chapter 13 : Constituency Parsing

(§13·3)

```
function CKY-PARSE (words, grammar) returns table

    for j ← from 1 to LENGTH(words) do
        for all {A | A → words[j] ∈ grammar}
            table[j-1, j] ← CHECK-UNITS(table[j-1, j] ∪ A)

        for i ← from j-2 downto 0 do
            for k ← i+1 to j-1 do
                for all {A | A → BC ∈ grammar and B ∈ table[i,k]
                                                and C ∈ table[k,j]}
                    table[i,j] ← CHECK-UNITS(table[i,j] ∪ A)
```

```
function CHECK-UNITS (cell) return updated_cell

    updated_cell = cell
    for i ← from 1 to LENGTH(cell)
        B ← cell[i]
        for all {A | $A \overset{*}{\Rightarrow} B$ ∈ grammar}
            updated_cell ← cell ∪ A
```

S → NP VP
S → X1 VP
X1 → Aux NP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → X2 PP
X2 → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP
Det → that | this | the | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

Book	the	flight	through	Houston
Noun, Verb ↓ ↓ Nominal VP, S 0,1	0,2	VP, X_2 ↓ S 0,3	0,4	VP VP X_2 ↓ ↓ S S 0,5
	Det 1,2	NP 1,3	1,4	NP 1,5
		Noun ↓ Nominal 2,3	2,4	Nominal 2,5
			Prep 3,4	PP 3,5
				Proper-Noun ↓ NP 4,5

Fig: Running above Algo.

(Q13.4) Partial parsing is mostly used in information extracal systems and its main advantage is the accelerated processing speed. Since you are only parsing chunks of data instead of each word, partial parsing is much faster. Also, as mentioned in Section 13.6, eliminating post-head modifiers obviates the need to resolve attachment ambiguities.

But a major disadvantage is probably the fact that you end up loosing a lot of valuable information.

(Q13.5) We can do the following things:

(i) In case of spelling mistakes, we can flag the incorrect words and use string matching algorithm to correct the word.

(ii) or, we can learn the n-gram probabilities, and use them to predict the closest word to replace the incorrect word.

(iii) We can extend (ii) to produce a set of candidate sentences, and choose one with the highest probability.