CS3200 - CYK & CNF (cont.)

HW due Saturday
Last time:

Chomsky Normal Form:
\[ A \rightarrow BC \]
\[ X \rightarrow x \]
and only from start state

Why?

CYK algorithm requires CNF.
Conversion example:

\[
\begin{align*}
S & \rightarrow ASB \\
A & \rightarrow aAS | aA \epsilon \\
B & \rightarrow SBS | A | bb \\
\end{align*}
\]

4 steps:
- remove ε transitions & get new start state
- remove unit parses
- need only 2 non-terminals
- replace terminals in pairs w/ dedicated non-terminal
(1) New start state, & eliminate $\varepsilon$ rules:

$S_0 \rightarrow S$
$S \rightarrow ASB | SB$
$A \rightarrow aAS | aA | aS$
$B \rightarrow SbS | A | 1bb$

$S_0 \rightarrow S$
$S \rightarrow ASB | SB$
$A \rightarrow aAS | aA | aS$
$B \rightarrow SbS | A | 1bb$
2. Remove Unit rules

\[ S_0 \rightarrow ASB | SB | AS \]

\[ S \rightarrow ASB | SB | AS \] (Crossed out)

\[ A \rightarrow aA | S | aAS \]

\[ B \rightarrow SbS | A | bb \]

\[ B \rightarrow SbS | bb | aAS | a | AS \]
③ Fix so that we have all pairs

\[ S_0 \rightarrow ASB \mid SB \mid AS \]
\[ S \rightarrow ASB \mid SB \mid AS \]
\[ A \rightarrow aAS \mid a \mid aS \]
\[ B \rightarrow SbS \mid bb \mid aAS \mid a \mid aS \]

\[ S_0 \rightarrow AU_1 \mid SB \mid AS \]
\[ U_1 \rightarrow SB \]
\[ S \rightarrow AU_1 \mid SB \mid AS \]
\[ A \rightarrow aU_2 \mid a \mid aS \]
\[ U_2 \rightarrow AS \]
\[ B \rightarrow SU_3 \mid bb \mid aU_2 \mid a \mid aS \]
\[ U_3 \rightarrow bS \]
Finally, need only non-terminal pairs

\[
S_0 \rightarrow AU_1 | SB | AS
\]
\[
U_1 \rightarrow SB
\]
\[
S \rightarrow AU_1 | SB | AS
\]
\[
A \rightarrow U_4 U_2 | a | U_4 S
\]
\[
U_2 \rightarrow AS
\]
\[
B \rightarrow SU_3 | U_5 U_5
\]
\[
U_3 \rightarrow X S
\]

\[
U_4 \rightarrow a
\]
\[
U_5 \rightarrow b
\]
Running time:

Actually depends on the order steps are performed in, since some operations undo others.

i.e.: deleting ε-rules
then eliminate right hands > 2

→ exponential blow-up

but reverse is a linear operation

Bottom line: $O(n^2)$
And the why: CYK algorithm in CNF.

An algorithm which, given a grammar and a word, decides if the word can be produced by the grammar.

Runtime: $O(n^3)$
How?

$S \rightarrow NP \ VP$
$VP \rightarrow VP \ PP$
$VP \rightarrow V \ NP$
$VP \rightarrow eats$
$PP \rightarrow P \ NP$
$NP \rightarrow Det \ N$
$NP \rightarrow she$
$V \rightarrow eats$
$P \rightarrow with$
$N \rightarrow fish$
$N \rightarrow fork$
$Det \rightarrow a$

CYK table

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>VP</td>
<td>PP</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>VP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>NP</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>V, VP</td>
<td>Det.</td>
<td>N</td>
</tr>
<tr>
<td>she</td>
<td>eats</td>
<td>a</td>
<td>fish</td>
</tr>
</tbody>
</table>
**Pseudo code:**

\[
\text{(length of word)}^3 \text{(no of productions)}
\]

let the input be a string \( S \) consisting of \( n \) characters: \( a_1 \ldots a_n \).

let the grammar contain \( r \) nonterminal symbols \( R_1 \ldots R_r \).

This grammar contains the subset \( R_s \) which is the set of start symbols.

let \( P[n,n,r] \) be an array of booleans. Initialize all elements of \( P \) to false.

for each \( i = 1 \) to \( n \)

for each unit production \( R_j \rightarrow a_i \)

set \( P[1,i,j] = \text{true} \)

for each \( i = 2 \) to \( n \) -- Length of span

for each \( j = 1 \) to \( n-i+1 \) -- Start of span

for each \( k = 1 \) to \( i-1 \) -- Partition of span

for each production \( R_A \rightarrow R_B R_C \)

if \( P[k,j,B] \) and \( P[i-k,j+k,C] \) then set \( P[i,j,A] = \text{true} \)

if any of \( P[n,1,x] \) is true (\( x \) is iterated over the set \( s \), where \( s \) are all the indices for \( R_s \)) then

\( S \) is member of language

else

\( S \) is not member of language