Probabilistic CFG

Including materials from Andrew McCallum - David Rodriguez-Velazquez

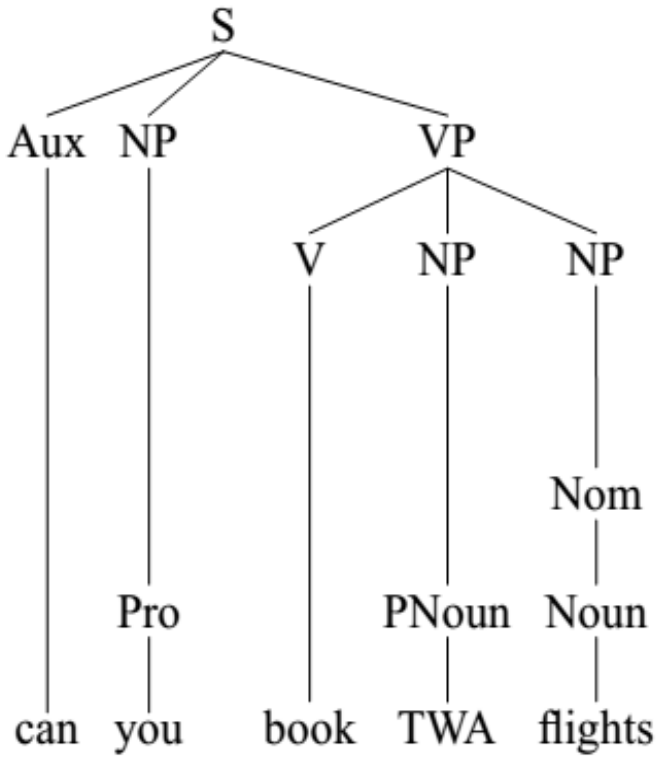
Ambiguity in Parsing

- Time flies like an arrow.
- Fruit flies like a banana.
- I saw the man with the telescope.

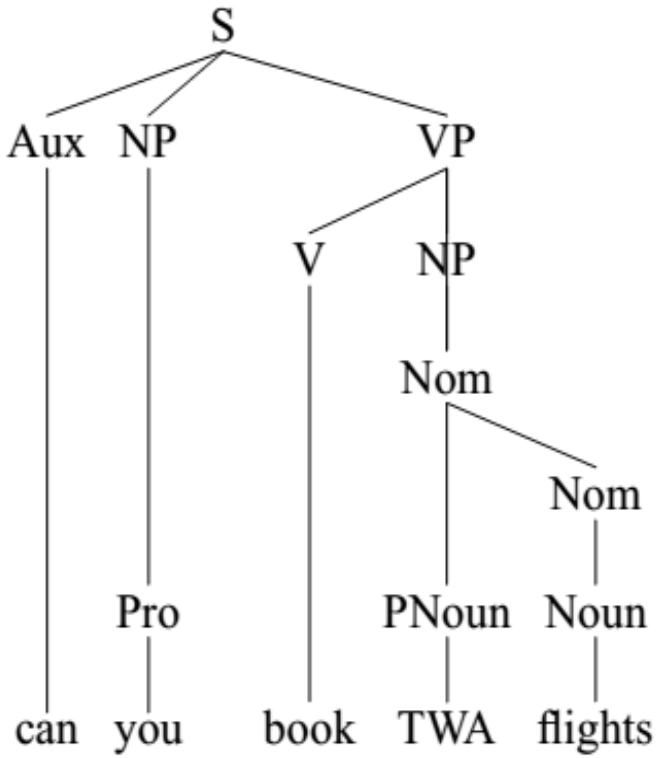
Probabilistic Augmentation

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.05] | the [.80] | a [.15]	
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.35]	$Verb \rightarrow book$	[.30]
$NP \rightarrow Nom$	[.05]	$Verb \rightarrow include$	[.30]
$NP \rightarrow Pronoun$	[.40]	$Verb \rightarrow want$	[.40]
$Nom \rightarrow Noun$	[.75]	$Aux \rightarrow can$	[.40]
$Nom \rightarrow Noun\ Nom$	[.20]	$Aux \rightarrow does$	[.30]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[.40]
$VP \rightarrow Verb\ NP\ NP$	[.05]	$Pronoun \rightarrow you$ [.40] | I [.60]	

(a)



(b)



Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP NP	.05
NP	→	Nom	.05
NP	→	PNoun	.35
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
PNoun	→	TWA	.40
Noun	→	flights	.50

Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP	.40
NP	→	Nom	.05
Nom	→	PNoun Nom	.05
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
Pnoun	→	TWA	.40
Noun	→	flights	.50

CYK ALGORITHM

The CYK Algorithm

- *The membership problem*:
  - Problem:
    - Given a context-free grammar **G** and a string **w**
      - **G** = (V, ∑ ,P , S) where
        - » V finite set of variables
        - » ∑ (the alphabet) finite set of terminal symbols
        - » P finite set of rules
        - » S start symbol (distinguished element of V)
        - » V and ∑ are assumed to be disjoint
      - **G** is used to generate the string of a language
  - Question:
    - Is **w** in **L**(**G**)?

The CYK Algorithm

- J. Cocke
- D. Younger,
- T. Kasami

  – Independently developed an algorithm to answer this question.

The CYK Algorithm Basics

- The Structure of the rules in a Chomsky Normal Form grammar
- Uses a “dynamic programming” or “table-filling algorithm”

Chomsky Normal Form

- *Normal Form* is described by a set of conditions that each rule in the grammar must satisfy
- Context-free grammar is in CNF if each rule has one of the following forms:
  - $A \rightarrow BC$ at most 2 symbols on right side
  - $A \rightarrow a$, or terminal symbol
  - $S \rightarrow \lambda$ null string

  where $B, C \in V - \{S\}$

Construct a Triangular Table

- Each row corresponds to one length of substrings
  - Bottom Row – Strings of length 1
  - Second from Bottom Row – Strings of length 2

    .

    .
  - Top Row – string ‘w’

Construct a Triangular Table

- $\mathbf{X_{i,\,i}}$ is the set of variables A such that
  A → w_i is a production of G

- Compare at most n pairs of previously computed sets:
  $(\mathbf{X_{i,\,i}}, \mathbf{X_{i+1,\,j}})$, $(\mathbf{X_{i,\,i+1}}, \mathbf{X_{i+2,\,j}})$ … $(\mathbf{X_{i,\,j-1}}, \mathbf{X_{j,\,j}})$

Construct a Triangular Table

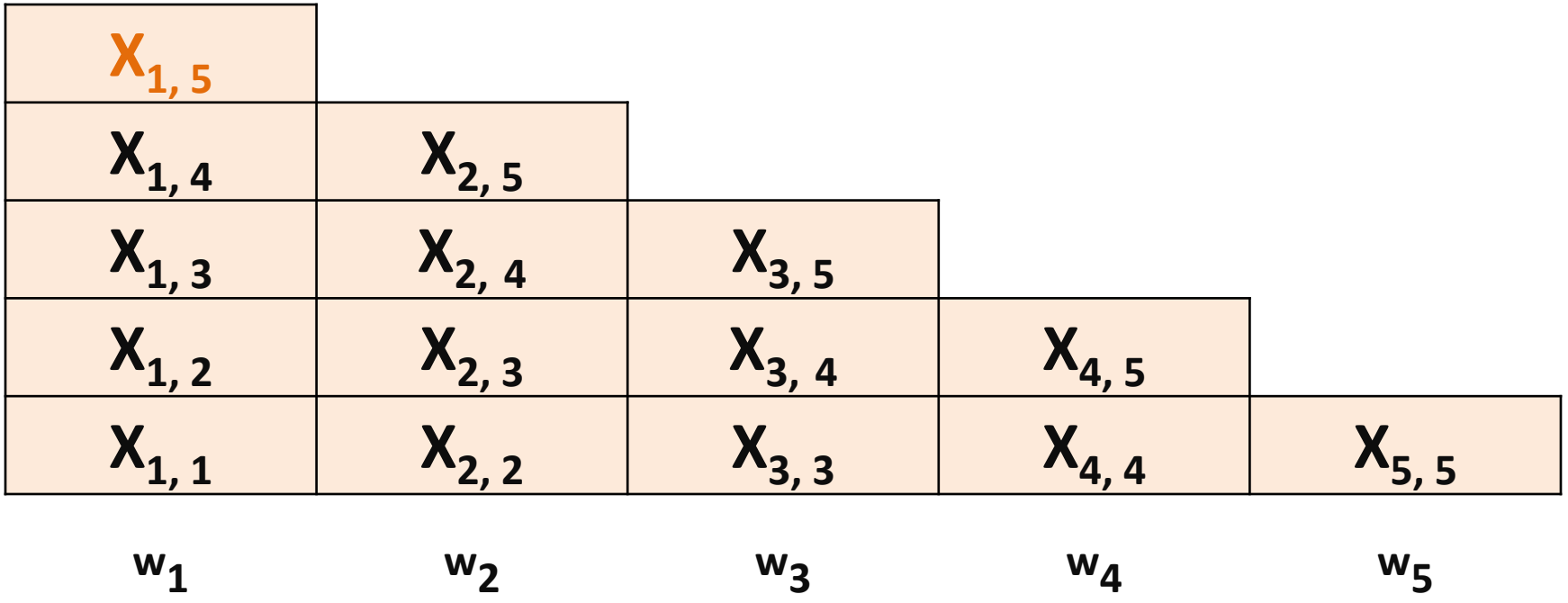
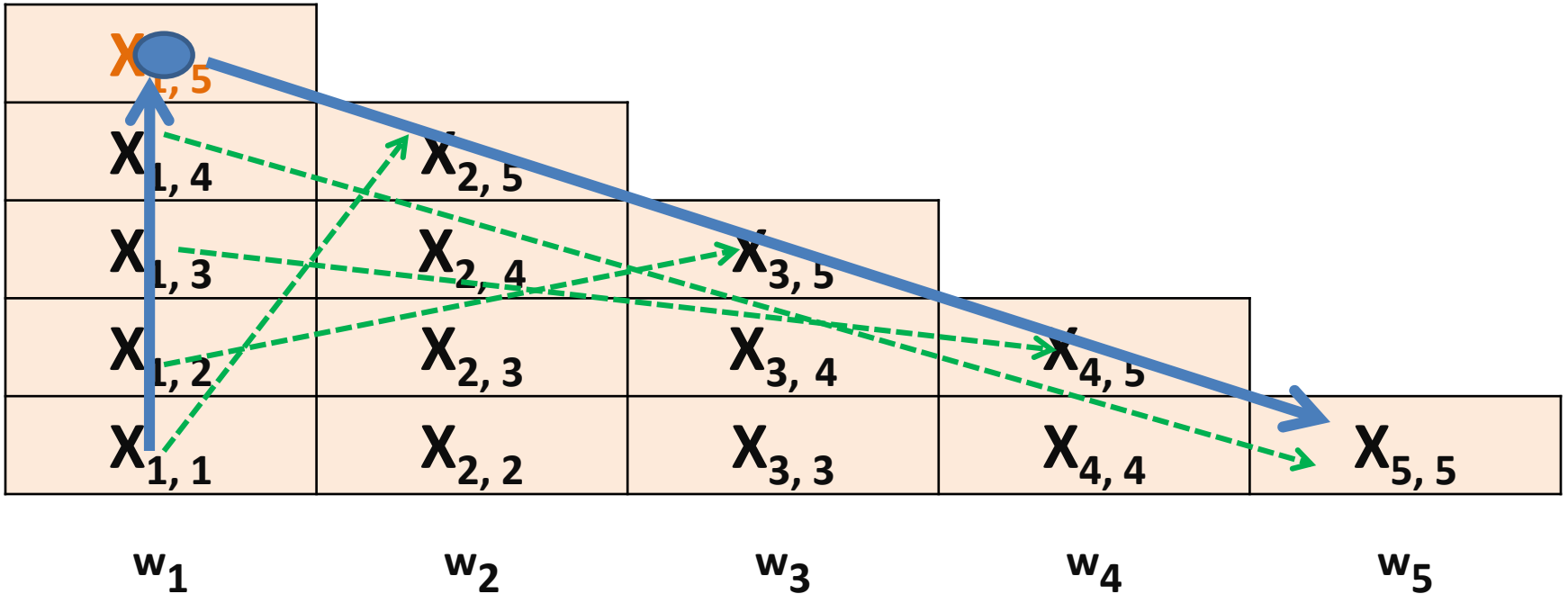


Table for string '**w**' that has length 5

Construct a Triangular Table



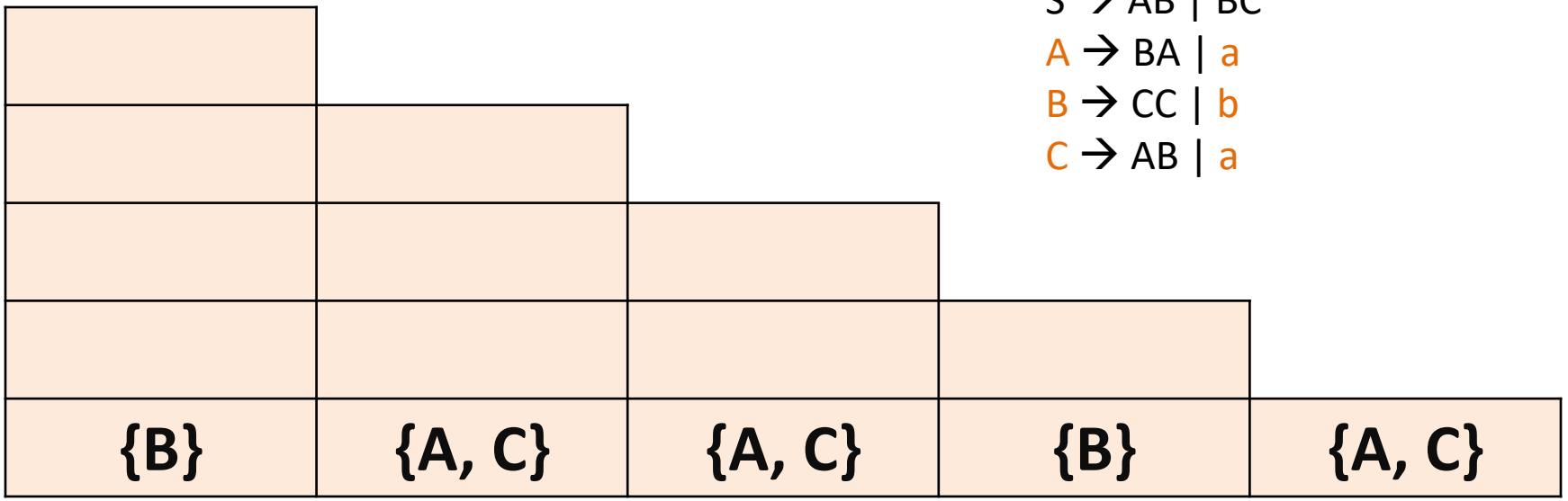
Looking for pairs to compare

Example CYK Algorithm

- Show the CYK Algorithm with the following example:
  - CNF grammar **G**
    - S → AB | BC
    - A → BA | a
    - B → CC | b
    - C → AB | a
  - **w** is baaba
  - Question Is **baaba** in L(G)?

Constructing The Triangular Table

S → AB | BC
A → BA | a
B → CC | b
C → AB | a



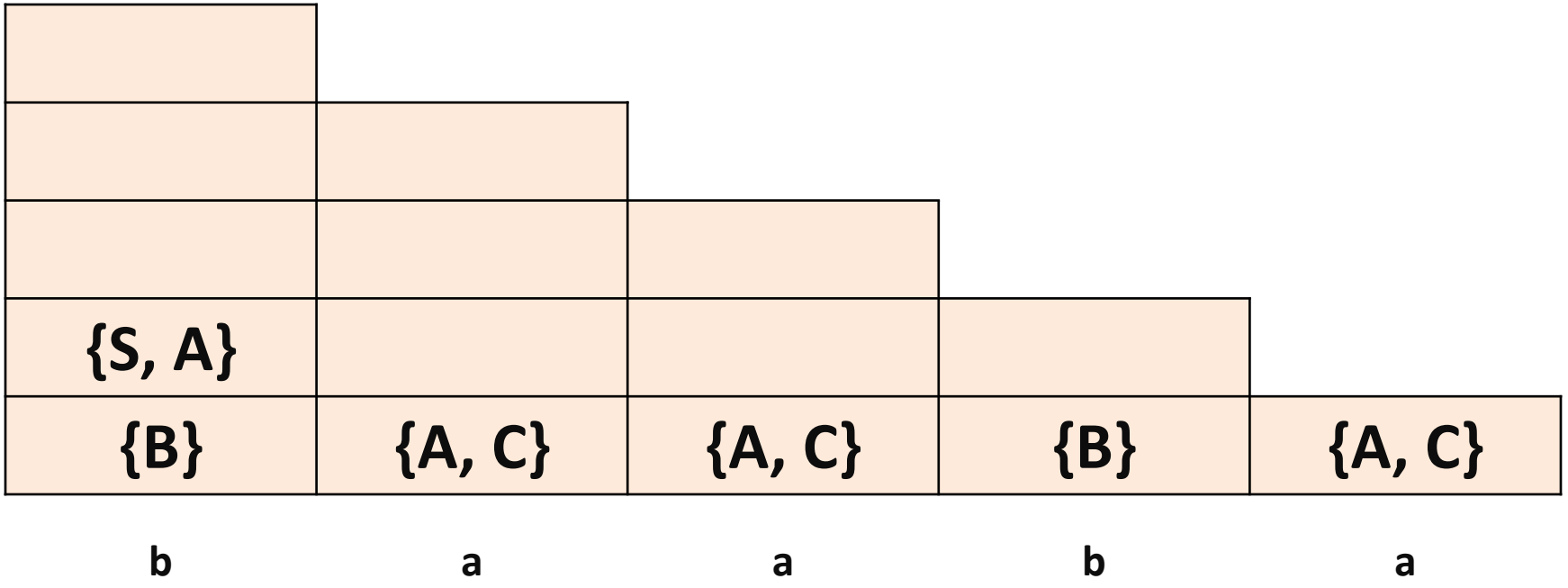
Calculating the Bottom ROW

Constructing The Triangular Table

- $X_{1,2} = (X_{i,i}, X_{i+1,j}) = (X_{1,1}, X_{2,2})$
- ➔ {B}{A,C} = {BA, BC}
- Steps:
  - Look for production rules to generate BA or BC
  - There are two: S and A
  - $X_{1,2} = \{S, A\}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

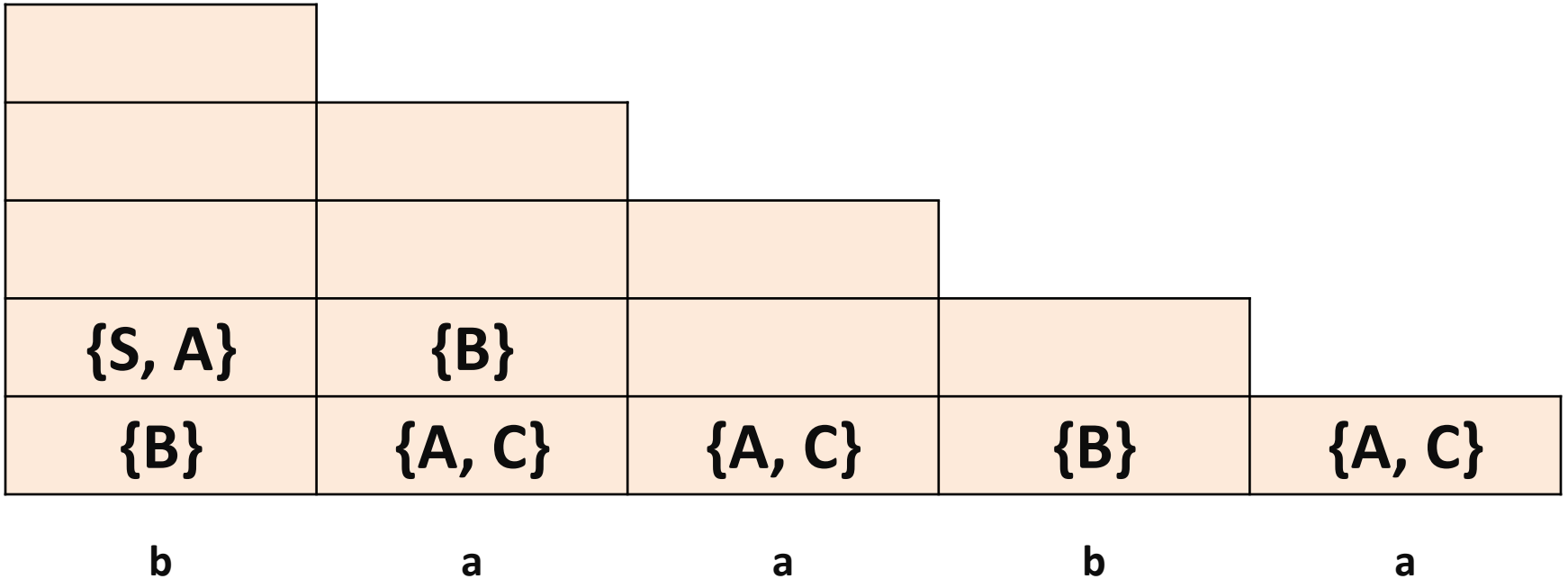


Constructing The Triangular Table

- $\mathbf{X_{2,3} = (X_{i,i}, X_{i+1,j}) = (X_{2,2}, X_{3,3})}$
- ➔ {A, C}{A,C} = {AA, AC, CA, CC} = Y
- Steps:
  - Look for production rules to generate Y
  - There is one: B
  - $\mathbf{X_{2,3} = \{B\}}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

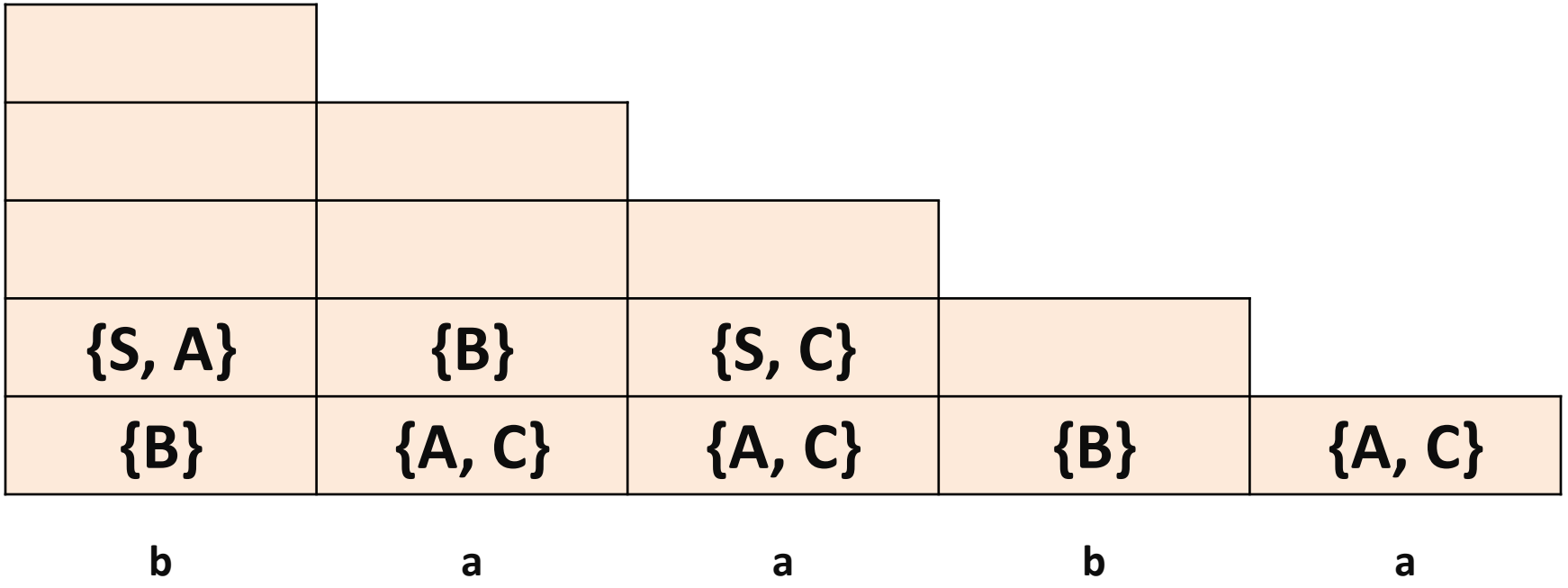


Constructing The Triangular Table

- $X_{3,4} = (X_{i,i}, X_{i+1,j}) = (X_{3,3}, X_{4,4})$
- ➔ {A, C}{B} = {AB, CB} = Y
- Steps:
  - Look for production rules to generate Y
  - There are two: S and C
  - $X_{3,4} = \{S, C\}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

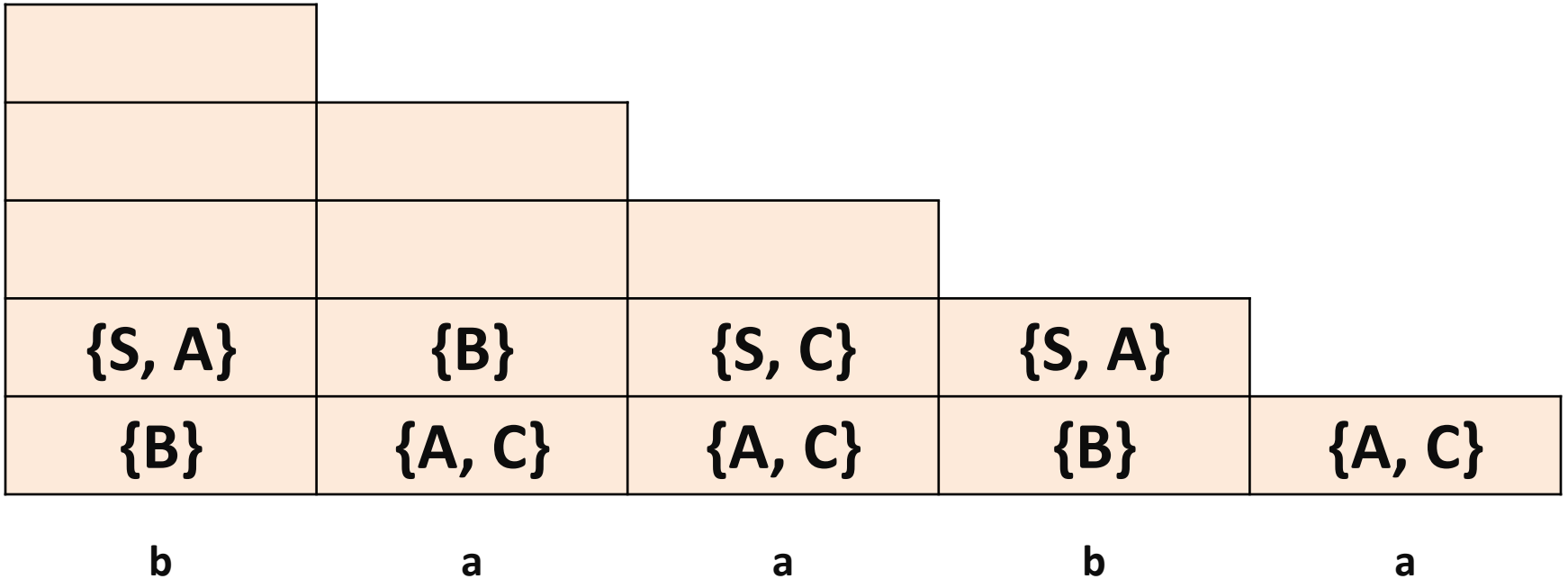


Constructing The Triangular Table

- $X_{4,5} = (X_{i,i}, X_{i+1,j}) = (X_{4,4}, X_{5,5})$
- ➔ {B}{A, C} = {BA, BC} = Y
- Steps:
  - Look for production rules to generate Y
  - There are two: S and A
  - $X_{4,5} = \{S, A\}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table



Constructing The Triangular Table

- $X_{1,3} = (X_{i,i}, X_{i+1,j})\,(X_{i,i+1}, X_{i+2,j})$
  $= (X_{1,1}, X_{2,3}), (X_{1,2}, X_{3,3})$
- ➔ {B}{B} **U** {S, A}{A, C}= {BB, SA, SC, AA, AC} = Y
- Steps:
  - Look for production rules to generate Y
  - There are NONE: S and A
  - $X_{1,3} = \emptyset$
  - **no elements in this set (empty set)**

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

Ø				
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

Constructing The Triangular Table

- $\mathbf{X_{2,4}} = (\mathbf{X_{i,i}}, \mathbf{X_{i+1,j}})\ (\mathbf{X_{i,i+1}}, \mathbf{X_{i+2,j}})$
  $= (\mathbf{X_{2,2}}, \mathbf{X_{3,4}}), (\mathbf{X_{2,3}}, \mathbf{X_{4,4}})$
- ➔ {A, C}{S, C} **U** {B}{B}= {AS, AC, CS, CC, BB} = Y
- Steps:
  - Look for production rules to generate Y
  - There is one: B
  - $\mathbf{X_{2,4}} = \mathbf{\{B\}}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

Ø	{B}			
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

Constructing The Triangular Table

- $X_{3,5} = (X_{i,i}, X_{i+1,j})\,(X_{i,i+1}, X_{i+2,j})$
  $= (X_{3,3}, X_{4,5}), (X_{3,4}, X_{5,5})$
- → {A,C}{S,A} **U** {S,C}{A,C}
  = {AS, AA, CS, CA, SA, SC, CA, CC} = Y
- Steps:
  - Look for production rules to generate Y
  - There is one: B
  - $X_{3,5} = \{B\}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

Ø	{B}	{B}		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

Final Triangular Table

{S, A, C}	← $X_{1,5}$			
Ø	{S, A, C}			
Ø	{B}	{B}		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

- Table for string '**w**' that has length 5
- The algorithm populates the triangular table

Example (Result)

- Is baaba in L(G)?

Yes

We can see the S in the set X_{1n} where ‘n’ = 5

We can see the table

the cell X_{15} = (S, A, C) then

if S ∈ X_{15} then baaba ∈ L(G)

Try this one

$$
\begin{array}{lcl}
S & \rightarrow & NP\ VP \\
VP & \rightarrow & VP\ PP \\
VP & \rightarrow & V\ NP \\
VP & \rightarrow & eats \\
PP & \rightarrow & P\ NP \\
NP & \rightarrow & Det\ N \\
NP & \rightarrow & she \\
V & \rightarrow & eats \\
P & \rightarrow & with \\
N & \rightarrow & fish \\
N & \rightarrow & fork \\
Det & \rightarrow & a
\end{array}
$$

- She eats a fish with a fork

Theorem

- The CYK Algorithm correctly computes X_{ij} for all i and j; thus w is in L(G) if and only if S is in X_{1n}.
- The running time of the algorithm is $O(n^3)$.

References

- J. E. Hopcroft, R. Motwani, J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Second Edition, Addison Wesley, 2001
- T.A. Sudkamp, An Introduction to the Theory of Computer Science Languages and Machines, Third Edition, Addison Wesley, 2006

Question

- Show the CYK Algorithm with the following example:
  - CNF grammar **G**
    - S → AB | BC
    - A → BA | a
    - B → CC | b
    - C → AB | a
  - **w** is ababa
  - Question Is **ababa** in L(G)?
- Basics of CYK Algorithm
  - The Structure of the rules in a Chomsky Normal Form grammar
  - Uses a “dynamic programming” or “table-filling algorithm”
- Complexity O(n3)