in L_2$$

NP-completeness

A language L in a complexity class (P or NP) is complete if any other language in that complexity class can be reduced to L.

Thm [Restatement of Cook-Levin Thm]

SAT is NP-complete.

1) SAT $\in$ NP
2) $\forall L \in NP$ is reducible to SAT. (NP-hard)

Other NP-complete problems

1) A 3CNF-formula is a Boolean of the form

$\varphi(x_1, \dots, x_n) = (y_1 \vee y_2 \vee y_3) \wedge \cdots \wedge (y_{k-2} \vee y_{k-1} \vee y_k)$

where y_j are literals i.e. x_i or $\neg x_i$.

$\boxed{3SAT} = \{ \langle \phi \rangle \mid \phi$ is a satisfiable 3CNF-formula$\}$

SAT is reducible to 3SAT ($\Rightarrow$ 3SAT is NP-complete)

Given ϕ use CNF to write it as

$\phi(x_1, \dots, x_n) = (y_1 \vee \cdots \vee y_e) \wedge \cdots \wedge (\cdots)$

each $(y_1 \vee \cdots \vee y_e)$ term can be written as a 3CNF formula

Ex 3CNF for $x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4$

$x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4 =$

$(x_1 \vee \neg x_2 \vee z) \wedge (\neg x_3 \vee x_4 \vee \neg z)$

where z is a new variable.

2) HAMPATH is NP-complete

3) Circuit version of SAT

$CSAT = \{ \langle C \rangle \mid C$ is a satisfiable Boolean circuit$\}$

Circuit complexity

a) The size of a circuit is the number of gates it contains.

The size complexity of a circuit family $\{C_n\}_{n \in \mathbb{N}}$ is the function $f : \mathbb{N} \to \mathbb{N}$ where

$$f(n) = \text{size of } C_n.$$

The circuit complexity of L is the size complexity of a minimal (in size) circuit family deciding L.

$$\Big(\underset{|\sigma| = n}{\sigma \in L} \iff C_n(\sigma) = 1 \Big)$$

b) The depth of a circuit is the length of the longest path from an input variable to the output. (Think of C as a acyclic directed graph)

(We can define depth complexity similarly.)

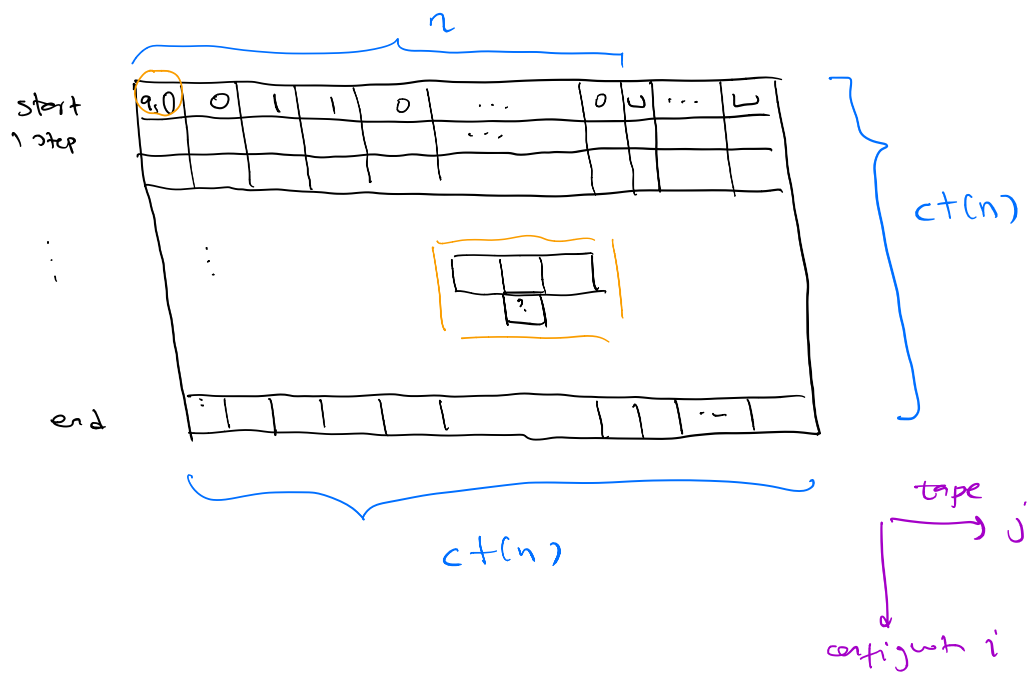
Thm Let $t: \mathbb{N} \to \mathbb{N}$ be a function such that $t(n) \geq n \;\; \forall n$. (to allow TM to read the input)

If $L \in \text{Time}(t(n))$ then L has circuit complexity $O(t(n)^2)$.

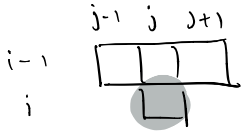
Proof: Let $M = (Q, \Gamma, \delta)$ decide L in time $\leq c\,t(n)$ for some constant c.

Let σ be an input $\underbrace{|\sigma|}_{\text{input length}} = n$.

Tableau for M



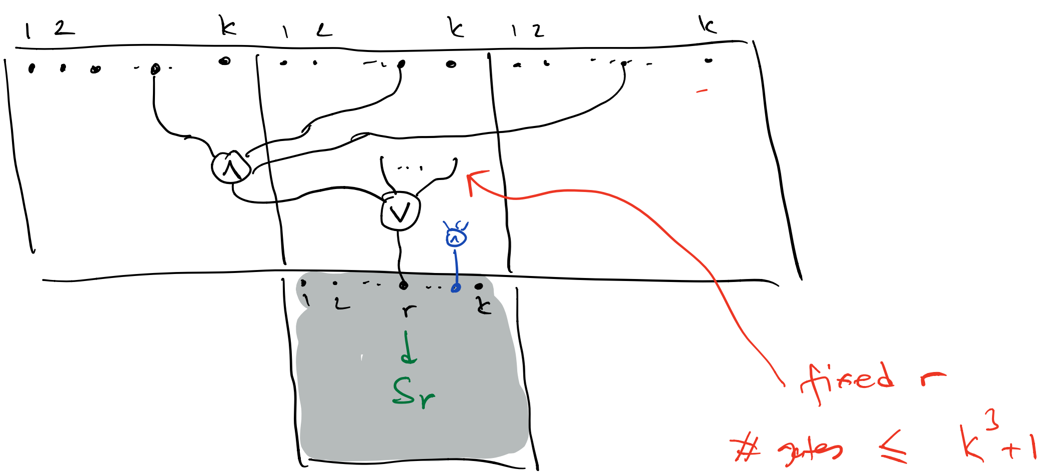
$$\text{cell}[i,j] = \begin{cases} s & s \in \Gamma \quad \text{if head} \neq j \\ qs & q \in Q \quad s \in \Gamma \quad \text{if head} = j \end{cases}$$



cell $[i, j]$ is determined by cell $[i-1, j-1]$, cell $[i-1, j]$, cell $[i-1, j+1]$ via the transition function δ.

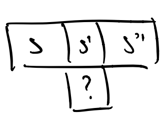
Let $k = |\Gamma| + |Q \times \Gamma|$

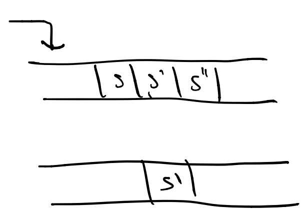
$$\Gamma \cup Q \times \Gamma = \{ \underset{\substack{\| \\ 0}}{s_1}, \underset{\substack{\| \\ 1}}{s_2}, \dots, s_k \}$$



Some cases demonstrating how δ is encoded

1) $s, s', s'' \in \Gamma$

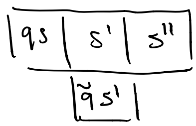


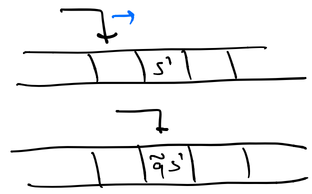


$s' \in \Gamma$

the symbol does not change

2)

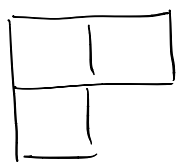




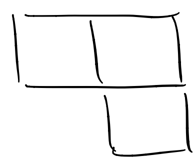
$(\tilde{q}, s') \in Q \times \Gamma$

On the sides of the tableau the picture is slightly different:

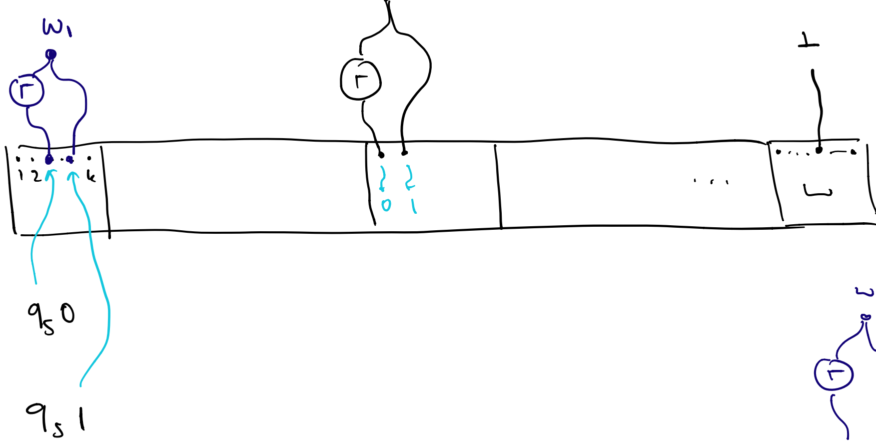
Left side

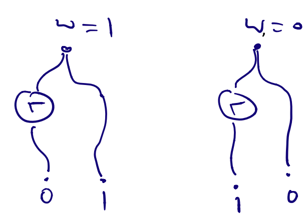


right side



Start configuration



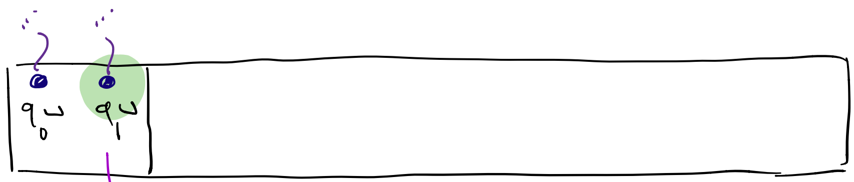


End configuration

The TM M before going into the accept state (q_1) it moves its head to the first cell & writes a $\sqcup$ symbol.

We can always modify a given TM in this way.

Then the circuit at the last row looks like



we pick this as the output.

Then the circuit C constructed this way satisfies

$$M(\sigma) = 1 \iff C(\sigma) = 1.$$

In other words C decides L.

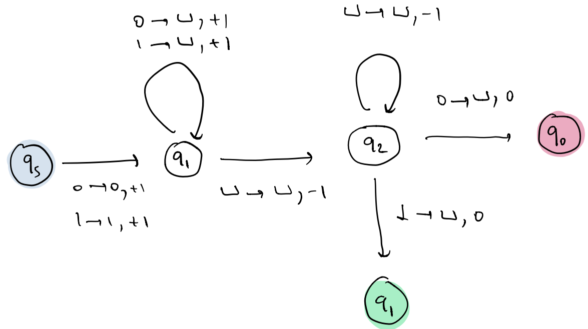
The total # of gates in the circuit:

$$\text{Total \# gates} \leq \underbrace{(k^3+1)\, c^2\, t(n)^2}_{O(t(n)^2)}\, k$$

Circuit complexity is $O(t(n)^2)$. $\blacksquare$

Ex) $M = (\Gamma, Q, \delta)$ $\quad \Gamma = \mathbb{B} \sqcup \{\sqcup\}$

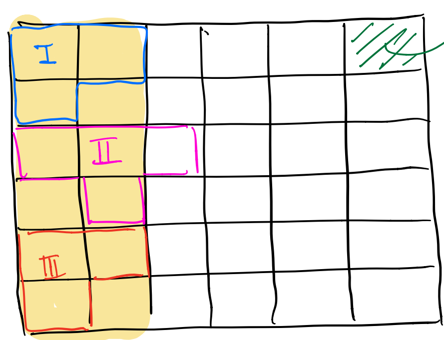
1. $|a_1|a_2|\sqcup|$ $\quad q_s$
2. $|a_1|a_2|\sqcup|$ $\quad q_1$
3. $|a_1|\sqcup|\sqcup|$ $\quad q_1$
4. $|a_1|\sqcup|\sqcup|$ $\quad q_2$
5. $|a_1|\sqcup|\sqcup|$ $\quad q_2$
6. $|\sqcup|\sqcup|\sqcup|$ $\quad q_{a_1}$

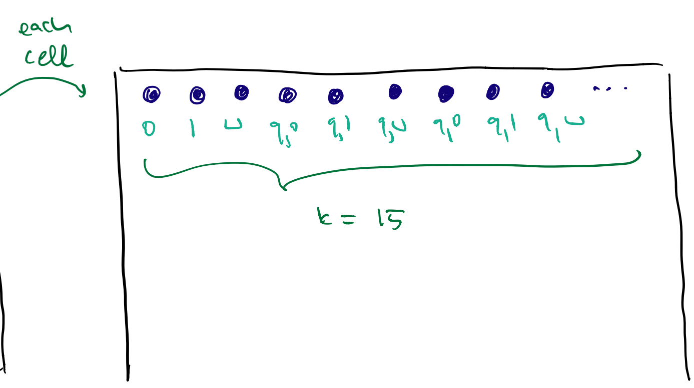


$k = |\Gamma| + |Q \times \Gamma|$

$= 3 + 5 \times 3 = 18$

Tableau for M :

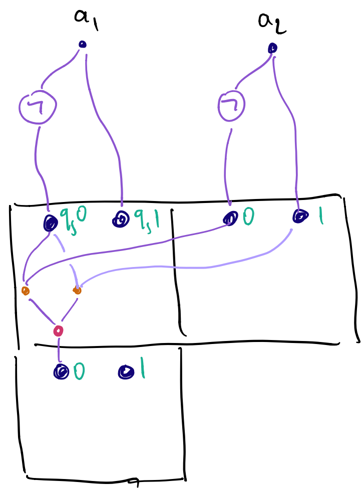




$L = \{ a_1 a_2 \cdots a_n \mid a_1 = 1 \ \& \ n \geq 1 \}$

M decides L : $\sigma \in L \iff M(\sigma) = 1$.

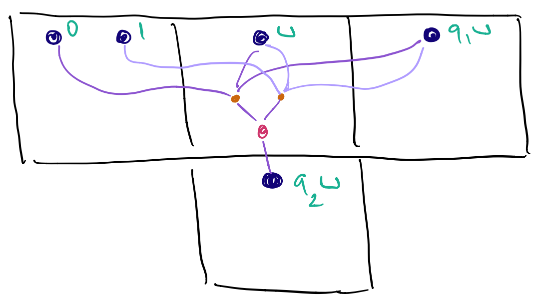
I)



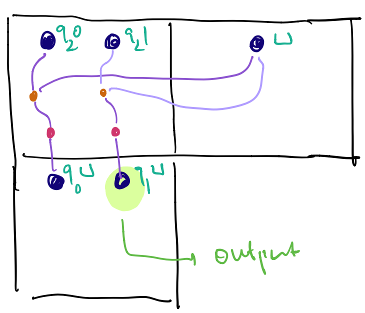
● : AND

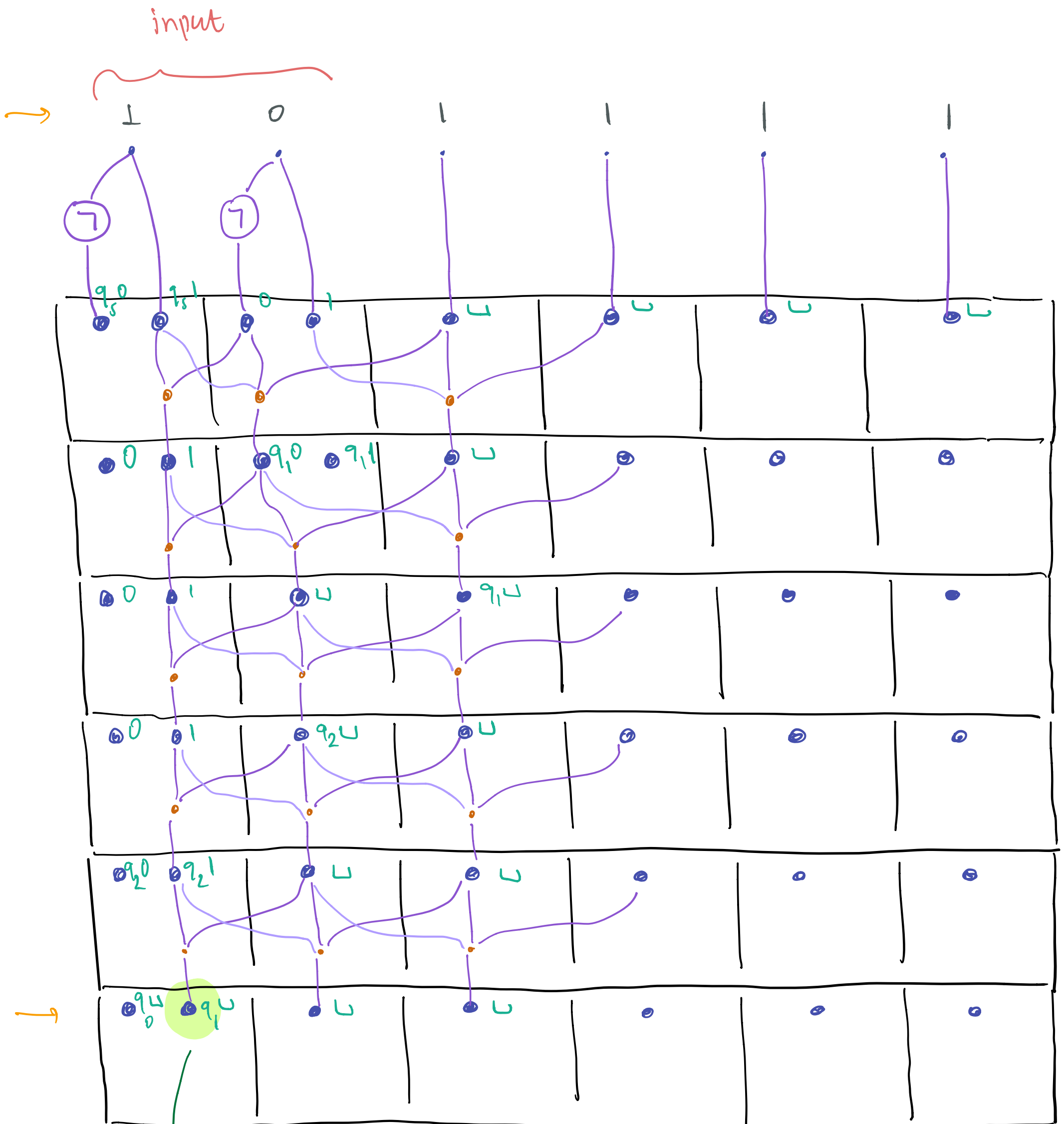
● : OR

II)



III)





we see an output of $\perp$.

$M(\sigma) = 1 \iff C(\sigma) = 1.$

Given $L \in NP$ with polynomial time verifier V construct the circuit C as in the proof of the Theorem.

We have

$$V(\langle \underbrace{G}_{\text{input}}, \overbrace{w}^{\text{witness}} \rangle) = 1 \iff C(G, w) = 1$$

→ Let M be a TM that constructs C in polynomial time.

Then

$$G \in L \iff \varphi_M(G) = \langle C \rangle \in CSAT$$

i.e. L is reducible to $CSAT$.

We have sketched a proof of the following:

Theorem [Circuit version of Cook-Levin]

$CSAT$ is NP-complete.

Complexity class BPP bounded error probabilistic polynomial time

a) Probabilistic TM $M = (\Gamma, Q, \delta_0, \delta_1)$

At each step of computation δ_0 or δ_1 is applied with probability $1/2$.

The machine either accepts or rejects

$$M(\sigma) \in \{0, 1\} \quad \text{(i.e. does not loop)}$$

We define probability that M accepts σ:

$$p(M(\sigma)=1) = \sum_b p(b)$$

where b runs over accepting branches and

$$p(b) = 2^{-k} \qquad k = \#\text{ coin flips.}$$

Probability that M rejects σ:

$$p(M(\sigma)=0) = 1 - p(M(\sigma)=1).$$

b) M decides L with probability $0 \le \varepsilon < \frac{1}{2}$ if

$$\forall \sigma \in L, \quad p(M(\sigma)=1) \geqslant 1-\varepsilon$$
$$\forall \sigma \notin L, \quad p(M(\sigma)=0) \geqslant 1-\varepsilon.$$

c) BPP is the class of languages decided by a probabilistic polynomial time TM with error probability of $1/3$.

complexity is measured similar to non-det TM

This number is arbitrary. Any $0 < \epsilon < 1/2$ works

Nielsen – Chuang takes $\epsilon = 1/4$.

Amplification lemma: The error probability can be made arbitrarily small by repeating the algorithm. (Can be deduced from Chernoff bound, see Box 3.4 from textbook.)

Strong Church–Turing thesis

Any model of computation can be simulated on a probabilistic TM with at most a polynomial increase in the number of elementary operations required.

FACTORING $\in$ NP, it is believed that $\notin$ P.

Moreover, FACTORING believed to be $\notin$ BPP.

Shor shows that

FACTORING $\in$ BQP.

This is a potential threat to strong CT thesis

We have $\overbrace{\underbrace{P \subseteq BPP} \subseteq BQP}$

det. TM is a

prob. TM with $\delta_0 = \delta_1$

It is not known if $P \subsetneq BPP$ or $\underbrace{P = BPP}_{\text{current belief}}$

PRIMES $= \{ n \mid n \text{ is a prime} \}$

belongs to BPP.

Recent progress: PRIMES $\in P$

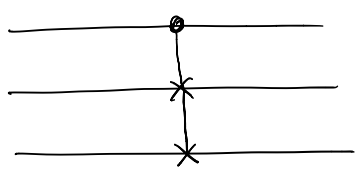
Reversible computation

A logic gate $f : \mathbb{B}^n \longrightarrow \mathbb{B}^n$ is called reversible if f is a permutation, i.e. isomorphism of sets.

1) Fredkin gate

$$\mathrm{CSWAP} : \mathbb{B}^3 \longrightarrow \mathbb{B}^3$$

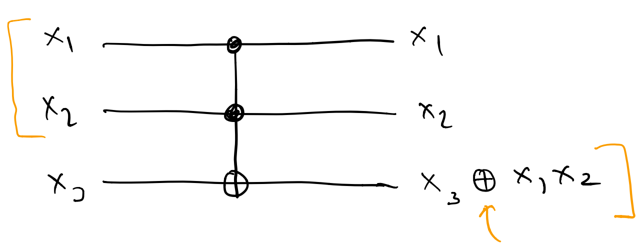
$$(x_1, x_2, x_3) \longmapsto \begin{cases} (0, x_2, x_3) & \text{if } x_1 = 0 \\ (1, x_3, x_2) & \text{if } x_1 = 1. \end{cases}$$



Thm $\mathcal{A} = \{\mathrm{CSWAP}\}$ is universal.

2) Toffoli gate

$$\mathrm{CCNOT} : \mathbb{B}^3 \longrightarrow \mathbb{B}^3$$



Thm $\mathcal{A} = \{\mathrm{CCNOT}\}$ is universal

Rem: Let $f : \mathbb{B}^n \to \mathbb{B}^m$ be a logic gate.

This can be turned into a reversible logic gate

$$\vec{f} : \mathbb{B}^{n+m} \longrightarrow \mathbb{B}^{n+m}$$

where

$$\vec{f}(x, \underbrace{0 \cdots 0}_{m}) = (x, f(x))$$

distinct for each x.

and extend to an isomorphism on $\mathbb{B}^{n+m}$.

More generally, it is possible to arrange

$$\vec{f}(x, y) = (x, y \oplus f(x))$$

by using ancilla bits & ignoring garbage bits.