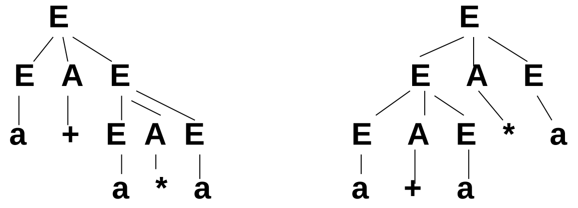
Ambiguity of a Grammar

Ex. a+a*a



Leaves the same: derive same string

*Correspond to a + (a*a) and (a + a) * a*

May yield different answers if evaluated!

Def. G is *ambiguous* **if there is some string w in** Σ^* **with two different parse trees.**

Ambiguity of a Grammar?

Ex. S → (S) | SS | ε where Σ is {ε, (,)}

What is L(G)?

Is G ambiguous?

Ex. E → E+E | ExE | (E) |a

Is G ambiguous?

Ex. S → if E then S | if E then S else S | other where Σ is { if, then, else, other}

Is G ambiguous?

Disambiguation of CFG

E → E+E | E x E |(E) | a *Problem: Precedence of +, x??*

Rewrite: E → E+T | T *1 new variable for each*
T → TxF | F
F → (E) | a *level of precedence*

Expression is a list of terms (T) separated by +
Term is a list of factors (F) separated by x

Ex. S → if E then S | if E then S else S | other
where Σ is { if, then, else, other}

Want to match else to closest previous unmatched else (C)

Add 2 new variables, MS (matched) and US (unmatched)

S → MS | US
MS → if E then MS else MS
US → if E then S | if E then MS else US

Disambiguation of CFG

Ex. S → (S) | SS | ε *What is the problem?*

Ex. B → B and B | a | b

Problem: How associate--to left or right?

Make right associative:

Regular Languages and CFG's

Given NFA, can construct CFG with same language

State of NFA $\longleftrightarrow$ *Variable*

$\Sigma \longleftrightarrow \Sigma$

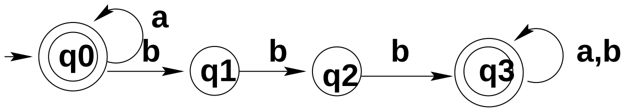
Edge $q1 \xrightarrow{a} q2 \longleftrightarrow Q1 \rightarrow a\ Q2$ *Rule*

$q1 \xrightarrow{\varepsilon} q2$ $\quad Q1 \rightarrow Q2$ *Rule*

$q0$ *(start)* $\quad Q0$ *(start)*

$q2$ *is final* $\quad Q2 \rightarrow \varepsilon$ *Rule*

Ex.



Chomsky Normal Form

Def. **A CFG is in** *CNF* **if all rules are of the form**

A $\rightarrow$ **BC** **B, C variables, not S**

A $\rightarrow$ **a** **a terminal**

S $\rightarrow$ ε

Th. **For any CFG G, there is an equivalent CFG G' in Chomsky normal form.**

Proof Idea: Assume no ε *rules.*

1. *Add new start symbol S0 and new rule S0* $\rightarrow$ *S.*
   (So S0 not on any RHS)
2. *Eliminate unit rules A* $\rightarrow$ *B.*
3. *Replace other rules A* $\rightarrow$ *B1 B2 .... Bm*
   First, get all variables on RHS, then get in form

Example, Chomsky Normal Form

V → bA V → aB
A → a B → b
A → aV B → bV
A → bAA B → aBB

1. Add S → V, S new start variable.
2. Eliminate unit rule S → V:

   V appears on LHS of rules V → bA, V → aB
   Add the rules S → bA, S → aB.
   (Only add non-unit rules.)

3. RHS all variables: For any rule with non-terms on the RHS, replace with variables:

   A → bAA replace with A → QAA, Q → b

Example, Chomsky Normal Form

S → B1A B1 → b S → A1B A1 → b
A → a B → b
A → A2V A2 → a B → B2S B2 → b
A → B3AA B3 → b B → A3BB A3 → a

4. Rewrite long rules:
   A → B3AA to A → B3Z Z → AA

   B → A3BB to B → A3W W → BB

In step 2: eliminate all unit rules. Determine for which variables you have $A \overset{*}{\Rightarrow} B$ *(how?). For each rule*

B → u |u| > 1

add rule A → u

Why Chomsky Normal Form ?

How big can grammar grow?

Step 1: Add new variable, rule

Step 2: One new rule with original start V on LHS

Step 3: Without ε rules, at most 2 extra rules for every occurrence of a nonterminal

Step 4: At most n new rules, for each RHS of n-1 symbols

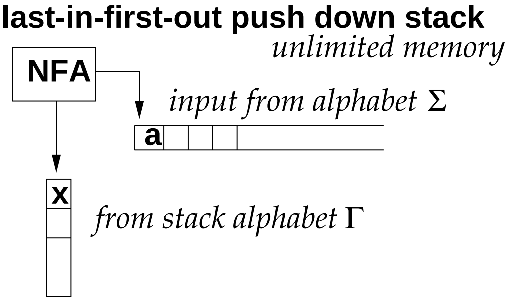
So polynomial in size of original grammar

Have simple and easily bounded algorithm to determine if string is generated by CNF form

How?

Pushdown Automata

NFA with single, last-in-first-out push down stack



Stack operations: **Read top**

Remove top *(Pop)*

Write top *(Push)*

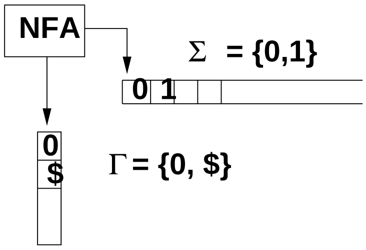
a, x → y *Reading input* **a** *with* **x** *on top, replace* **x** *with* **y**

(a, x = ε)

PDA ⟷ **CFG**

Example PDA

PDA to recognize $\{ 0^n 1^n \mid n \geq 0\}$ *(non-regular)*



Push $ on stack.

Start reading input. As read 0's, push them on stack.

When reach 1, for each 1, pop a 0 off stack. If no 0 to pop, reject.

If reach another 0, after you've read a 1, reject.

Accept if finish input, and stack has $ on top.

Finish by popping $ off stack. **(Deterministic)**