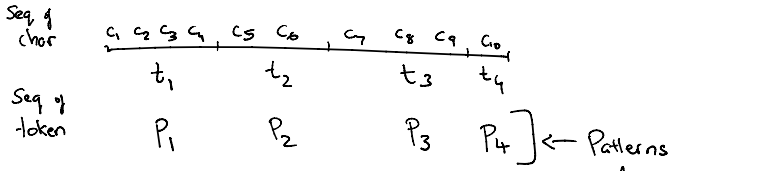
Lecture 18

Tuesday, October 26, 2021 2:04 PM



Regular expressions are a Domain Specific Language (DSL)

r ::= constant string

match this string & nothing else

| $r_1 + r_2$

match this or match that

| $r_1 \cdot r_2$

match the prefix against this & suffix against that

| r^* (Kleene - *)

match this again & again & again

| Bottom $\perp$ $\varnothing$

match nothing at all.

| Intersection $r_1 \cap r_2$

Complement $\overline{r}$

Problem 1: Match sentences where the word "John" appears at least twice.

Problem 0: "John" appears at least once

string _____ pattern _____

"John" matches Constant "John"

"John went to the market" ~~matches~~ ∈ "John"

P0 : ['a'-'z']* · "John" · [a-z]*

P1 : [a-z]* "John" [a-z]* "John" [a-z]*

∪ ~~[a-z]* John John [a-z]*~~

P2 : Contains "John" exactly once.

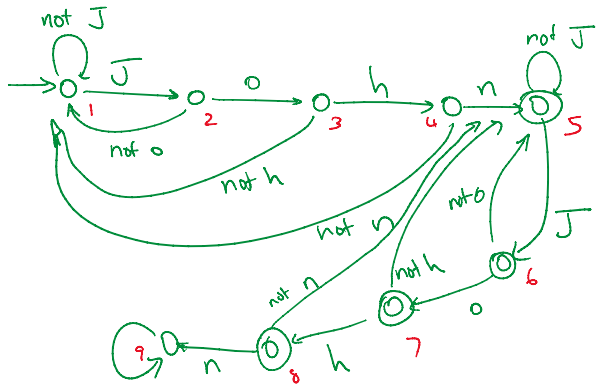
P2' : Does not contain John at all

¬([a-z]* John [a-z]*) · John ·

→ ([a-z]* John [a-z]*)

John

P2 :



Curr state	Curr symbol	Next state
1	J	2
1	not J	1
2	o	3
2	not o	1
⋮	⋮	⋮

q = 1

for each char a in the string:

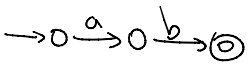
q := table[q, a]

2 [not o] 1

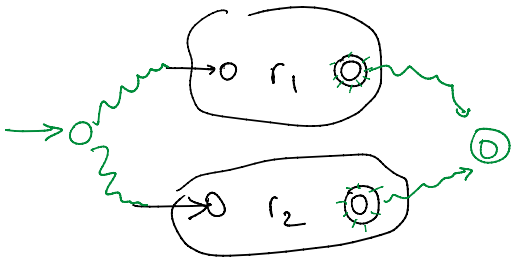
1 [move [q, a]

Games to match regular expressions

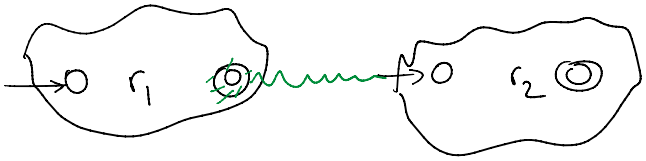
① Constant "ab"



② $r_1 + r_2$



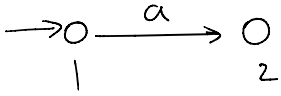
③ Concatenation $r_1 \cdot r_2$



$a^* \cdot b^*$ "Sequence of a-s followed by a sequence of b-s"

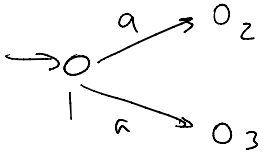
(M_1 M_2)

a a a a a b b b

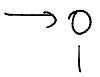


: If token in 1
& next char is a,

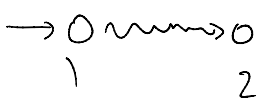
move token to 2



: If token in 1 & next char is a kill token in 1 & create tokens in 2 & 3.

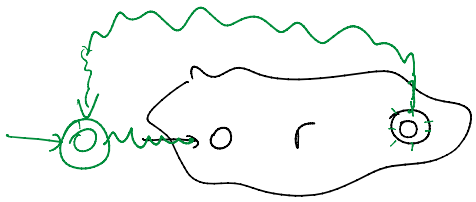


: If token in 1, & no outgoing transition on a, it dies

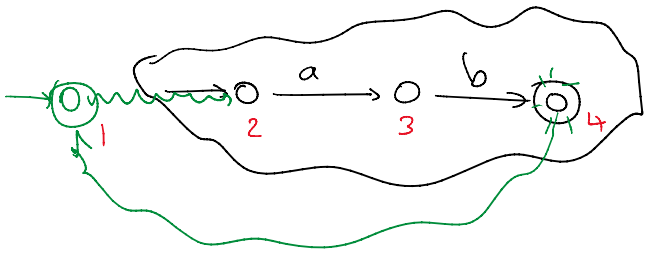


: If token in 1, then also, for free, create token in 2.

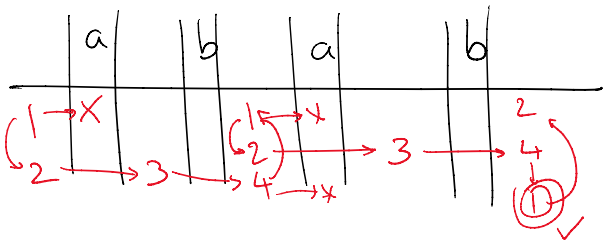
④ Kleene $*$, r^*

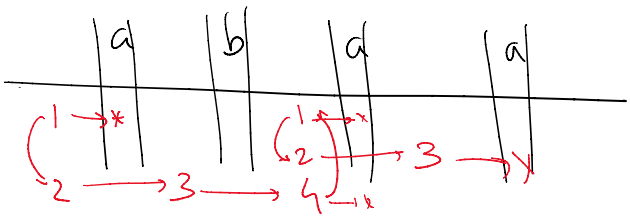


Ex: Pattern $(ab)^*$



|a| |b| |a| |b|





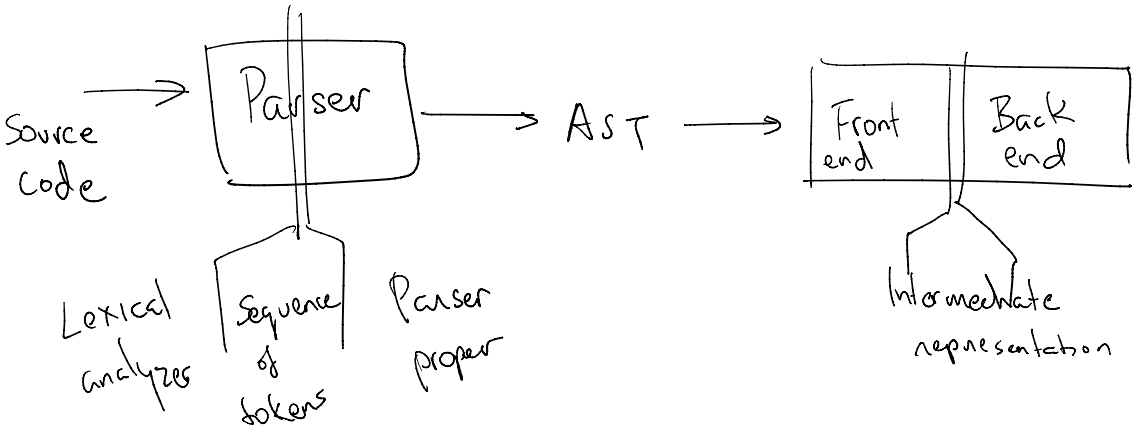
Machines with states & arrows : automata

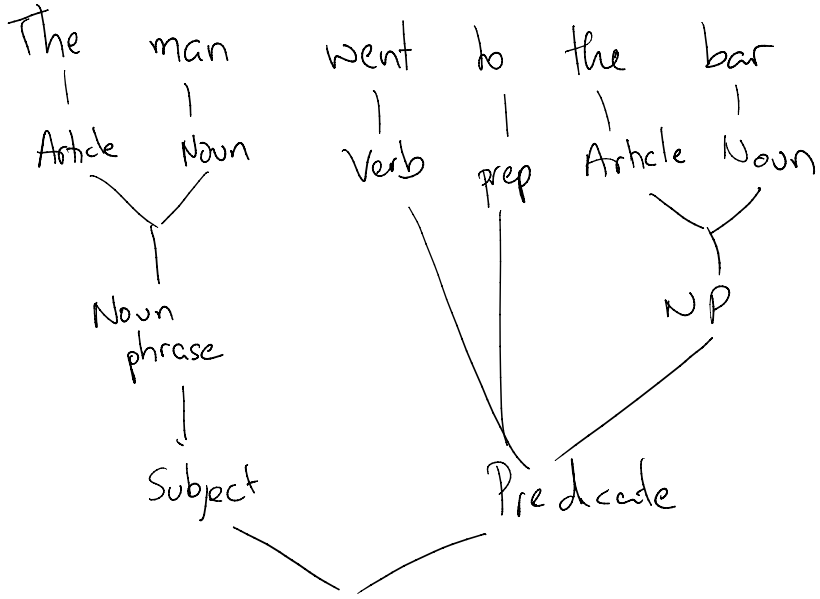
finitely many states : finite automata

multiple tokens simultaneously in play : non-deterministic

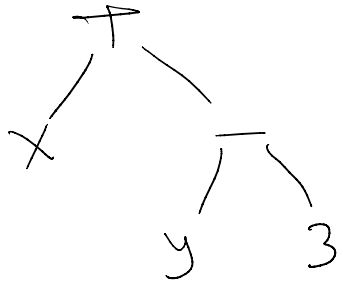
Subset Construction

deterministic





"x + (y - 3)" ⟶



Problem: Given a sequence of tokens, w
& a context-free grammar, G

(a) Find if w is a legal production according to G.

(b) If it is, construct the parse tree.

G

Sent ::= Subj · Obj | Sent ::= Sent · And · Sent

Subj ::= Article · Noun

Article ::= A | An | The | Some

Noun ::= Man | Woman | Child | Bar

Obj ::= Verb · Prep · Article · Noun

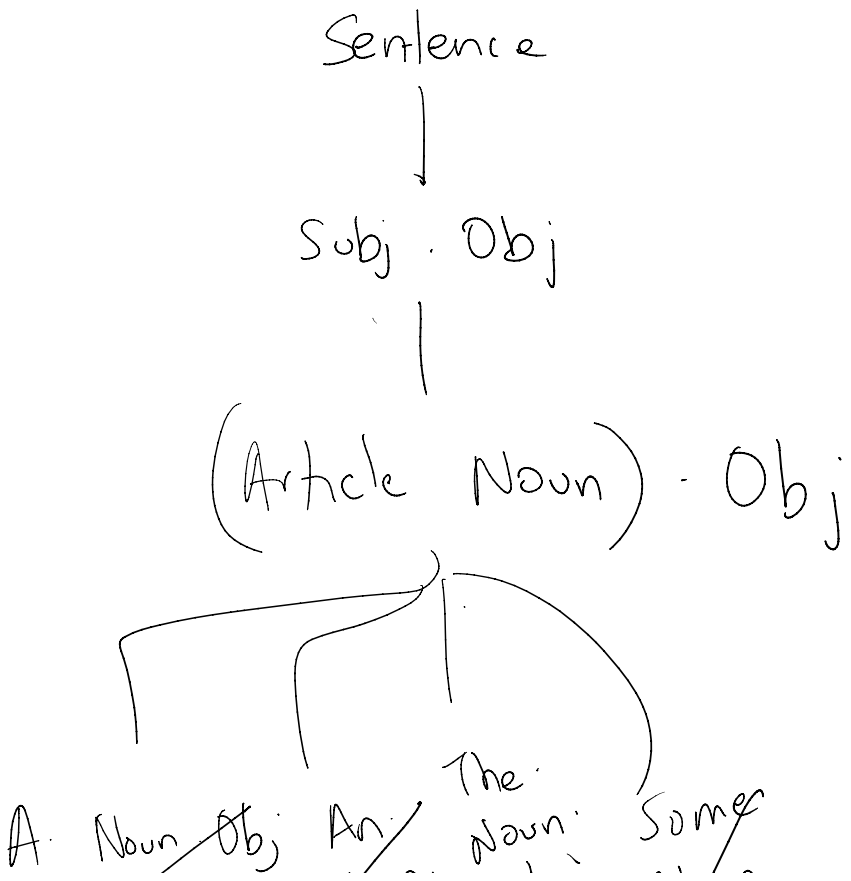
Verb ::= Goes

Prep ::= To

Concrete words "Terminal symbols"

The Man Goes To The Bar

Non-terminal symbols



A Noun Obj An Noun Obj The Noun Obj Some Noun Obj

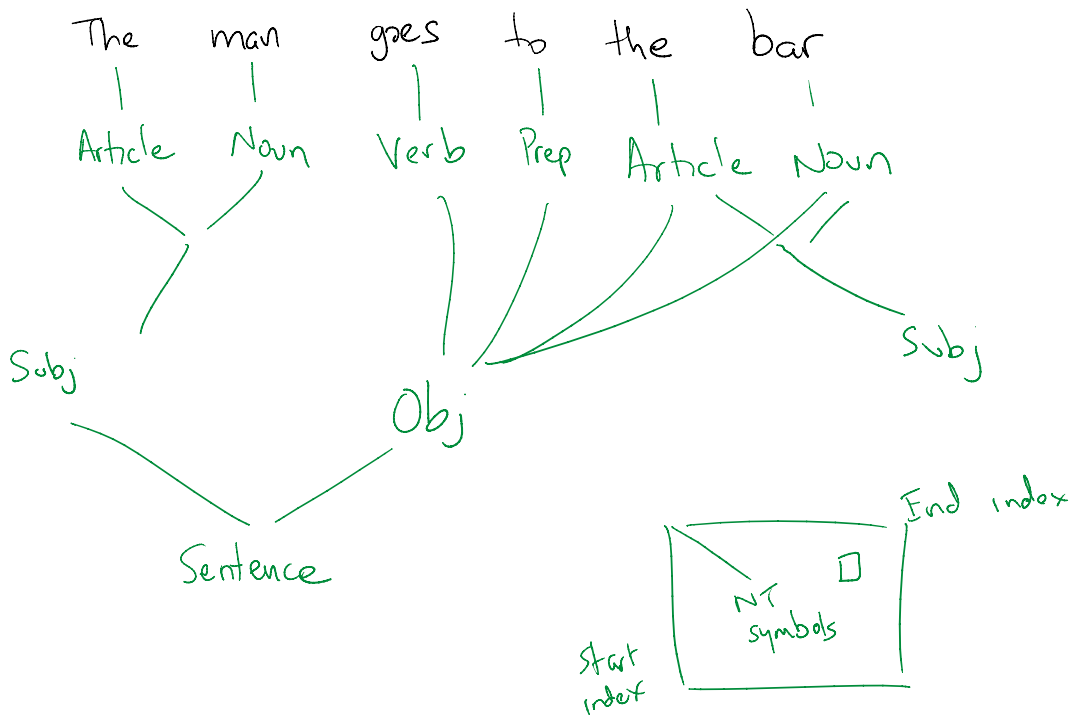
SLD decomposition

Algorithm using Dynamic Programming

For each substring — w —

for each non-terminal symbol S

in memoization table, note if w matches S.



$O(n^3 |G|)$ time

Cocke - Young - Kasami (CYK) parser

LL(k), LR(1), LALR(*),

...