XI

Regular languages

- Regular languages are the class of languages recognized by finite automata.

Regular languages can be denoted by regular expressions.

Regular Expressions // A recursive (or inductive) definition

- 1) $\emptyset$, ϵ are regular expressions
- 2) a is regular, $\forall a \in \Sigma$
- 3) Given that α, β are regular expressions, then so are:

  a) $\alpha + \beta$ // sometimes written as $\alpha \mid \beta$, $\alpha \vee \beta$, α or β

  b) $\alpha \circ \beta$

  c) α^*, β^*

- 4) Furthermore, any expression which is regular may be obtained in a finite number of applications of 1) → 3)

Regular Sets

examples

Regular expressions	Regular sets
00	{00}
(0+1)*	{ϵ, 0, 1, 00, 01, 10, 11, 000 ...}
(0+1)*00(0+1)*	{00, 1001, 0001, 1000, 1100 ...} "Any binary string containing '00' as a subword"
(0+1)*011	All binary strings ending with '011'
0*1*2*	{012, 222, 01, ...}

not a regular expression

$0^n 1^n 2^n$ denotes $L = \{012, 001122\}$

Regular expression for the set of all strings over {b, c} with an even number of b's?

(bb)* // no ... why not?

(bb)* // no " " ?

c*(bb)*c* // no " " ?

c*(bc*b)*c* // well? ??

Regular expressions and fa

- Regular expressions $\cong$ fa

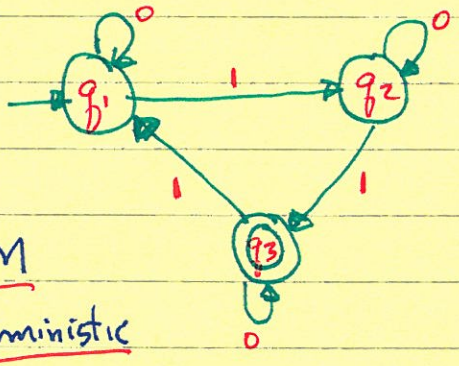
What does this mean?

And how would you prove this?

- Part I of proof

fa → regular expression

- wlog we let M be deterministic



$0^*10^*10^*(10^*10^*10^*)^*$

this was ad hoc

An algorithm is required

$R^0_{11} = \epsilon + 0$

$R^0_{12} = 1$

Dynamic Programming Approach

Given a problem of size n, solve all subproblems

Inductive step: $R_{ij}^{k} = R_{ik}^{k-1}\left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1} + R_{ij}^{k-1}$

$L(M) = R^3_{13} = R^2_{13}\left(R^2_{33}\right)^* R^2_{33} + R^2_{13}$

Notation R_{ij}^{k} — k = index of highest numbered state we may pass through

j is destination state

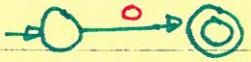
i is state started in

XI

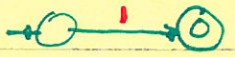
Regular expressions and fa

- Part II of proof (Equivalence of regular expressions and fa)
- Given an arbitrary regular expression E, construct an fa M such that L(M) = L(E).
- $E = (01 + 10)^*$

Construct a machine for 0:



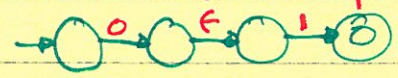
and one for 1:



- Build a '01' machine

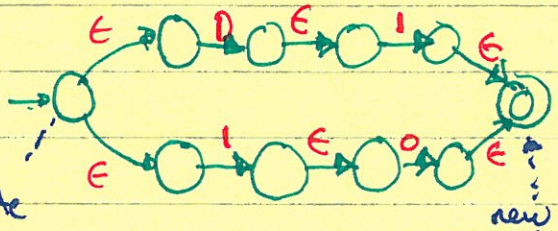
we cancel its accept status

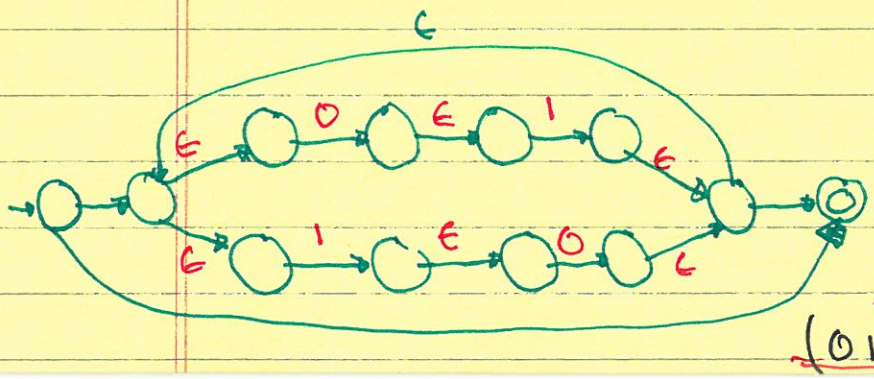
yielding:



- Similarly, construct a '10' machine

- Next, build a '01 + 10' fa



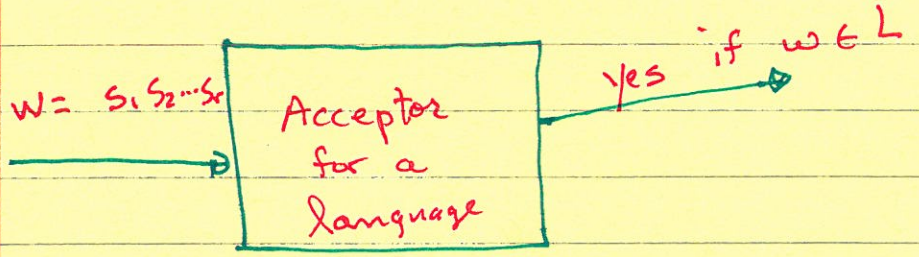


And finally, an fa for $(01 + 10)^*$

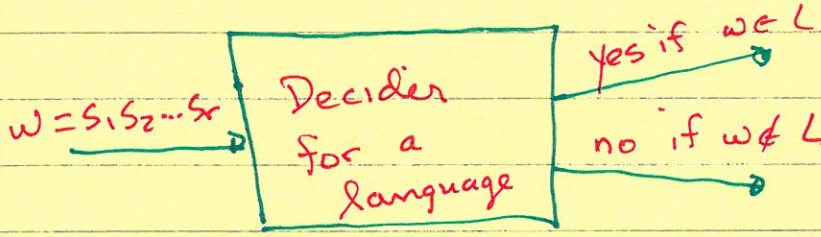
Grammars

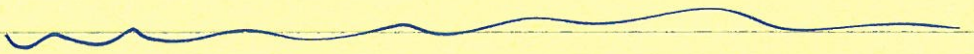
- A grammar is a generative procedure for a language.

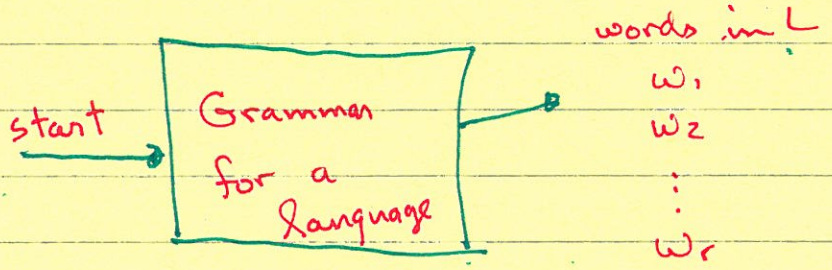
as opposed to an acceptor



and a decider







Grammers - con't.

- A grammar is a four-tuple.

$$G = \langle N, T, S, R \rangle$$

- N is a set of non-terminals - they are not actual words in a language but rather blueprints for words.
- T - set of terminals - the actual words in L.
- S - the start symbol : $S \in N$
- R - set of replacement rules (sometimes denoted as P for production)

An example (from English)

"They are flying planes"

<Sentence> → <Subject><Predicate>

brackets denote these are not actual words in the language.

left hand side (lhs)

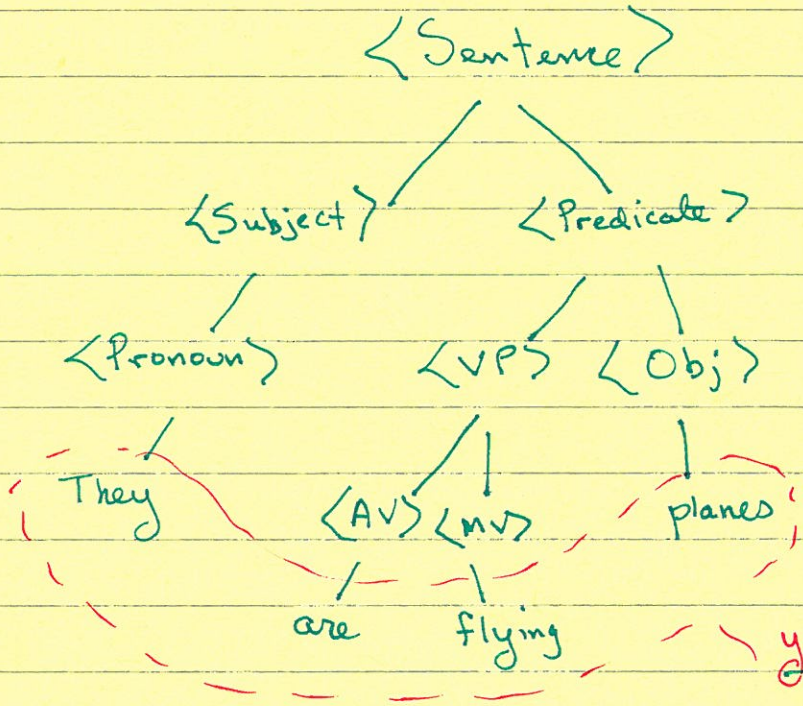
right hand side

Grammars in English

example con't

"They are flying planes"

1) <Sentence> → <Subject><Predicate>
2) <Subject> → <Pronoun>
3) <Pronoun> → They
4) <Predicate> → <Verb Phrase> <Object>
5) <Verb Phrase> → <Auxiliary Verb> <Main Verb>
6) <Auxiliary Verb> → are
7) <Main Verb> → flying
8) <Object> → planes



A parse tree or derivation tree for above sentence.

yield of tree

Indicates the semantics of the sentence (i.e. meaning)

And, what is its meaning?

Can you give an alternate tree??

XI

Grammars in English con't.

- We can also give a derivation for a sentence.

note - a double-edged arrow is using in a derivation

$\langle Sentence \rangle \overset{(1)}{\Rightarrow} \langle Subject \rangle \langle Predicate \rangle$

$\overset{(2)}{\Rightarrow} \langle Pronoun \rangle \langle Predicate \rangle$

$\overset{(3)}{\Rightarrow}$ They $\langle Predicate \rangle$

$\overset{(4)}{\Rightarrow}$ They $\langle VP \rangle \langle Obj \rangle$

$\overset{(5)}{\Rightarrow}$ They $\langle AV \rangle \langle MV \rangle \langle Obj \rangle$

$\overset{(6)}{\Rightarrow}$ They are $\langle MV \rangle \langle Obj \rangle$

$\overset{(7)}{\Rightarrow}$ They are flying $\langle Obj \rangle$

$\overset{(8)}{\Rightarrow}$ They are flying planes.

Notice that whenever a choice was present, we replaced the leftmost non-terminal.

A mixture of terminal and non-terminals is known as a sentential form.

This is known as a leftmost derivation

- If we had always replaced the rightmost non-terminal - we would have a rightmost derivation.

$\langle Sentence \rangle \Rightarrow \langle Subject \rangle \langle Predicate \rangle$

$\Rightarrow \langle Subject \rangle \langle VP \rangle \langle Obj \rangle$

$\Rightarrow \langle Subject \rangle \langle VP \rangle$ planes.

- The beginning of a rightmost derivation.

Grammars

- Corresponding to each parse tree, there may exist many distinct derivations.

- However, corresponding to each parse tree, there is
  - one unique leftmost derivation

  and
  - one unique rightmost derivation.

Regular Grammars

- In a regular grammar all productions are of the form:

$A \rightarrow a$ // right linear

$A \rightarrow aA$

or

$A \rightarrow a$

$A \rightarrow Aa$ // left linear

- In formal grammars (grammars for formal languages)

lower-case letters denote terminals whereas, capital letters stand for non-terminals.

Regular Grammars

examples:

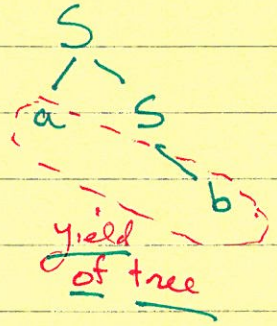
- Construct a regular grammar for $E = a^*b$ s.t. $L(G) = L(E)$.

Grammar: $S \to b$

$S \to aS$

To generate 'b'

To generate 'ab'



- Give a regular grammar for $xy(xx)^*y$

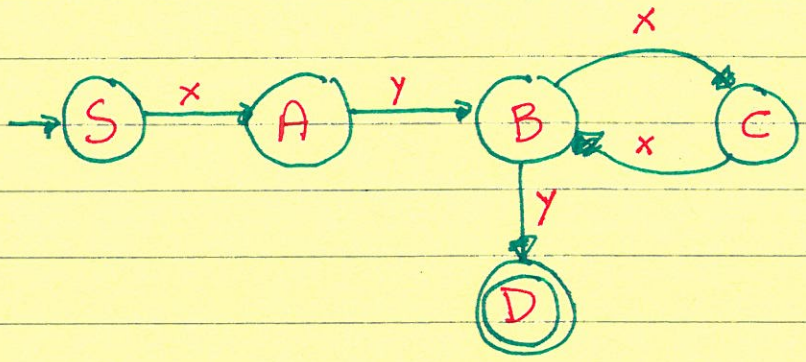
- $L(E) = \{ xyy, xyxxy, xyxxxxy, \ldots \}$

- one approach - • first find an fa^M for expression E
  • then glean the grammar G from M

Regular Grammars con't.

- $E = xy(xx)^* y$

M =



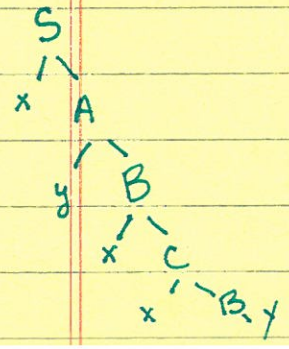
An nfa

- We observe that $L(M) = L(E)$

- A regular grammar G s.t. $L(G) = L(M)$ and hence equals $L(E)$

- G: $S \to xA$
  $A \to yB$
  $B \to xC \mid yD \mid y$ // these rules have the same l.h.s.
  $C \to xB$

- Observe: $xyxxy \in L(E)$



$xyxxy \in L(M)$ // follow path thru M.

$xyxxy \in L(G)$ // consult tree at left.

XI

Regular Grammars con't.

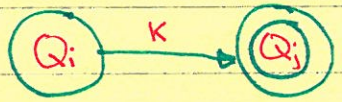
Relationship between fa's and regular rules.

(1) If in an fa M we have:

then the corresponding regular grammar will contain the rule:

$$Q_i \to K Q_j$$

(2)



corresponds to:

$Q_i \to K Q_j$ and $Q_i \to K$

or

$Q_i \to K Q_j$ and $Q_j \to \epsilon$

Characterizations for Regular Sets

- i) Recognizable by fa's
- ii) Denoted by regular expressions
- iii) Generated by regular grammars