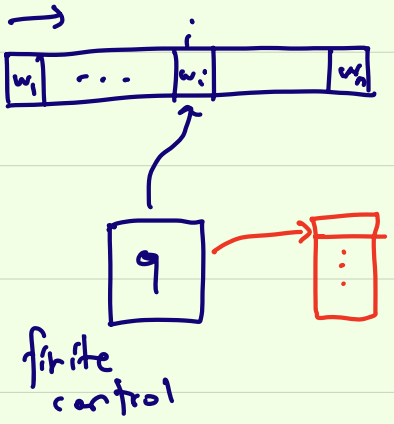
Lecture 19

PUSHDOWN AUTOMATA (PDA)

- Machine model for CFLs.
- Recall that CFLs form a superset of regular languages. That is, every regular language is a CFL, but not vice versa.

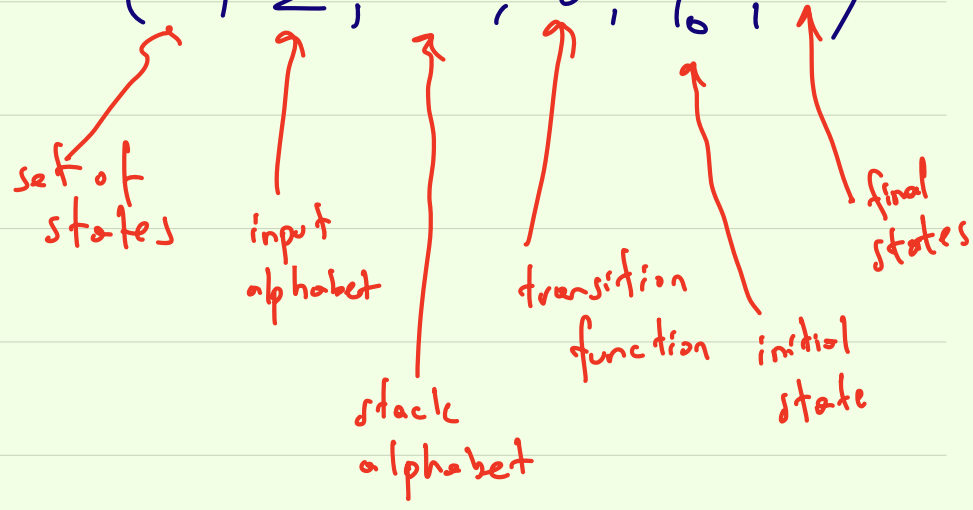
  $\Rightarrow$ We extend finite automata with a storage device called "stack".

Physical representation:



- read only
- tape moves only right (1-way)

Definition: A pushdown automaton (PDA) is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$



- $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

  $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$

Definition: Given a state $q \in Q$, an input symbol $a \in \Sigma_\varepsilon$ and a stack symbol $X \in \Gamma_\varepsilon$, the transition function δ returns a set of pairs of

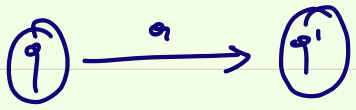
- a symbol $Y \in \Gamma_\varepsilon$ to replace X
- next state q'

i.e.

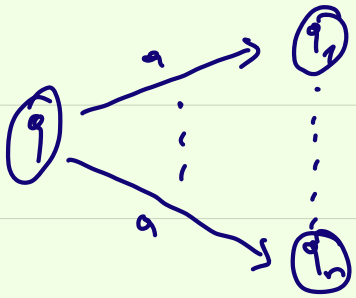
$$\delta(q, a, X) = \{(q_1, Y_1), (q_2, Y_2), \ldots, (q_n, Y_n)\}$$

Remark: PDAs are nondeterministic.

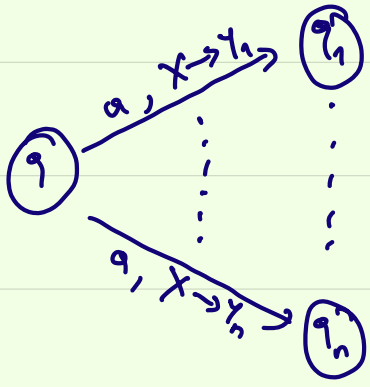
DFA:



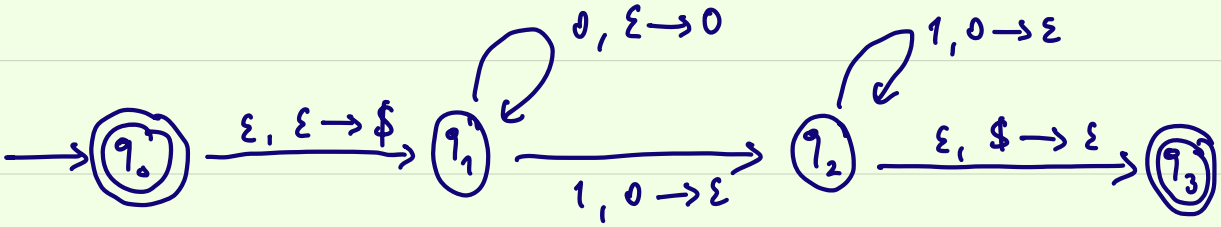
NFA:



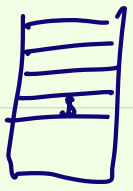
PDA:



Example: $L = \{0^n 1^n \mid n \geq 0\}$ G: $S \to 0S1 \mid \varepsilon$



1100 ✗
0101
001100



$\underline{0^3 1^3}$

Alg. idea:

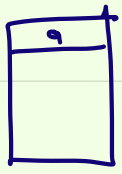
- Push $ to the stack
  (to mark its bottom.)



- Push all the 0's to the stack
- For each 1, pop a 0 from the stack.

- If the stack is empty (i.e. only $ is left), accept.

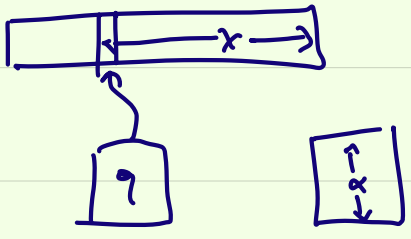
Note:
$\varepsilon \to a$ ... push
$a \to \varepsilon$ ... pop

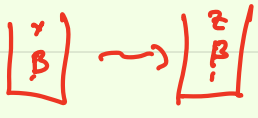


$X \to Y$ ... $X \to \varepsilon$, $\varepsilon \to Y$

Definition: A **configuration** of a PDA m is a triple $(q, x, \alpha) \in Q \times \Sigma^* \times \Gamma^*$

where q is a state, x is the unused portion of the input and α is current stack content.





- For a configuration $(q, ay, X\beta)$ if $(q', z) \in \delta(q, a, X)$, we write

$$(q, ay, X\beta) \vdash_m (q', y, z\beta)$$

current config.

the next configuration.

- If $(q_1, y_1, \alpha_1) \vdash_m \cdots \vdash_m (q_n, y_n, \alpha_n)$,

we write $(q_1, y_1, \alpha_1) \vdash_m^* (q_n, y_n, \alpha_n)$.

- A string $w \in \Sigma^*$ is accepted by PDA M if there is a computation

$$(q_0, w, \varepsilon) \vdash_m^* (q_f, \varepsilon, \alpha)$$

(note: $q_f \in F$, $\alpha \in \Gamma^*$)

- The language of a PDA M is

$$L(M) = \{ w \in \Sigma^* \mid w \text{ is accepted by } M \}$$

Lecture 20

Designing PDAs

Goal: Given a language L, construct a PDA M s.t. $L = L(M)$

Two design ideas:

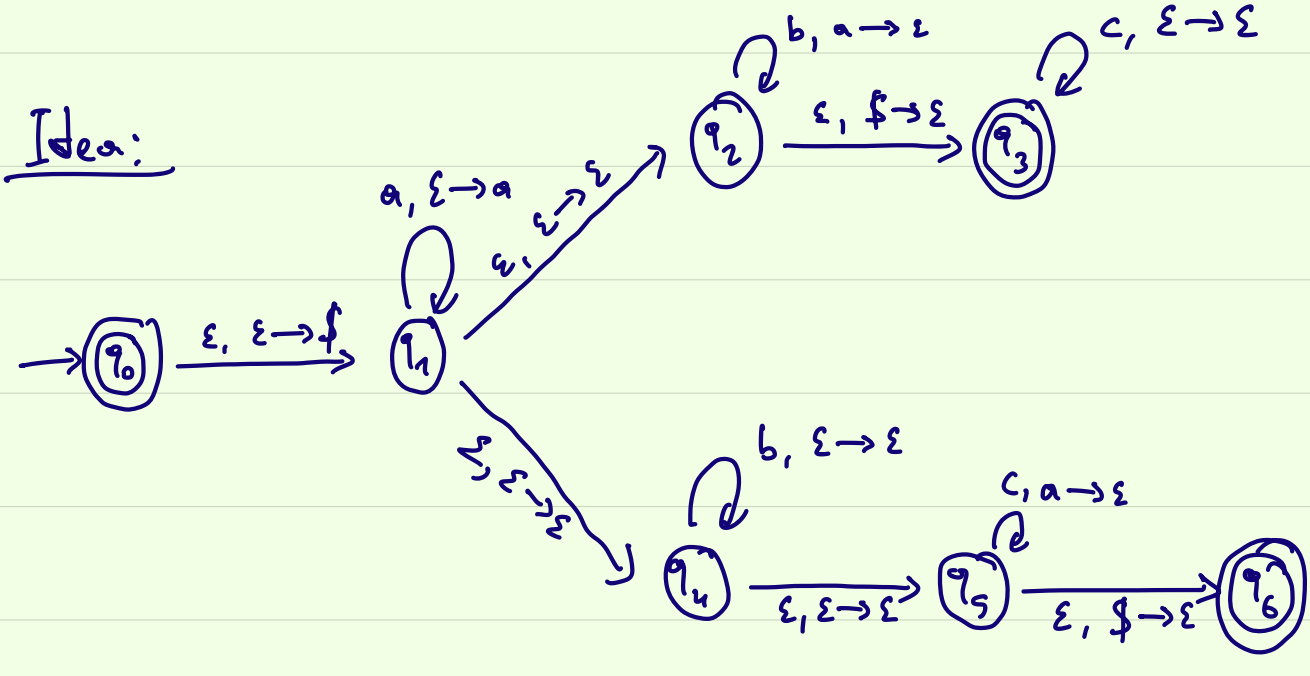
1) Come up with a strategy to use the stack.
2) Exploit nondeterminism.

Example: A PDA M to accept

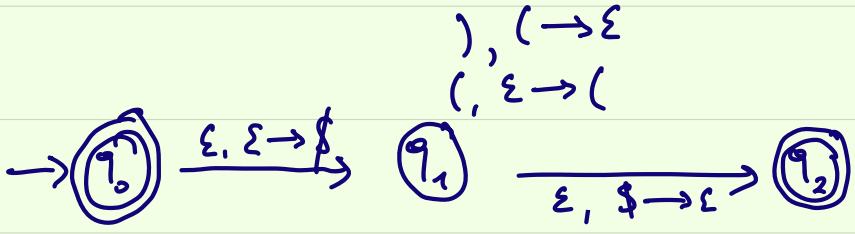
$$L = \{ a^i b^j c^k \mid i = j \text{ or } i = k \}$$

$\underbrace{abccc}_{\text{L}}$, $\underbrace{\varepsilon}_{\text{V}}$, $\underbrace{a^2bc^2}_{\text{V}}$, $\underbrace{a^2b^3c^3}_{\text{No}}$, b^2, c^2

Idea:



Example: PDA for the language of balanced parentheses. e.g. (()), ()(), ε,)((No)

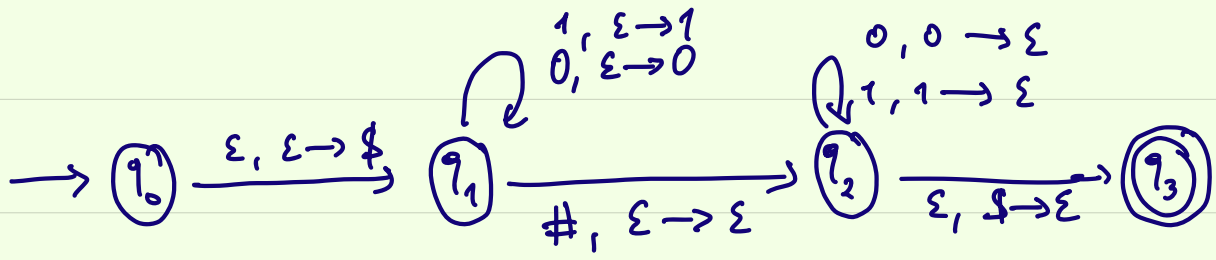


Example: $L = \{ w \# w^R \mid w \in \{0,1\}^* \}$

e.g. 001#100

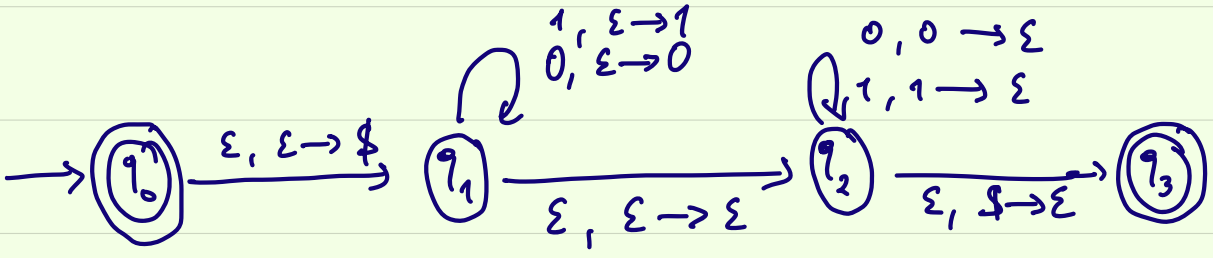
Alg. idea:

- Scan the symbol before # and push each of them.
- Scan # and do nothing to the stack.
- (After # is scanned), if the input symbol and the top stack symbol are identical, pop.
- If the stack is empty at the end, accept.



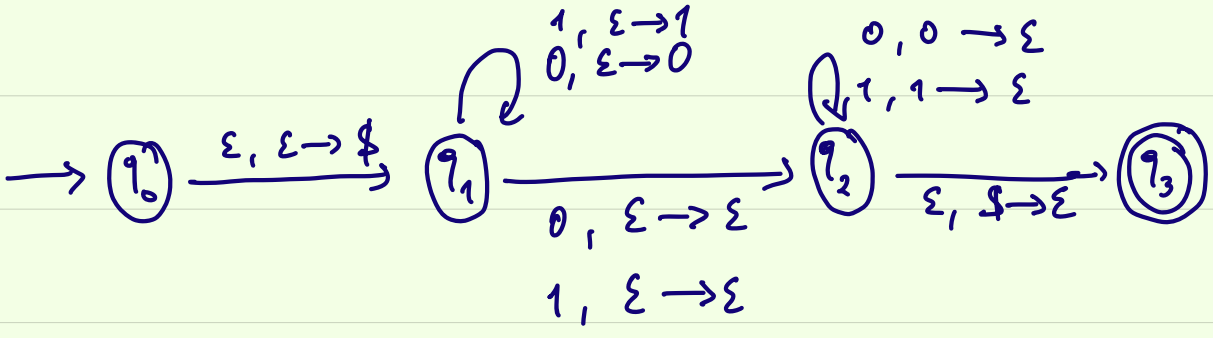
Example: $L = \{ w w^R \mid w \in \{0,1\}^* \}$

e.g. 001100 i.e. even length palindromes



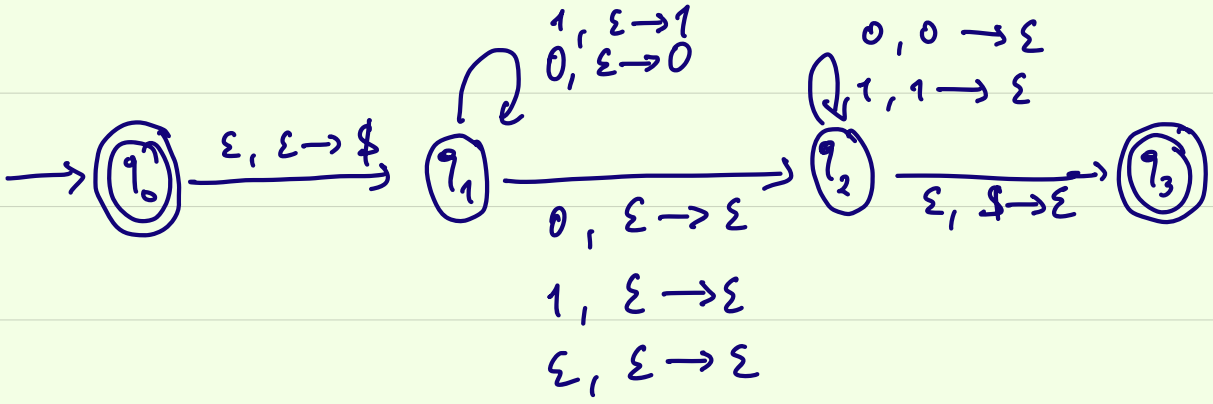
Example: $L = \{ w k w^R \mid w \in \{0,1\}^*, k \in \{0,1\} \}$

e.g. 001100 i.e. odd length palindromes



00100

Example: $L = \{ w \in \{0,1\}^* \mid w = w^R \}$



Lecture 21

Pumping Lemma for CFLs

Let $L \subseteq \Sigma^*$ be a CFL. Then there exists a constant $n > 0$ s.t. for any $z \in L$ with $|z| \geq n$, we can factorize z as $z = u \cdot v \cdot w \cdot x \cdot y$ satisfying the following properties:

1) $|v \cdot w \cdot x| \leq n$
2) $|v \cdot x| \geq 1$
3) $\forall i \geq 0$, $uv^i wx^i y \in L$.

Application of Pumping Lemma

As in the case of regular sets, we perform 5 steps to show that a given a language L is not context-free:

(1) Let $n > 0$ be an arbitrary constant.

(2) Choose a string z s.t $z \in L \wedge |z| \geq n$.

(3) Consider an arbitrary factorization of z:
$z = uvwxy$ with $|vwx| \leq n$ and $|vx| \geq 1$

(4) Find an integer $i_0 \geq 0$ s.t.
$z' = uv^{i_0} wx^{i_0} y \notin L$

(5) Conclude that L is not context-free (CF).

Example: Show that $L = \{0^{k^2} \mid k \geq 0\}$ is not CFL.

1) Let $n > 0$.

2) Let $z = 0^{n^2}$. $z \in L$ and $|z| \geq n$.

3) Let $z = u \cdot v \cdot w \cdot x \cdot y$ s.t. $|v \cdot w \cdot x| \leq n$ and $|vx| \geq 1$.

4) Let $i = 2$. Let $|v \cdot x| = \ell$

Then $z' = u v^2 w \cdot x^2 \cdot y$

$= 0^{n^2 + \ell}$

$n^2 \leq n^2 + \ell \leq n^2 + n < n^2 + n + n + 1 = (n+1)^2$

$\Rightarrow n^2 < n^2 + \ell < (n+1)^2$

$\Rightarrow n^2 + \ell$ is not a perfect square

because it is between two consecutive perfect squares.

$\Rightarrow z' \notin L$.

5) L is not context-free.

Exercise:

- In step 4, try $i = 0$.
- Show that $L = \{0^{k^3} \mid k \geq 0\}$ is not CFL.
- Show that $L = \{0^{k^{2+1}} \mid k \geq 0\}$ is not CFL.

Example: Show that $L = \{0^p \mid p \text{ is a prime}\}$ is not a CFL.

1) Let $n \geq 0$.

2) Let $z = 0^p$ such that p is prime and $p \geq n$.

Then $z \in L$ and $|z| \geq n$.

3) Let $z = u \cdot v \cdot w \cdot x \cdot y$ s.t $|v \cdot w \cdot x| \leq n$ and $|v \cdot x| \geq 1$.

4) Let $i = p+1$.

Then $z' = u \cdot v^{p+1} \cdot w \cdot x^{p+1} \cdot y$

$$\Rightarrow |z'| = |z| + p \cdot |v| + p \cdot |x|$$

$$= p + p|v| + p|x|$$

$$= p(1 + |v| + |x|)$$

$\Rightarrow |z'|$ is not a prime.

$\Rightarrow z' \notin L$.

5) L is not a CFL.

Lecture 22

Pumping Lemma for CFLs cont...

A non-CFL that satisfies the pumping lemma:

Let $L = L_1 \cup L_2$ be a language where

$L_1 = \{ (ab)^{j^2} \mid j \geq 0 \}$ ... note: L_1 isn't CFL.

$L_2 = L\left((a+b)^*(aa+bb)(a+b)^*\right)$

Remark: $L_1 \cap L_2 = \emptyset$.

Fact: If L is a CFL and L' is regular, then $L \cap L'$ is a CFL.

Claim: $L = L_1 \cup L_2$ is not a CFL.

Proof: Suppose the opposite. That is, L is a CFL. Then $L_1 = L \cap L((ab)^*)$ would be a CFL as well. But it is known that L_1 is not a CFL. Contradiction.

Claim: L satisfies the pumping lemma for CFLs.

Proof:

- Let $n = 4$
- Consider $z \in L$ with $|z| \geq 4$.

  $\Rightarrow$ Since $L_1 \cap L_2 = \emptyset$, we get either $z \in L_1$ or $z \in L_2$.

- Case 1: $z \in L_2$.

  W.l.o.g assume $z = z_1 aa z_2$

- Since $|z| \geq 4$, either $z_1 \neq \varepsilon$ or $z_2 \neq \varepsilon$

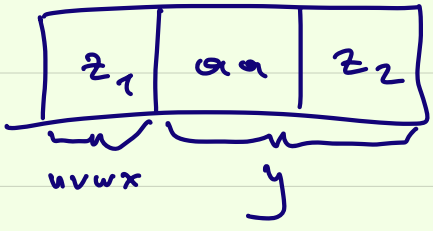
  W.l.o.g $z_1 \neq \varepsilon$.

- Let us factorize z as

  $z = uvwxy$ where

  $z_1 = uvwx$ and $aa z_2 = y$

  i.e.



- It is clear that,

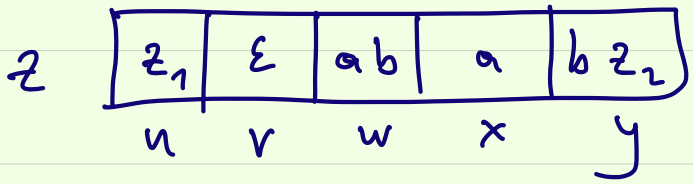
  $\forall i \geq 0$, $z' = uv^i w x^i y \in L_2$

  $\Rightarrow z' \in L$.

- Case 2: $z \in L_1$

Since $|z| \geq 4$, let us take $z = z_1 abab z_2$

- Factorize $z = uvwxy$ where



- Let $i = 0$. Then $z' = uv^0wx^0y$

$= z_1 abb z_2 \in L_2$

$\Rightarrow z' \in L$.

- Let $i \geq 2$. Then $z' = uv^iwx^iy$

$= z_1 ab a^i y$

$\Rightarrow z'$ includes aa as a substring

$\Rightarrow z' \in L_2$

$\Rightarrow z' \in L$.

Closure Properties of CFLs

Claim: CFLs are closed under $\cup$, $\cdot$ and *.

Proof: Let L_1, L_2 be CFLs with CFGs $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$.

Case 1: $L_1 \cup L_2$

To construct a CFG G for $L_1 \cup L_2$,

i) Add a new nonterminal S

ii) Add a new rule $S \to S_1 \mid S_2$

iii) Take the union of components of G_1 and G_2

Therefore $G = (V_1 \cup V_2 \cup \{S\}, \Sigma, R_1 \cup R_2 \cup \{S \to S_1 \mid S_2\}, S)$

Case 2: $L_1 \cdot L_2$

To construct a CFG G for $L_1 \cdot L_2$,

i) Add a new nonterminal S

ii) Add a new rule $S \to S_1 \cdot S_2$

iii) Take the union of components of G_1 and G_2

Therefore $G = (V_1 \cup V_2 \cup \{S\}, \Sigma, R_1 \cup R_2 \cup \{S \to S_1 \cdot S_2\}, S)$

Case 3: L_1^*

To construct a CFG G for L_1^*

i) Add a new nonterminal S

ii) Add two new rules $S \to S_1 \cdot S \mid \varepsilon$

Therefore $G = (V_1 \cup V_2 \cup \{S\}, \Sigma, R_1 \cup R_2 \cup \{S \to S_1 S \mid \varepsilon\}, S)$

Claim: CFLs are not closed under intersection.

Proof: by counter example

Let $L_1 = \{a^i b^i c^j \mid i, j \geq 0\}$.

$\Rightarrow$ L_1 is a CFL:

$S \to AB$ CFG for L_1

$A \to aAb \mid \varepsilon$

$B \to cB \mid \varepsilon$

Let $L_2 = \{ a^i b^j c^j \mid i,j \geq 0 \}$

$S' \to AB$
$A \to aA \mid \varepsilon$
$B \to bBc \mid \varepsilon$

CFG for L_2

However, $L = L_1 \cap L_2 = \{ a^i b^i c^i \mid i \geq 0 \}$ is not a CFL.

Claim: CFLs are not closed under complement.

Proof: Suppose the opposite. Then CFLs would be closed under intersection as well.

Note: $L_1 \cap L_2 = \overline{\overline{L_1 \cap L_2}}$
$= \overline{\overline{L_1} \cup \overline{L_2}}$

Recall:
$\overline{\overline{A}} = A$
$\overline{A \cap B} = \overline{A} \cup \overline{B}$

Claim: CFLs are closed under intersection with regular languages.

Proof: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA and $m' = (Q', \Sigma, \delta', q_0', F')$

be a DFA.

$L(M)$ ... CFL

$L(M')$ ... Regular

To construct a PDA $\hat{M}$ for $L(M) \cap L(M')$ we use M and M':

$\hat{M} = (\hat{Q}, \Sigma, \Gamma, \hat{\delta}, \hat{q}_0, \hat{F})$ where

- $\hat{Q} = Q \times Q'$
- $\hat{q}_0 = (q_0, q_1)$
- $\hat{F} = F \times F'$
- Given a state $(p, p') \in \hat{Q}$ $\left(\text{i.e } \begin{matrix} p \in Q \\ p' \in Q' \end{matrix}\right)$

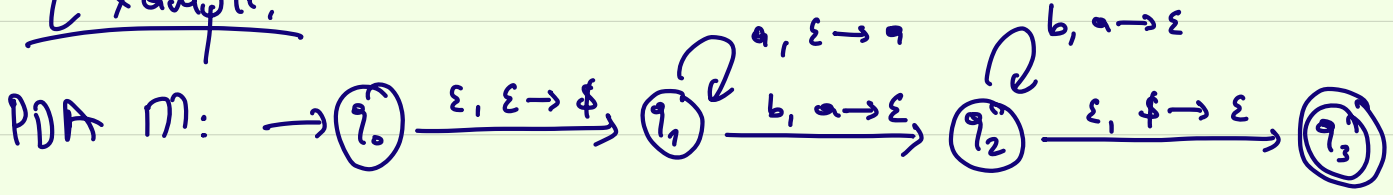
  $\hat{\delta}$ should simulate M on the state p and should simulate M' on the state p'.

i.e.

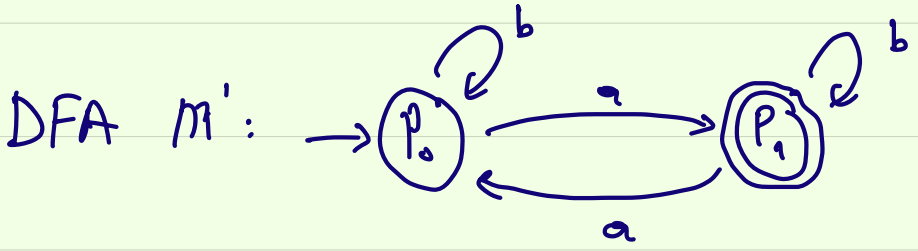
$$\hat{\delta}((p,p'), a, X) = \left\{ ((q,q'), \gamma) \;\middle|\; \begin{matrix} (q, \gamma) \in \delta(p, a, X) \\ \text{and } q' = \delta'(p', a) \end{matrix} \right\}$$

Example:

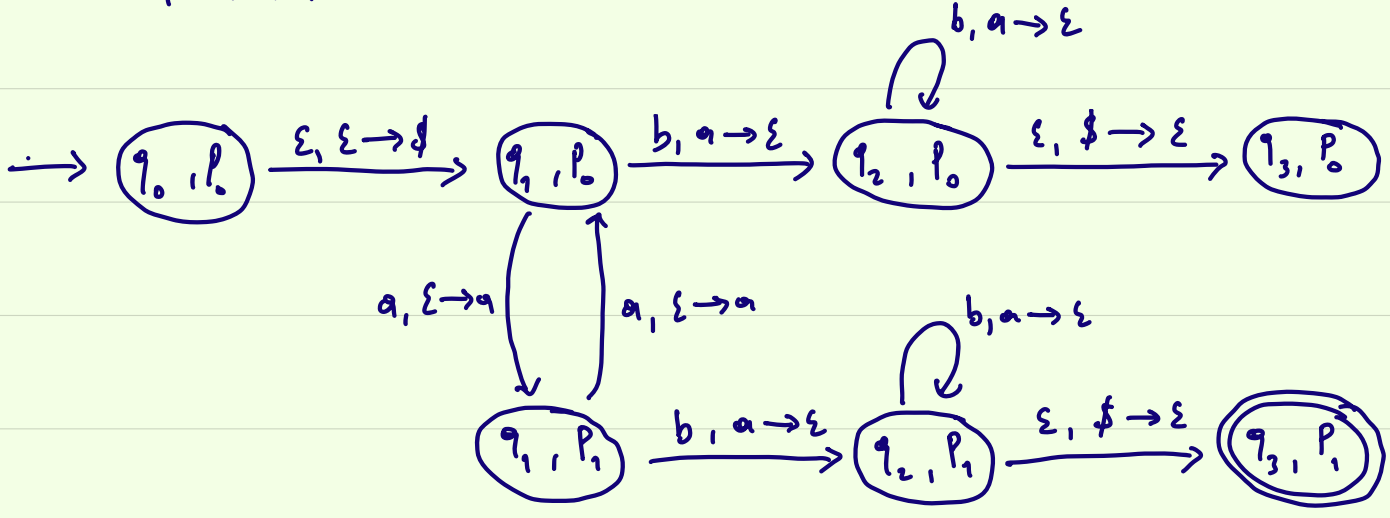
PDA M:



DFA M':



$M \times M'$:



Lecture 23

Decision Problems for CFLs

Emptiness Problem:

Input: A CFG G.

Decide: Is $L(G) = \emptyset$?

Claim: The emptiness problem is decidable.

Proof: Let $G = (V, \Sigma, R, S)$ be a CFG.

Note that $L(G) \neq \emptyset$ iff S is productive.

Therefore, to decide if $L(G) \neq \emptyset$, compute the set of productive nonterminals (see Lec. 16) and check if S is in the output.

Finiteness Problem:

Input: A CFG G

Decide: If $L(G)$ is finite?

Claim: The finiteness problem is decidable.

Proof: Let $G = (V, \Sigma, R, S)$ be a CFG.

It is known that $L(G)$ is infinite iff there exists a nonterminal $A \in V$ s.t. $A \Rightarrow^* \alpha A \beta$ for some $\alpha, \beta \in (\Sigma \cup V)^*$.

e.g. $G: S \to aSb \mid \varepsilon$, $L(G)$ is infinite.

For each $A \in V$, define a set

$R(A) = \{ B \in V \mid A \Rightarrow^* \alpha B \gamma$ for some $\alpha, \gamma \in (\Sigma \cup V)^* \}$.

To compute $R(A)$:

- $R_0(A) = \{ B \in V \mid \exists$ a rule $A \to \alpha B \gamma$ for some $\alpha, \gamma \in (\Sigma \cup V)^* \}$
- Set $i = 0$
- Repeat:

  $i = i + 1$

  $R_i(A) = R_{i-1}(A) \cup \{ B \in V \mid \exists C \in R_{i-1}(A)$ and a rule $C \to \alpha B \gamma$ for some $\alpha, \gamma \in (\Sigma \cup V)^* \}$

  until $R_i(A) = R_{i-1}(A)$
- Return $R_i(A)$.

To verify if $L(G)$ is infinite, compute $R(A)$ for each $A \in V$ and check if $A \in R(A)$.

Membership Problem:

Input: A CFL $L \subseteq \Sigma^*$, a string $w \in \Sigma^*$.

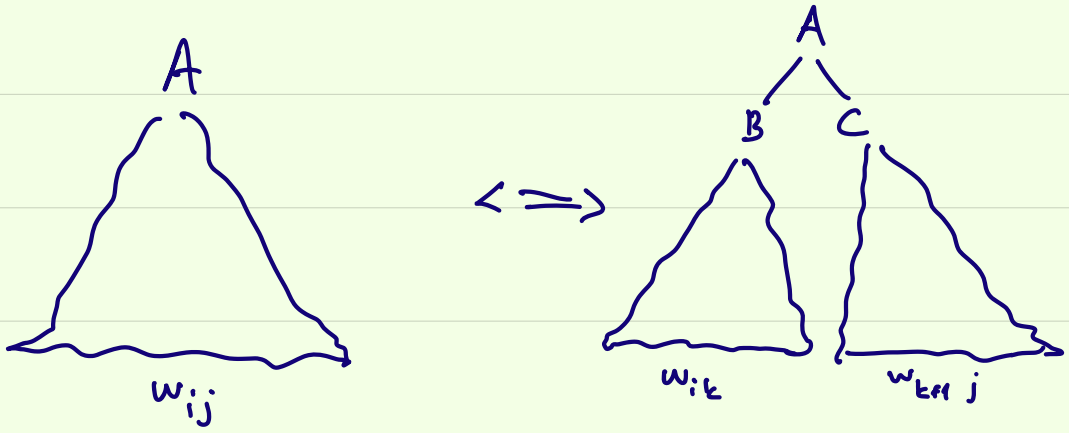
Decide: Is $w \in L$?

$A \to a$
$A \to BC$

The CYK Algorithm:

- Let $G = (V, \Sigma, R, S)$ be a CFG in CNF.
  Assume $L = L(G)$.
- Let $|w| = n$ and $w = w_1 w_2 \dots w_n$
  $(w_i \in \Sigma, \ i = 1, \dots, n)$
- Let $w_{ij} = w_i \dots w_j$, $1 \le i, j \le n$
  Note: $w = w_{1n}$

- $w \in L \iff w \in L(G) \iff S \Rightarrow^* w \iff S \Rightarrow^* w_{1n}$.
- Q: How to check in general if $A \Rightarrow^* w_{ij}$?



$(i \le k < j)$

$A \Rightarrow^* w_{ij}$ iff $\exists k (i \le k < j)$ $\exists A \to BC$ s.t.

$B \Rightarrow^* w_{ik}$ and $C \Rightarrow^* w_{k+1\,j}$

Define $V_{ij} = \{ A \in V \mid A \Rightarrow^* w_{ij} \}$

Then, we get

$A \in V_{ij}$ iff $\exists k (i \le k < j)$ $\exists A \to BC$ s.t.

$B \in V_{ik}$ and $C \in V_{k+1\,j}$.

The CYK algorithm:

Input: $w = w_1 \dots w_n$

Output: The sets V_{ij}, $1 \le i \le j \le n$ stored in a 2D table

```
for i := 1 to n do
    V_ii = { A ∈ V | ∃ A → w_i ∈ R }
for l := 2 to n do
    for i := 1 to n - l + 1 do
        j := i + l - 1;
        V_ij := ∅
        for k := i to j - 1 do
            V_ij = V_ij ∪ { A ∈ V | ∃ A → BC ∈ R : B ∈ V_ik ∧ C ∈ V_{k+1,j} }.
end
```

Running Time: $O(n^3)$.

Lecture 24

TURING MACHINES

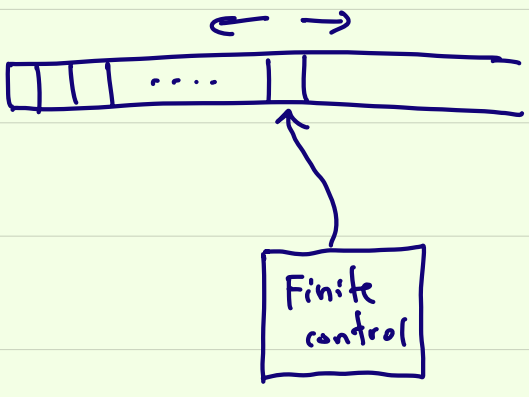
A Turing Machine (TM) M is a 7-tuple

$M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ where

- Q is the set of states
- Σ is the input alphabet
- Γ is the tape alphabet (includes Σ and extra symbol(s))

  e.g. $\Gamma = \Sigma \cup \{\sqcup\}$ → blank symbol.
- δ is the transition function

  $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$
- q_0 is the initial state
- q_a is the accept state
- q_r is the reject state

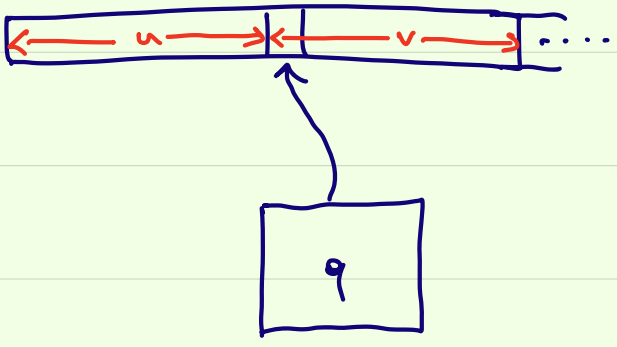


- read-write tape
- Left or right moves

A configuration of TM M is a string of the form $uqv \in \Gamma^* Q \Gamma^*$ $(u, v \in \Gamma^*, q \in Q)$

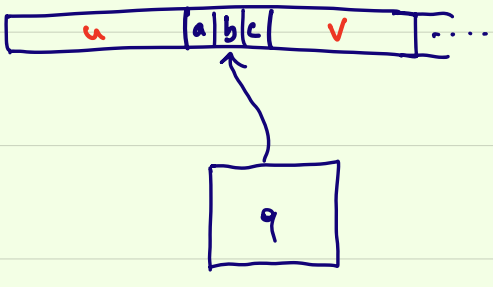
where

- q is the current state
- u is the portion of the tape to the left of the tape head
- v is the ↑ nonblank portion of the tape to the right of the tape head, including the symbol being scanned.

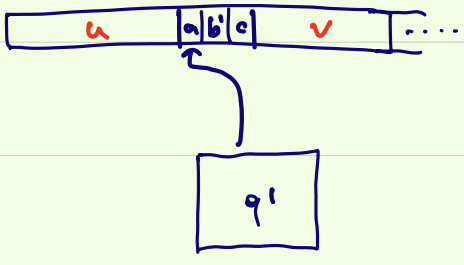


uqv

Left move: $\delta(q, b) = (q', b', L)$

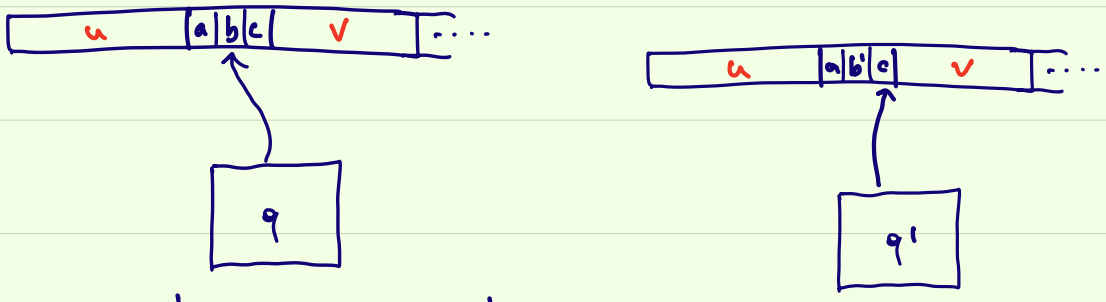


$uaqbcv$ $\vdash_M$



$uq'ab'cv$

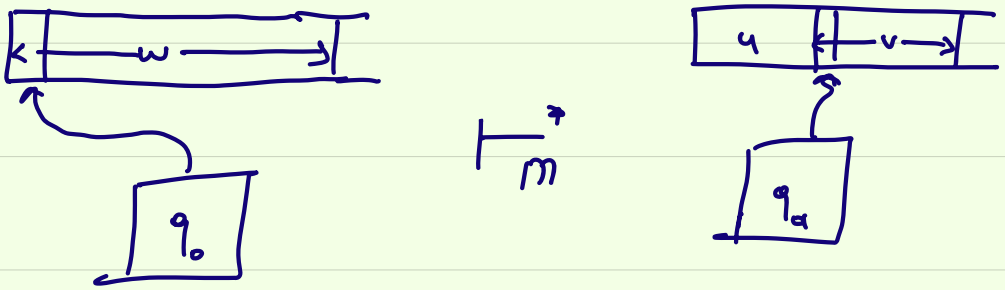
Right move: $\delta(q, b) = (q', b', R)$



If $C_0 \vdash_m C_1 \vdash_m C_2 \vdash_m \cdots \vdash_m C_k$ where $C_0, \ldots, C_k$ are configurations of a TM m, then we write $C_0 \vdash_m^* C_k$.

An input string $w \in \Sigma^*$ is <u>accepted</u> by a TM m, if there is a computation $q_0 w \vdash_m^* u q_a v$ for some $u, q \in \Gamma^*$.

i.e.



The language of a TM M is

$$L(M) = \{ w \in \Sigma^* \mid q_0 w \vdash_M^* u q_a v \}$$

$$= \{ w \in \Sigma^* \mid M \text{ accepts } w \}.$$

A language $L \subseteq \Sigma^*$ is Turing-recognizable if $L = L(M)$ for some TM M.

Remark: On an input w, a TM M may have three possible outcomes:

1) It enters q_a; halts and accepts w.
2) It enter q_r; halts and rejects w.
3) It does not enter q_a or q_r; it does not halt.

A TM M that always halts on its inputs is called a decider.

A language $L \subseteq \Sigma^*$ is Turing-decidable (or decidable) if $L = L(M)$ for some decider M.

Note:

- Every decider is a TM, but not vice versa.
- Every Turing-decidable language is Turing-recognizable, but not vice versa.



- Let $L \subseteq \Sigma^*$ be Turing recognizable:
  - There exist a TM M s.t. $L = L(M)$.
  - If $w \in L$, then we know that M accepts w.
  - If $w \notin L$, then we don't know whether M rejects w. It might loop.

- Let $L \subseteq \Sigma^*$ be Turing-decidable:
  - There exist a TM M s.t. $L = L(M)$.
  - If $w \in L$, then we know that M accepts w

- If $w \notin L$, then we know that TM rejects w.

- Let $L \subseteq \Sigma^*$ be a non-Turing-recognizable:
  - There does not exist a TM s.t. $L = L(M)$.
  - If $w \in L$, we don't know if if is accepted
  - If $w \notin L$, " " " if it is rejected.

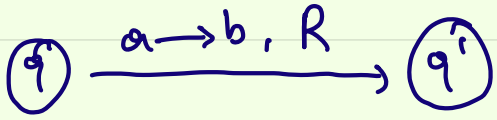
Church-Turing Thesis

class of computable functions $\simeq$ class Turing machine.

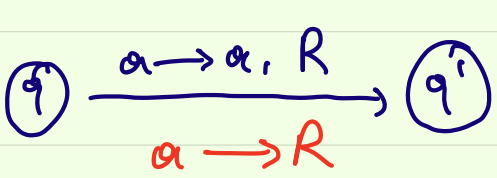
Lecture 25

Turing Machines ... cont.

Notation:



$\sim \quad \delta(q, a) = (q', b, R)$

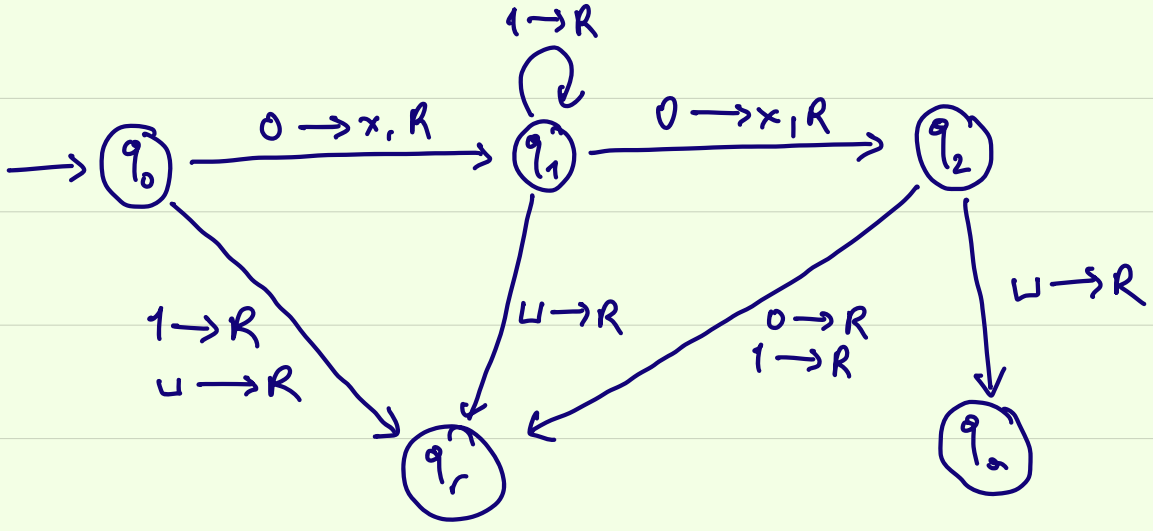


$\sim \quad \delta(q, a) = (q', a, R)$

Similarly, for left moves.

Example: A TM for $L = L(01^*0)$

$\Sigma = \{0, 1\} \quad \Gamma = \{0, 1, \sqcup\}$



Note: This is indeed a decider for L

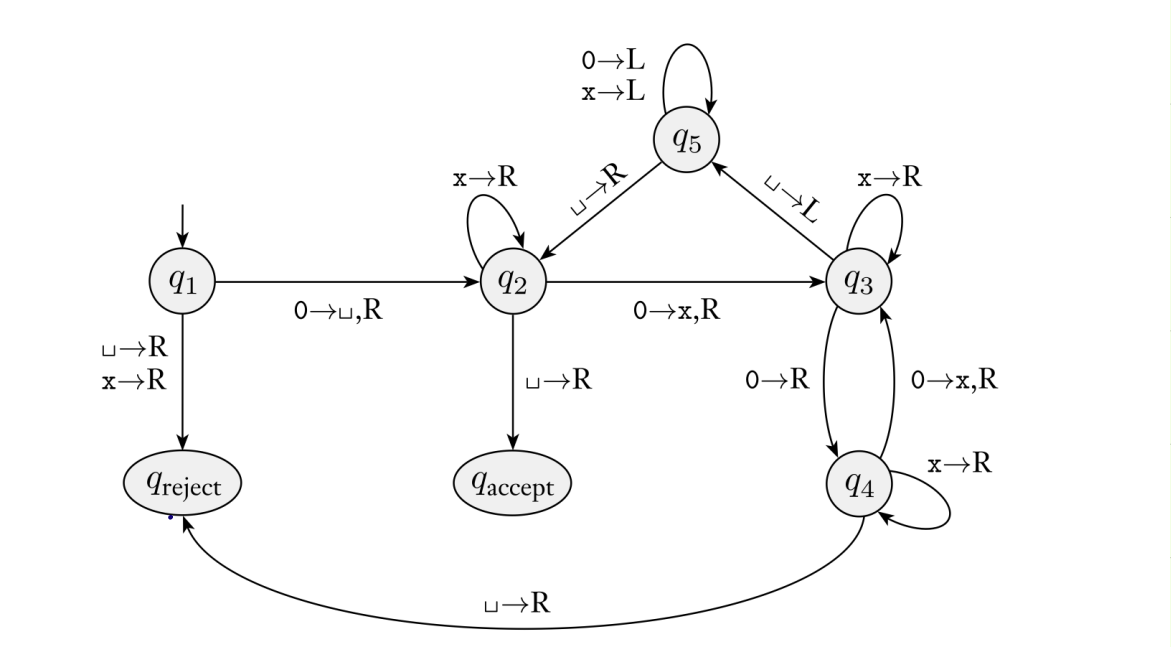
Example: A TM for $L = \{0^{2^n} \mid n \geq 0\}$

e.g. $0, 0^2, 0^4, 0^8 \ldots$

$0^3, 0^5, 0^6 \ldots$

M_2 = "On input string w:

1. Sweep left to right across the tape, crossing off every other 0.
2. If in stage 1 the tape contained a single 0, *accept*.
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, *reject*.
4. Return the head to the left-hand end of the tape.
5. Go to stage 1."



0 ✓ — ␣

00 ✓ — ␣x

0000 ✓ — ␣x0x

00000␣ ✗ — ␣x0x0

000000 ✗ — ␣x0x0x␣

␣xxx0x␣

Exercise: Run on input $w = 0^8$.

Run on input $w = 0^9$.

Decidable Languages

Decision problems can be formulated as languages over an alphabet.

Example: Consider the equivalence problem for DFAs.

Input: Two DFAs M_1 and M_2

Decide: Is $L(M_1) = L(M_2)$?

Let $\langle M_1, M_2 \rangle$ be a string over an alphabet Σ and describe the DFAs M_1 and M_2

$$M = (Q, \Sigma, \delta, q_0, F) \xrightarrow[\cdots]{\text{Encoding}} \langle M \rangle$$

Now we formulate the problem as a language

$$EQ_{DFA} = \{ \langle M_1, M_2 \rangle \mid \langle M_1, M_2 \rangle \text{ is an encoding of } M_1 \text{ and } M_2 \text{ and } L(M_1) = L(M_2) \}$$

Fact: EQ_{DFA} is Turing-decidable. (See Thm 4.5)

To solve the equivalence problem using a decider for EQ_{DFA}:

On an input M_1 and M_2

1) Convert M_1 and M_2 into a string

$w = \langle M_1, M_2 \rangle$

2) Input w to a decider M of EQ_{DFA}

3) If M accepts w, return yes.

4) If M rejects w, return no.

Undecidable Languages

Let M be a TM and w a string.

We can encode M and w as a string $\langle M, w \rangle$

Define a language (problem)

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ halts and accepts } w \}$$

a version of halting problem.

Theorem: A_{TM} is Turing-recognizable.

Proof: Construct a TM M' s.t. $L(M') = A_{TM}$.

M' on input $\langle M, w \rangle$ where $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$:

1) Simulate M on w.

2) If M accepts w, then halt and accept.

Note that $\langle M, w \rangle \in L(M')$ iff M accepts w iff $\langle M, w \rangle \in A_{TM}$.

$\Rightarrow A_{TM} = L(M')$

1. Convert the following grammar to Chomsky normal form.

$S \to ASA \mid A \mid \varepsilon$
$A \to aa \mid \varepsilon$

1) Elim. ε-transition

Nullable: S, A

$S \to ASA \mid AS \mid AA \mid SA \mid A$

$A \to aa$

2) Elim. chain productions

$S \to ASA \mid AS \mid AA \mid SA \mid aa$

$A \to aa$

3) Elim. useless symbols:

No change is required.

4) Convert to CNF

Step 1:

$S \to ASA \mid AS \mid AA \mid SA \mid D_a D_a$
$A \to D_a D_a$
$D_a \to a$

Step 2:

$S \to AE_1 \quad E_1 \to SA$
$S \to AS \mid AA \mid SA \mid D_a D_a$
$A \to D_a D_a$
$D_a \to a$

2. Show that the language $K = \{\langle M \rangle \mid M \text{ is a TM and does not accept } \langle M \rangle\}$ is **not Turing-recognizable**. Note: Prove by contradiction.

Proof: by contradiction.

- Suppose K is Turing-recognizable.

$\Rightarrow \exists$ a TM M_K s.t. $L(M_K) = K$.

- Consider M_K on input $\langle M_K \rangle$

- M_K accepts $\langle M_K \rangle$:

Then $\langle M_K \rangle \in K$.

However, by def of K, M_K doesn't accept $\langle M_K \rangle$. Contradiction.

- M_K doesn't accept $\langle M_K \rangle$:

$\langle M_K \rangle \in K$.

$\Rightarrow \langle M_K \rangle \in L(M_K)$ because

$L(M_K) = K$.

$\Rightarrow M_K$ accepts $\langle M_K \rangle$.

Contradiction.

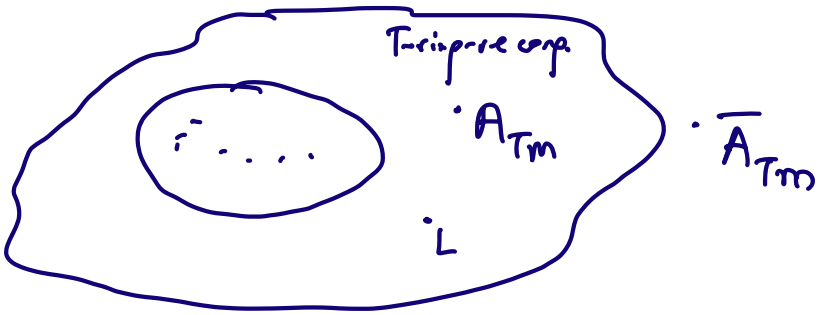
$\Rightarrow M_K$ doesn't exist.

$\Rightarrow K$ is not Turing-recognizable.

Cont.

3. Prove that if A_{TM} were decidable, then every Turing-recognizable language would be decidable.

$A_{TM} = \{\langle M, w\rangle \mid M \text{ halts and accepts } w\}$



Proof:

- Suppose A_{TM} is decidable.
  $\Rightarrow \exists$ a TM O that decides A_{TM}.

- Let L be a Turing-recognizable language.
  $\Rightarrow \exists$ a TM M_L s.t. $L(M_L) = L$.

- Construct a decider M' for L:

  M' on w:
  - Run O on $\langle M_L, w\rangle$
  - If O accepts (which happens iff M_L accepts w)
    then accept
    else reject.

- M' accepts w if $w \in L(M_L) = L$
- M' rejects w if $w \notin L(M_L) = L$

$\Rightarrow L(M') = L \Rightarrow$ M' decides L.

Cont.

4. Let $\Sigma = \{a\}$ be a unary alphabet. Explain that there exists a language $L \subseteq \Sigma^*$ that is undecidable.

Σ^* : set of all strings over Σ

$2^{|\Sigma^*|}$: set of all lang. over Σ.

To prove: $\exists\ L \subseteq \Sigma^*$ s.t. L is undecidable.

- Let A be the set of decidable languages over Σ.
  - A is infinite and countable.
- $2^{|\Sigma^*|}$ is infinite and uncountable.

$\Longrightarrow A \subset 2^{|\Sigma^*|}$

$\Longrightarrow \exists$ an undecidable lang. over Σ.

The End.

Lecture 26

Undecidable Languages... cont.

Recall:

Let M be a TM and w a string.

We can encode M and w as a string $\langle M, w \rangle$

Define a language (problem)

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ halts and accepts } w \}$$

Theorem: A_{TM} is undecidable.

Proof: by contradiction and using a technique called diagonalization.

- Assume: A_{TM} is decidable.
- Then there must exists a decider H for A_{TM}.

$$H(\langle M, w \rangle) = \begin{cases} \text{Accept} & \text{if } M \text{ accepts } w. \\ \text{Reject} & \text{if } M \text{ rejects } w. \end{cases}$$

- Now let's define a new decider D using H:

D on input $\langle M \rangle$ where M is a TM:

1) Simulate H on $\langle M, \langle M \rangle \rangle$
2) Accept if H rejects
   Reject if H accepts

i.e.

$$D(\langle M \rangle) = \begin{cases} \text{Accept} & \text{if } M \text{ rejects } \langle M \rangle \\ \text{Reject} & \text{if } M \text{ accepts } \langle M \rangle \end{cases}$$

Q: What does D return on $\langle D \rangle$?

$$D(\langle D \rangle) = \begin{cases} \text{Accept} & \text{if } D \text{ rejects } \langle D \rangle \text{ ... contradiction} \\ \text{Reject} & \text{if } D \text{ accepts } \langle D \rangle \text{ .. Contradiction} \end{cases}$$

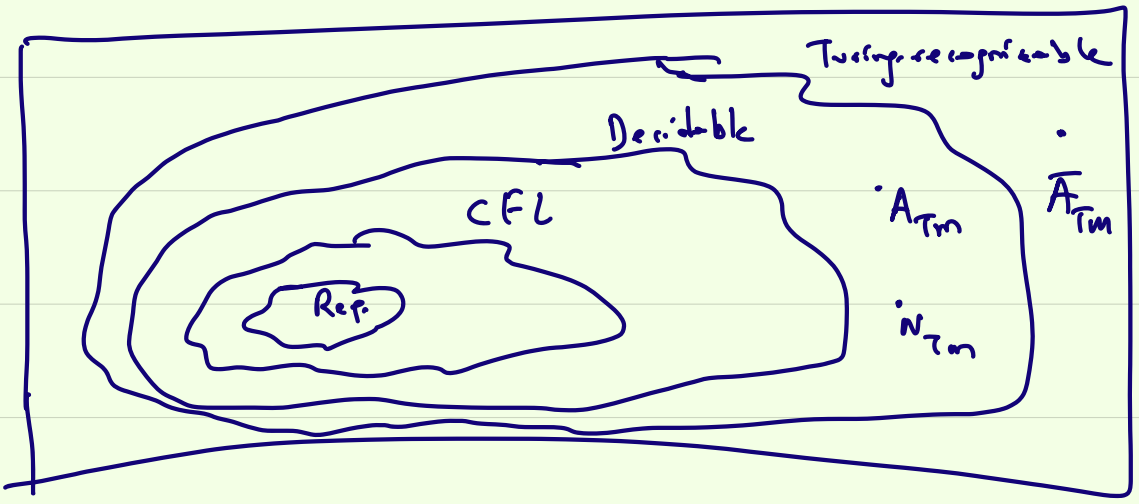
$\Rightarrow$ D does not halt on $\langle D \rangle$

$\Rightarrow$ D is not a decider

$\Rightarrow$ H is not a decider, either.

$\Rightarrow$ There exists no decider for A_{TM}.

Contradiction. ●



Theorem: If both L and $\bar{L}$ are Turing-recognizable, then both L and $\bar{L}$ are decidable.

Proof: Assume L and $\bar{L}$ are Turing-recognizable

$\Rightarrow$ There exist TMs M_1 and M_2 s.t.

$L = L(M_1)$ and $\bar{L} = L(M_2)$

Now we construct a decider M' that decides L:

M' on input x:

- Simulate M_1 and M_2 on x in parallel
- If M_1 accepts, return accept.
- If M_2 accepts, return reject.

(Recall: $L \cap \bar{L} = \emptyset$. So either $x \in L$ or $x \in \bar{L}$ is true)

Similarly define M'' that decides $\bar{L}$:

M'' on input x:
- Simulate M_1 and M_2 on x in parallel
- If M_1 accepts, return reject
- If M_2 accepts, return accept.

$\Longrightarrow$ Both L and $\bar{L}$ are decidable. ■

Corollary: $\bar{A}_{TM}$ is not Turing-recognizable.

Proof: If $\bar{A}_{TM}$ were Turing-recognizable, then both A_{TM} and $\bar{A}_{TM}$ would be decidable. This contradicts the fact that A_{TM} is undecidable

Reduction

Example: Consider the problem of nonemptiness for TMs:

$$NE_{TM} = \{ \langle M \rangle \mid L(M) \neq \emptyset \}$$

Theorem: NE_{TM} is undecidable.

Proof: by reduction from A_{TM}.

We take an instance $\langle M, w \rangle$ of A_{TM} and convert it into an instance $\langle M' \rangle$ of NE_{TM}:

M' on input x:

1) Simulate M on w.
2) If M accepts w, then accept.

Now we need to prove the following:

"$\langle M, w \rangle \in A_{TM}$ iff $\langle M' \rangle \in NE_{TM}$"

i) If $\langle M, w \rangle \in A_{TM}$, then M accept w.

This implies that M' accepts any input x.

Therefore $L(M') = \Sigma^* \neq \emptyset$. Hence $\langle M' \rangle \in N_{TM}$.

ii) To prove "if $\langle M' \rangle \in NE_{TM}$, then $\langle M, w \rangle \in A_{TM}$"

we take its contrapositive:

Assume $\langle M, w \rangle \notin A_{TM}$. Then M does not accept w which in turn implies that M' does not accept any input x.

Therefore $L(M') = \emptyset$ and we get $\langle M' \rangle \notin NE_{TM}$.

To decide A_{TM}:

- Construct $\langle M' \rangle$ from $\langle M, w \rangle$
- Input $\langle M' \rangle$ into a decider of NE_{TM}.
- Return accept if $\langle M' \rangle$ is accepted.
  Reject otherwise.

The above is a decider for A_{TM}. However it is known that there exists no decider for A_{TM}. Hence, there is no decider for NE_{TM} either