Natural Language Processing (2)

Formal Language Theory

Daisuke Kawahara

Department of Communications and Computer Engineering,
Waseda University

Lecture Plan

1. Overview of Natural Language Processing
2. Formal Language Theory
3. Word Senses and Embeddings
4. Topic Models
5. Collocations, Language Models, and Recurrent Neural Networks
6. Sequence Labeling and Morphological Analysis
7. Parsing (1)
8. Parsing (2)
9. Transfer Learning
10. Knowledge Acquisition
11. Information Retrieval, Question Answering, and Machine Translation
12. Guest Talk (1)
13. Guest Talk (2)
14. Project: Survey or Programming
15. Project Presentation

Grading

- In-class quizzes or assignments: 50% (several times)
- A report for your project: 50%
  - Project: programming or survey (will be announced later)
- No exam

Assignment

1. Describe your research theme or interest.
2. Choose one (or more) service(s) or system(s) based on NLP. Describe problems that arise with these services or systems and how they try to solve the problems.
3. Describe any thoughts about this class if you have.

Deadline: April 21 (Thu) 23:59

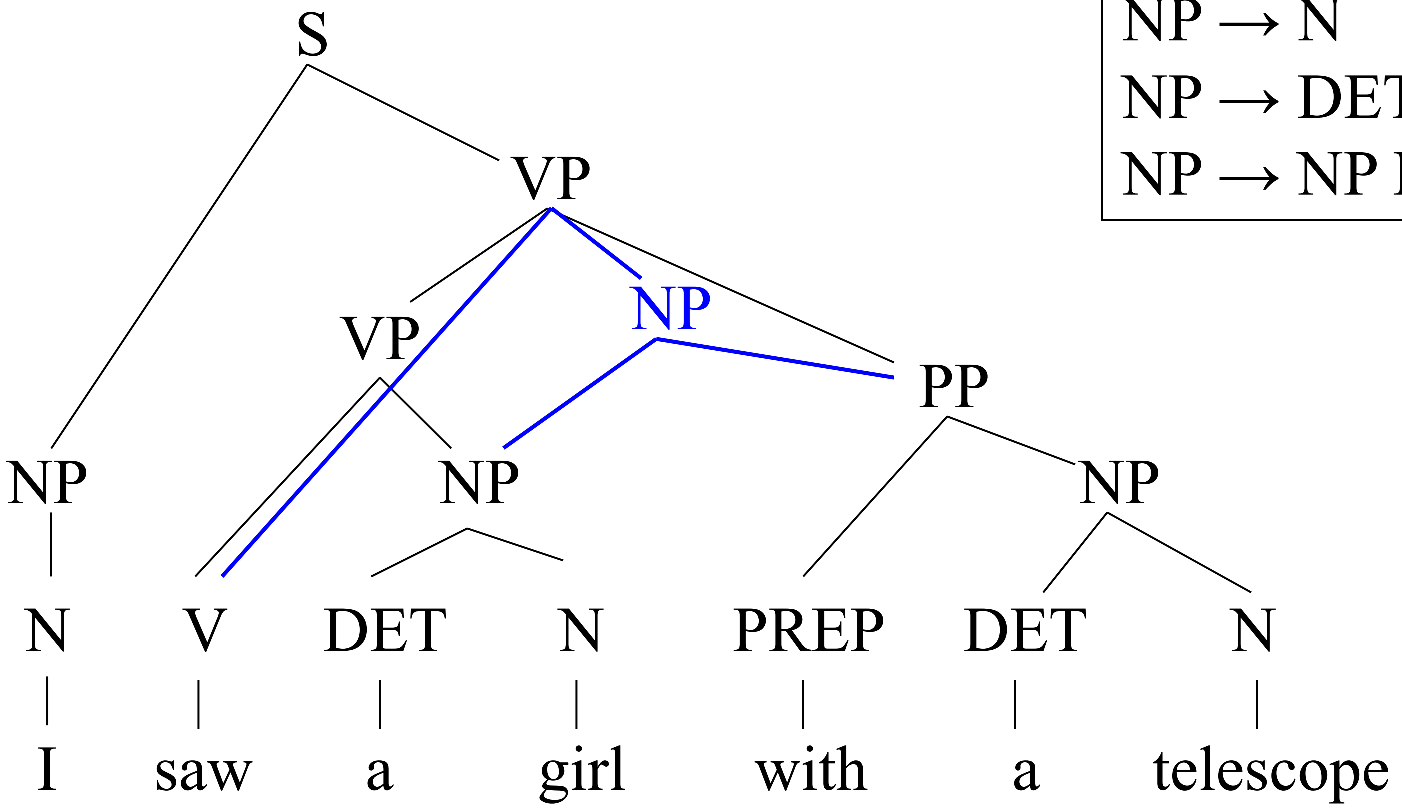
※ You can write it in English or Japanese.

Formal Language Theory

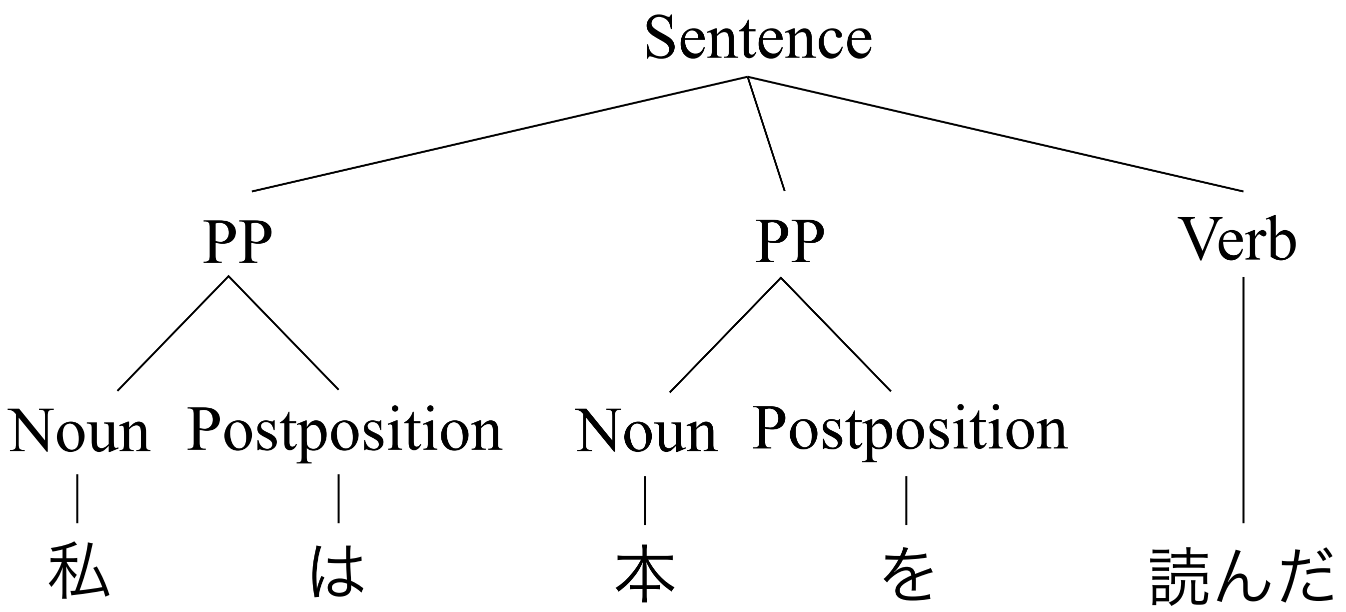
- Formal Definition of Languages
- Formal Grammars
  - Phrase Structure Grammars
- Regular Grammars
- Finite Automata
- Context-free Grammars

Syntactic Structure (English)

S → NP VP	VP → V
NP → N	VP → V NP
NP → DET N	VP → VP PP
NP → NP PP	PP → PREP NP



Syntactic Structure (Japanese)



Sentence → PP PP Verb
PP → Noun Postposition

Formal Definition of Languages

- Alphabet: a finite set of symbols (basic units of language)

  e.g., $\Sigma = \{0, 1\}$, phonemes, English alphabet, English words

- String: a finite sequence of symbols

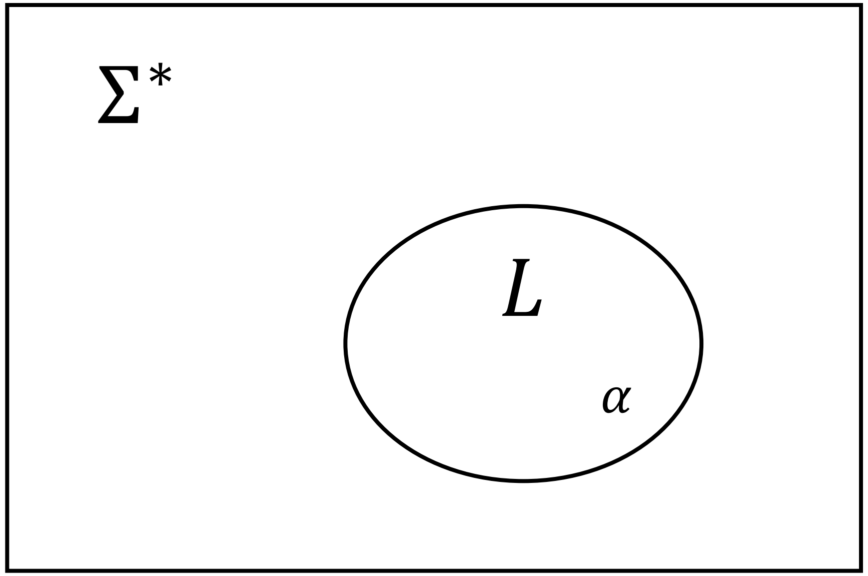
  e.g., in the case of $\Sigma = \{0, 1\}$:

  - $\Sigma^0 = \{\varepsilon\}$
  - $\Sigma^1 = \{0, 1\}$
  - $\Sigma^2 = \{00, 01, 10, 11\}$, ...
  - $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

    The set of all finite length strings over Σ,
    consisting of a countably infinite number of elements.

Formal Definition of Languages

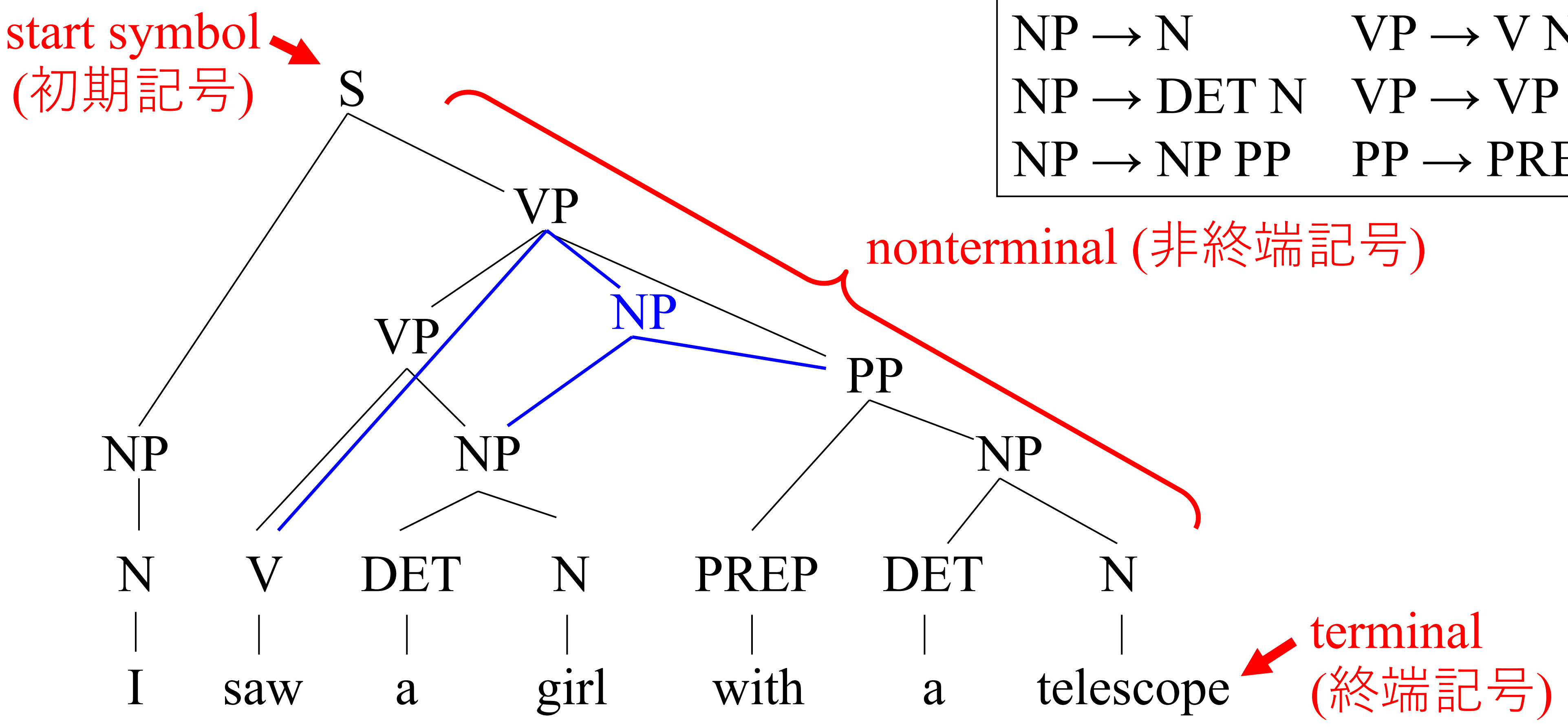
- Language L
  - A subset of Σ^* ($L \subset \Sigma^*$)
- Sentence α
  - An element of L ($\alpha \in L$)
- Grammar G
  - Specify a language



Syntactic Structure (English)

production rule (生成規則/書換え規則)

S → NP VP	VP → V
NP → N	VP → V NP
NP → DET N	VP → VP PP
NP → NP PP	PP → PREP NP



Phrase Structure Grammars (PSG)

- $G = < N, T, P, S >$
  - N: a finite set of nonterminals
  - T: a finite set of terminals
  - P: a finite set of productions
    $\alpha \to \beta \qquad \alpha \in (N \cup T)^+, \beta \in (N \cup T)^*$
  - S: the start symbol

finite

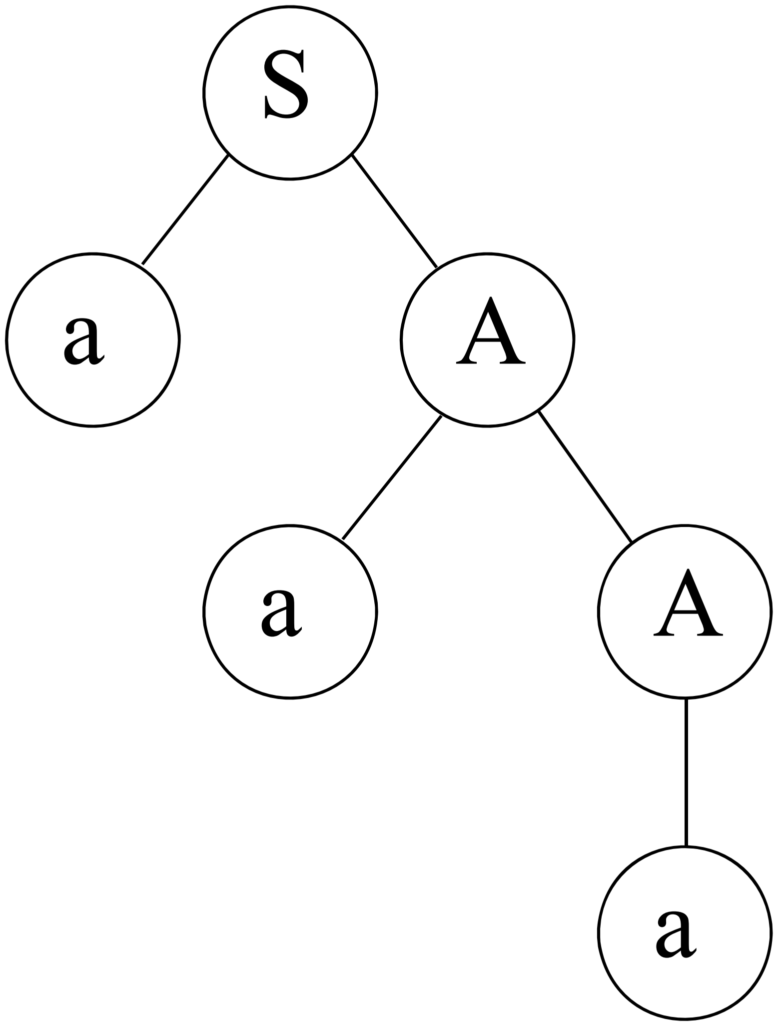
- $\underbrace{L(G)}_{\text{infinite}} = \{w | w \in T^*, S \overset{*}{\Rightarrow} w\}$

$\Rightarrow$: derivation
$\overset{*}{\Rightarrow}$: iteration of derivations

An Example of PSG

$$
\boxed{\begin{array}{l} S \rightarrow aA \\ A \rightarrow aA \\ A \rightarrow a \end{array}}
$$

$S \Rightarrow aA \Rightarrow aaA \Rightarrow aaa$



derivation tree

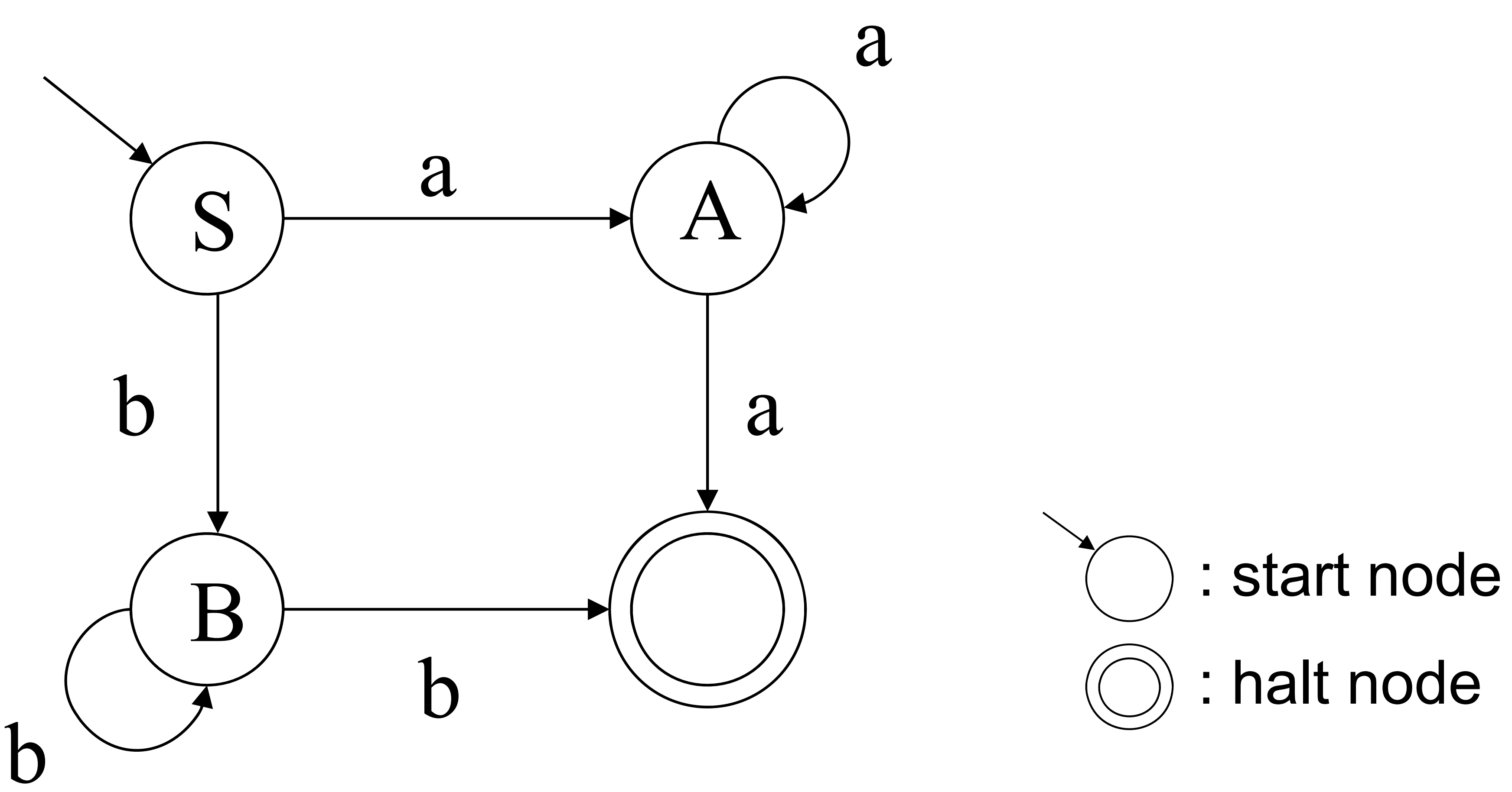
Regular Grammars (正規文法)

- $A \rightarrow aB$, $A \rightarrow a$ $\quad$ $A, B \in N$, $a \in T$

- Example:

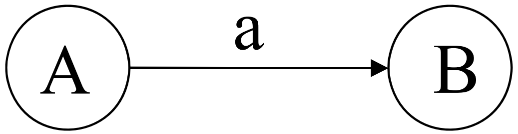
$$\boxed{\begin{array}{ll} S \rightarrow aA & S \rightarrow bB \\ A \rightarrow aA & A \rightarrow a \\ B \rightarrow bB & B \rightarrow b \end{array}}$$

Finite Automata (有限オートマトン)

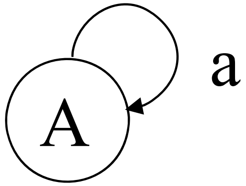


Finite Automata (有限オートマトン)

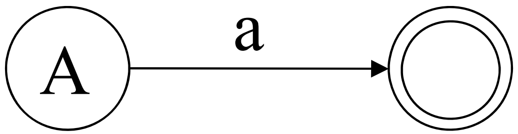
- Production rules
  - A → aB



  - A → aA



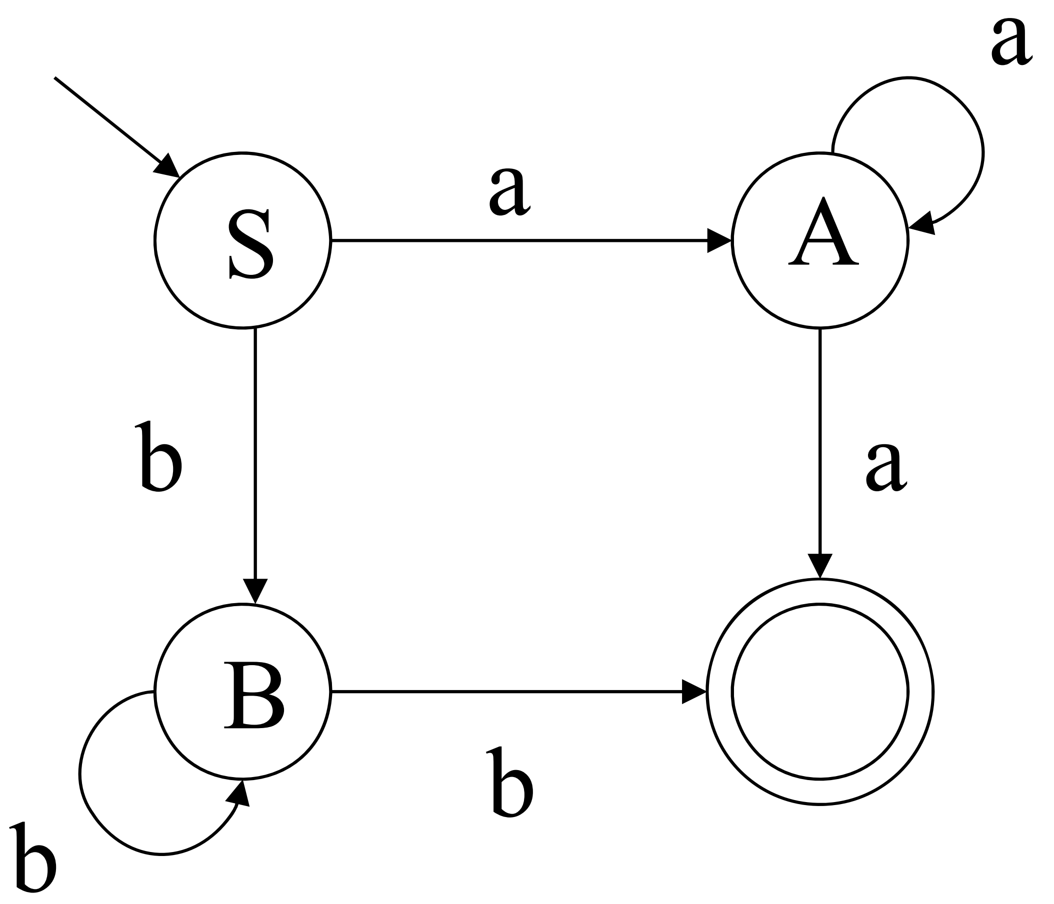
  - A → a



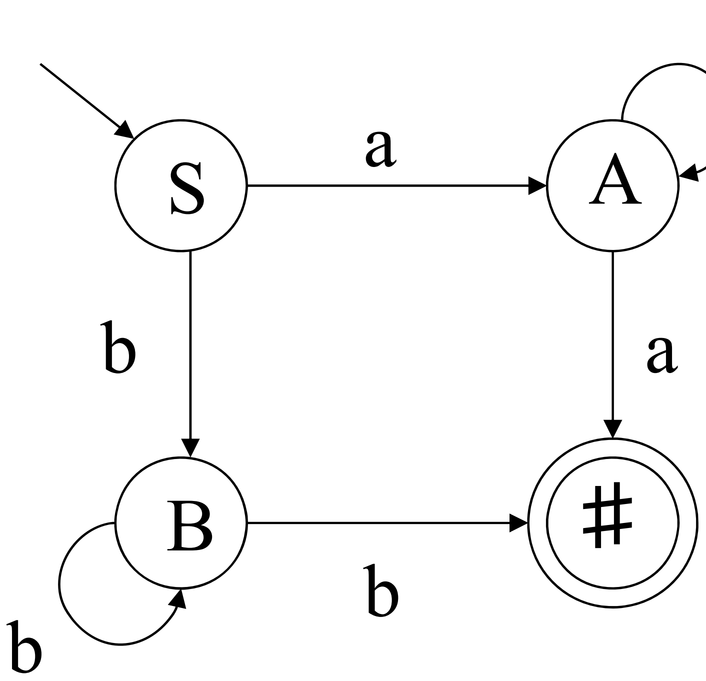
Acceptance of Regular Languages (正規言語の受理)

S → aA	S → bB
A → aA	A → a
B → bB	B → b

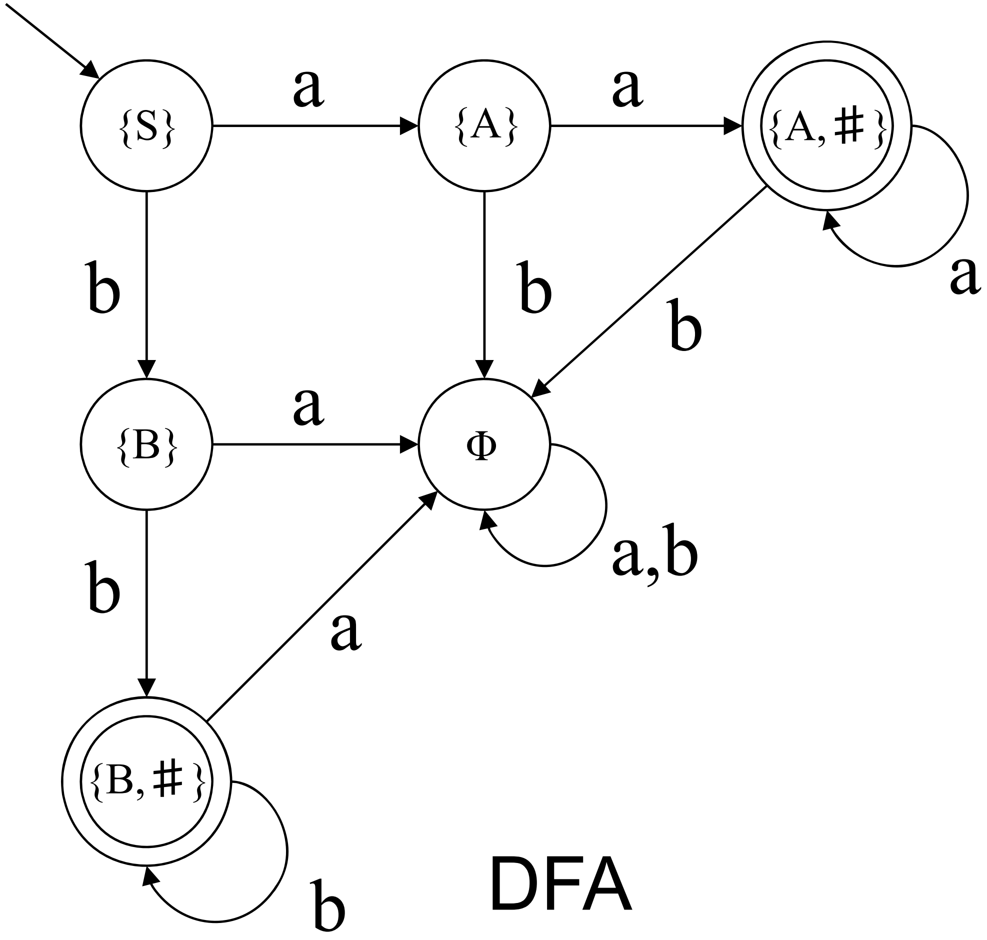
S → aA | bB
A → aA | a
B → bB | b



Nondeterministic FA → Deterministic FA



NFA



DFA

Practice

- For the alphabet $\Sigma = \{0,1\}$, design a DFA that accepts only strings that contain an even number of 0s.
  - For example, this DFA accepts 001010, 00, and 111.

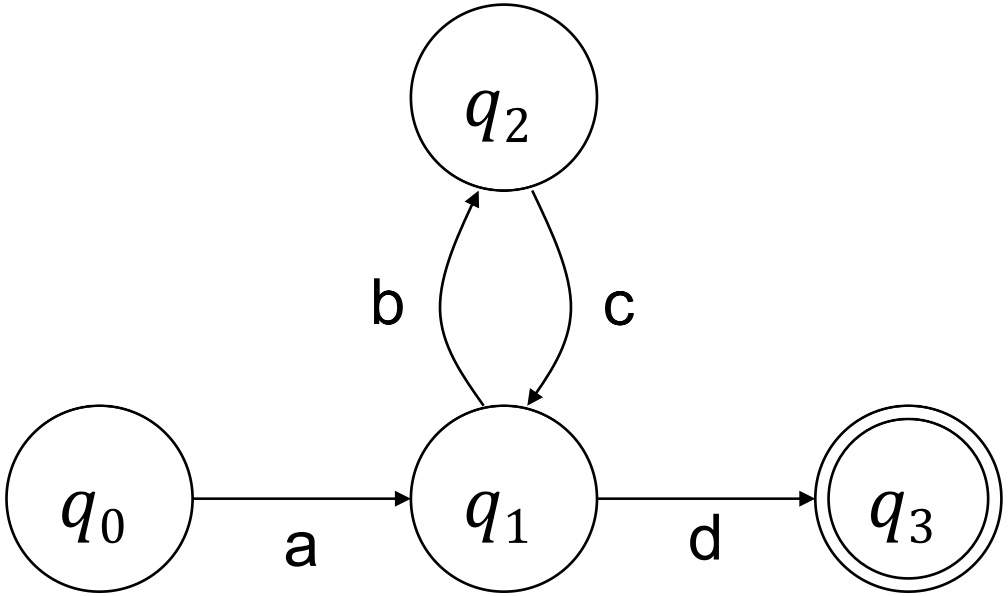
Practice: FA accepting “1月” – “12月”

NFA:

DFA:

Pumping Lemma for Regular Languages

- Let L be a regular language.
- Then there exists an integer $p \geq 1$ depending only on L such that every string w in L of length at least p can be written as $w = xyz$, satisfying the following conditions:
  - $|y| \geq 1$
  - $|xy| \leq p$
  - $(\forall n \geq 0)(xy^n z \in L)$



a bc d
a bcbc d
a bcbcbc d
:

Does a regular grammar generate $a^n b^n$?

Prove that the language $L = \{a^n b^n : n \geq 0\}$ over the alphabet $\sum = \{a, b\}$ is not regular

Context Free Grammars (文脈自由文法)

- $A \rightarrow \beta \qquad A \in N, \beta \in (N \cup T)^*$

- Example:

$$\boxed{S \rightarrow aSb \quad S \rightarrow ab}$$

Grammar for Arithmetic Expressions

- Consider a grammar that interprets the priority of operators in arithmetic expressions properly.
  - Terminals: +, -, x, /, (,), a, b, c
  - Grammar 1

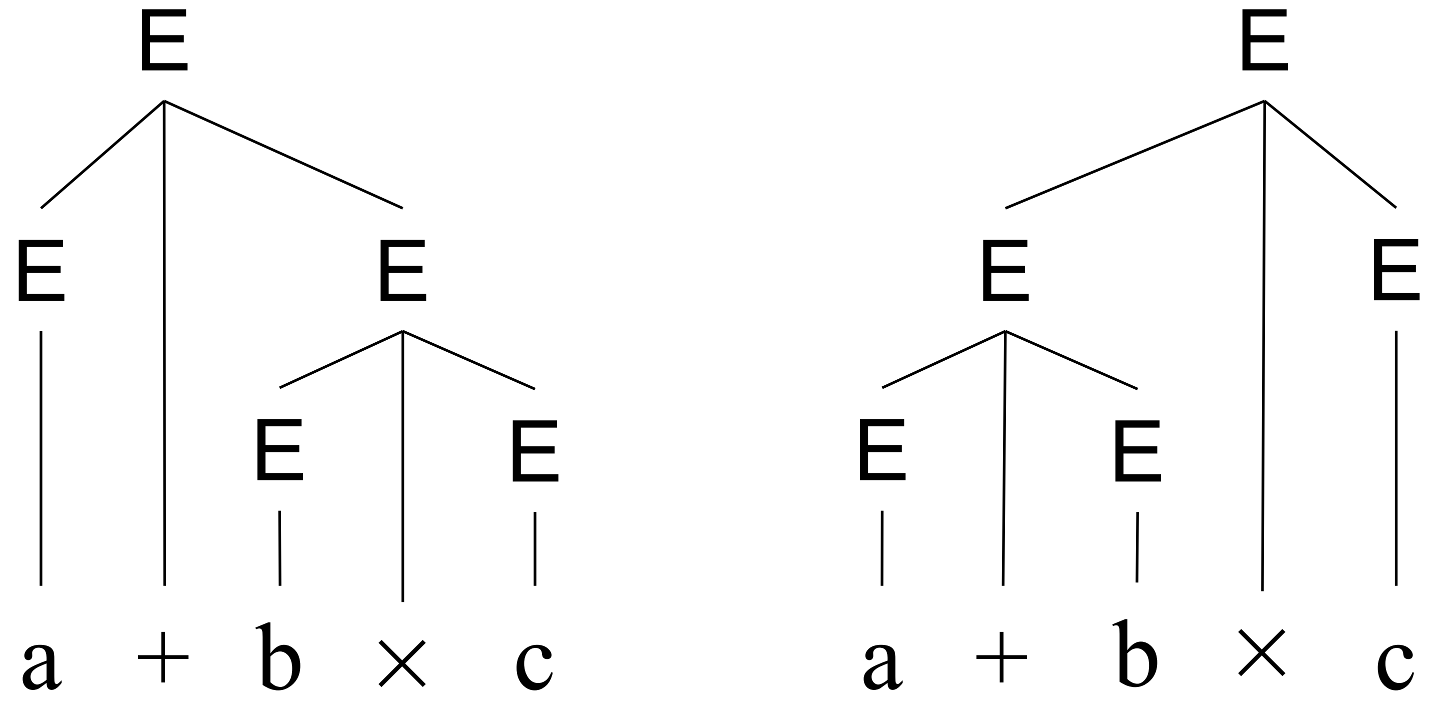
$$E \rightarrow E+E \mid E-E \mid ExE \mid E/E \mid a \mid b \mid c \mid (E)$$

  - Grammar 2

$$\begin{aligned} E &\rightarrow T \mid E+T \mid E-T \\ T &\rightarrow F \mid TxF \mid T/F \\ F &\rightarrow a \mid b \mid c \mid (E) \end{aligned}$$

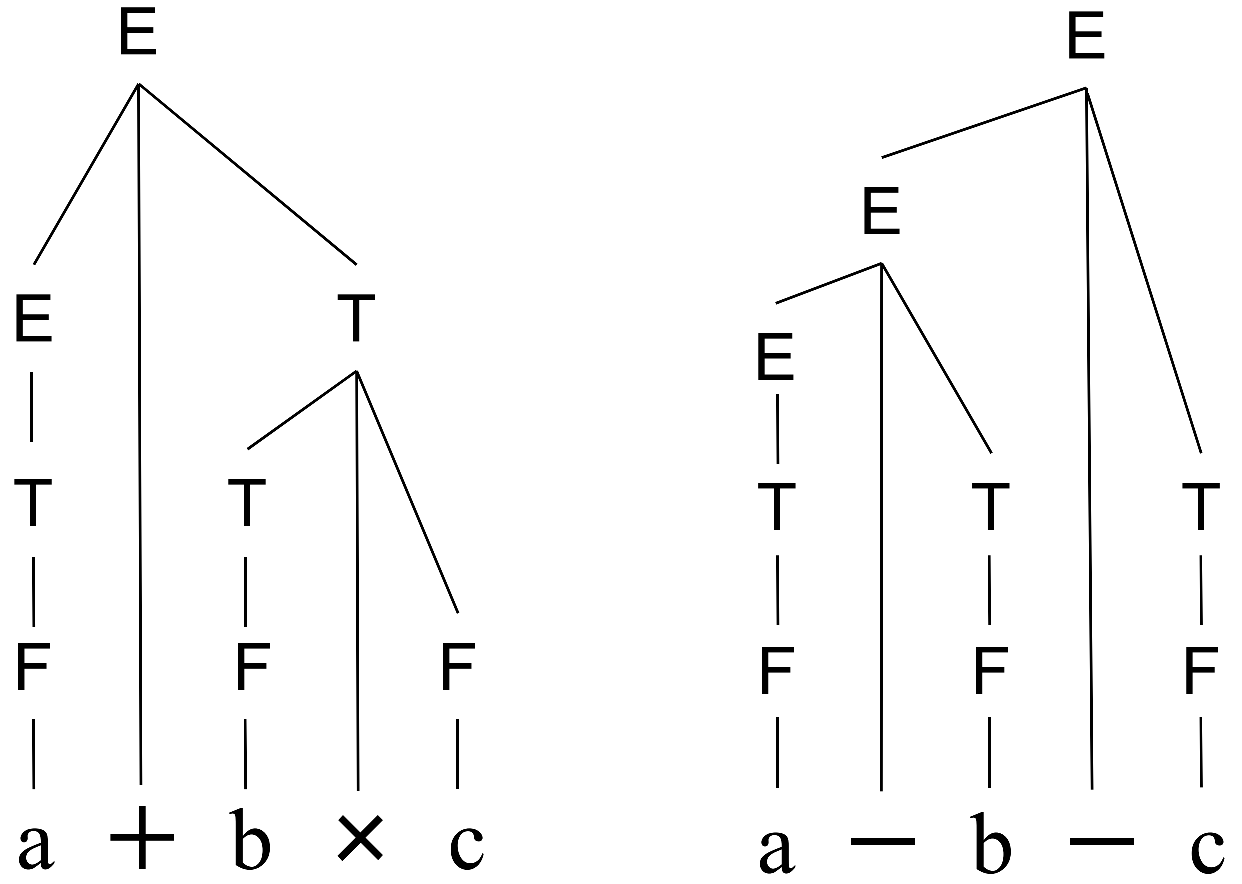
Grammar for Arithmetic Expressions

$E \to E{+}E \mid E{-}E \mid E{\times}E \mid E/E \mid a \mid b \mid c \mid (E)$



Grammar for Arithmetic Expressions

$E \rightarrow T \mid E+T \mid E-T$
$T \rightarrow F \mid T \times F \mid T/F$
$F \rightarrow a \mid b \mid c \mid (E)$



Normal Forms for CFG

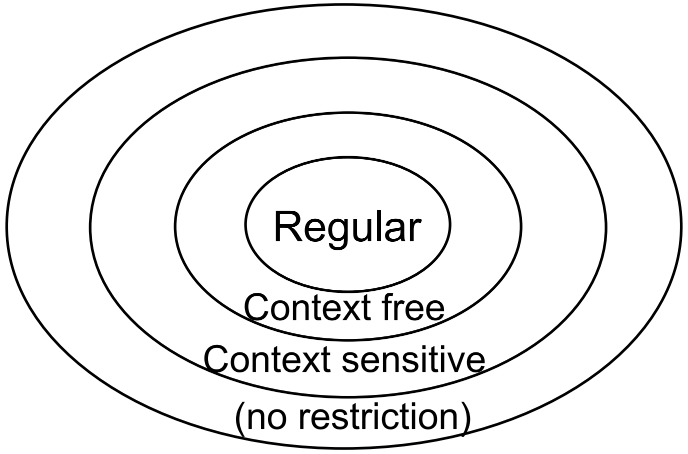
- Chomsky Normal Form
  - $A \rightarrow BC \mid a$
- Greibach Normal Form
  - $A \rightarrow a\alpha,\ \alpha \in N^*$

Practice

- Convert the following CFG grammar to Chomsky Normal Form

$$
\begin{array}{|l|}
\hline
S \to bA \mid aB \\
A \to bAA \mid aS \mid a \\
B \to aBB \mid bS \mid b \\
\hline
\end{array}
$$

Chomsky Hierarchy



Grammar	Language	Production Rules	Automaton
Type-0		no constraints	Turing machine
Type-1	context sensitive	$\alpha \rightarrow \beta$ $\|\alpha\| \leqq \|\beta\|$	Linear-bounded automaton
Type-2	context free	$A \rightarrow \beta$ $A \in N, \beta \in (N \cup T)^*$	Pushdown automaton
Type-3	regular	$A \rightarrow aB, A \rightarrow a$ $A, B \in N, a \in T$	Finite automaton

Practice

- When the grammar G has the following production rules, what type of grammar is G?
  1. $S \to aA,\ A \to aAB,\ B \to b,\ A \to a$
  2. $S \to aAB,\ AB \to bB,\ B \to b,\ A \to aB$
  3. $S \to aAB,\ AB \to c,\ A \to b,\ B \to AB$
  4. $S \to aB,\ B \to bA,\ B \to b,\ B \to a,\ A \to aB,\ A \to a$

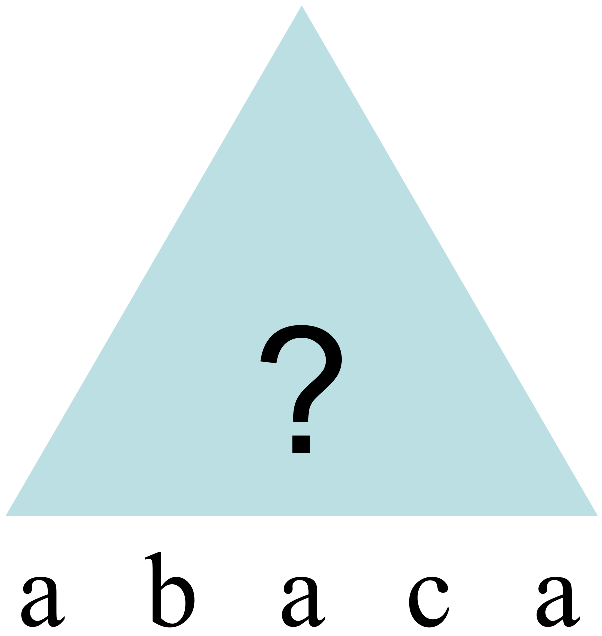
Ambiguous Grammar

- Two or more derivation trees exist for a sentence

$S \rightarrow SbS$

$S \rightarrow ScS$

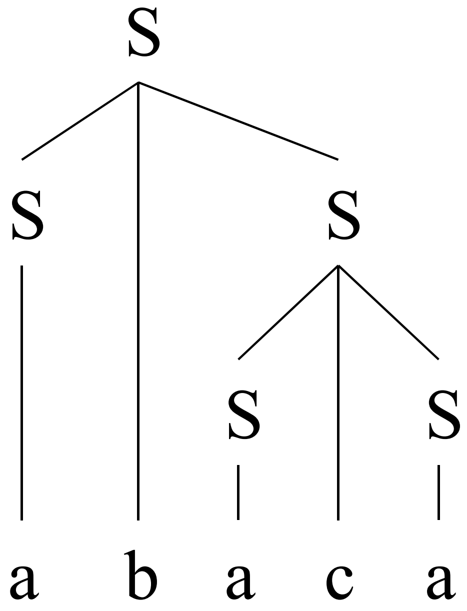
$S \rightarrow a$

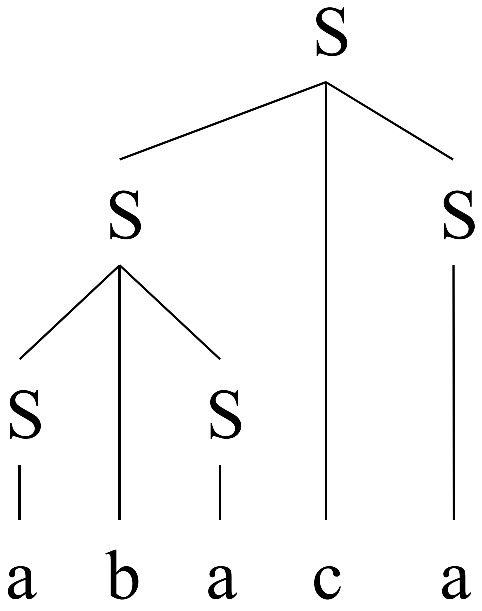


Ambiguous Grammar

- Two or more derivation trees exist for a sentence

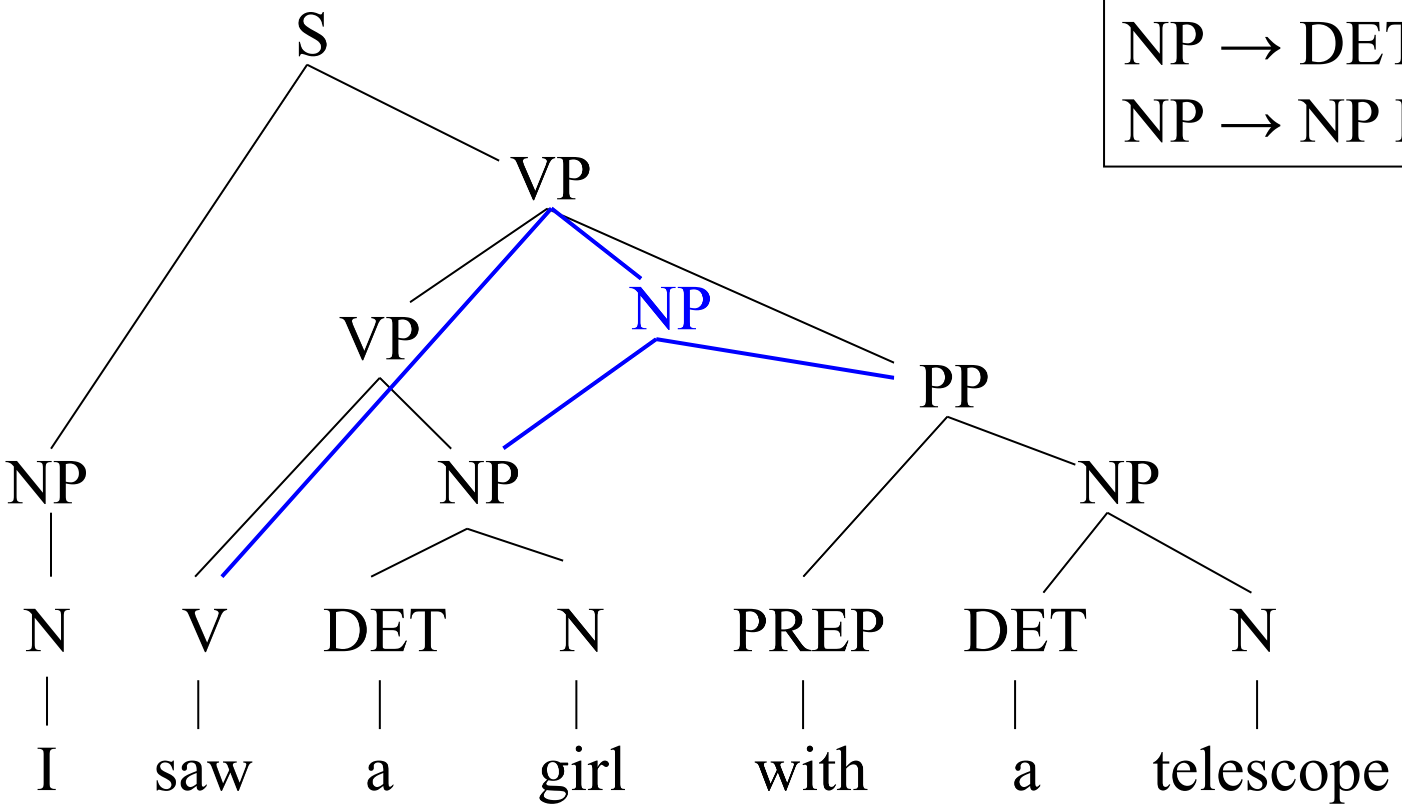
$S \rightarrow SbS$
$S \rightarrow ScS$
$S \rightarrow a$





Ambiguities in Natural Language

S $\rightarrow$ NP VP	VP $\rightarrow$ V
NP $\rightarrow$ N	VP $\rightarrow$ V NP
NP $\rightarrow$ DET N	VP $\rightarrow$ VP PP
NP $\rightarrow$ NP PP	PP $\rightarrow$ PREP NP



Naive Parsing Method

- Derivation starting with S (in a top-down manner)
- If there is an ambiguity, apply rules in some fixed order and do a **backtracking** if it fails
- Exponential explosion by repeating the same calculation

S → NP VP	VP → V	PP → PREP NP
NP → N	VP → V NP	
NP → DET N	VP → V VP PP	

Naive Parsing Method

S → NP VP	VP → V	PP → PREP NP
NP → N	VP → V NP	
NP → DET N	VP → V NP PP	

I/N saw/V a/DET girl/N with/PREP a/DET telescope/N

$S \Rightarrow NP\ VP \Rightarrow N\ VP \Rightarrow N\ V$ ✗

$\Rightarrow N\ V\ NP \Rightarrow N\ V\ N$ ✗

$\Rightarrow N\ V\ DET\ N$ ✗

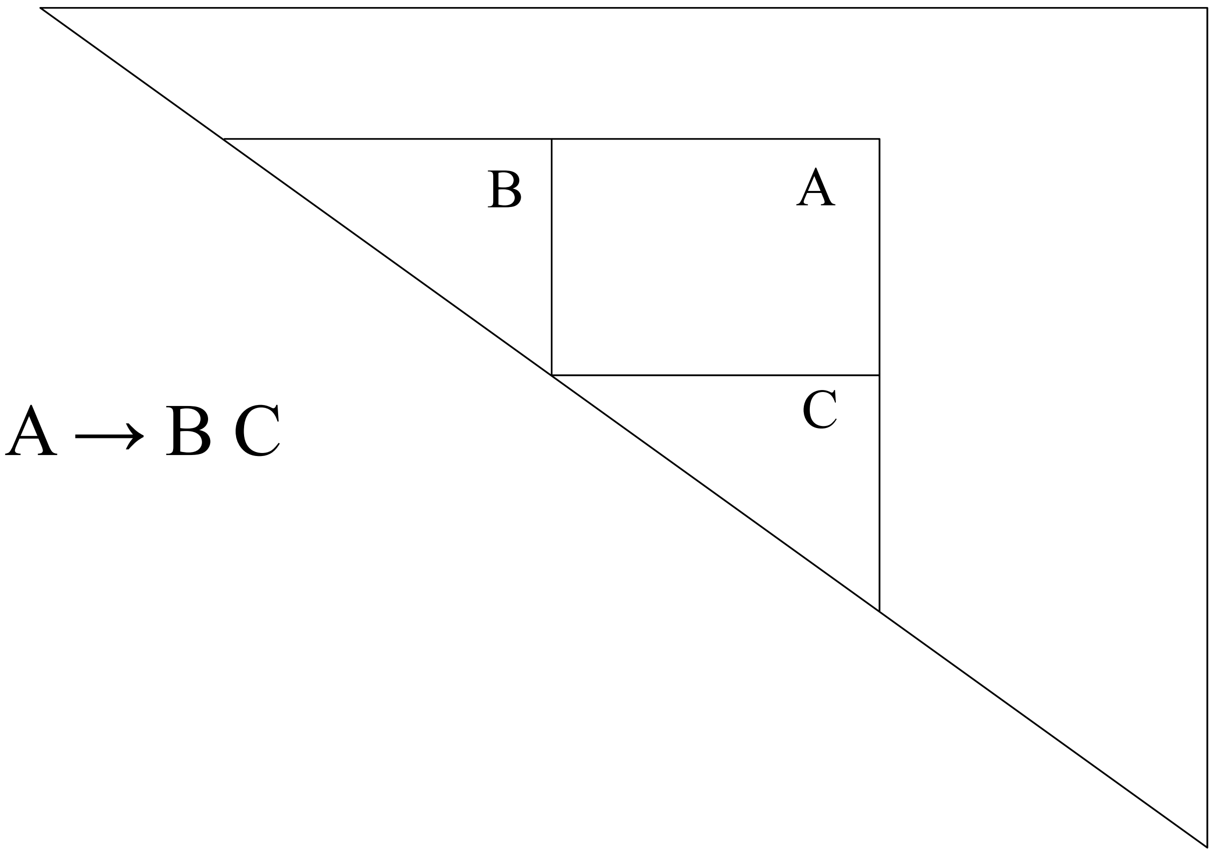
$\Rightarrow N\ V\ NP\ PP \Rightarrow N\ V\ N\ PP$ ✗

$\Rightarrow N\ V\ DET\ N\ PP$ ...

CKY (Cocke-Kasami-Younger) Algorithm

- Produce S from the input in a bottom-up manner
- Keep the intermediate results in a table
- Work with Chomsky Normal Form

S → NP VP	VP → V NP
NP → DET N	VP → VP PP
NP → NP PP	PP → PREP NP



CKY Algorithm

Input := $w_1\, w_2\ \ldots\ w_n$

for i := 1 to n do

    a(i, i) = { A | A → w_i ∈ Lexical Rules }

for d := 1 to n – 1 do

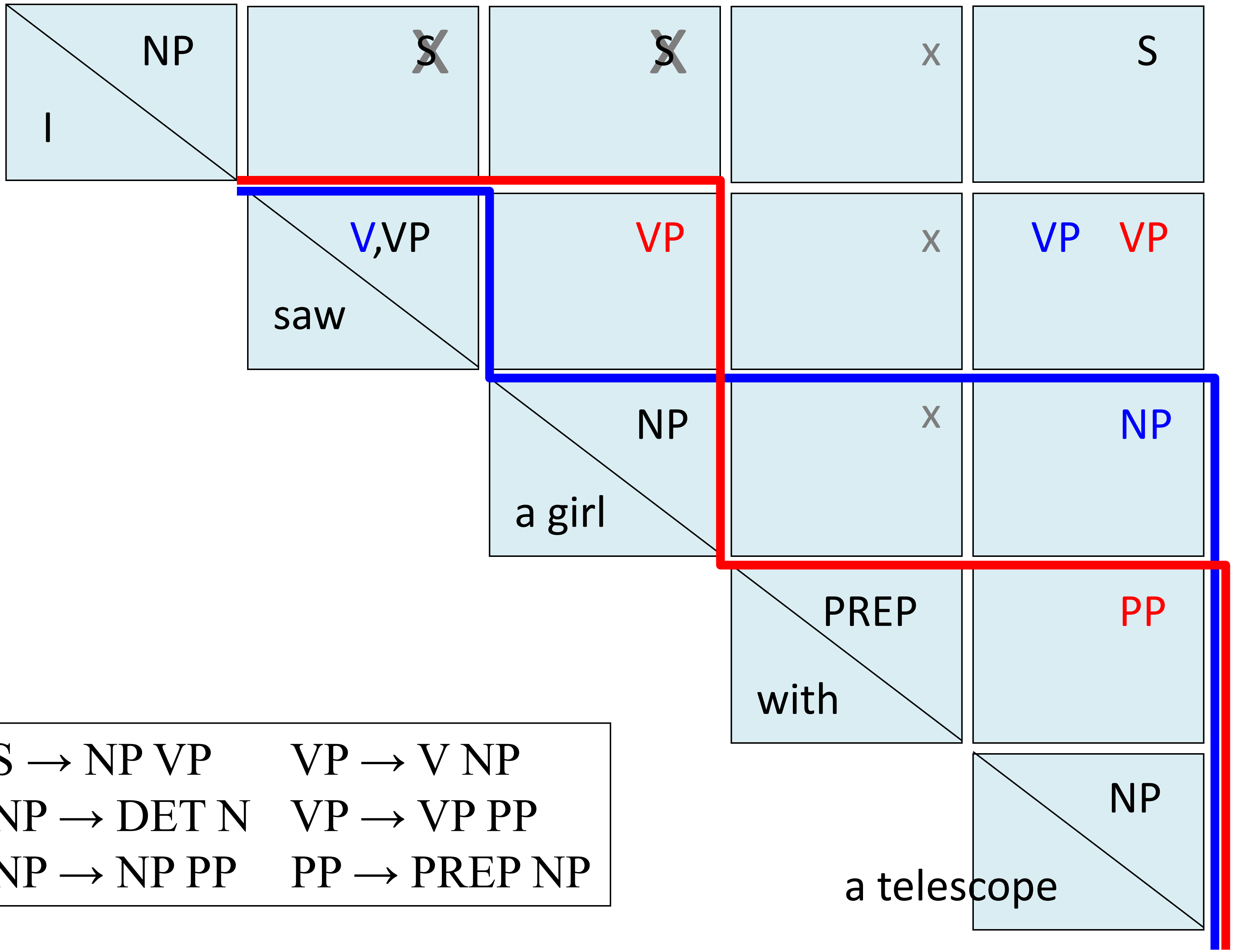
    for i := 1 to n – d do

        j = i + d

        for k := i to j – 1 do

            a(i, j) = a(i, j) ∪ { A | A → BC ∈ Production Rules, B ∈ a(i, k), C ∈ a(k+1, j) }

if (S ∈ a(1, n)) then accept else reject



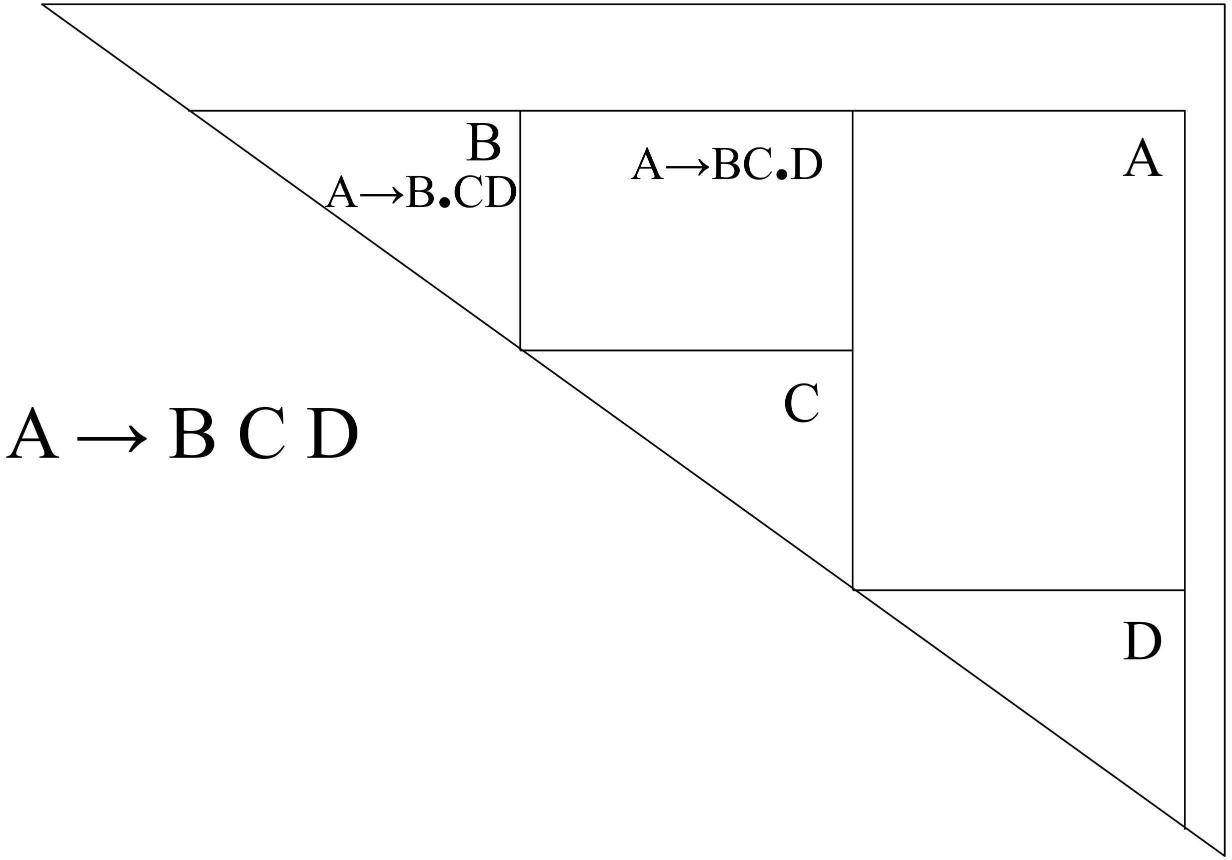
$S \rightarrow NP\ VP$ $\quad VP \rightarrow V\ NP$

$NP \rightarrow DET\ N$ $\quad VP \rightarrow VP\ PP$

$NP \rightarrow NP\ PP$ $\quad PP \rightarrow PREP\ NP$

Chart Parsing

- Generalize the CKY algorithm for All CFGs
- Keep which symbols were rewritten in the right-hand side string of the production rule



Formal Language Theory

- Formal Definition of Languages
- Formal Grammars
  - Phrase Structure Grammars
- Regular Grammars
- Finite Automata
- Context-free Grammars