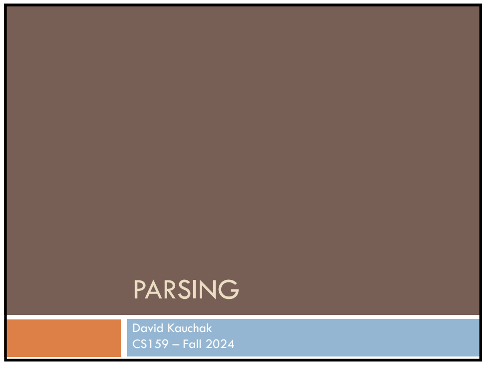
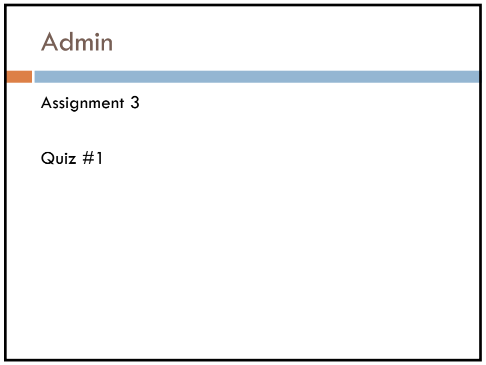


PARSING

David Kauchak
CS159 – Fall 2024

1

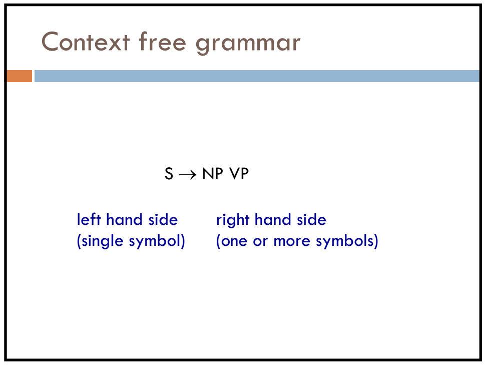


Admin

Assignment 3

Quiz #1

2



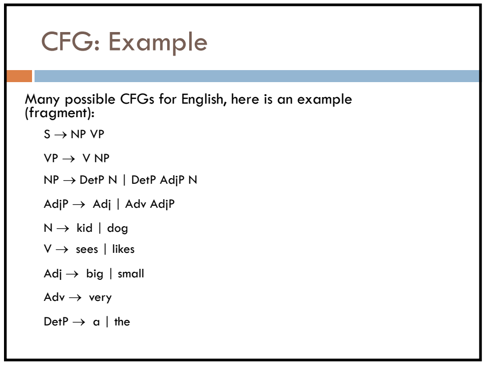
Context free grammar

S → NP VP

left hand side
(single symbol)

right hand side
(one or more symbols)

3

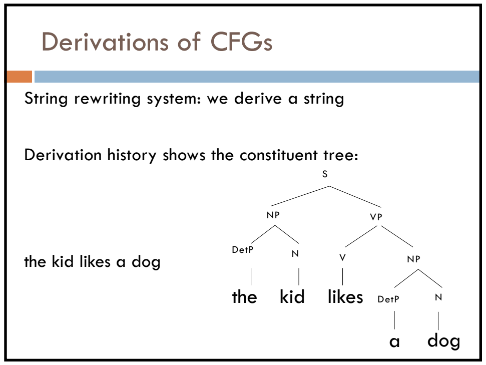


CFG: Example

Many possible CFGs for English, here is an example (fragment):

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → kid | dog
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

4



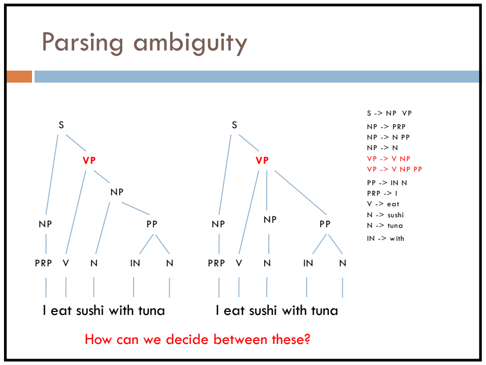
Derivations of CFGs

String rewriting system: we derive a string

Derivation history shows the constituent tree:

the kid likes a dog

5

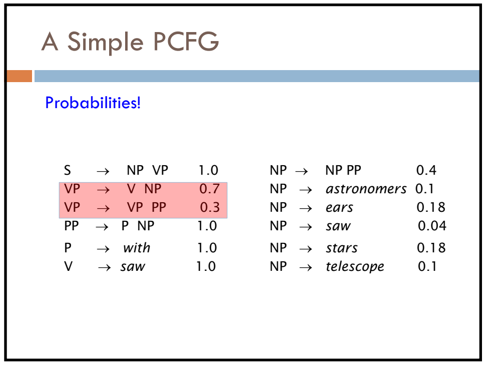


Parsing ambiguity

S -> NP VP
NP -> PRP
NP -> N PP
NP -> N
VP -> V NP
VP -> V NP PP
PP -> IN N
PRP -> I
V -> eat
N -> sushi
N -> tuna
IN -> with

How can we decide between these?

6

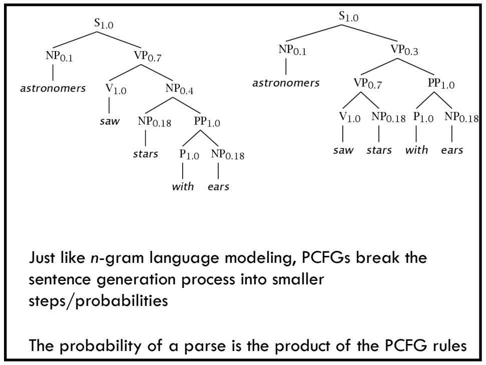


A Simple PCFG

Probabilities!

S	→	NP VP	1.0	NP	→	NP PP	0.4
VP	→	V NP	0.7	NP	→	*astronomers*	0.1
VP	→	VP PP	0.3	NP	→	*ears*	0.18
PP	→	P NP	1.0	NP	→	*saw*	0.04
P	→	*with*	1.0	NP	→	*stars*	0.18
V	→	*saw*	1.0	NP	→	*telescope*	0.1

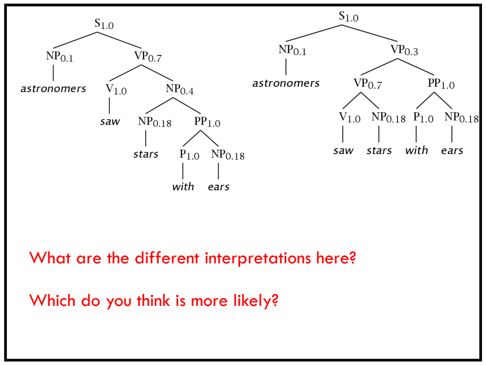
7



Just like *n*-gram language modeling, PCFGs break the sentence generation process into smaller steps/probabilities

The probability of a parse is the product of the PCFG rules

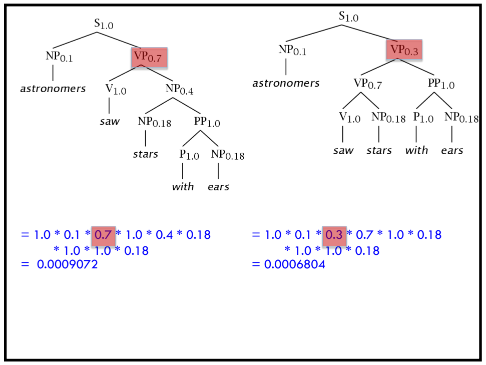
8



What are the different interpretations here?

Which do you think is more likely?

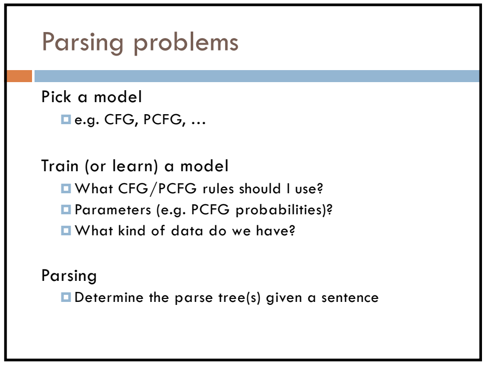
9



= 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18
* 1.0 * 1.0 * 0.18
= 0.0009072

= 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18
* 1.0 * 1.0 * 0.18
= 0.0006804

10



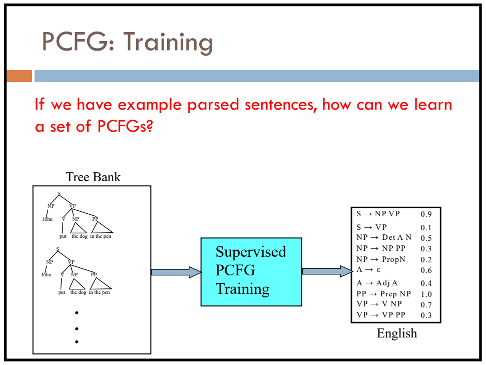
Parsing problems

Pick a model
- e.g. CFG, PCFG, …

Train (or learn) a model
- What CFG/PCFG rules should I use?
- Parameters (e.g. PCFG probabilities)?
- What kind of data do we have?

Parsing
- Determine the parse tree(s) given a sentence

11



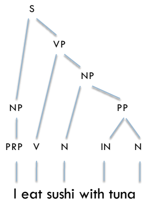
PCFG: Training

If we have example parsed sentences, how can we learn a set of PCFGs?

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

12

Extracting the rules



S → NP VP
NP → PRP
PRP → I
VP → V NP
V → eat
NP → N PP
N → sushi
PP → IN N
IN → with
N → tuna

What CFG rules occur in this tree?

13

Estimating PCFG Probabilities

We can extract the rules from the trees

S → NP VP
NP → PRP
PRP → I
VP → V NP
V → eat
NP → N PP
N → sushi
…

S	→	NP VP	1.0
VP	→	V NP	0.7
VP	→	VP PP	0.3
PP	→	P NP	1.0
P	→	*with*	1.0
V	→	*saw*	1.0

How do we go from the extracted CFG rules to PCFG rules?

14

Estimating PCFG Probabilities

Extract the rules from the trees

Calculate the probabilities using MLE

$$\alpha \rightarrow \beta \quad \Rightarrow \quad p(\alpha \rightarrow \beta \mid \alpha)$$

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

15

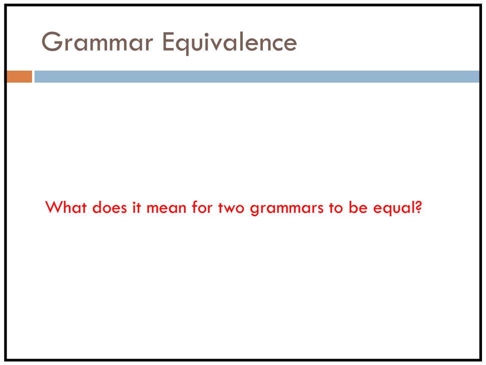
Estimating PCFG Probabilities

	Occurrences
S → NP VP	10
S → V NP	3
S → VP PP	2
NP → N	7
NP → N PP	3
NP → DT N	6

P(S → V NP) = ?

$$\text{P(S} \rightarrow \text{V NP)} = \text{P(S} \rightarrow \text{V NP} \mid \text{S)} = \frac{\text{count(S} \rightarrow \text{V NP)}}{\text{count(S)}} = \frac{3}{15}$$

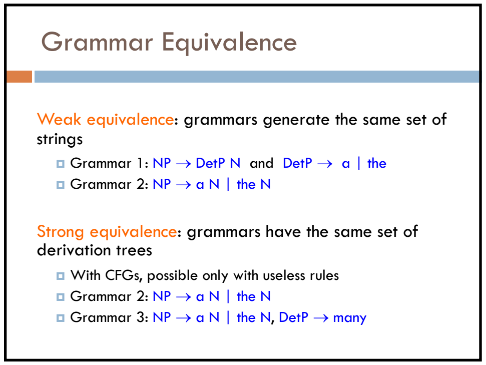
16



Grammar Equivalence

What does it mean for two grammars to be equal?

17



Grammar Equivalence

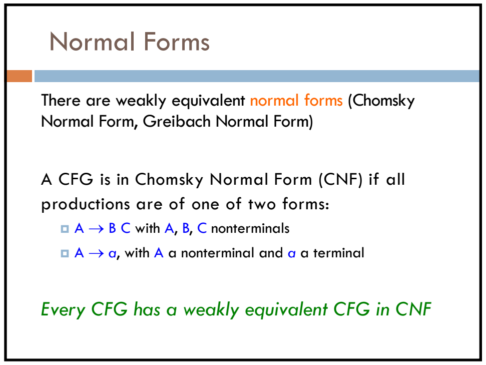
Weak equivalence: grammars generate the same set of strings

- Grammar 1: NP → DetP N and DetP → a | the
- Grammar 2: NP → a N | the N

Strong equivalence: grammars have the same set of derivation trees

- With CFGs, possible only with useless rules
- Grammar 2: NP → a N | the N
- Grammar 3: NP → a N | the N, DetP → many

18



Normal Forms

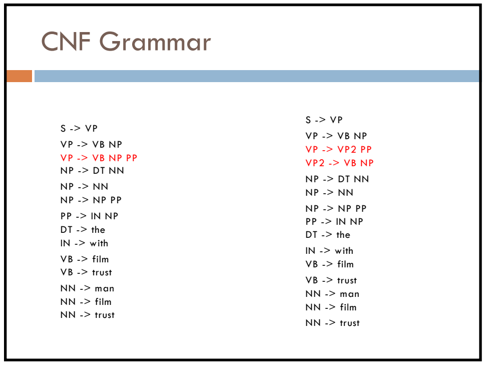
There are weakly equivalent normal forms (Chomsky Normal Form, Greibach Normal Form)

A CFG is in Chomsky Normal Form (CNF) if all productions are of one of two forms:

- A → B C with A, B, C nonterminals
- A → a, with A a nonterminal and a a terminal

Every CFG has a weakly equivalent CFG in CNF

19

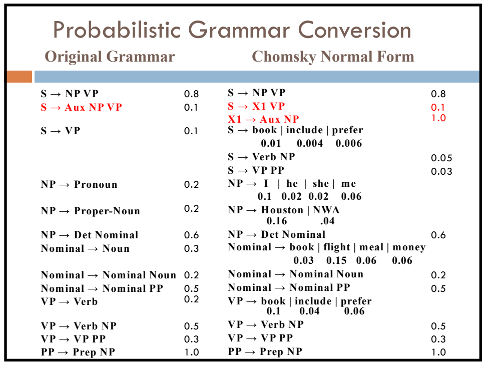


CNF Grammar

```
S -> VP
VP -> VB NP
VP -> VB NP PP
NP -> DT NN
NP -> NN
NP -> NP PP
PP -> IN NP
DT -> the
IN -> with
VB -> film
VB -> trust
NN -> man
NN -> film
NN -> trust
```

```
S -> VP
VP -> VB NP
VP -> VP2 PP
VP2 -> VB NP
NP -> DT NN
NP -> NN
NP -> NP PP
PP -> IN NP
DT -> the
IN -> with
VB -> film
VB -> trust
NN -> man
NN -> film
NN -> trust
```

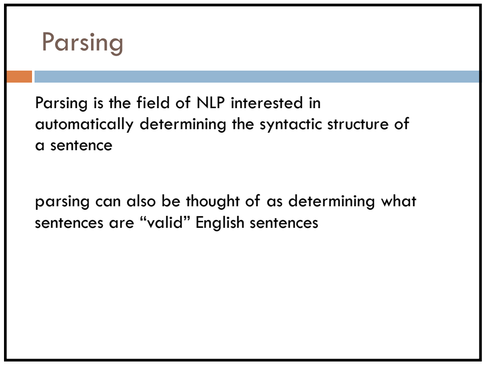
20



Probabilistic Grammar Conversion

Original Grammar		Chomsky Normal Form	
S → NP VP	0.8	S → NP VP	0.8
S → Aux NP VP	0.1	S → X1 VP	0.1
		X1 → Aux NP	1.0
S → VP	0.1	S → book | include | prefer 0.01 0.004 0.006	
		S → Verb NP	0.05
		S → VP PP	0.03
NP → Pronoun	0.2	NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Proper-Noun	0.2	NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6	NP → Det Nominal	0.6
Nominal → Noun	0.3	Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2	Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5	Nominal → Nominal PP	0.5
VP → Verb	0.2	VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5	VP → Verb NP	0.5
VP → VP PP	0.3	VP → VP PP	0.3
PP → Prep NP	1.0	PP → Prep NP	1.0

21

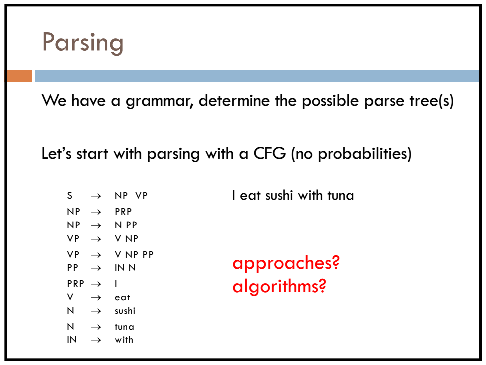


Parsing

Parsing is the field of NLP interested in automatically determining the syntactic structure of a sentence

parsing can also be thought of as determining what sentences are "valid" English sentences

22



Parsing

We have a grammar, determine the possible parse tree(s)

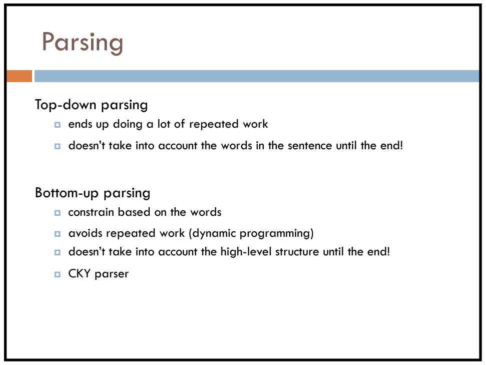
Let's start with parsing with a CFG (no probabilities)

S	→	NP VP
NP	→	PRP
NP	→	N PP
VP	→	V NP
VP	→	V NP PP
PP	→	IN N
PRP	→	I
V	→	eat
N	→	sushi
N	→	tuna
IN	→	with

I eat sushi with tuna

approaches?
algorithms?

23

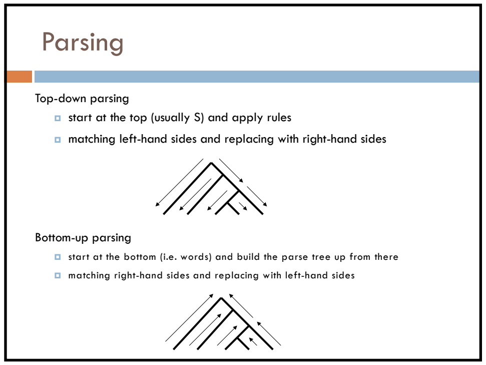


Parsing

Top-down parsing
- ends up doing a lot of repeated work
- doesn't take into account the words in the sentence until the end!

Bottom-up parsing
- constrain based on the words
- avoids repeated work (dynamic programming)
- doesn't take into account the high-level structure until the end!
- CKY parser

24



Parsing

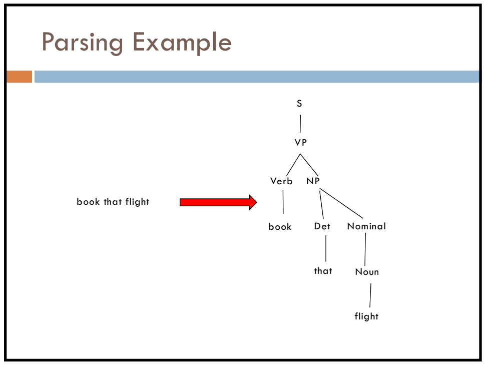
Top-down parsing

- start at the top (usually S) and apply rules
- matching left-hand sides and replacing with right-hand sides

Bottom-up parsing

- start at the bottom (i.e. words) and build the parse tree up from there
- matching right-hand sides and replacing with left-hand sides

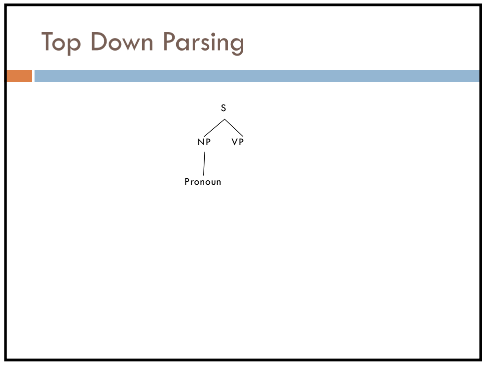
25



Parsing Example

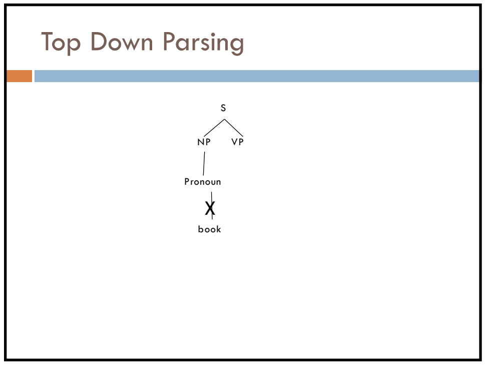
book that flight

26



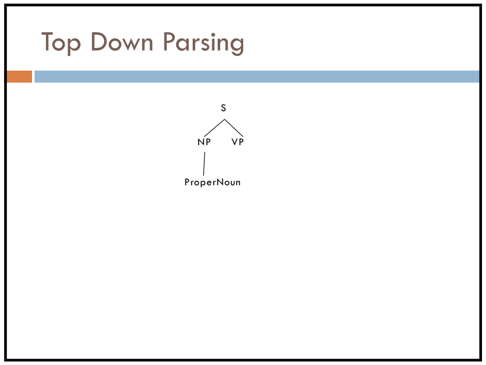
Top Down Parsing

27



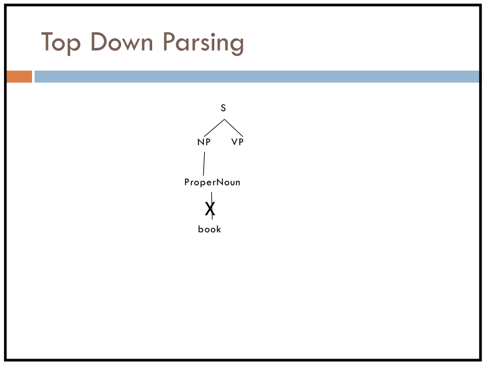
Top Down Parsing

28



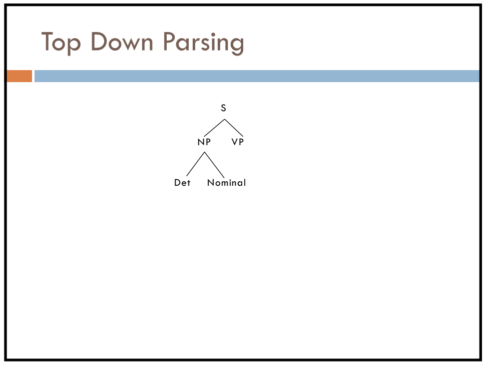
Top Down Parsing

29



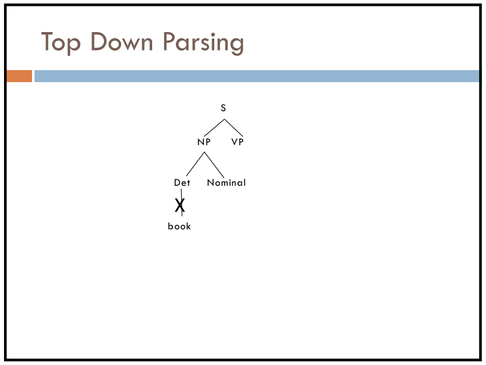
Top Down Parsing

30



Top Down Parsing

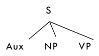
31



Top Down Parsing

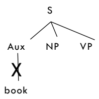
32

Top Down Parsing



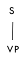
33

Top Down Parsing



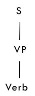
34

Top Down Parsing



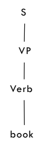
35

Top Down Parsing



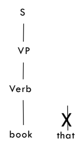
36

Top Down Parsing



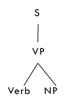
37

Top Down Parsing



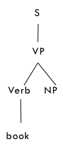
38

Top Down Parsing



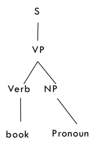
39

Top Down Parsing



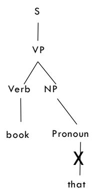
40

Top Down Parsing



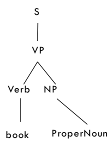
41

Top Down Parsing



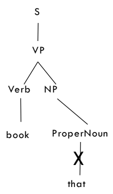
42

Top Down Parsing



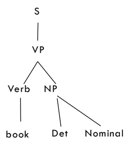
43

Top Down Parsing



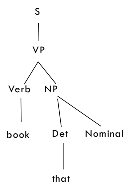
44

Top Down Parsing



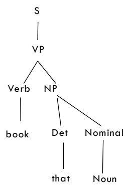
45

Top Down Parsing



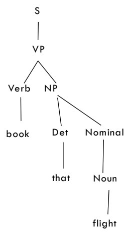
46

Top Down Parsing

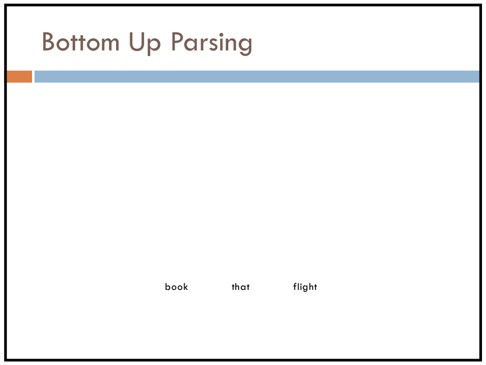


47

Top Down Parsing

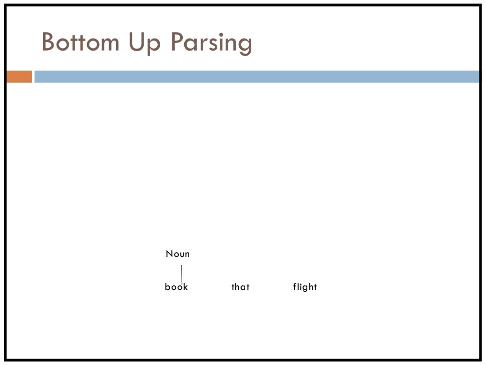


48



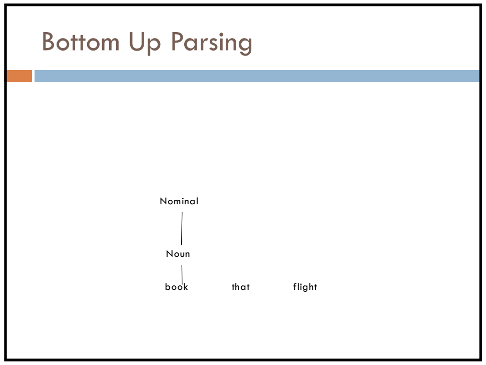
Bottom Up Parsing

49



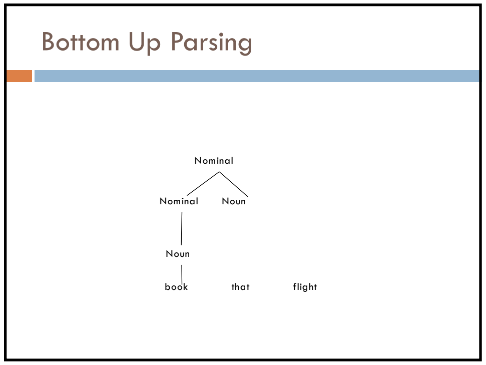
Bottom Up Parsing

50



Bottom Up Parsing

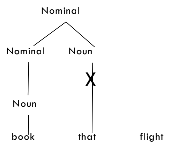
51



Bottom Up Parsing

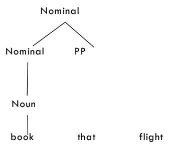
52

Bottom Up Parsing



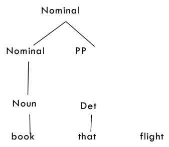
53

Bottom Up Parsing



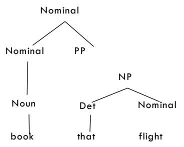
54

Bottom Up Parsing

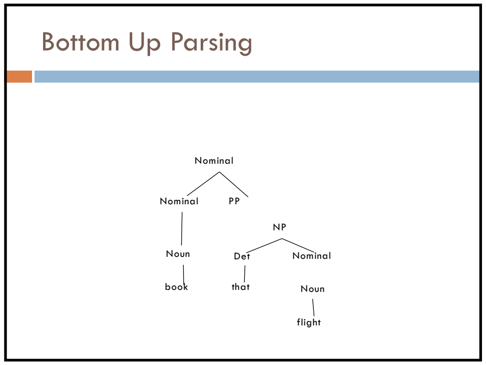


55

Bottom Up Parsing

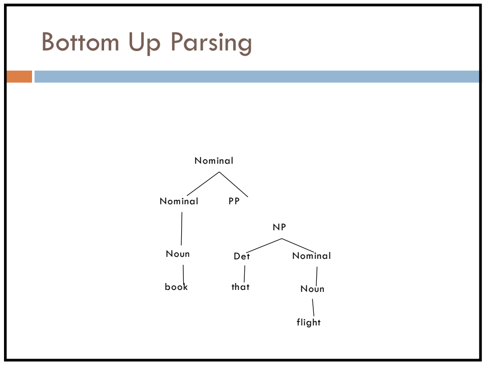


56



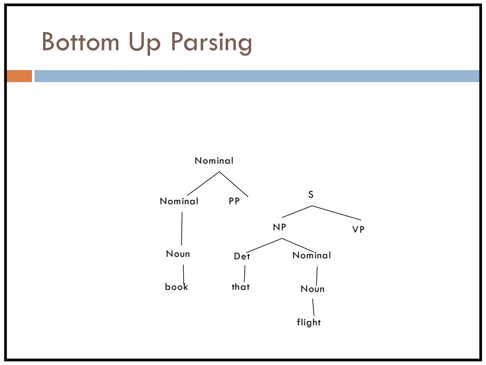
Bottom Up Parsing

57



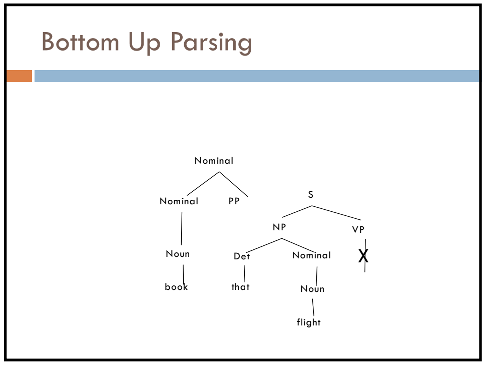
Bottom Up Parsing

58



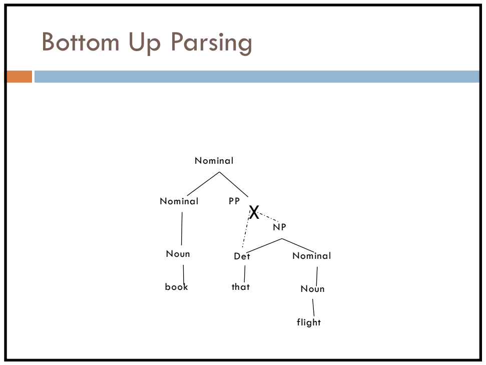
Bottom Up Parsing

59



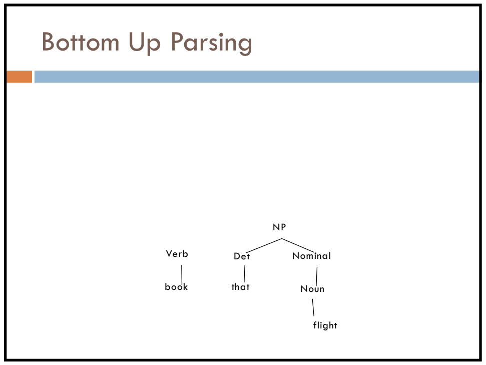
Bottom Up Parsing

60



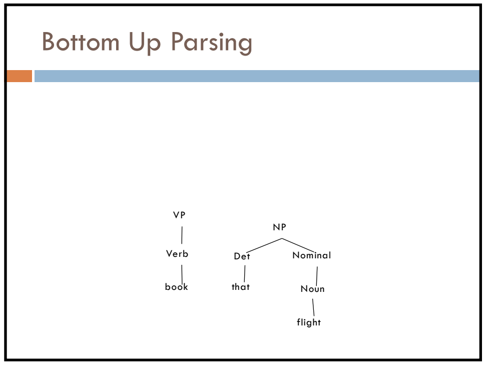
Bottom Up Parsing

61



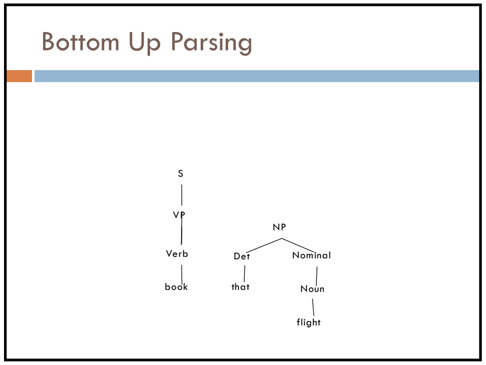
Bottom Up Parsing

62



Bottom Up Parsing

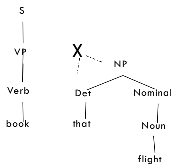
63



Bottom Up Parsing

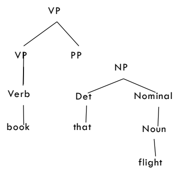
64

Bottom Up Parsing



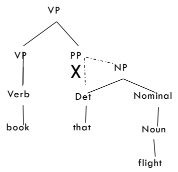
65

Bottom Up Parsing



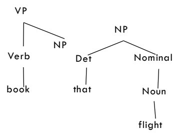
66

Bottom Up Parsing

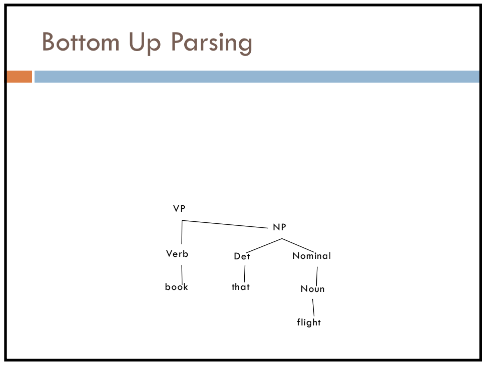


67

Bottom Up Parsing

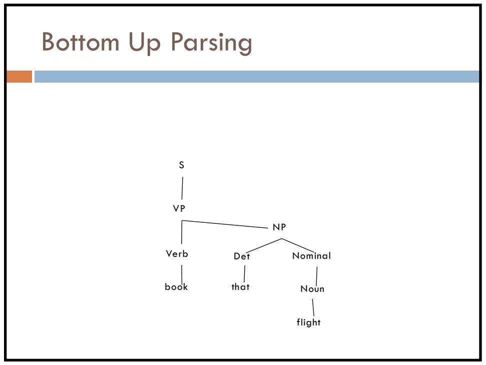


68



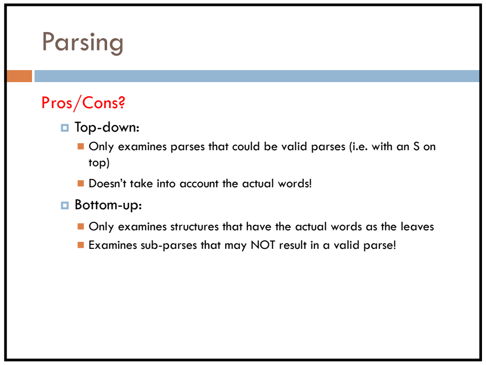
Bottom Up Parsing

69



Bottom Up Parsing

70

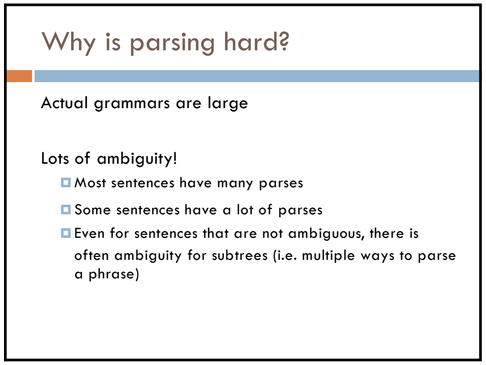


Parsing

Pros/Cons?

- Top-down:
  - Only examines parses that could be valid parses (i.e. with an S on top)
  - Doesn't take into account the actual words!
- Bottom-up:
  - Only examines structures that have the actual words as the leaves
  - Examines sub-parses that may NOT result in a valid parse!

71



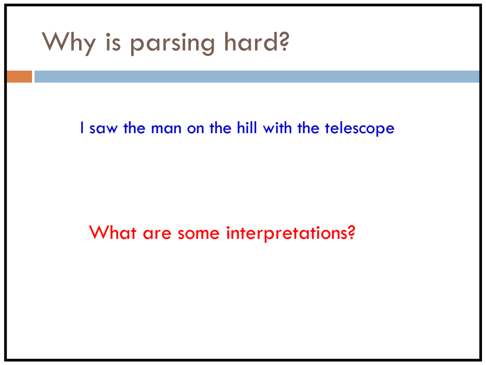
Why is parsing hard?

Actual grammars are large

Lots of ambiguity!

- Most sentences have many parses
- Some sentences have a lot of parses
- Even for sentences that are not ambiguous, there is often ambiguity for subtrees (i.e. multiple ways to parse a phrase)

72

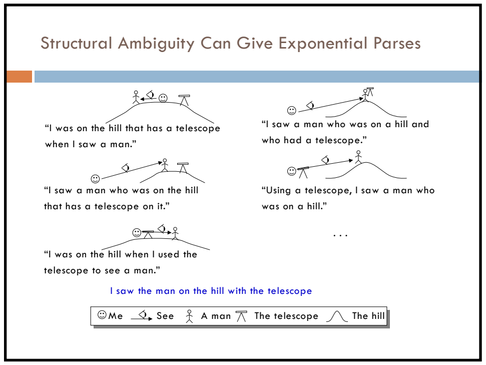


Why is parsing hard?

I saw the man on the hill with the telescope

What are some interpretations?

73



Structural Ambiguity Can Give Exponential Parses

"I was on the hill that has a telescope when I saw a man."

"I saw a man who was on a hill and who had a telescope."

"I saw a man who was on the hill that has a telescope on it."

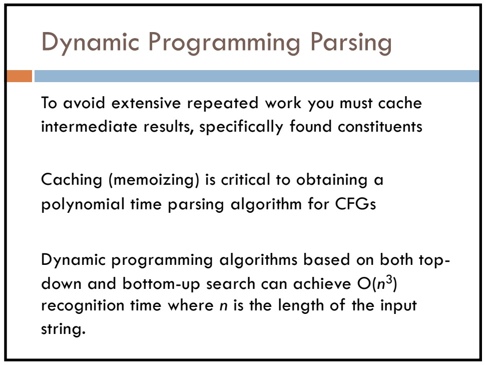
"Using a telescope, I saw a man who was on a hill."

"I was on the hill when I used the telescope to see a man."

. . .

I saw the man on the hill with the telescope

74



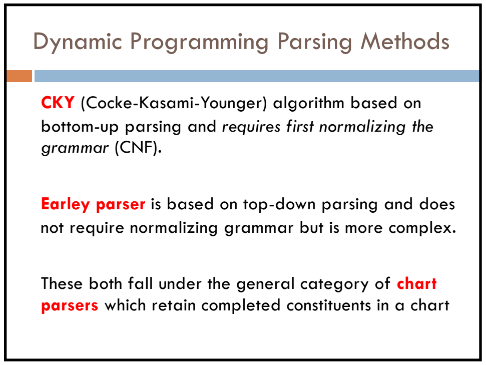
Dynamic Programming Parsing

To avoid extensive repeated work you must cache intermediate results, specifically found constituents

Caching (memoizing) is critical to obtaining a polynomial time parsing algorithm for CFGs

Dynamic programming algorithms based on both top-down and bottom-up search can achieve O(n^3) recognition time where *n* is the length of the input string.

75



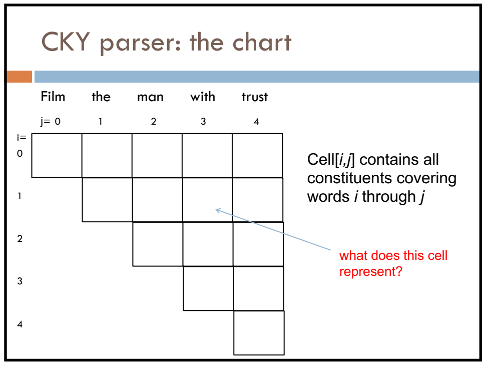
Dynamic Programming Parsing Methods

CKY (Cocke-Kasami-Younger) algorithm based on bottom-up parsing and *requires first normalizing the grammar* (CNF).

Earley parser is based on top-down parsing and does not require normalizing grammar but is more complex.

These both fall under the general category of **chart parsers** which retain completed constituents in a chart

76



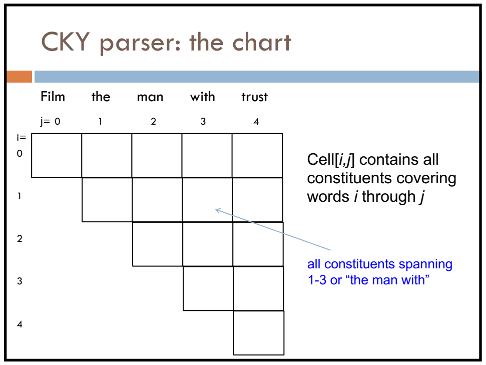
CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0					
1					
2					
3					
4					

Cell[*i*,*j*] contains all constituents covering words *i* through *j*

what does this cell represent?

77



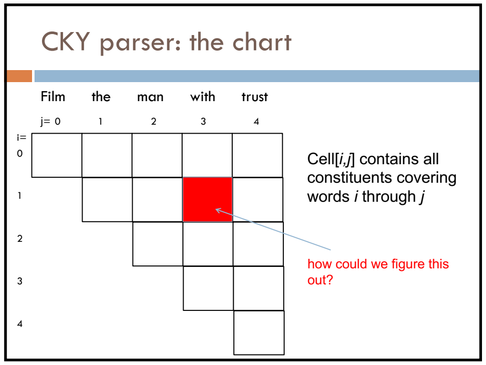
CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0					
1					
2					
3					
4					

Cell[*i*,*j*] contains all constituents covering words *i* through *j*

all constituents spanning 1-3 or "the man with"

78



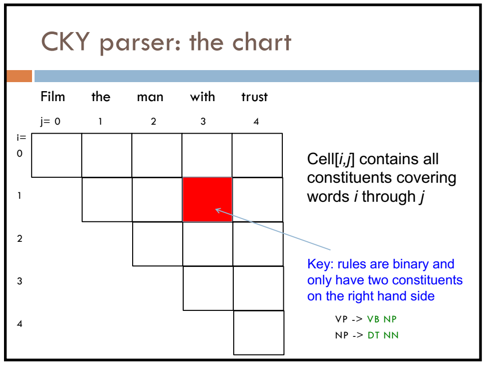
CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0					
1				🟥	
2					
3					
4					

Cell[*i*,*j*] contains all constituents covering words *i* through *j*

how could we figure this out?

79



CKY parser: the chart

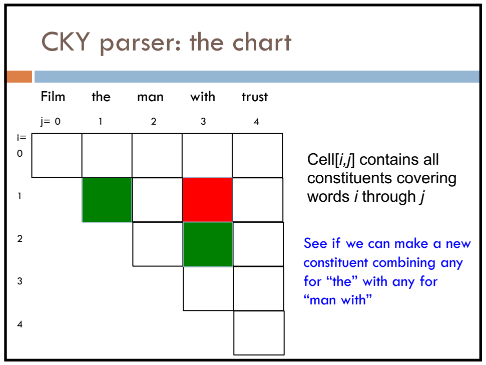
	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0					
1				🟥	
2					
3					
4					

Cell[*i*,*j*] contains all constituents covering words *i* through *j*

Key: rules are binary and only have two constituents on the right hand side

VP -> VB NP
NP -> DT NN

80



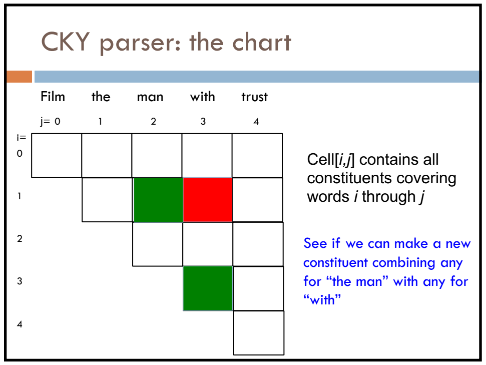
CKY parser: the chart

	Film	the	man	with	trust
i=	j= 0	1	2	3	4
0					
1		🟩		🟥	
2				🟩	
3					
4					

Cell[*i*,*j*] contains all constituents covering words *i* through *j*

See if we can make a new constituent combining any for "the" with any for "man with"

81



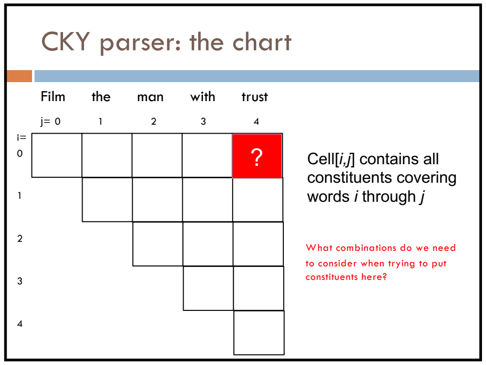
CKY parser: the chart

	Film	the	man	with	trust
i=	j= 0	1	2	3	4
0					
1			🟩	🟥	
2					
3				🟩	
4					

Cell[*i*,*j*] contains all constituents covering words *i* through *j*

See if we can make a new constituent combining any for "the man" with any for "with"

82



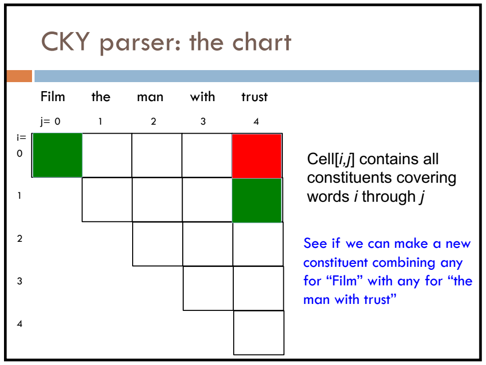
CKY parser: the chart

	Film	the	man	with	trust
i=	j= 0	1	2	3	4
0					🟥 ?
1					
2					
3					
4					

Cell[*i*,*j*] contains all constituents covering words *i* through *j*

What combinations do we need to consider when trying to put constituents here?

83



CKY parser: the chart

	Film	the	man	with	trust
i=	j= 0	1	2	3	4
0	🟩				🟥
1					🟩
2					
3					
4					

Cell[*i*,*j*] contains all constituents covering words *i* through *j*

See if we can make a new constituent combining any for "Film" with any for "the man with trust"

84

CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0		🟩			🟥
1					
2					🟩
3					
4					

Cell[*i,j*] contains all constituents covering words *i* through *j*

See if we can make a new constituent combining any for “Film the” with any for “man with trust”

85

CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0			🟩		🟥
1					
2					
3					🟩
4					

Cell[*i,j*] contains all constituents covering words *i* through *j*

See if we can make a new constituent combining any for “Film the man” with any for “with trust”

86

CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0				🟩	🟥
1					
2					
3					
4					🟩

Cell[*i,j*] contains all constituents covering words *i* through *j*

See if we can make a new constituent combining any for “Film the man with” with any for “trust”

87

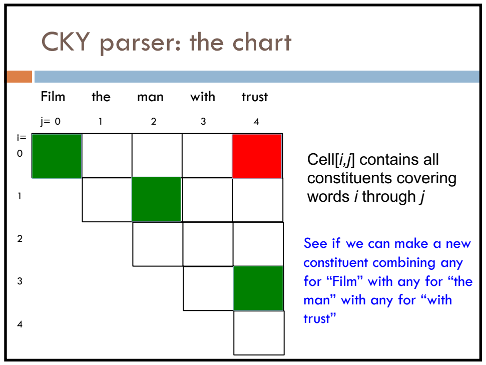
CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0					🟥
1					
2					
3					
4					

Cell[*i,j*] contains all constituents covering words *i* through *j*

What if our rules weren’t binary?

88



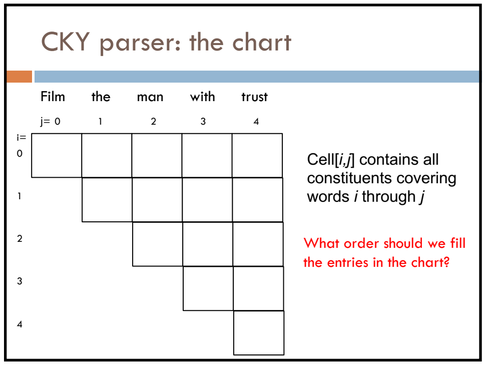
CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0	🟩				🟥
1			🟩		
2					
3					🟩
4					

Cell[*i,j*] contains all constituents covering words *i* through *j*

See if we can make a new constituent combining any for "Film" with any for "the man" with any for "with trust"

89



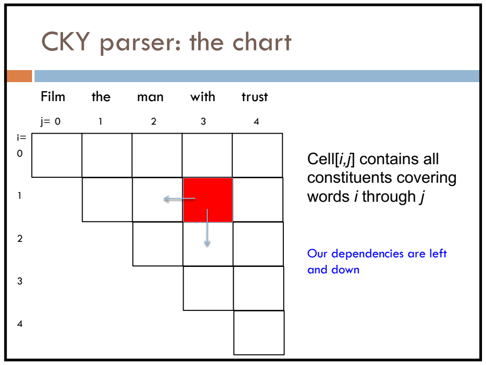
CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0					
1					
2					
3					
4					

Cell[*i,j*] contains all constituents covering words *i* through *j*

What order should we fill the entries in the chart?

90



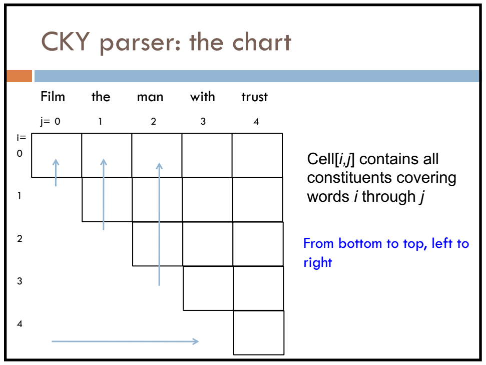
CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0					
1				🟥	
2					
3					
4					

Cell[*i,j*] contains all constituents covering words *i* through *j*

Our dependencies are left and down

91



CKY parser: the chart

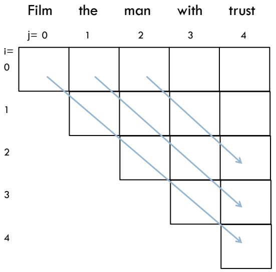
	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0					
1					
2					
3					
4					

Cell[*i,j*] contains all constituents covering words *i* through *j*

From bottom to top, left to right

92

CKY parser: the chart



Cell[*i*,*j*] contains all constituents covering words *i* through *j*

Top-left along the diagonals moving to the right

93

CKY parser: unary rules

S -> VP
VP -> VB NP
VP -> VP2 PP
VP2 -> VB NP
NP -> DT NN
NP -> NN
NP -> NP PP
PP -> IN NP
DT -> the
IN -> with
VB -> film
VB -> trust
NN -> man
NN -> film
NN -> trust

Often, we will leave unary rules rather than converting to CNF

Do these complicate the algorithm?

Must check whenever we add a constituent to see if any unary rules apply

94

CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0					
1					
2					
3					
4					

S → VP
VP → VB NP
VP → VP2 PP
VP2 → VB NP
NP → DT NN
NP → NN
NP → NP PP
PP → IN NP
DT → the
IN → with
VB → film
VB → man
VB → trust
NN → man
NN → film
NN → trust

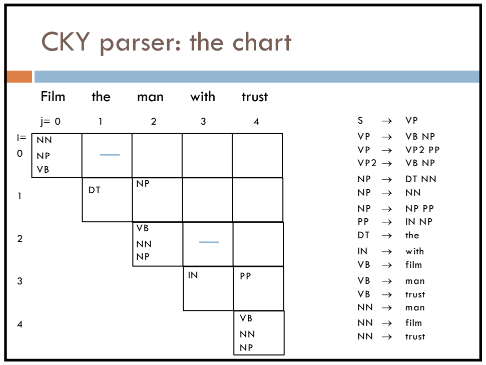
95

CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0	NN NP VB				
1		DT			
2			VB NN NP		
3				IN	
4					VB NN NP

S → VP
VP → VB NP
VP → VP2 PP
VP2 → VB NP
NP → DT NN
NP → NN
NP → NP PP
PP → IN NP
DT → the
IN → with
VB → film
VB → man
VB → trust
NN → man
NN → film
NN → trust

96

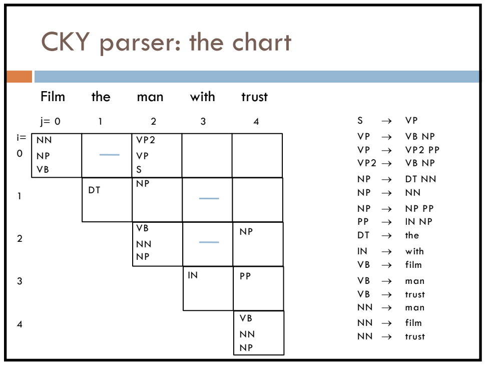


CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0	NN NP VB	—			
1		DT	NP		
2			VB NN NP	—	
3				IN	PP
4					VB NN NP

S	→	VP
VP	→	VB NP
VP	→	VP2 PP
VP2	→	VB NP
NP	→	DT NN
NP	→	NN
NP	→	NP PP
PP	→	IN NP
DT	→	the
IN	→	with
VB	→	film
VB	→	man
VB	→	trust
NN	→	man
NN	→	film
NN	→	trust

97

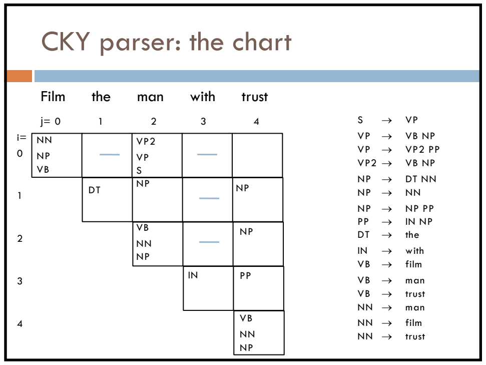


CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0	NN NP VB	—	VP2 VP S		
1		DT	NP	—	
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

S	→	VP
VP	→	VB NP
VP	→	VP2 PP
VP2	→	VB NP
NP	→	DT NN
NP	→	NN
NP	→	NP PP
PP	→	IN NP
DT	→	the
IN	→	with
VB	→	film
VB	→	man
VB	→	trust
NN	→	man
NN	→	film
NN	→	trust

98

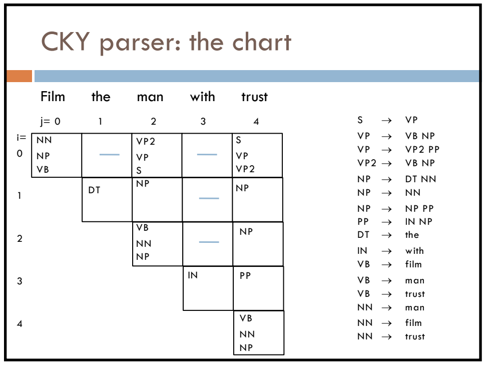


CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0	NN NP VB	—	VP2 VP S	—	
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

S	→	VP
VP	→	VB NP
VP	→	VP2 PP
VP2	→	VB NP
NP	→	DT NN
NP	→	NN
NP	→	NP PP
PP	→	IN NP
DT	→	the
IN	→	with
VB	→	film
VB	→	man
VB	→	trust
NN	→	man
NN	→	film
NN	→	trust

99

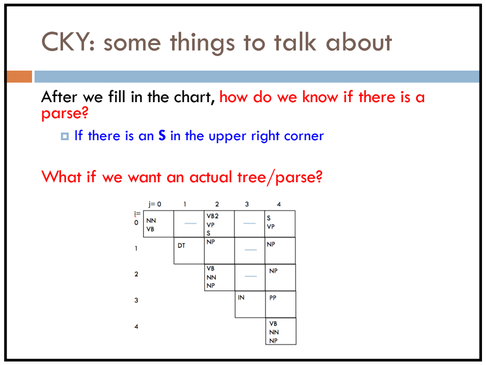


CKY parser: the chart

	Film	the	man	with	trust
	j= 0	1	2	3	4
i= 0	NN NP VB	—	VP2 VP S	—	S VP VP2
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

S	→	VP
VP	→	VB NP
VP	→	VP2 PP
VP2	→	VB NP
NP	→	DT NN
NP	→	NN
NP	→	NP PP
PP	→	IN NP
DT	→	the
IN	→	with
VB	→	film
VB	→	man
VB	→	trust
NN	→	man
NN	→	film
NN	→	trust

100



CKY: some things to talk about

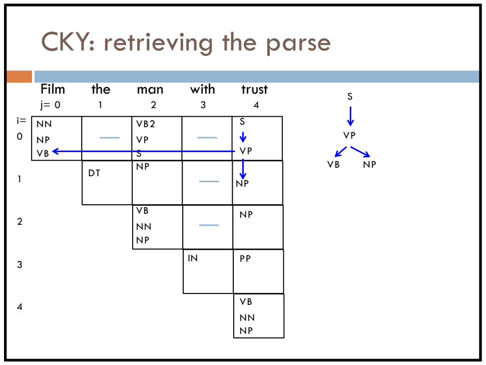
After we fill in the chart, how do we know if there is a parse?

- If there is an **S** in the upper right corner

What if we want an actual tree/parse?

i=	j= 0	1	2	3	4
0	NN VB	—	VB2 VP S	—	S VP
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

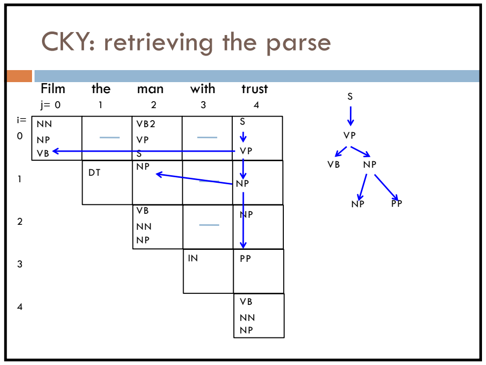
101



CKY: retrieving the parse

i=	Film j= 0	the 1	man 2	with 3	trust 4
0	NN NP VB	—	VB2 VP S	—	S VP
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

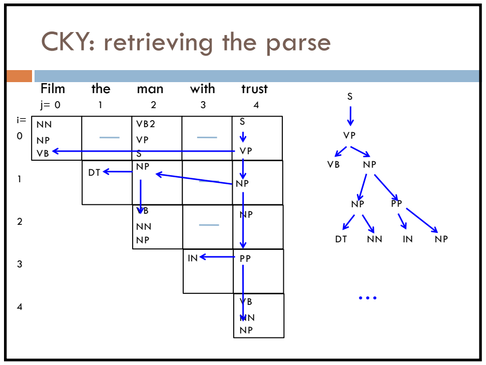
102



CKY: retrieving the parse

i=	Film j= 0	the 1	man 2	with 3	trust 4
0	NN NP VB	—	VB2 VP S	—	S VP
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

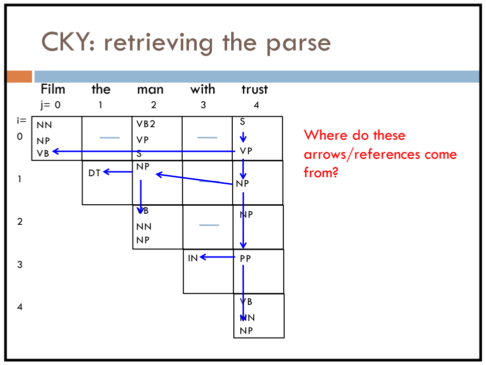
103



CKY: retrieving the parse

i=	Film j= 0	the 1	man 2	with 3	trust 4
0	NN NP VB	—	VB2 VP S	—	S VP
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

104

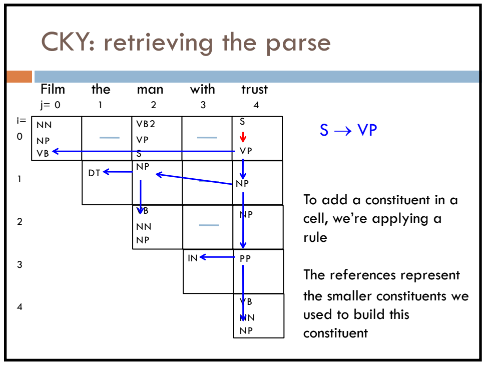


CKY: retrieving the parse

i=	Film j= 0	the 1	man 2	with 3	trust 4
0	NN NP VB	—	VB2 VP S	—	S VP
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

Where do these arrows/references come from?

105



CKY: retrieving the parse

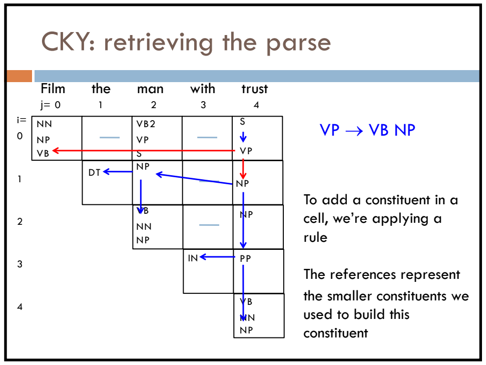
i=	Film j= 0	the 1	man 2	with 3	trust 4
0	NN NP VB	—	VB2 VP S	—	S VP
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

S → VP

To add a constituent in a cell, we're applying a rule

The references represent the smaller constituents we used to build this constituent

106



CKY: retrieving the parse

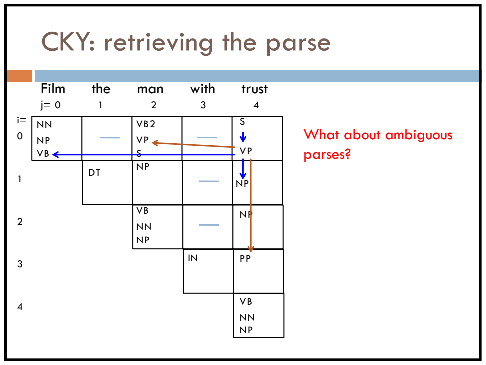
i=	Film j= 0	the 1	man 2	with 3	trust 4
0	NN NP VB	—	VB2 VP S	—	S VP
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

VP → VB NP

To add a constituent in a cell, we're applying a rule

The references represent the smaller constituents we used to build this constituent

107

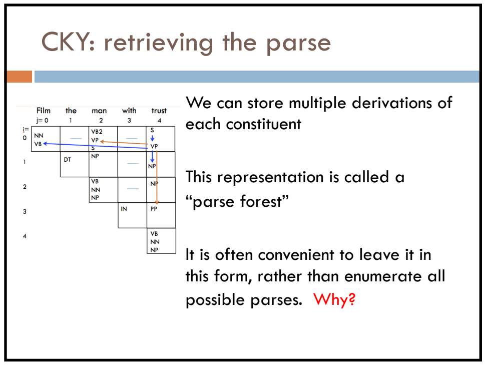


CKY: retrieving the parse

i=	Film j= 0	the 1	man 2	with 3	trust 4
0	NN NP VB	—	VB2 VP S	—	S VP
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

What about ambiguous parses?

108



CKY: retrieving the parse

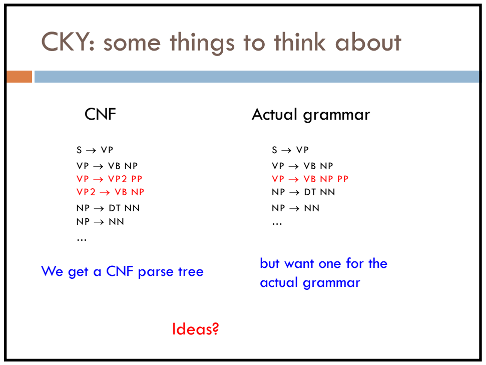
i=	Film j= 0	the 1	man 2	with 3	trust 4
0	NN VB	—	VB2 VP S	—	S VP
1		DT	NP	—	NP
2			VB NN NP	—	NP
3				IN	PP
4					VB NN NP

We can store multiple derivations of each constituent

This representation is called a "parse forest"

It is often convenient to leave it in this form, rather than enumerate all possible parses. Why?

109



CKY: some things to think about

CNF

S → VP
VP → VB NP
VP → VP2 PP
VP2 → VB NP
NP → DT NN
NP → NN
…

Actual grammar

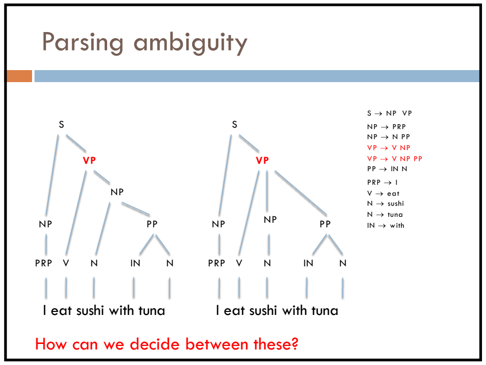
S → VP
VP → VB NP
VP → VB NP PP
NP → DT NN
NP → NN
…

We get a CNF parse tree

but want one for the actual grammar

Ideas?

110

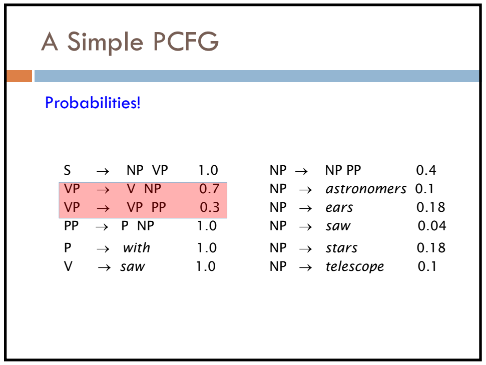


Parsing ambiguity

S → NP VP
NP → PRP
NP → N PP
VP → V NP
VP → V NP PP
PP → IN N
PRP → I
V → eat
N → sushi
N → tuna
IN → with

How can we decide between these?

111

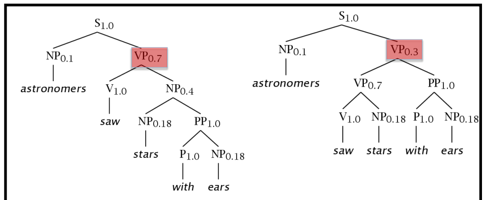


A Simple PCFG

Probabilities!

S	→	NP VP	1.0	NP	→	NP PP	0.4
VP	→	V NP	0.7	NP	→	*astronomers*	0.1
VP	→	VP PP	0.3	NP	→	*ears*	0.18
PP	→	P NP	1.0	NP	→	*saw*	0.04
P	→	*with*	1.0	NP	→	*stars*	0.18
V	→	*saw*	1.0	NP	→	*telescope*	0.1

112



= 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18
* 1.0 * 1.0 * 0.18
= 0.0009072

= 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18
* 1.0 * 1.0 * 0.18
= 0.0006804

113

Parsing with PCFGs

How does this change our CKY algorithm?

- We need to keep track of the probability of a constituent

How do we calculate the probability of a constituent?

- Product of the PCFG rule times the product of the probabilities of the sub-constituents (right hand sides)
- Building up the product from the bottom-up

What if there are multiple ways of deriving a particular constituent?

- max: pick the most likely derivation of that constituent

114

Probabilistic CKY

Include in each cell a probability for each non-terminal

Cell[*i*,*j*] must retain the *most probable* derivation of each constituent (non-terminal) covering words *i* through *j*

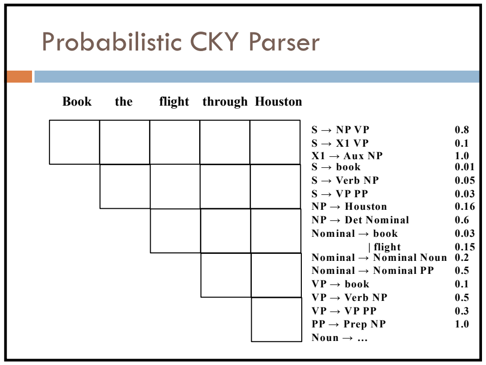
When transforming the grammar to CNF, must set production probabilities to preserve the probability of derivations

115

Probabilistic Grammar Conversion

Original Grammar		Chomsky Normal Form	
S → NP VP	0.8	S → NP VP	0.8
S → Aux NP VP	0.1	S → X1 VP X1 → Aux NP	0.1 1.0
S → VP	0.1	S → book | include | prefer 0.01 0.004 0.006	
		S → Verb NP	0.05
		S → VP PP	0.03
NP → Pronoun	0.2	NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Proper-Noun	0.2	NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6	NP → Det Nominal	0.6
Nominal → Noun	0.3	Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2	Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5	Nominal → Nominal PP	0.5
VP → Verb	0.2	VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5	VP → Verb NP	0.5
VP → VP PP	0.3	VP → VP PP	0.3
PP → Prep NP	1.0	PP → Prep NP	1.0

116

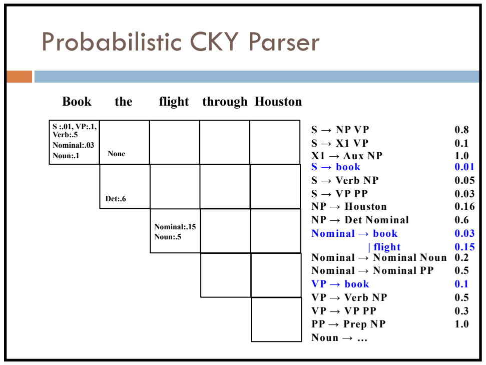


Probabilistic CKY Parser

Book	the	flight	through	Houston

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

117

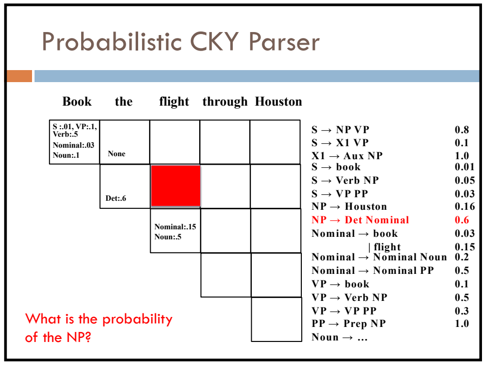


Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None			
	Det:.6			
		Nominal:.15 Noun:.5		

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

118



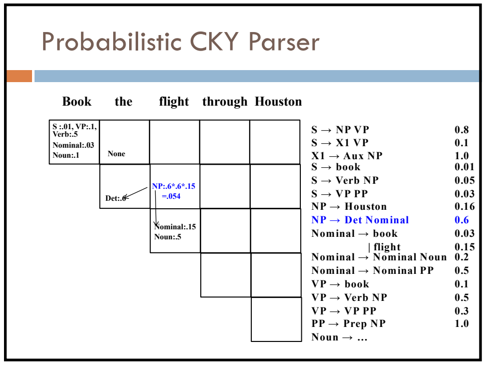
Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None			
	Det:.6	🟥		
		Nominal:.15 Noun:.5		

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

What is the probability of the NP?

119

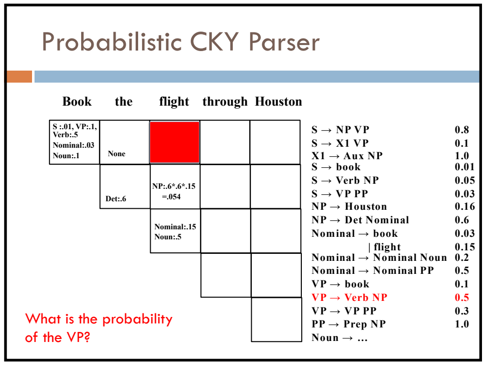


Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None			
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

120



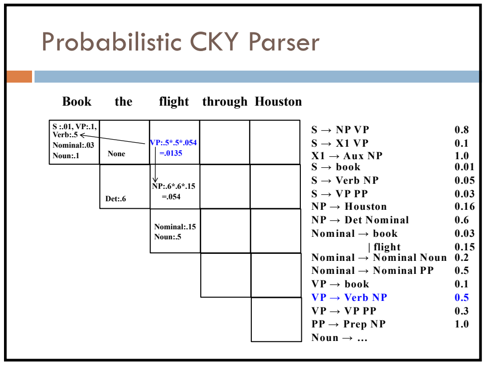
Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	🟥		
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

What is the probability of the VP?

121

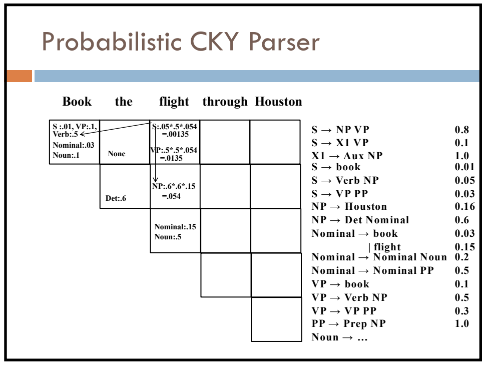


Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	VP:.5*.5*.054 =.0135		
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

122

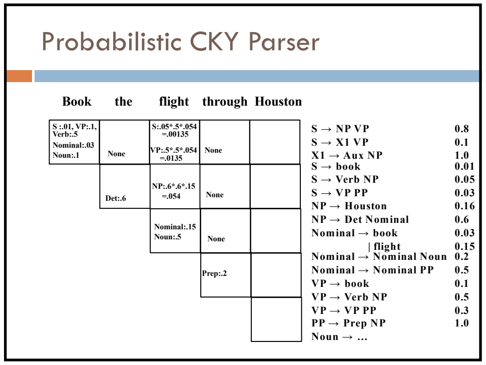


Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135		
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

123

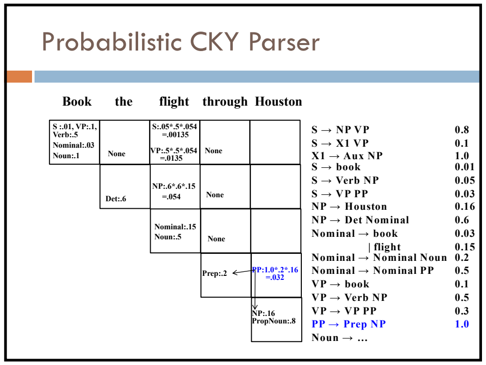


Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

124

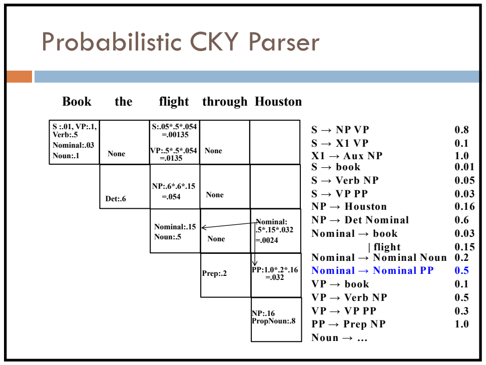


Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

125

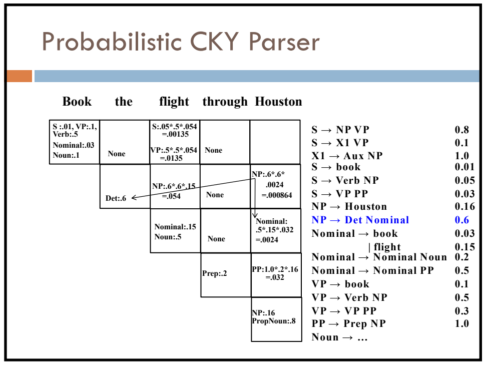


Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

126

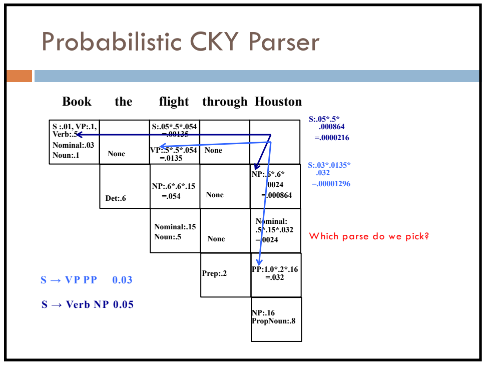


Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → ...	

127



Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S:.05*.5*
.000864
=.0000216

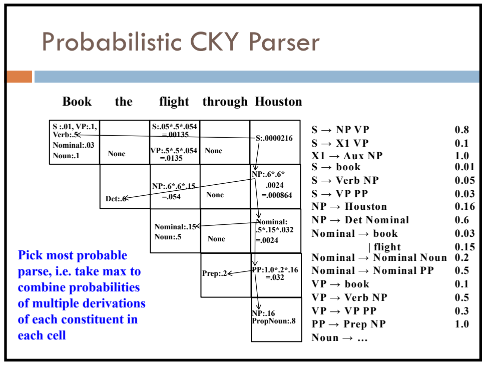
S:.03*.0135*
.032
=.00001296

Which parse do we pick?

S → VP PP 0.03

S → Verb NP 0.05

128



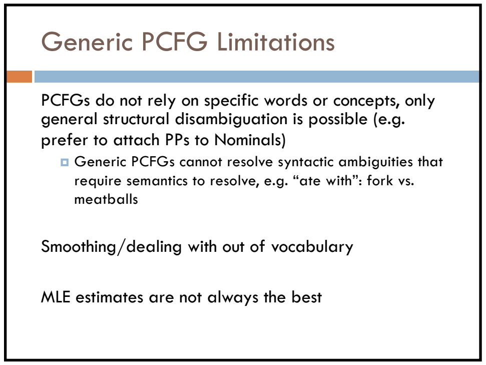
Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book	0.01
S → Verb NP	0.05
S → VP PP	0.03
NP → Houston	0.16
NP → Det Nominal	0.6
Nominal → book	0.03
| flight	0.15
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book	0.1
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0
Noun → …	

Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell

129



Generic PCFG Limitations

PCFGs do not rely on specific words or concepts, only general structural disambiguation is possible (e.g. prefer to attach PPs to Nominals)

- Generic PCFGs cannot resolve syntactic ambiguities that require semantics to resolve, e.g. “ate with”: fork vs. meatballs

Smoothing/dealing with out of vocabulary

MLE estimates are not always the best

130