COMP2022: Formal Languages and Logic

2017, Semester 1, Week 4

Joseph Godbehere

Adapted from slides by A/Prof Kalina Yacef

March 28, 2017

THE UNIVERSITY OF SYDNEY

Outline

- Grammars
- Context-Free Grammars (CFG)
- Context-Free Languages (CFL)
- Parsing (introduction)
- Ambiguity
- Recursive Grammars
- Clean Grammars
- Types of Grammar

Introduction

So far we have seen two different, but equivalent, methods of describing languages: finite automata and regular expressions, which describe *regular languages*

We have already proven that some languages, such as $\{0^n 1^n \mid n \geq 0\}$, cannot be described using FA or RE.

Today we will introduce *context-free grammars (CFG)*, which describe the next category of languages, the *context-free languages*

Later, will see grammars called *regular grammars*, which describe exactly *regular languages*

Grammars

Grammars are another way to describe a language

A *grammar* is a set of rules which can be used to *generate* a language

The language generated is the set of all strings which can be *derived* from the grammar

Introductory example (G_1)

$S \to 01$ Base case: $01 \in L$

$S \to 0S1$ Recursive case: if $S \in L$ then $0S1 \in L$

G_1 generates the language $L = \{0^n1^n \mid n > 0\}$, which we already know is not regular

How does it derive 000111?

Introductory example (G_1)

$S \to 01$ Base case: $01 \in L$

$S \to 0S1$ Recursive case: if $S \in L$ then $0S1 \in L$

G_1 generates the language $L = \{0^n 1^n \mid n > 0\}$, which we already know is not regular

How does it derive 000111?

S

Introductory example (G_1)

$$S \to 01 \qquad \text{Base case: } 01 \in L$$

$$S \to 0S1 \qquad \text{Recursive case: if } S \in L \text{ then } 0S1 \in L$$

G_1 generates the language $L = \{0^n 1^n \mid n > 0\}$, which we already know is not regular

How does it derive 000111?

$$S \Rightarrow 0S1 \qquad \text{using rule } S \to 0S1$$

Introductory example (G_1)

$S \to 01$ Base case: $01 \in L$

$S \to 0S1$ Recursive case: if $S \in L$ then $0S1 \in L$

G_1 generates the language $L = \{0^n 1^n \mid n > 0\}$, which we already know is not regular

How does it derive 000111?

$$
\begin{aligned}
S &\Rightarrow 0S1 && \text{using rule } S \to 0S1 \\
&\Rightarrow 00S11 && \text{using rule } S \to 0S1
\end{aligned}
$$

Introductory example (G_1)

$S \to 01$ — Base case: $01 \in L$

$S \to 0S1$ — Recursive case: if $S \in L$ then $0S1 \in L$

G_1 generates the language $L = \{0^n 1^n \mid n > 0\}$, which we already know is not regular

How does it derive 000111?

$$
\begin{aligned}
S &\Rightarrow 0S1 && \text{using rule } S \to 0S1 \\
&\Rightarrow 00S11 && \text{using rule } S \to 0S1 \\
&\Rightarrow 000111 && \text{using rule } S \to 01
\end{aligned}
$$

Introductory example (G_2)

$$
\begin{aligned}
S &\rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase} \\
\mathit{NounPhrase} &\rightarrow \text{the } \mathit{Noun} \\
\mathit{VerbPhrase} &\rightarrow \mathit{Verb}\ \mathit{NounPhrase} \\
\mathit{Noun} &\rightarrow \text{girl} \mid \text{ball} \\
\mathit{Verb} &\rightarrow \text{likes} \mid \text{sees}
\end{aligned}
$$

What language does G_2 generate?

Introductory example (G_2)

$$
\begin{aligned}
S &\rightarrow NounPhrase\ \ VerbPhrase \\
NounPhrase &\rightarrow \textsf{the}\ Noun \\
VerbPhrase &\rightarrow Verb\ \ NounPhrase \\
Noun &\rightarrow \textsf{girl} \mid \textsf{ball} \\
Verb &\rightarrow \textsf{likes} \mid \textsf{sees}
\end{aligned}
$$

What language does G_2 generate?

{ the girl likes the girl, the girl likes the ball,
the girl sees the girl, the girl sees the ball,
the ball likes the girl, the ball likes the ball,
the ball sees the girl, the ball sees the ball }

Definitions

Terminals

- The finite set of symbols which make up strings of the language

Non-terminals / Variables

- A finite set of symbols used to generate the strings.
- They never appear in the language.

Start symbol

- The variable used to start every derivation

Definitions

Production rules

- Sometimes called substitution or derivation rules
- Define strings of *variables* and *terminals* which can be substituted for a *variable*:

$$Variable \rightarrow <\text{string of } Variables \text{ and } Terminals>$$

Definitions

Production rules

- Sometimes called substitution or derivation rules
- Define strings of *variables* and *terminals* which can be substituted for a *variable*:

$$Variable \rightarrow <\text{string of } Variables \text{ and } Terminals>$$

A variable can have many rules:

$$Noun \rightarrow \text{girl}$$
$$Noun \rightarrow \text{ball}$$
$$Noun \rightarrow \text{quokka}$$

Definitions

Production rules

- Sometimes called substitution or derivation rules
- Define strings of *variables* and *terminals* which can be substituted for a *variable*:

$$Variable \rightarrow <\text{string of } Variables \text{ and } Terminals>$$

A variable can have many rules:

$$Noun \rightarrow \text{girl}$$
$$Noun \rightarrow \text{ball}$$
$$Noun \rightarrow \text{quokka}$$

They can be written together:

$$Noun \rightarrow \text{girl} \mid \text{ball} \mid \text{quokka}$$

Some common notational conventions

If not stated otherwise:

- $A, B, C, \ldots$ and S are variables
- S is the start variable
- $a, b, c, \ldots$ are terminals
- $\ldots, X, Y, Z$ are either terminals or variables
- $\ldots w, x, y, z$ are strings of terminals *only*
- $\alpha, \beta, \gamma, \ldots$ are strings of terminals and/or variables

Context-free Grammar (CFG)

A *context-free grammar* is a grammar where every production rule has the form $A \to \alpha$

- A is a variable
- α is a string of terminals and/or variables (possibly ϵ)

Context-free Grammar (CFG)

A *context-free grammar* is a grammar where every production rule has the form $A \to \alpha$

- A is a variable
- α is a string of terminals and/or variables (possibly ϵ)

Context-free grammars describe *context-free languages*

Context-free Grammar (CFG)

A *context-free grammar* is a grammar where every production rule has the form $A \to \alpha$

- A is a variable
- α is a string of terminals and/or variables (possibly ϵ)

Context-free grammars describe *context-free languages*

Example:
$\{a^n b^n \mid n \in N\}$ is not a regular language (no finite automata exists recognising it), but we can prove that it is a context-free language, because the following grammar generates it:

$$S \to aSb \mid \varepsilon$$

CFG: Formal Definition

A *context-free grammar* G is a 4-tuple (V, T, P, S) where:

- V is a finite set of *variables*
- T is a finite set of *terminals*
- P is a finite set of *production rules* in the form $\alpha \to \beta$ where $\alpha \in V$ and $\beta \in \{V \cup T \cup \{\varepsilon\}\}^\star$
- $S \in V$ is a special variable called the *Start Symbol*

Example G_1

$$S \to 01$$
$$S \to 0S1$$

More formally, $G_1 = (T, V, S, P)$ where:

$$T =$$
$$V =$$
$$S =$$
$$P =$$

Example G_1

$$S \to 01$$
$$S \to 0S1$$

More formally, $G_1 = (T, V, S, P)$ where:

$$T = \{0, 1\}$$
$$V =$$
$$S =$$
$$P =$$

Example G_1

$$S \to 01$$
$$S \to 0S1$$

More formally, $G_1 = (T, V, S, P)$ where:

$$T = \{0, 1\}$$
$$V = \{S\}$$
$$S =$$
$$P =$$

Example G_1

$$S \to 01$$
$$S \to 0S1$$

More formally, $G_1 = (T, V, S, P)$ where:

$$T = \{0, 1\}$$
$$V = \{S\}$$
$$S = S$$
$$P =$$

Example G_1

$$S \to 01$$
$$S \to 0S1$$

More formally, $G_1 = (T, V, S, P)$ where:

$$T = \{0, 1\}$$
$$V = \{S\}$$
$$S = S$$
$$P = \{$$
$$S \to 01$$
$$S \to 0S1$$
$$\}$$

Example G_2

$$
\begin{aligned}
S &\rightarrow NounPhrase\ \ VerbPhrase \\
NounPhrase &\rightarrow \textsf{the}\ Noun \\
VerbPhrase &\rightarrow Verb\ \ NounPhrase \\
Noun &\rightarrow \textsf{girl} \mid \textsf{ball} \\
Verb &\rightarrow \textsf{likes} \mid \textsf{sees}
\end{aligned}
$$

More formally, $G_2 = (T, V, S, P)$ where:

$$
\begin{aligned}
T &= \\
V &= \\
S &= \\
P &=
\end{aligned}
$$

Example G_2

$$
\begin{aligned}
S &\to NounPhrase\ \ VerbPhrase \\
NounPhrase &\to \textsf{the}\ Noun \\
VerbPhrase &\to Verb\ \ NounPhrase \\
Noun &\to \textsf{girl} \mid \textsf{ball} \\
Verb &\to \textsf{likes} \mid \textsf{sees}
\end{aligned}
$$

More formally, $G_2 = (T, V, S, P)$ where:

$$
\begin{aligned}
T &= \{\textsf{the, girl, ball, likes, sees}\} \\
V &= \\
S &= \\
P &=
\end{aligned}
$$

Example G_2

$$
\begin{aligned}
S &\rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase} \\
\mathit{NounPhrase} &\rightarrow \textsf{the}\ \mathit{Noun} \\
\mathit{VerbPhrase} &\rightarrow \mathit{Verb}\ \mathit{NounPhrase} \\
\mathit{Noun} &\rightarrow \textsf{girl} \mid \textsf{ball} \\
\mathit{Verb} &\rightarrow \textsf{likes} \mid \textsf{sees}
\end{aligned}
$$

More formally, $G_2 = (T, V, S, P)$ where:

$$
\begin{aligned}
T &= \{\textsf{the, girl, ball, likes, sees}\} \\
V &= \{S, \mathit{NounPhrase}, \mathit{VerbPhrase}, \mathit{Noun}, \mathit{Verb}\} \\
S &= \\
P &=
\end{aligned}
$$

Example G_2

$$
\begin{aligned}
S &\rightarrow \mathit{NounPhrase}\ \ \mathit{VerbPhrase} \\
\mathit{NounPhrase} &\rightarrow \text{the } \mathit{Noun} \\
\mathit{VerbPhrase} &\rightarrow \mathit{Verb}\ \ \mathit{NounPhrase} \\
\mathit{Noun} &\rightarrow \text{girl} \mid \text{ball} \\
\mathit{Verb} &\rightarrow \text{likes} \mid \text{sees}
\end{aligned}
$$

More formally, $G_2 = (T, V, S, P)$ where:

$$
\begin{aligned}
T &= \{\text{the, girl, ball, likes, sees}\} \\
V &= \{S, \mathit{NounPhrase}, \mathit{VerbPhrase}, \mathit{Noun}, \mathit{Verb}\} \\
S &= S \\
P &=
\end{aligned}
$$

Example G_2

$$
\begin{aligned}
S &\rightarrow \mathit{NounPhrase}\ \ \mathit{VerbPhrase} \\
\mathit{NounPhrase} &\rightarrow \textsf{the}\ \mathit{Noun} \\
\mathit{VerbPhrase} &\rightarrow \mathit{Verb}\ \ \mathit{NounPhrase} \\
\mathit{Noun} &\rightarrow \textsf{girl} \mid \textsf{ball} \\
\mathit{Verb} &\rightarrow \textsf{likes} \mid \textsf{sees}
\end{aligned}
$$

More formally, $G_2 = (T, V, S, P)$ where:

$$
\begin{aligned}
T &= \{\textsf{the, girl, ball, likes, sees}\} \\
V &= \{S, \mathit{NounPhrase}, \mathit{VerbPhrase}, \mathit{Noun}, \mathit{Verb}\} \\
S &= S \\
P &= (\textsf{set of seven rules above})
\end{aligned}
$$

Language of a Grammar

Let w be a string over T.
$w \in L(G)$ if and only if it is possible to derive w from S

Language of a Grammar

Let w be a string over T.
$w \in L(G)$ if and only if it is possible to derive w from S

Notation:
$\alpha \Rightarrow \beta$ denotes that α derives β in one step
$\alpha \Rightarrow^{+} \beta$ means it derives it in one *or more* steps

Language of a Grammar

Let w be a string over T.
$w \in L(G)$ if and only if it is possible to derive w from S

Notation:
$\alpha \Rightarrow \beta$ denotes that α derives β in one step
$\alpha \Rightarrow^{+} \beta$ means it derives it in one *or more* steps

The *language* of a grammar $G = (V, T, P, S)$ is the set
$L(G) = \{s \mid s \in T^{\star} \text{ and } S \Rightarrow^{+} s\}$

Language of a Grammar

Let w be a string over T.
$w \in L(G)$ if and only if it is possible to derive w from S

Notation:
$\alpha \Rightarrow \beta$ denotes that α derives β in one step
$\alpha \Rightarrow^{+} \beta$ means it derives it in one *or more* steps

The *language* of a grammar $G = (V, T, P, S)$ is the set
$L(G) = \{s \mid s \in T^{\star}$ and $S \Rightarrow^{+} s\}$

If two grammars generate the same langauge, then they are *equivalent*.

Derivation of a string

- Begin with the start symbol
- Repeatedly replace one variable with the right hand side of one of it's productions
- ... until the string is composed only of terminal symbols

Derivation of a string

- Begin with the start symbol
- Repeatedly replace one variable with the right hand side of one of it's productions
- … until the string is composed only of terminal symbols

Example, derivation of 000111 from this grammar:

$$S \to 0S1 \mid \varepsilon \qquad\qquad S \Rightarrow$$

Derivation of a string

- Begin with the start symbol
- Repeatedly replace one variable with the right hand side of one of it's productions
- ... until the string is composed only of terminal symbols

Example, derivation of 000111 from this grammar:

$$S \to 0S1 \mid \varepsilon$$

$$\begin{aligned} S &\Rightarrow 0S1 \\ &\Rightarrow \end{aligned}$$

Derivation of a string

- Begin with the start symbol
- Repeatedly replace one variable with the right hand side of one of it's productions
- ... until the string is composed only of terminal symbols

Example, derivation of 000111 from this grammar:

$$S \to 0S1 \mid \varepsilon$$

$$\begin{aligned} S &\Rightarrow 0S1 \\ &\Rightarrow 00S11 \\ &\Rightarrow 000S111 \\ &\Rightarrow \end{aligned}$$

Derivation of a string

- Begin with the start symbol
- Repeatedly replace one variable with the right hand side of one of it's productions
- ... until the string is composed only of terminal symbols

Example, derivation of 000111 from this grammar:

$$S \to 0S1 \mid \varepsilon$$

$$\begin{aligned} S &\Rightarrow 0S1 \\ &\Rightarrow 00S11 \\ &\Rightarrow 000S111 \\ &\Rightarrow 000111 \end{aligned}$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$S \Rightarrow NounPhrase\ \ VerbPhrase$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$S \Rightarrow NounPhrase\ \ VerbPhrase$$
$$\Rightarrow \text{the}\ Noun\ \ VerbPhrase$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$\begin{aligned} S &\Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase} \\ &\Rightarrow \text{the}\ \mathit{Noun}\ \mathit{VerbPhrase} \\ &\Rightarrow \text{the girl}\ \mathit{VerbPhrase} \end{aligned}$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$
\begin{aligned}
S &\Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase} \\
&\Rightarrow \text{the } \mathit{Noun}\ \mathit{VerbPhrase} \\
&\Rightarrow \text{the girl } \mathit{VerbPhrase} \\
&\Rightarrow \text{the girl } \mathit{Verb}\ \mathit{NounPhrase}
\end{aligned}
$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$S \Rightarrow NounPhrase\ \ VerbPhrase$

$\Rightarrow$ the $Noun\ \ VerbPhrase$

$\Rightarrow$ the girl $VerbPhrase$

$\Rightarrow$ the girl $Verb\ \ NounPhrase$

$\Rightarrow$ the girl sees $NounPhrase$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$S \Rightarrow NounPhrase\ \ VerbPhrase$

$\Rightarrow$ the $Noun\ \ VerbPhrase$

$\Rightarrow$ the girl $VerbPhrase$

$\Rightarrow$ the girl $Verb\ \ NounPhrase$

$\Rightarrow$ the girl sees $NounPhrase$

$\Rightarrow$ the girl sees the $Noun$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$S \Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase}$
$\Rightarrow$ the $\mathit{Noun}\ \mathit{VerbPhrase}$
$\Rightarrow$ the girl $\mathit{VerbPhrase}$
$\Rightarrow$ the girl $\mathit{Verb}\ \mathit{NounPhrase}$
$\Rightarrow$ the girl sees $\mathit{NounPhrase}$
$\Rightarrow$ the girl sees the Noun
$\Rightarrow$ the girl sees the ball

$S \Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase}$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$
\begin{aligned}
S &\Rightarrow NounPhrase\ \ VerbPhrase \\
&\Rightarrow \text{the } Noun\ \ VerbPhrase \\
&\Rightarrow \text{the girl } VerbPhrase \\
&\Rightarrow \text{the girl } Verb\ \ NounPhrase \\
&\Rightarrow \text{the girl sees } NounPhrase \\
&\Rightarrow \text{the girl sees the } Noun \\
&\Rightarrow \text{the girl sees the ball}
\end{aligned}
$$

$$
\begin{aligned}
S &\Rightarrow NounPhrase\ \ VerbPhrase \\
&\Rightarrow NounPhrase\ \ Verb\ \ NounPhrase
\end{aligned}
$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$
\begin{aligned}
S &\Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase} \\
&\Rightarrow \text{the } \mathit{Noun}\ \mathit{VerbPhrase} \\
&\Rightarrow \text{the girl } \mathit{VerbPhrase} \\
&\Rightarrow \text{the girl } \mathit{Verb}\ \mathit{NounPhrase} \\
&\Rightarrow \text{the girl sees } \mathit{NounPhrase} \\
&\Rightarrow \text{the girl sees the } \mathit{Noun} \\
&\Rightarrow \text{the girl sees the ball}
\end{aligned}
$$

$$
\begin{aligned}
S &\Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase} \\
&\Rightarrow \mathit{NounPhrase}\ \mathit{Verb}\ \mathit{NounPhrase} \\
&\Rightarrow \mathit{NounPhrase}\ \mathit{Verb} \text{ the } \mathit{Noun}
\end{aligned}
$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$\begin{aligned} S &\Rightarrow NounPhrase\ \ VerbPhrase \\ &\Rightarrow \text{the } Noun\ \ VerbPhrase \\ &\Rightarrow \text{the girl } VerbPhrase \\ &\Rightarrow \text{the girl } Verb\ \ NounPhrase \\ &\Rightarrow \text{the girl sees } NounPhrase \\ &\Rightarrow \text{the girl sees the } Noun \\ &\Rightarrow \text{the girl sees the ball} \end{aligned}$$

$$\begin{aligned} S &\Rightarrow NounPhrase\ \ VerbPhrase \\ &\Rightarrow NounPhrase\ \ Verb\ \ NounPhrase \\ &\Rightarrow NounPhrase\ \ Verb \text{ the } Noun \\ &\Rightarrow NounPhrase\ \ Verb \text{ the ball} \end{aligned}$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$\begin{aligned} S &\Rightarrow NounPhrase\ \ VerbPhrase \\ &\Rightarrow \text{the } Noun\ \ VerbPhrase \\ &\Rightarrow \text{the girl } VerbPhrase \\ &\Rightarrow \text{the girl } Verb\ \ NounPhrase \\ &\Rightarrow \text{the girl sees } NounPhrase \\ &\Rightarrow \text{the girl sees the } Noun \\ &\Rightarrow \text{the girl sees the ball} \end{aligned}$$

$$\begin{aligned} S &\Rightarrow NounPhrase\ \ VerbPhrase \\ &\Rightarrow NounPhrase\ \ Verb\ \ NounPhrase \\ &\Rightarrow NounPhrase\ \ Verb \text{ the } Noun \\ &\Rightarrow NounPhrase\ \ Verb \text{ the ball} \\ &\Rightarrow NounPhrase \text{ sees the ball} \end{aligned}$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$\begin{aligned} S &\Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase} \\ &\Rightarrow \text{the } \mathit{Noun}\ \mathit{VerbPhrase} \\ &\Rightarrow \text{the girl } \mathit{VerbPhrase} \\ &\Rightarrow \text{the girl } \mathit{Verb}\ \mathit{NounPhrase} \\ &\Rightarrow \text{the girl sees } \mathit{NounPhrase} \\ &\Rightarrow \text{the girl sees the } \mathit{Noun} \\ &\Rightarrow \text{the girl sees the ball} \end{aligned}$$

$$\begin{aligned} S &\Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase} \\ &\Rightarrow \mathit{NounPhrase}\ \mathit{Verb}\ \mathit{NounPhrase} \\ &\Rightarrow \mathit{NounPhrase}\ \mathit{Verb} \text{ the } \mathit{Noun} \\ &\Rightarrow \mathit{NounPhrase}\ \mathit{Verb} \text{ the ball} \\ &\Rightarrow \mathit{NounPhrase} \text{ sees the ball} \\ &\Rightarrow \text{the } \mathit{Noun} \text{ sees the ball} \end{aligned}$$

Leftmost and Rightmost Derivations

Leftmost derivation: always derive the leftmost variable first
Rightmost derivation: always derive the rightmost variable first

Example: "the girl sees the ball"

$$\begin{aligned} S &\Rightarrow NounPhrase\ \ VerbPhrase \\ &\Rightarrow \text{the } Noun\ \ VerbPhrase \\ &\Rightarrow \text{the girl } VerbPhrase \\ &\Rightarrow \text{the girl } Verb\ \ NounPhrase \\ &\Rightarrow \text{the girl sees } NounPhrase \\ &\Rightarrow \text{the girl sees the } Noun \\ &\Rightarrow \text{the girl sees the ball} \end{aligned}$$

$$\begin{aligned} S &\Rightarrow NounPhrase\ \ VerbPhrase \\ &\Rightarrow NounPhrase\ \ Verb\ \ NounPhrase \\ &\Rightarrow NounPhrase\ \ Verb \text{ the } Noun \\ &\Rightarrow NounPhrase\ \ Verb \text{ the ball} \\ &\Rightarrow NounPhrase \text{ sees the ball} \\ &\Rightarrow \text{the } Noun \text{ sees the ball} \\ &\Rightarrow \text{the girl sees the ball} \end{aligned}$$

Context-Free Languages

A language is *context-free* if it is generated by a CFG

Context-Free Languages

A language is *context-free* if it is generated by a CFG

The syntax of most programming languages are context-free.

$$S \to \text{while } E \text{ do } S$$
$$S \to \text{if } E \text{ then } S \text{ else } S$$
$$S \to I := E$$
$$S \to \{SL\}$$
$$L \to ;SL \mid \varepsilon$$
$$E \to \text{ ... (description of an expression)}$$
$$I \to \text{ ... (description of an identifier)}$$

Context-Free Languages

A language is *context-free* if it is generated by a CFG

$\{$Regular Languages$\} \subset \{$Context-Free Languages$\}$

- The *union* of two CFL is also context-free
- The *concatenation* of two CFL is also context-free
- The *star closure* of a CFL is also context-free

Example

Consider the grammar G:

$$S \to AB \qquad\qquad S \Rightarrow AB$$
$$A \to \varepsilon \mid aA$$
$$B \to \varepsilon \mid bB$$

What is $L(G)$?

Example

Consider the grammar G:

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow \varepsilon \mid aA \\
B &\rightarrow \varepsilon \mid bB
\end{aligned}
$$

$$
\begin{aligned}
S &\Rightarrow AB \\
&\Rightarrow aAB \\
&\Rightarrow^{+} aaaaAB \\
&\Rightarrow aaaaB
\end{aligned}
$$

What is $L(G)$?

Example

Consider the grammar G:

$$
\begin{aligned}
S &\to AB \\
A &\to \varepsilon \mid aA \\
B &\to \varepsilon \mid bB
\end{aligned}
$$

What is $L(G)$?

$$
\begin{aligned}
S &\Rightarrow AB \\
&\Rightarrow aAB \\
&\Rightarrow^{+} aaaaAB \\
&\Rightarrow aaaaB \\
&\Rightarrow aaaabB \\
&\Rightarrow^{+} aaaabbbbbB \\
&\Rightarrow aaaabbbbb
\end{aligned}
$$

Example

Consider the grammar G:

$$
\begin{aligned}
S &\to AB \\
A &\to \varepsilon \mid aA \\
B &\to \varepsilon \mid bB
\end{aligned}
$$

What is $L(G)$?

$$
\begin{aligned}
S &\Rightarrow AB \\
&\Rightarrow aAB \\
&\Rightarrow^{+} aaaaAB \\
&\Rightarrow aaaaB \\
&\Rightarrow aaaabB \\
&\Rightarrow^{+} aaaabbbbbB \\
&\Rightarrow aaaabbbbb
\end{aligned}
$$

i.e. $L(G) = L(a^{\star}b^{\star}) = \{a^n b^m \mid n \geq 0, m \geq 0\}$

More examples

Describe the language generated

1. $S \to aSa \mid bSb \mid \varepsilon$
2. $S \to aS \mid bS \mid a$
3. $S \to SS \mid bS \mid a$
4. $S \to aT \mid bT \mid \varepsilon$
   $T \to aS \mid bS$
5. $S \to aSa \mid bSb \mid a \mid b$

More examples

Give grammars generating these languages

1. $\{ba^{n+1}b \mid n \geq 0\}$
2. Odd-length strings in $\{a, b\}^{\star}$ with middle symbol a
3. Even-length strings in $\{a, b\}^{\star}$ with matching middle symbols
4. Binary strings containing more 0's than 1's
5. Strings over $\{a, b\}$ with at least three a's

Constructing grammars

Let M and N be two languages whose grammars have disjoint sets of non-terminals (rename them if necessary). Let S_M and S_N be their start symbols.

Constructing grammars

Let M and N be two languages whose grammars have disjoint sets of non-terminals (rename them if necessary). Let S_M and S_N be their start symbols.

Then we can construct a grammar recognising the following languages, with a new start symbol S:

Constructing grammars

Let M and N be two languages whose grammars have disjoint sets of non-terminals (rename them if necessary). Let S_M and S_N be their start symbols.

Then we can construct a grammar recognising the following languages, with a new start symbol S:

- Union: the grammar for $M \cup N$ starts with $S \to S_M \mid S_N$

All other productions remain unchanged (aside for renaming of variables as needed)

Constructing grammars

Let M and N be two languages whose grammars have disjoint sets of non-terminals (rename them if necessary). Let S_M and S_N be their start symbols.

Then we can construct a grammar recognising the following languages, with a new start symbol S:

- Union: the grammar for $M \cup N$ starts with $S \to S_M \mid S_N$
- Concatenation: the grammar for $M \cup N$ starts with $S \to S_M S_N$

All other productions remain unchanged (aside for renaming of variables as needed)

Constructing grammars

Let M and N be two languages whose grammars have disjoint sets of non-terminals (rename them if necessary). Let S_M and S_N be their start symbols.

Then we can construct a grammar recognising the following languages, with a new start symbol S:

- Union: the grammar for $M \cup N$ starts with $S \to S_M \mid S_N$
- Concatenation: the grammar for $M \cup N$ starts with $S \to S_M S_N$
- Star closure: the grammar for $M^\star$ starts with $S \to S_M S \mid \varepsilon$

All other productions remain unchanged (aside for renaming of variables as needed)

Using the union rule

Let $L = \{\varepsilon, a, b, aa, bb, ..., a^n, b^n, ...\}$

Then $L = M \cup N$ where $M = \{a^n \mid n \geq 0\}, N = \{b^n \mid n \geq 0\}$

So a grammar G_M of M is
and a grammar G_N of N is

Using the union rule

Let $L = \{\varepsilon, a, b, aa, bb, ..., a^n, b^n, ...\}$

Then $L = M \cup N$ where $M = \{a^n \mid n \geq 0\}, N = \{b^n \mid n \geq 0\}$

So a grammar G_M of M is $S_M \to \varepsilon \mid aS_M$
and a grammar G_N of N is

Using the union rule

Let $L = \{\varepsilon, a, b, aa, bb, ..., a^n, b^n, ...\}$

Then $L = M \cup N$ where $M = \{a^n \mid n \geq 0\}, N = \{b^n \mid n \geq 0\}$

So a grammar G_M of M is $S_M \rightarrow \varepsilon \mid aS_M$
and a grammar G_N of N is $S_N \rightarrow \varepsilon \mid bS_N$

Using the union rule

Let $L = \{\varepsilon, a, b, aa, bb, ..., a^n, b^n, ...\}$

Then $L = M \cup N$ where $M = \{a^n \mid n \geq 0\}, N = \{b^n \mid n \geq 0\}$

So a grammar G_M of M is $S_M \rightarrow \varepsilon \mid aS_M$
and a grammar G_N of N is $S_N \rightarrow \varepsilon \mid bS_N$

Using the union rule we get:
$S \rightarrow S_M \mid S_N$
$S_M \rightarrow \varepsilon \mid aS_M$
$S_N \rightarrow \varepsilon \mid bS_N$

Using the concatenation rule

Let $L = \{a^m b^n \mid m \geq 0, n \geq 0\}$

Then $L = MN$ where $M = \{a^m \mid m \geq 0\}, N = \{b^n \mid n \geq 0\}$

So a grammar G_M of M is $S_M \to \varepsilon \mid aS_M$
and a grammar G_N of N is $S_N \to \varepsilon \mid bS_N$

Using the concatenation rule

Let $L = \{a^m b^n \mid m \geq 0, n \geq 0\}$

Then $L = MN$ where $M = \{a^m \mid m \geq 0\}, N = \{b^n \mid n \geq 0\}$

So a grammar G_M of M is $S_M \to \varepsilon \mid aS_M$
and a grammar G_N of N is $S_N \to \varepsilon \mid bS_N$

Using the concatenation rule we get:
$S \to S_M S_N$
$S_M \to \varepsilon \mid aS_M$
$S_N \to \varepsilon \mid bS_N$

Using the star closure rule

Let L be strings consisting of 0 or more occurrences of aa or bb, i.e. $(aa \mid bb)^\star$

Then $L = M^\star$ where $M = \{aa, bb\}$

So a grammar G_M of M is

Using the star closure rule

Let L be strings consisting of 0 or more occurrences of aa or bb, i.e. $(aa \mid bb)^\star$

Then $L = M^\star$ where $M = \{aa, bb\}$

So a grammar G_M of M is $S_M \to aa \mid bb$

Using the star closure rule

Let L be strings consisting of 0 or more occurrences of aa or bb, i.e. $(aa \mid bb)^\star$

Then $L = M^\star$ where $M = \{aa, bb\}$

So a grammar G_M of M is $S_M \rightarrow aa \mid bb$

Using the star closure rule we get:
$S \rightarrow S_M S \mid \varepsilon$
$S_M \rightarrow aa \mid bb$

Parsing

Given a sentence, the problem of *parsing* is determining *how* the grammar generates it.

i.e. To discover the *correct* derivation of the sentence, or the correct parse tree

Parse Tree

A *parse tree* is a tree labelled by symbols from the CFG

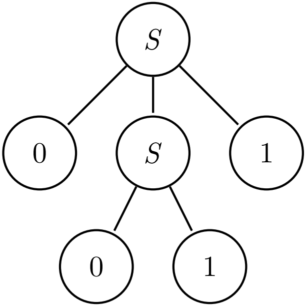
- root $=$ the start symbol
- interior node $=$ a variable
- leaf node $=$ a terminal or ε
- children of X $=$ the right hand side of a production rule for X, in order

Parse Tree

A *parse tree* is a tree labelled by symbols from the CFG

- root = the start symbol
- interior node = a variable
- leaf node = a terminal or ε
- children of X = the right hand side of a production rule for X, in order

Example parse tree for "0011" in
$S \to 0S1 \mid 01$
An in-order traversal of the leaf nodes retrieves the string

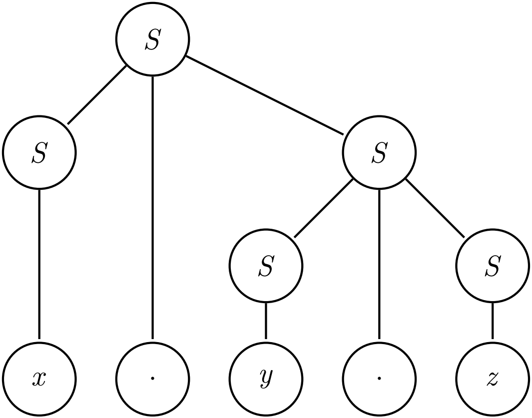


Parse Tree or Derivation Tree

The parse tree defines the (syntactic) *meaning* of a string in the grammar's language

Parse Tree or Derivation Tree

The parse tree defines the (syntactic) *meaning* of a string in the grammar's language



$$S \to S \cdot S$$
$$S \to x \mid y \mid z$$

This parse tree implies that the expression means $x \cdot (y \cdot z)$

Natural Language Processing (NLP) example

$$
\begin{aligned}
S &\rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase} \\
\mathit{NounPhrase} &\rightarrow \mathit{ComplexNoun} \mid \mathit{ComplexNoun}\ \mathit{PrepPhrase} \\
\mathit{VerbPhrase} &\rightarrow \mathit{ComplexVerb} \mid \mathit{ComplexVerb}\ \mathit{PrepPhrase} \\
\mathit{PrepPhrase} &\rightarrow \mathit{Prep}\ \mathit{ComplexNoun} \\
\mathit{ComplexNoun} &\rightarrow \mathit{Article}\ \mathit{Noun} \\
\mathit{ComplexVerb} &\rightarrow \mathit{Verb} \mid \mathit{Verb}\ \mathit{NounPhrase} \\
\mathit{Article} &\rightarrow \mathsf{a} \mid \mathsf{the} \\
\mathit{Noun} &\rightarrow \mathsf{girl} \mid \mathsf{dog} \mid \mathsf{stick} \mid \mathsf{ball} \\
\mathit{Verb} &\rightarrow \mathsf{chases} \mid \mathsf{sees} \\
\mathit{Prep} &\rightarrow \mathsf{with}
\end{aligned}
$$

Simple example

Is the string "a ball" accepted by this grammar?

Simple example

Is the string “a ball” accepted by this grammar?

We have no choice but start by deriving
$S \Rightarrow NounPhrase\ \ VerbPhrase$

Simple example

Is the string "a ball" accepted by this grammar?

We have no choice but start by deriving
$S \Rightarrow NounPhrase\ \ VerbPhrase$

All the production rules for $VerbPhrase$ produce a $ComplexVerb$, which in turn must produce a $Verb$.

Simple example

Is the string “a ball” accepted by this grammar?

We have no choice but start by deriving
$S \Rightarrow NounPhrase\ \ VerbPhrase$

All the production rules for $VerbPhrase$ produce a $ComplexVerb$, which in turn must produce a $Verb$.

Therefore all strings in the language contain a verb. “a ball” does not contain a verb, so it cannot be accepted by the grammar.

Ambiguity: example

Ambiguity: several meanings for the same sentence.

"The girl chases the dog with a stick" has *two leftmost derivations*

$$
\begin{aligned}
Sentence &\Rightarrow NounPhrase\ VerbPhrase \\
&\Rightarrow ComplexNoun\ VerbPhrase \\
&\Rightarrow Article\ Noun\ VerbPhrase \\
&\Rightarrow \text{the } Noun\ VerbPhrase \\
&\Rightarrow \text{the girl } VerbPhrase \\
&\Rightarrow \underline{\text{the girl } ComplexVerb} \\
&\Rightarrow \text{the girl } Verb\ NounPhrase \\
&\Rightarrow \text{the girl chases } NounPhrase \\
&\Rightarrow \text{the girl chases } ComplexNoun\ PrepPhrase \\
&\Rightarrow \text{the girl chases } Article\ Noun\ PrepPhrase \\
&\Rightarrow \text{the girl chases the } Noun\ PrepPhrase \\
&\Rightarrow \text{the girl chases the dog } PrepPhrase \\
&\Rightarrow \text{the girl chases the dog } Prep\ ComplexNoun \\
&\Rightarrow \text{the girl chases the dog with } ComplexNoun \\
&\Rightarrow \text{the girl chases the dog with } Article\ Noun \\
&\Rightarrow \text{the girl chases the dog with a } Noun \\
&\Rightarrow \text{the girl chases the dog with a stick}
\end{aligned}
$$

$$
\begin{aligned}
Sentence &\Rightarrow NounPhrase\ VerbPhrase \\
&\Rightarrow ComplexNoun\ VerbPhrase \\
&\Rightarrow Article\ Noun\ VerbPhrase \\
&\Rightarrow \text{the } Noun\ VerbPhrase \\
&\Rightarrow \text{the girl } VerbPhrase \\
&\Rightarrow \underline{\text{the girl } ComplexVerb\ PrepPhrase} \\
&\Rightarrow \text{the girl } Verb\ NounPhrase\ PrepPhrase \\
&\Rightarrow \text{the girl chases } NounPhrase\ PrepPhrase \\
&\Rightarrow \text{the girl chases } Article\ Noun\ PrepPhrase \\
&\Rightarrow \text{the girl chases the } Noun\ PrepPhrase \\
&\Rightarrow \text{the girl chases the dog } PrepPhrase \\
&\Rightarrow \text{the girl chases the dog Prep } ComplexNoun \\
&\Rightarrow \text{the girl chases the dog with } ComplexNoun \\
&\Rightarrow \text{the girl chases the dog with } Article\ Noun \\
&\Rightarrow \text{the girl chases the dog with a } Noun \\
&\Rightarrow \text{the girl chases the dog with a stick}
\end{aligned}
$$

Ambiguity: example

Ambiguity: several meanings for the same sentence.

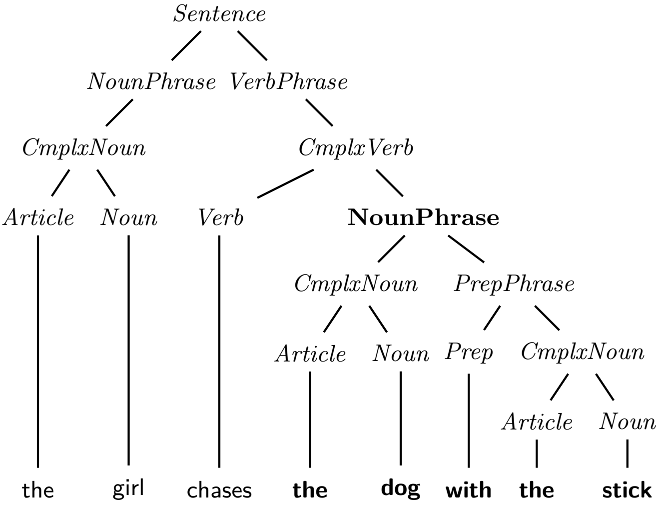
"The girl chases the dog with a stick" has *two leftmost derivations*

$$\begin{aligned} \mathit{Sentence} &\Rightarrow^{+} \text{the girl } \mathit{VerbPhrase} \\ &\Rightarrow \text{the girl } \mathit{ComplexVerb} \\ &\Rightarrow^{+} \text{the girl chases } \underline{\text{the dog with a stick}} \end{aligned}$$

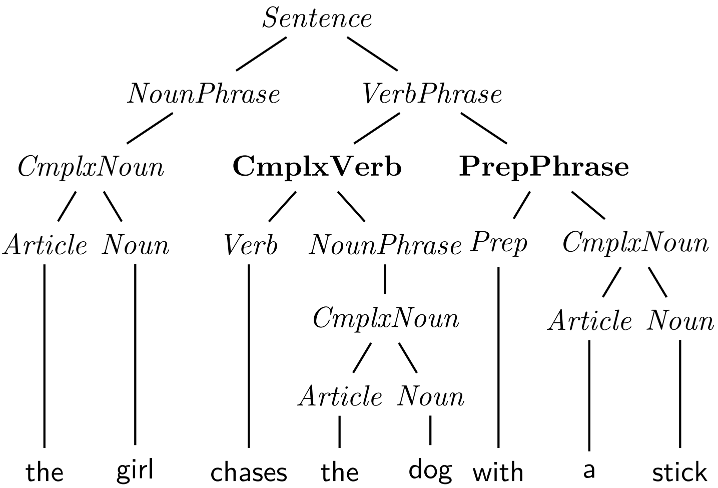
$$\begin{aligned} \mathit{Sentence} &\Rightarrow^{+} \text{the girl } \mathit{VerbPhrase} \\ &\Rightarrow \text{the girl } \mathit{ComplexVerb}\ \mathit{PrepPhrase} \\ &\Rightarrow^{+} \text{the girl chases } \underline{\text{the dog}} \text{ with a stick} \end{aligned}$$

Who has the stick?

First Leftmost Derivation Tree



Second Leftmost Derivation Tree



Ambiguous Grammars

Definition:
A string is *ambiguous* on a given grammar if it has two different parse trees. Otherwise, it is unambigous.

Definition:
A grammar is *ambiguous* if it contains an ambiguous string.

Ambiguous Grammars

Definition:
A string is *ambiguous* on a given grammar if it has two different parse trees. Otherwise, it is unambigous.

Definition:
A grammar is *ambiguous* if it contains an ambiguous string.

Each parse tree has only one leftmost derivation, so this is equivalent to saying that the string has two distinct leftmost derivations.

Ambiguous Grammars

Definition:
A string is *ambiguous* on a given grammar if it has two different parse trees. Otherwise, it is unambigous.

Definition:
A grammar is *ambiguous* if it contains an ambiguous string.

Each parse tree has only one leftmost derivation, so this is equivalent to saying that the string has two distinct leftmost derivations.

Similarly for rightmost derivations.

Is this grammar ambiguous?

$$E \to E - E$$
$$E \to a \mid b \mid c$$

Rightmost derivations of $a - b - c$:

Is this grammar ambiguous?

$$E \to E - E$$
$$E \to a \mid b \mid c$$

Rightmost derivations of $a - b - c$:

$$\begin{aligned} E &\Rightarrow E - E \\ &\Rightarrow E - c \\ &\Rightarrow E - E - c \\ &\Rightarrow E - b - c \\ &\Rightarrow a - b - c \end{aligned}$$

i.e. $(a - b) - c$

Is this grammar ambiguous?

$$E \to E - E$$
$$E \to a \mid b \mid c$$

Rightmost derivations of $a - b - c$:

$$\begin{aligned} E &\Rightarrow E - E \\ &\Rightarrow E - c \\ &\Rightarrow E - E - c \\ &\Rightarrow E - b - c \\ &\Rightarrow a - b - c \end{aligned}$$

$$\begin{aligned} E &\Rightarrow E - E \\ &\Rightarrow E - E - E \\ &\Rightarrow E - E - c \\ &\Rightarrow E - b - c \\ &\Rightarrow a - b - c \end{aligned}$$

i.e. $(a - b) - c$

i.e. $a - (b - c)$

Removing ambiguity (example)

Suppose we want $a - b - c$ to always mean $(a - b) - c$?

Removing ambiguity (example)

Suppose we want $a - b - c$ to always mean $(a - b) - c$?

Introduce a new nonterminal symbol T:

$$E \to E - T \mid T$$

$$T \to a \mid b \mid c$$

Removing ambiguity (example)

Suppose we want $a - b - c$ to always mean $(a - b) - c$?
Introduce a new nonterminal symbol T:

$$E \to E - T \mid T$$
$$T \to a \mid b \mid c$$

Now the only rightmost derivation is:

$$E \Rightarrow E - T$$

Removing ambiguity (example)

Suppose we want $a - b - c$ to always mean $(a - b) - c$?

Introduce a new nonterminal symbol T:

$$E \to E - T \mid T$$

$$T \to a \mid b \mid c$$

Now the only rightmost derivation is:

$$\begin{aligned} E &\Rightarrow E - T \\ &\Rightarrow E - c \end{aligned}$$

Removing ambiguity (example)

Suppose we want $a - b - c$ to always mean $(a - b) - c$?
Introduce a new nonterminal symbol T:

$$E \to E - T \mid T$$
$$T \to a \mid b \mid c$$

Now the only rightmost derivation is:

$$\begin{aligned} E &\Rightarrow E - T \\ &\Rightarrow E - c \\ &\Rightarrow E - T - c \end{aligned}$$

Removing ambiguity (example)

Suppose we want $a - b - c$ to always mean $(a - b) - c$?
Introduce a new nonterminal symbol T:

$$E \to E - T \mid T$$
$$T \to a \mid b \mid c$$

Now the only rightmost derivation is:

$$\begin{aligned} E &\Rightarrow E - T \\ &\Rightarrow E - c \\ &\Rightarrow E - T - c \\ &\Rightarrow E - b - c \end{aligned}$$

Removing ambiguity (example)

Suppose we want $a - b - c$ to always mean $(a - b) - c$?

Introduce a new nonterminal symbol T:

$$E \to E - T \mid T$$
$$T \to a \mid b \mid c$$

Now the only rightmost derivation is:

$$\begin{aligned} E &\Rightarrow E - T \\ &\Rightarrow E - c \\ &\Rightarrow E - T - c \\ &\Rightarrow E - b - c \\ &\Rightarrow T - b - c \end{aligned}$$

Removing ambiguity (example)

Suppose we want $a - b - c$ to always mean $(a - b) - c$?
Introduce a new nonterminal symbol T:

$$E \to E - T \mid T$$
$$T \to a \mid b \mid c$$

Now the only rightmost derivation is:

$$\begin{aligned} E &\Rightarrow E - T \\ &\Rightarrow E - c \\ &\Rightarrow E - T - c \\ &\Rightarrow E - b - c \\ &\Rightarrow T - b - c \\ &\Rightarrow a - b - c \end{aligned}$$

Recursion

If a variable X can generate a string containing X itself, then it is recursive

Recursion

If a variable X can generate a string containing X itself, then it is recursive

- left-recursive: it occurs at the start of the string $X \Rightarrow^{+} X\beta$

Recursion

If a variable X can generate a string containing X itself, then it is recursive

- left-recursive: it occurs at the start of the string $X \Rightarrow^{+} X\beta$
- right-recursive: it occurs at the end of the string $X \Rightarrow^{+} \alpha X$

Recursion

If a variable X can generate a string containing X itself, then it is recursive

- left-recursive: it occurs at the start of the string $X \Rightarrow^{+} X\beta$
- right-recursive: it occurs at the end of the string $X \Rightarrow^{+} \alpha X$
- self-embedding: it occurs in between: $X \Rightarrow^{+} \alpha X\beta$

Recursion

If a variable X can generate a string containing X itself, then it is recursive

- left-recursive: it occurs at the start of the string $X \Rightarrow^{+} X\beta$
- right-recursive: it occurs at the end of the string $X \Rightarrow^{+} \alpha X$
- self-embedding: it occurs in between: $X \Rightarrow^{+} \alpha X \beta$

A grammar is recursive if any of its variables is recursive

Recursion

If a variable X can generate a string containing X itself, then it is recursive

- left-recursive: it occurs at the start of the string $X \Rightarrow^+ X\beta$
- right-recursive: it occurs at the end of the string $X \Rightarrow^+ \alpha X$
- self-embedding: it occurs in between: $X \Rightarrow^+ \alpha X \beta$

A grammar is recursive if any of its variables is recursive

A grammar for an infinite language must contain at least one recursive variable

Balanced Parentheses

This grammar generates the language of balanced parentheses:

$$B \to (B) \mid BB \mid \varepsilon$$

Show that it is ambiguous.

Remove left recursion

Original grammar is left-recursive: $B \to (B) \mid BB \mid \varepsilon$

An equivalent grammar without left-recursion: $B \to (B)B \mid \varepsilon$

Left parsing of $()()()$ is now deterministic:

Remaining input	Derivation steps	
$()()()$	B	start symbol
$()()()$	$(B)B$	$B \to (B)B$
$)()()$	$B)B$	matching terminals
$)()()$	$)B$	$B \to \varepsilon$
$()()$	B	matching terminals
...	...	...

Clean grammars

- No circular definitions: $A_1 \Rightarrow A_2 \Rightarrow ... \Rightarrow A_n \Rightarrow A_1$
  All the A's can generate the same set of strings, therefore there is no reason to distinguish between them. They should be reduced to a single variable.

Clean grammars

- No circular definitions: $A_1 \Rightarrow A_2 \Rightarrow ... \Rightarrow A_n \Rightarrow A_1$
  All the A's can generate the same set of strings, therefore there is no reason to distinguish between them. They should be reduced to a single variable.

- No useless variables:
  A variable is useless if it cannot appear in the derivation of any string. i.e. there is no derivation $S \Rightarrow^+ \alpha X \beta \Rightarrow^+ \sigma$ where σ is a string of terminals. Useless variables can be removed without affecting the language generated by the grammar.

Clean grammars

- No circular definitions: $A_1 \Rightarrow A_2 \Rightarrow ... \Rightarrow A_n \Rightarrow A_1$
  All the A's can generate the same set of strings, therefore there is no reason to distinguish between them. They should be reduced to a single variable.

- No useless variables:
  A variable is useless if it cannot appear in the derivation of any string. i.e. there is no derivation $S \Rightarrow^{+} \alpha X \beta \Rightarrow^{+} \sigma$ where σ is a string of terminals. Useless variables can be removed without affecting the language generated by the grammar.

- No null productions (except for the start symbol)

Types of grammars

We are interested in 4 classes of grammars, depending on the type of production rules that they allow:

Type 0 (unrestricted)	$\chi \to \alpha$
Type 1 (context-sensitive)	$\chi \to \alpha$ where $1 \leq \lvert\chi\rvert \leq \lvert\alpha\rvert$
Type 2 (context-free)	$A \to \alpha$
Type 3 (regular)	$A \to \omega B$ and $A \to \omega$

χ	arbitrary string of one or more symbols
α	arbitrary string of symbols, possibly null
A, B	non-terminal symbols
ω	arbitrary string of terminal symbols

Recall the Chomsky Hierarchy from week 1!

Context-free Grammars

- Generate Context-Free Languages
  - Very important class of languages in CS (compilers, NLP, etc.)
- All rules are in the form $A \to \alpha$
- Closed under Union, Concatenation and Star Closure
- String derivation (left-most, right-most)
- Ambiguous grammars
- Clean grammars

Next lecture:

- Push-Down Automata
- Parsing