A CYK for Any Grammar

grammar G, non-terminals $A_1,\ldots,A_K$, tokens $t_1,\ldots.t_L$

input word: $w = w_{(0)}w_{(1)} \ldots w_{(N-1)}$

$w_{p..q} = w_{(p)}w_{(p+1)} \ldots w_{(q-1)}$

Triple (A, p, q) means: $A \Rightarrow^* w_{p..q}$, A can be: A_i, t_j, or ε

$P = \{(w_{(i)},i,i+1) \mid 0 \leq i < N-1\}$

repeat {

choose rule $(A::=B_1\ldots B_m) \in G$

if $((A,p_0,p_m) \notin P$ &&

$((m=0 \text{ \&\& } p_0=p_m) \mid\mid (B_1,p_0,p_1), \ldots,(B_m,p_{m-1},p_m) \in P))$

$P := P \cup \{(A,p_0,p_m)\}$

} until no more insertions possible

What is the maximal number of steps?
How long does it take to check step for a rule?

} for grammar in given normal form

Strategy for Populating Table

- Which order to insert (A, p, q) tuples ?
  - all orders give correct result
  - efficiency differs
- Left-to-right scan of the input:
  - derive all A,p for given q
  - then increase q to q+1
- Consider only productive parse attempts
  - insert (A,p,q) only if we can prove that
    S =>* $w_{0..p-1}$ A Y (Y is any string of symbols)

Dotted Rules Like Nonterminals

$$X ::= Y_1\ Y_2\ Y_3$$

Chomsky transformation is (a simplification of) this:

$$\begin{aligned} X &::= W_{123} \\ W_{123} &::= W_{12}\ Y_3 \\ W_{12} &::= W_1\ Y_2 \\ W_1 &::= W_\varepsilon\ Y_1 \\ W_\varepsilon &::= \varepsilon \end{aligned}$$

Early parser: dotted RHS as names of fresh non-terminals:

$$\begin{aligned} X &::= (Y_1 Y_2 Y_3 .) \\ (Y_1 Y_2 Y_3 .) &::= (Y_1 Y_2 . Y_3)\ \ Y_3 \\ (Y_1 Y_2 . Y_3) &::= (Y_1 . Y_2 Y_3)\ \ Y_2 \\ (Y_1 . Y_2 Y_3) &::= (. Y_1 Y_2 Y_3)\ \ Y_3 \\ (. Y_1 Y_2 Y_3) &::= \varepsilon \end{aligned}$$

Earley Parser

- group the triples by last element: S(q) ={(A,p)|(A,p,q)∈P}
- dotted rules effectively make productions at most binary

Steps of Earley Parsing Algorithm

Initially, let $S(0) = \{(D' ::= .D\ \mathrm{EOF},\ 0)\}$

When scanning input at position j, parser does the following operations (p, q, r are sequences of terminals and non-terminals):

Prediction

If $(X ::= p.Yq, i) \in S(j)$ and $Y ::= r$ is a grammar rule, then

$$S(j) = S(j) \cup \{(Y ::= .r, j)\}$$

Scanning

If $w(j) = a$ and $(X ::= p.aq, i) \in S(j)$ then (we can skip a):

$$S(j+1) = S(j+1) \cup \{(X ::= pa.q, i)\}$$

Completion

If $(X ::= p., i) \in S(j)$ and $(Y ::= q.Xr, k) \in S(i)$ then

$$S(j) = S(j) \cup \{(Y ::= qX.r, k)\}$$

sketch of completion:

```
w(0) ... w(k)   ...   w(i)    ...   w(j)
           |      q     |      p      |
                    Y::=q.Xr        X::=p.
                                    Y::=qX.r
```

		ID (s_1)	- (s_2)	ID (s_3)	==	ID	EOF
	ε . e EOF . ID . e - e . e == e	ID ID . e . EOF e . - e e . == e	ID- e - . e	ID-ID e - e . e . EOF e . - e e . == e	ID-ID== e == . e	ID-ID==ID e == e . e - e .	e . EOF
ID		ε	-	-ID	-ID==	-ID==ID	
-			ε . ID . e - e . e == e	ID ID . e . - e e . == e	ID== e == . e	ID==ID e == e .	
ID				ε	==	==ID	
==					ε . ID . e - e . e == e	ID ID . e . - e e . == e	
ID						ε	
EOF							ε

S ::= . e EOF | e . EOF | e EOF .

e ::= . ID | ID .
| . e – e | e . – e | e – . e | e – e .
| . e == e | e . == e | e == . e | e == e .