Dyn Prog 3

Thursday, March 12, 2020 3:31 PM

Parsing with the CYK alg.

Parsing?: Given a grammar G, and string S, determine whether G generates S.

ex. G = all syntactically correct english sentences

$S \leadsto^*$ "this dog is a great dane"

subject, verb, object

prep, noun phrase

adject noun

ex. G representing C code

hello.c

$S \leadsto^*$ "int main() { println(); }"

ex. $(3+(5\times4)+2)$

$G \rightarrow$ expr := expr + expr

| expr x expr

| (expr)

parser(G, S):

// 1 if $G \leadsto^* S$

0 otherwise.

Flow of Compilers

Lexing: Reg lang : String $\longrightarrow$ Tokens

parsing: Grammar : Tokens $\longrightarrow$ Parse Tree

{0,1} if prog. is valid

Grammar $\Downarrow$ CFG

for (i=0; i<num; i++);

literal

"for" $\rightarrow$ FOR

CYK alg. requires G to be in CNF

Recap: CNF means

i) $A \rightarrow \alpha$ (one terminal)

ii) $A \rightarrow BC$ (exactly 2 nonterminals)

Any CFG grammar can be converted to CNF

---

ex. $E := P_1 \; E$ $\quad T := 'x'$

$P_1 := \underline{E]}$

$E ::= \quad P_2 \; E \qquad P := '+'$

$P_2 ::= E\,P$

$E ::= \quad P_3 \; R \qquad L ::= '('$

$R ::= ')'$

$P_3 ::= L \; E$

ex. '(3+(5*4)+2)'

$\underline{\text{parse}}(N_s, \underline{N}, P, \underline{\underline{S}}) : \{0,1\}$

non terminals, starting non terminal, production rules, string of symbols in Σ or "terminals"

$\underline{S \to AB}$

$\underline{A} \to 0A1$

$B \to 0B1$

0011001

// base case

if $|S| = 1$:

return True is $\underline{\text{exist}}$ a rule in P of the form

$N \to S[0]$

alternately:

$\bigvee_{rule \in P} rule = (N \to S[0])$

// inductive case

$\bigvee_{rule \in P} rule = (N \to AB)$ for some $A, B, \ldots$

// $S = XY$

$\underline{A \leadsto X} \quad \overline{B \leadsto Y}$

$\bigvee_{K \in [1 .. |S|-1]}$ parse$(N_s, P, A, S[:k])$

and parse$(N_s, P, B, S[k:])$

each subproblem takes $O(|P| \times |S|)$ time.

$W \times W \times \phi$ subproblems

$O(|W|^3 \cdot |P| \cdot d)$

$G =$

Let N_s, P, S_t be fixed. Let S the string be fixed.

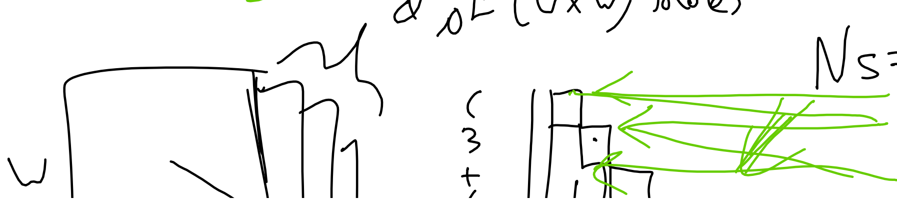
parse$(N^{\in N_s}, i, j)$: $\quad |S| = W, \quad d = |N_s|$

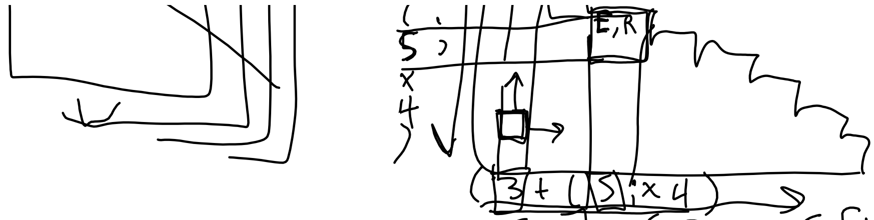
// parse$(G, N, S[i:j])$

$W \times W \times d$

d of $(W \times W)$ tables

$N_s = [E, P, L, R, \cancel{M}]$





"$E \in OPT[i,j]$ $\longleftrightarrow$ $S[i:j]$ can be generated starting from E