CSCI 5832
Natural Language Processing

Lecture 13
Jim Martin

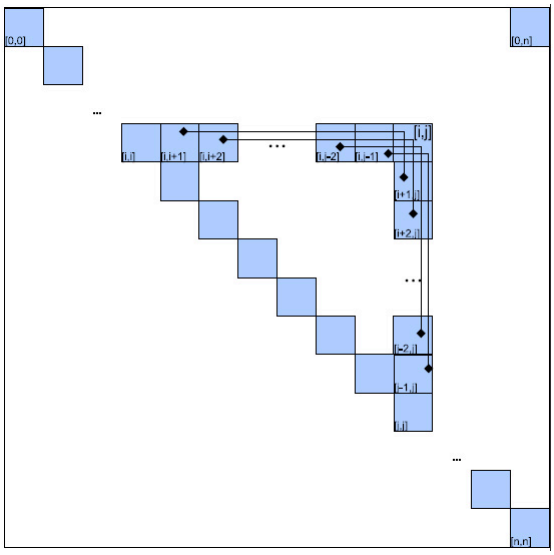
Today: 3/1

- **Review/Finish CKY**
- **Earley parsing**

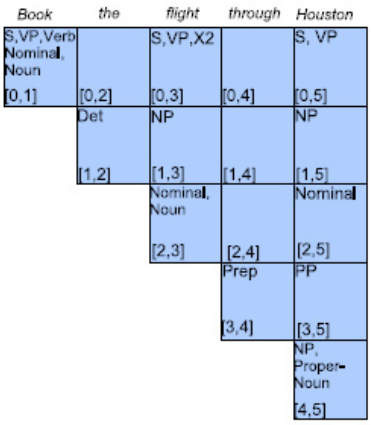
CKY Algorithm

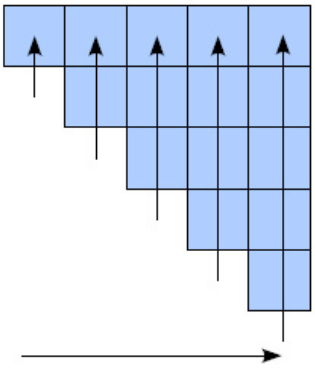
```
function CKY-PARSE(words, grammar) returns table

for j ← from 1 to LENGTH(words) do
    table[j − 1, j] ← {A | A → words[j] ∈ grammar }
    for i ← from j − 2 downto 0 do
        for k ← i + 1 to j − 1 do
            table[i, j] ← table[i, j] ∪
                          {A | A → BC ∈ grammar,
                               B ∈ table[i, k],
                               C ∈ table[k, j] }
```


CKY Table



Example

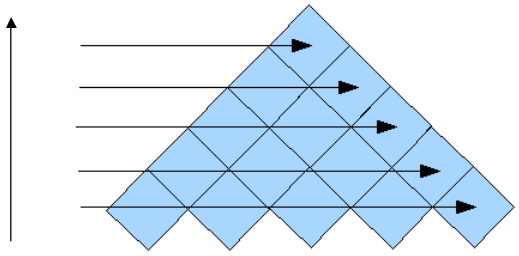




Other Ways to Do It?

- **Are there any other sensible ways to fill the table that still guarantee that the cells we need are already filled?**

Other Ways to Do It?



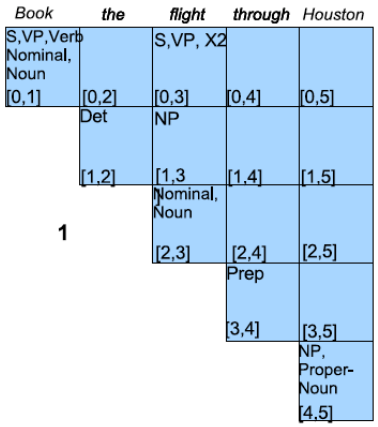
Sample Grammar

$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid TWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

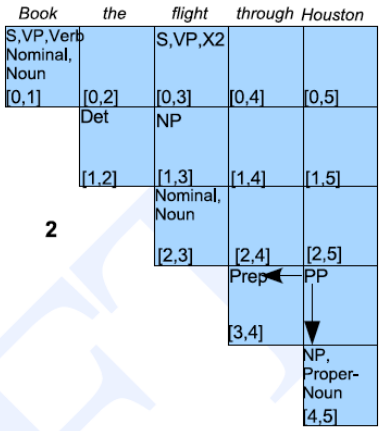
CNF Conversion

$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

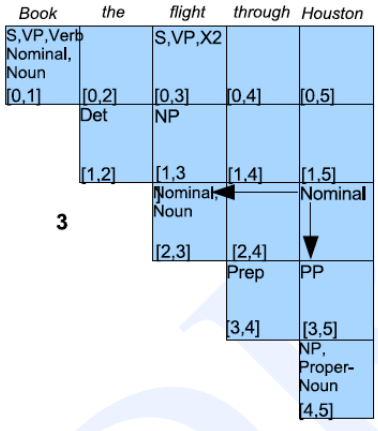
Example
Fill the Last Column



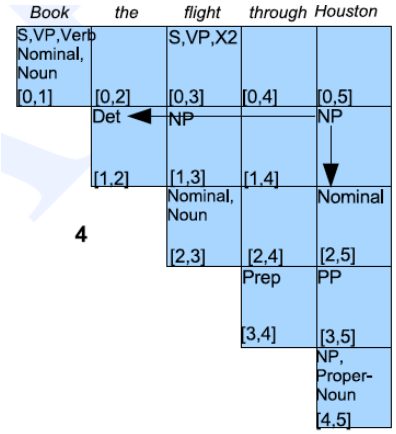
Example
Fill the Last Column



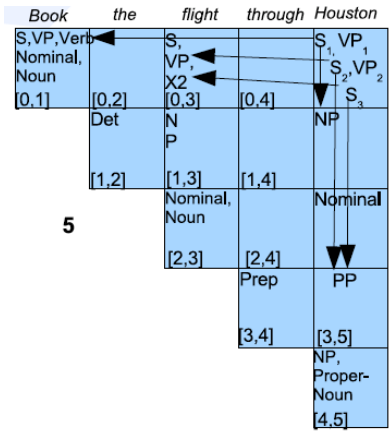
Example
Fill the Last Column



Example
Fill the Last Column



Example
Fill the Last Column



CKY Notes

- Since it's bottom up, CKY populates the table with a lot of phantom constituents.
  - Segments that by themselves are constituents but cannot really occur in the context in which they are being suggested.
  - To avoid this we can switch to a top-down control strategy or
  - We can add some kind of filtering that blocks constituents where they can not happen in a final analysis.

Earley Parsing

- Allows arbitrary CFGs
- Top-down control
- Fills a table in a single sweep over the input words
  - Table is length N+1; N is number of words
  - Table entries represent
    - Completed constituents and their locations
    - In-progress constituents
    - Predicted constituents

States

- The table-entries are called states and are represented with dotted-rules.

S -> ▪ VP	A VP is predicted
NP -> Det ▪ Nominal	An NP is in progress
VP -> V NP ▪	A VP has been found

States/Locations

- It would be nice to know where these things are in the input so…

S -> ● VP [0,0] A VP is predicted at the start of the sentence

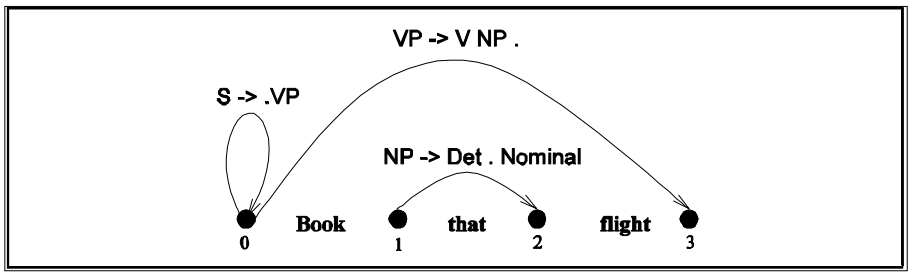
NP -> Det ● Nominal [1,2] An NP is in progress; the Det goes from 1 to 2

VP -> V NP ● [0,3] A VP has been found starting at 0 and ending at 3

States/Locations

- S -> • VP [0,0]
  - A VP is predicted at the start of the sentence
- NP -> Det • Nominal [1,2]
  - An NP is in progress; the Det goes from 1 to 2
- VP -> V NP • [0,3]
  - A VP has been found starting at 0 and ending at 3

Graphically



Earley

- As with most dynamic programming approaches, the answer is found by looking in the table in the right place.
- In this case, there should be an S state in the final column that spans from 0 to n and is complete.
- If that's the case you're done.
  - S –> α • [0,n]

Earley

- So sweep through the table from 0 to n…
  - New predicted states are created by starting top-down from S
  - New incomplete states are created by advancing existing states as new constituents are discovered
  - New complete states are created in the same way.

Earley

- **More specifically…**
  1. ***Predict*** **all the states you can upfront**
  2. **Read a word**
     1. **Extend states based on matches**
     2. **Generate new predictions**
     3. **Go to step 2**
  3. **Look at n to see if you have a winner**

Earley Code

```
function EARLEY-PARSE(words, grammar) returns chart

  ADDTOCHART((γ → • S, [0,0]), chart[0])
  for i ← from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
                NEXT-CAT(state) is not a part of speech then
          PREDICTOR(state)
      elseif INCOMPLETE?(state) and
                NEXT-CAT(state) is a part of speech then
          SCANNER(state)
      else
          COMPLETER(state)
    end
  end
  return(chart)
```

Earley Code

procedure PREDICTOR($(A \rightarrow \alpha \bullet B\ \beta,\ [i, j])$)
  for each $(B \rightarrow \gamma)$ **in** GRAMMAR-RULES-FOR(B, *grammar*) **do**
    ADDTOCHART($(B \rightarrow \bullet\ \gamma,\ [j, j])$, *chart*[*j*])
  end

procedure SCANNER($(A \rightarrow \alpha \bullet B\ \beta,\ [i, j])$)
  if $B \in$ PARTS-OF-SPEECH(*word*[*j*]) **then**
    ADDTOCHART($(B \rightarrow word[j] \bullet,\ [j, j+1])$, *chart*[*j*+1])

procedure COMPLETER($(B \rightarrow \gamma \bullet,\ [j,k])$)
  for each $(A \rightarrow \alpha \bullet B\ \beta,\ [i, j])$ **in** *chart*[*j*] **do**
    ADDTOCHART($(A \rightarrow \alpha\ B \bullet \beta,\ [i,k])$, *chart*[*k*])
  end

Example

- **Book that flight**
- **We should find… an S from 0 to 3 that is a completed state…**

Example

Chart[0]	S0	$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
	S1	$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
	S2	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
	S3	$S \rightarrow \bullet VP$	[0,0]	Predictor
	S4	$NP \rightarrow \bullet Pronoun$	[0,0]	Predictor
	S5	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
	S6	$NP \rightarrow \bullet Det\ Nominal$	[0,0]	Predictor
	S7	$VP \rightarrow \bullet Verb$	[0,0]	Predictor
	S8	$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor
	S9	$VP \rightarrow \bullet Verb\ NP\ PP$	[0,0]	Predictor
	S10	$VP \rightarrow \bullet Verb\ PP$	[0,0]	Predictor
	S11	$VP \rightarrow \bullet VP\ PP$	[0,0]	Predictor

Add To Chart

procedure ADDTOCHART(*state, chart-entry*)
  if *state* is not already in *chart-entry* **then**
    PUSH-ON-END(*state, chart-entry*)
  end

Example

Chart[1]	S12	*Verb* → *book* •	[0,1]	Scanner
	S13	*VP* → *Verb* •	[0,1]	Completer
	S14	*VP* → *Verb* • *NP*	[0,1]	Completer
	S15	*VP* → *Verb* • *NP PP*	[0,1]	Completer
	S16	*VP* → *Verb* • *PP*	[0,1]	Completer
	S17	*S* → *VP* •	[0,1]	Completer
	S18	*VP* → *VP* • *PP*	[0,1]	Completer
	S19	*NP* → • *Pronoun*	[1,1]	Predictor
	S20	*NP* → • *Proper-Noun*	[1,1]	Predictor
	S21	*NP* → • *Det Nominal*	[1,1]	Predictor
	S22	*PP* → • *Prep NP*	[1,1]	Predictor

Example

Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
	S24	*NP* → *Det* • *Nominal*	[1,2]	Completer
	S25	*Nominal* → • *Noun*	[2,2]	Predictor
	S26	*Nominal* → • *Nominal Noun*	[2,2]	Predictor
	S27	*Nominal* → • *Nominal PP*	[2,2]	Predictor
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	Completer
	S30	*NP* → *Det Nominal* •	[1,3]	Completer
	S31	*Nominal* → *Nominal* • *Noun*	[2,3]	Completer
	S32	*Nominal* → *Nominal* • *PP*	[2,3]	Completer
	S33	*VP* → *Verb NP* •	[0,3]	Completer
	S34	*VP* → *Verb NP* • *PP*	[0,3]	Completer
	S35	*PP* → • *Prep NP*	[3,3]	Predictor
	S36	*S* → *VP* •	[0,3]	Completer
	S37	*VP* → *VP* • *PP*	[0,3]	Completer

Efficiency

- **For such a simple example, there seems to be a lot of useless stuff in there.**
- **Why?**
  - It's predicting things that aren't consistent with the input
  - That's the flipside to the CKY problem.

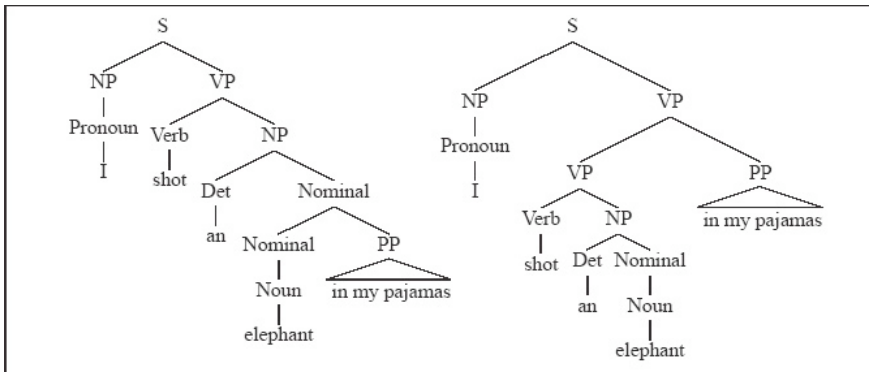
Details

- **As with CKY that isn't a parser until we add the backpointers so that each state knows where it came from.**

Back to Ambiguity

- **Did we solve it?**

Ambiguity



Ambiguity

- No…
  - Both CKY and Earley will result in multiple S structures for the [0,n] table entry.
  - They both efficiently store the sub-parts that are shared between multiple parses.
  - And they obviously avoid re-deriving those sub-parts.
  - But neither can tell us which one is right.

Ambiguity

- In most cases, humans don't notice incidental ambiguity (lexical or syntactic). It is resolved on the fly and never noticed.
- We'll try to model that with probabilities.
- But note something odd and important about the Groucho Marx example…

Next Time

- Read Section 12.5 (Partial Parsing)
- Then go back and start reading Sections 6.6, 6,7 and 6.8.
  - If you have no background in stats please slog through section 6.6 carefully.
- After that we'll move on to probabilistic parsing which is in the new draft Ch.13
  - Which isn't there yet. Hopefully it will be there when we get there.