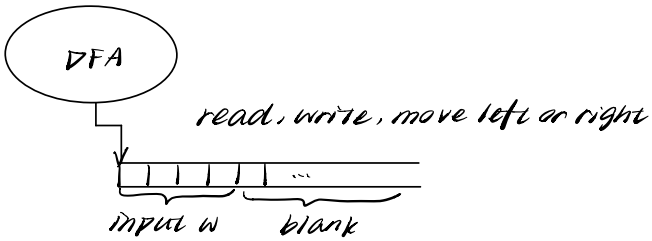
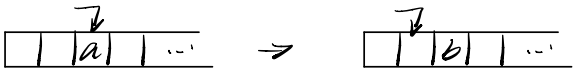
Turing Machine ♥

Consider a DFA with an infinite memory tape



- one accept state, one reject state.
  We stop only when we reach one of these, otherwise it can go forever
- transition $a \to b, L$ means if tape-head points at an "a" then replace it with "b" and move left.



DEF A turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

- Q : finite set of states
- Σ : input alphabet. We have a blank symbol that is not in Σ. "$\sqcup$" $\notin \Sigma$
- Γ : tape alphabet. $\Sigma \cup \{\sqcup\} \subseteq \Gamma$
- δ : $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- q_0 : start
- $q_{accept} \neq q_{reject}$

} different $\Rightarrow$ at least 3 states

<u>computation</u>

In the beginning, w is written in the beginning of the tape. The rest is filled with "$\sqcup$"s and tape head is on the first cell, we're at q_0.

- We follow the transition according to δ
  (If the tape head is on the first cell and we see an "L", we stay here)
- We stop when we reach q_{accept} or q_{reject}.

$\Rightarrow$ Accept Reject Loop

DEF $L(M) = \{w \mid M \text{ accepts } w\}$

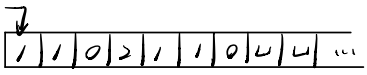
DEF A TM M that never loops is called **decider** (or algorithm)
In this case, we say M **decides** $L(M)$.

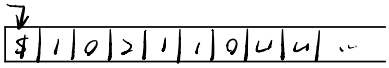
DEF A language L is called

$\begin{cases} w \in L \Leftrightarrow M \text{ accepts } w \\ w \notin L \Rightarrow M \text{ reject / loop} \end{cases}$

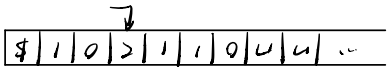
- **Turing-recognizable** if there is a TM M with $L(M) = L$
- **decidable** if there is a decider M with $L(M) = L$ $\begin{cases} w \in L \Rightarrow \text{accept} \\ w \notin L \Rightarrow \text{reject} \end{cases}$

Ex. Construct a TM that decides $\{w\#w \mid w \in \{0,1\}^*\}$ over $\Sigma = \{0,1,\#\}$
not context-free

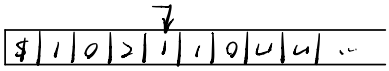




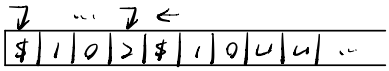
we remember we read 1

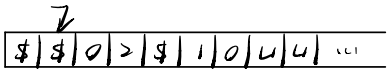


we move the tape until we get #

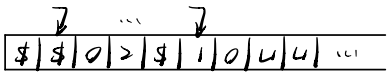


we expect to see a "1" (if not reject)





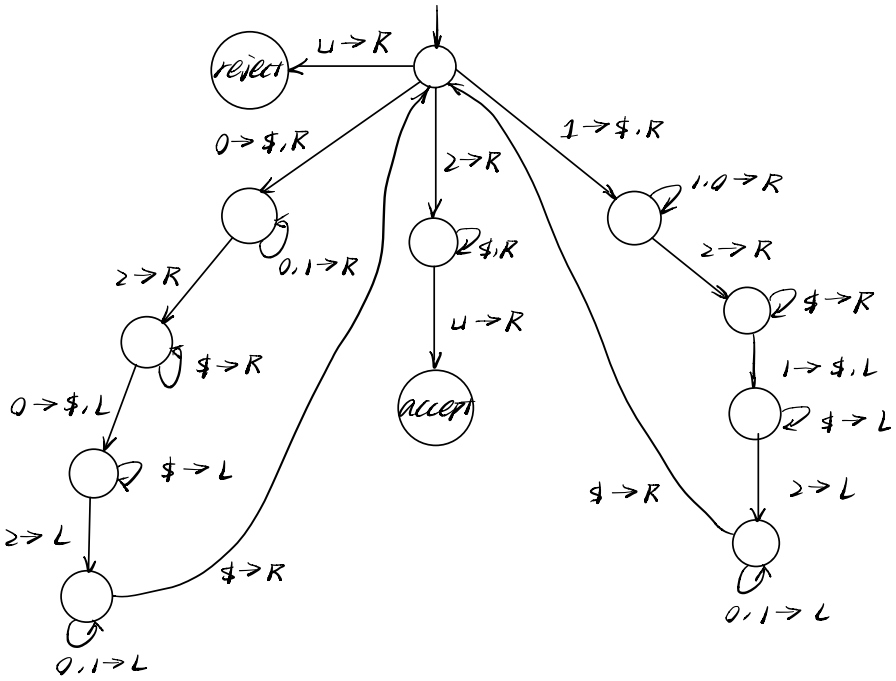
remember we read 1



we move to the first unread letter of the second part. If it's not 1 reject

To simplify the picture

- $a \to L$ means $a \to a, L$
  $a \to R$ means $a \to a, R$
- if an arrow for "a" is missing, it mean it exists and go to reject



Ex. Construct a TM that decides $L = \{0^{2^n} \mid n \geq 0\}$ over $\Sigma = \{0\}$

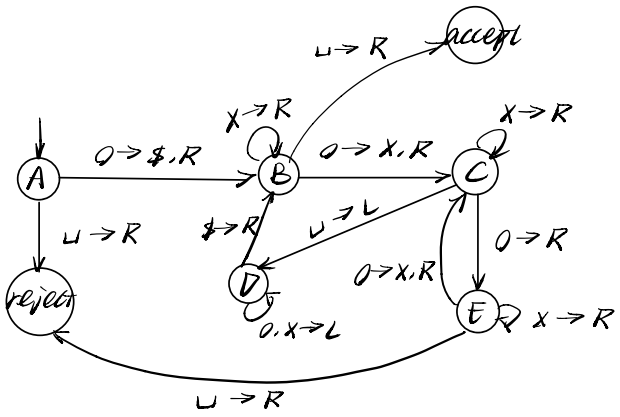
① Swap from left to right cross off every other "0".

② If in stage ① the tape contained an odd number of "0"s, then reject.

③ Return the head to the left of the tape. Go to stage ①

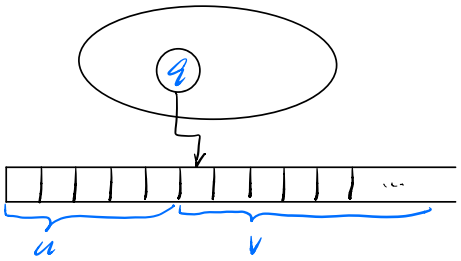
START

reject



Notation A configuration is a "snapshot" of the machine at a particular point in the computation. It contains

(i) The current state (ii) The position of head (iii) The current tape content



we write uqv to denote this config

e.g. run on $w = 0000$

A0000 $B000 $XC00 $X0E0 $X0XC

$X0DX $XD0X $DX0X D$X0X $BX0X

$XB0X $XXCX $XXXC $XXDX $XDXX

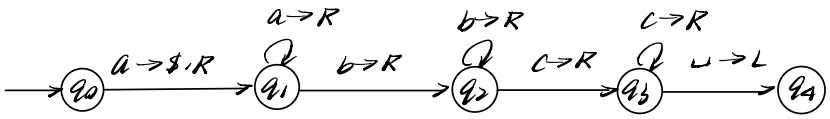
$DXXX D$XXX $BXXX $XBXX $XXBX

$XXXB $XXX␣q

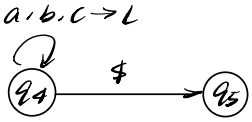
Ex. (Multiplication) Describe a TM that decides

$$L = \{ a^n b^m c^{mn} \mid m, n \geq 1 \}$$

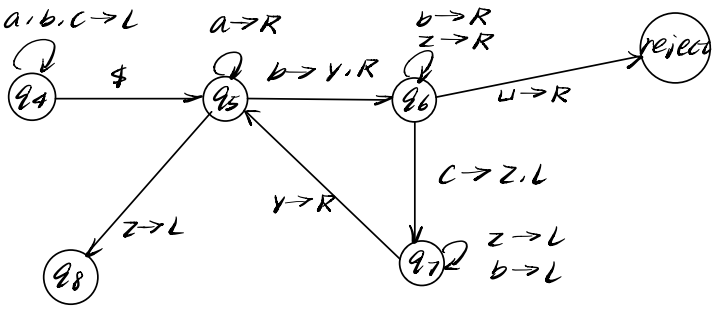
① Format check: $a^i b^j c^k$



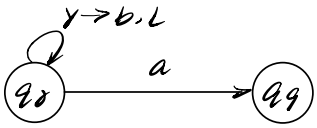
② Bring the tape to the left-end of the tape



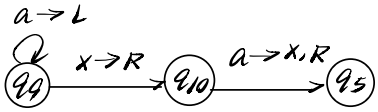
③ We'll use X, Y, Z to cross off a's, b's and c's. For every "a" that we cross off, we'll go over b's and c's and cross off a "c" for each "b" that we cross off. Afterwards, we recover all b's and repeat this for next "a".



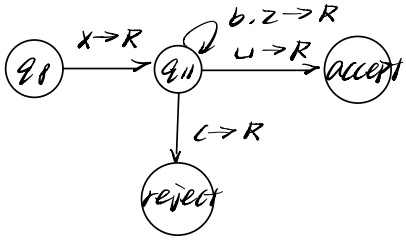
④ restore crossed off b's.



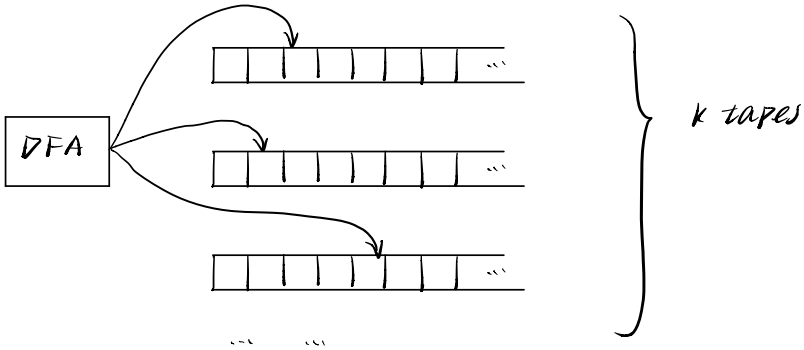
⑤ If there're a's left, cross off the left-most one and get to stage ③



⑥ If no a left, see if all c's are crossed off. If yes, accept; else, reject.



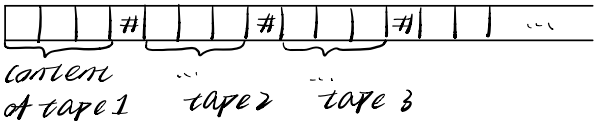
Multi-Tape Turing Machine



transition: $\delta: Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$

THM Every multi-tape TM is equivalent to some single tape TM.

It's possible to simulate several tapes on a single tape



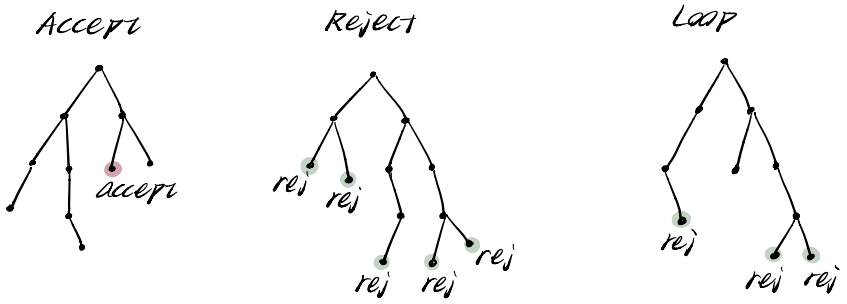
If we need more memory for a tape, we shift everything.

Nondeterministic Turing Machine

At every step during the computation, it may proceed according to several possibilities.

$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$

- It accepts w if there $\exists$ a computation path with input w that leads to accept
- It rejects w if $\forall$ computation path lead to reject.
- Otherwise loop



THM Every nondeterministic TM N has an equivalent TM M.

Let r be the max number of choices that we might face during a computation by M.

$$r = \max_{q \in Q, a \in \Gamma} |\delta(q, a)|$$ maximum # children in the parsing tree

Note that a string $y = y_1 y_2 \cdots y_m$ can tell us what choices to take in the first m steps of the computation. Here $y_i \in \{1, \cdots, r\}$ and means that in the i-step, take the y_i'th choice out of all the $\leq r$ available choices.

Consider the following algorithm:

① Set $m = 1$

② Set $y = \underbrace{111\cdots1}_{m}$

③ Run M on w for m steps and whenever we face multiple choices we pick one according to y. If we reach accept: accept w. If we didn't reach accept or reject, then remember this as potential "∞-loop".

④ Pick the next $y \in \{1, \cdots r\}^m$ and go to ③

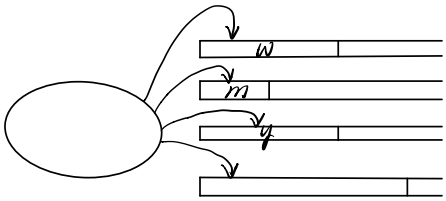
⑤ If we finish all the y's
- If we had not found a potential ∞-loop, then reject
- Else increase m and go to ②

N accept $\Rightarrow$ M accept
N reject $\Rightarrow$ M reject
N loop $\Rightarrow$ M loop

Equivalent to

for $m = 1$ to ∞
  for each $y \in \{1, \cdots r\}^m$ r^m
    run N on w for m steps
    making choices according to y
    if we reach accept $\Rightarrow$ accept, terminate
    if we didn't halt after m steps $\Rightarrow$ potential loop
  end for
  if no potential loop $\Rightarrow$ reject, halt

Implementation using 4-tape



- Tape 1: Contain w (never write on it)
- Tape 4: Simulate the tape of the original TM N. At every iteration, we clear this, copy w there and simulate N on it for m steps.

THM A language is decidable $\Leftrightarrow$ some non-deterministic TM decides it.

Enumerator

DEF An **enumerator** is similar to a TM except that it has an output device (printer) to print strings.

- They don't have accept and reject, but can halt.
- Starts with an empty string, keep printing strings.

THM A language is Turing recognizable ⇔ It can be enumerated by an enumerator.

⇒ A TM M recognizes L. Construct an enumerator E to print L

Let $s_1, s_2, s_3 \cdots$ be all the words in Σ^* (sorted according to length and then lexicographically) e.g. $\varepsilon, 0, 1, 00, 01 \cdots$

```
for i = 1, 2 ... ∞
    run M on s_i  ⇒ wrong! TM might loop
        if M accepts s_i then print(s_i)
end for

for k = 0, 1, ...        # steps
    for i = 1, ... k
        run M on s_i for k steps
        if M accepts s_i after exactly k steps
                print s_i
```

⇐ An enumerator E for a language L. to convert it into a TM, consider an input w, run the enumerator on an empty tape. If at any point it prints w, we will accept; if it halts without printing it, reject.

① $\{ \langle G, u, v \rangle \mid G$ is a finite undirected graph and there is a uv-path in $G \}$ decidable.

On input w, if w is not of the form $\langle G, u, v \rangle$, reject format check
Run a BFS starting at u and if we reach v, accept; else, reject

② $\{\langle w\rangle \mid w$ is an arrangement of 8×8 chess game. if white starts, it can win$\}$

decidable ⇒ finite ⇒ regular

(full algo for n×n in Lec 15)

for $k = 1, 2 \cdots$

try all the possible moves up to k steps

if all of them, white has a way of winning ⇒ accept

we will avoid repeating the same configuration (if same ⇒ halt, continue)

if all the branches stop without white winning ⇒ reject

key: finite amount of configurations.

③ $\{w\}$ where $w = \begin{cases} 1 & \text{Biden wins} \\ 0 & \text{Trump wins} \end{cases}$ decidable

When designing TM, assume L is given.

④ $L = \{\langle p(x)\rangle \mid$ $p(x)$ is a one variable polynomial with integer coefficients and has an integer root$\}$

Turing recognizable

$\langle x^2+1\rangle, x+1\rangle \notin L \quad \langle x-\frac{1}{5}\rangle \notin L \quad \langle x^2+1\rangle \notin L$

On input w

check to see if w is in the right format

for $i = 0, 1, 2 \cdots$

see if $p(i) = 0$ or $p(-i) = 0$ then "accept".

- if $\langle p(x)\rangle \in L$ ⇒ accept
- if $\langle p(x)\rangle \notin L$ ⇒ loop

$\Big\}\ L(M) = L$

⑤ $L = \{\langle A\rangle \mid L(A) = \phi$, A is a DFA$\}$ decidable

On input $\langle A\rangle$

if wrong format, reject

run a DFS starting from q_0 and if we ever reach any accept state, then reject;

else accept

⑥ $L = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFA's and } L(A) = L(B) \}$ decidable

On input $\langle A, B \rangle$
reject if wrong format
construct a DFA C for $L' = (L(A) \cap L^c(B)) \cup (L(B) \cap L^c(A))$
if $L(C) \neq \phi \Rightarrow$ reject
else $\Rightarrow$ accept
} use ⑤

⑦ $K = \{ \langle C, w \rangle \mid C \text{ is a CFG and } w \in L(C) \}$ decidable

Hint: Chomsky Normal Form

$B \to CD$ $(C, D \neq A)$
$B \to a$
$A \to \varepsilon$ Only available for start state

Note that if C is in the Chomsky Normal Form, then a left-most derivation for a word "w" is of length at most $2|w|$: Because we can apply at most $|w|$ rules of the form $B \to CD$, and at most $|w|+1$ rules of the form $B \to a$ or $A \to \varepsilon$.

On input $\langle w \rangle$
Convert C to Chomsky Normal Form C'
Generate all the possible left-most derivations of length at most $k = 2|w|+1$
if r is the number of rules in C', every such derivation can be encoded as $(i_1, i_2 \cdots i_{2|w|+1})$ where $i_j \in \{1, \cdots r\}$ tells us which rule to apply at step j. If any of these generates w, accept; else reject.

② $\{ \langle C \rangle \mid C \text{ is a CFG with } L(C) \neq \Sigma^* \}$ Turing recognizable

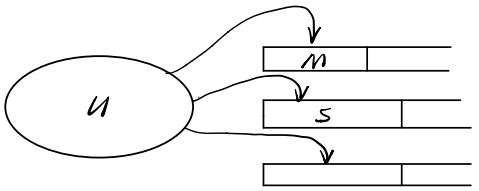
On input $\langle C \rangle$ $\{ C : L(C) = \Sigma^* \} \Rightarrow$ impossible
for $k = 0, 1, 2 \cdots \infty$
generate all the Σ^* words w of length k
if $w \notin L(C) \Rightarrow$ accept
use ⑦

⑨ $L = \{ \langle M, s \rangle \mid M \text{ is a TM that halts on } s \}$ Turing recognizable

Universal Turing Machine

It's possible to design a TM U that does the following
- Takes an input w
- If w is not in the form $\langle M, s \rangle \Rightarrow$ reject
- Else simulate the computation of M on s step by step
- If at any point
  - M accepts $s \Rightarrow U$ accepts $w = \langle M, s \rangle$
  - M rejects $s \Rightarrow U$ rejects $w = \langle M, s \rangle$
  - Otherwise as M loops on s, U naturally loops on $\langle M, s \rangle$

U is like the "program" that runs your (say) Java "program" on its input s.



description of M

tape of M

for U to simulate M (keep track of current state of U, the position of head ...)

On input $\langle M, s \rangle$
- run U on $\langle M, s \rangle$
- if it halts $\Rightarrow$ accept
- else $\Rightarrow$ reject.

⑩ $L = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$ Turing recognizable

If the input is not in the form $\langle M, w \rangle$, reject

Use the universal TM U to simulate M on w
- If at any point M accepts w, then stop and accept $\langle M, w \rangle$
- If at any point M rejects w, then stop and reject $\langle M, w \rangle$

$\langle M, w \rangle \in L \Rightarrow$ accept

$\langle M, w \rangle \notin L \Rightarrow$ reject / loop