COMP2022: Formal Languages and Logic

2018, Semester 2, Week 7

THE UNIVERSITY OF SYDNEY

Outline

- Grammars

- 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, whic

We ha
$\{0^n 1^n \mid n \geq 0\}$, cannot be described using FA o

Today we will introduce *context-free gr*
describe the next category of languages, the *context-free languages*

Grammars

Grammars are another way to describe a language

A *gr* a
lang

The language generated is the set of all strings whic
derived from the grammar

Introductory example (G_1)

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

L

G_1 g ... dy

kno

How does it derive 000111?

Introductory example (G_1)

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

L

G_1 g … dy

kno

How does it derive 000111?

S

Introductory example (G_1)

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

L

G_1 g dy
kno

How does it derive 000111?

$S \Rightarrow 0S1$ usi

Introductory example (G_1)

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

L

G_1 g dy

kno

How does it derive 000111?

$S \Rightarrow 0S1$ usi

$\Rightarrow 00S11$ using rule $S \to 0S1$

Introductory example (G_1)

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

L

G_1 g ... dy

kno

How does it derive 000111?

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

Introductory example (G_2)

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

$$\mathit{Verb} \to \text{likes} \mid \text{see}$$

What language does G_2 generate?

INTRODUCTORY EXAMPLE (G_2)

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

$$\mathit{Verb} \rightarrow \text{likes} \mid \text{see}$$

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-

- A finite set of symbols used to generate the stri
- 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

Definitions

Production rules

- Sometimes called substitution or derivation rules
- Define strings of *variables* and *terminals* which can be

A variable can have many rules:

$Noun \rightarrow$ girl

$Noun \rightarrow$ ball

$Noun \rightarrow$ quokka

Definitions

Production rules

- Sometimes called substitution or derivation rules
- Define strings of *variables* and *terminals* which can be

A variable can have many rules: Th

$Noun \rightarrow$ girl ka

$Noun \rightarrow$ ball

$Noun \rightarrow$ quokka

Another example

$$S \to T \mid (S \cdot S) \mid (\lambda T.S)$$

This is a grammar for lambda calculus expression

- The variables are S, T
- S is the start variable
- The terminals are $a, b, ..., (,), \lambda, ., \cdot$ (i.e. atoms and operators)

Some common notational conventions

If not stated otherwise:

-
-
-
- ..., X, Y, Z are either terminals or varia
- ..., w, x, y, z are strings of terminals
- $\alpha, \beta, \gamma, \ldots$ are strings of terminals and/o

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
-

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
-

Cont

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
-

Cont

Example:

$\{a^n b^n \mid n \in N\}$ is not a regular language (no fi
exists recognising it), but we can prove that it is a cont
language, because the following grammar gene

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

CFG: Formal Definition

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

-

-

- P is a finite set of *production rules* ... ere $\alpha \in V$ and $\beta \in \{V \cup T \cup \{\varepsilon\}\}^{\star}$

- $S \in V$ is a special variable called the *Start Symbol*

Example G_1

$$S \rightarrow 01$$
$$S \rightarrow 0S1$$

Mor

$V =$

$S =$

$P =$

Example G_1

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

Mor

$V =$

$S =$

$P =$

EXAMPLE G_1

$$S \rightarrow 01$$
$$S \rightarrow 0S1$$

Mor

$$V = \{S\}$$
$$\Sigma =$$
$$P =$$

Example G_1

$$S \rightarrow 01$$
$$S \rightarrow 0S1$$

Mor

$$V = \{S\}$$
$$S = S$$
$$P =$$

Example G_1

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

Mor

$$V = \{S\}$$
$$S = S$$

$$P = \{$$
$$S \to 01$$
$$S \to 0S1$$
$$\}$$

Example G_2

$$S \to NounPhrase\ VerbPhrase$$

$$NounPhrase \to \text{the}\ Noun$$

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

$T =$

$V =$

$S =$

$P =$

Example G_2

$$S \to NounPhrase\ VerbPhrase$$

$$NounPhrase \to \text{the}\ Noun$$

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

$$T = \{\text{the, girl, ball, likes, sees}$$

$$V =$$

$$S =$$

$$P =$$

Example G_2

$$S \rightarrow \textit{NounPhrase VerbPhrase}$$

$$\textit{NounPhrase} \rightarrow \text{the } \textit{Noun}$$

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

$T = \{$the, girl, ball, likes, sees

$V = \{S, \textit{NounPhrase}, \textit{VerbPhrase}, \textit{Noun}, \textit{Verb}\}$

$S =$

$P =$

Example G_2

$$S \to NounPhrase\ VerbPhrase$$

$$NounPhrase \to \text{the}\ Noun$$

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

$T = \{$the, girl, ball, likes, sees

$V = \{S, NounPhrase, VerbPhrase, Noun, Verb\}$

$S = S$

$P =$

Example G_2

$$S \rightarrow NounPhrase\ VerbPhrase$$

$$NounPhrase \rightarrow \text{the } Noun$$

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

$T = \{$the, girl, ball, likes, sees

$V = \{S, NounPhrase, VerbPhrase, Noun, Verb\}$

$S = S$

$P =$ (set of seven rules above)

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

Nota

$\alpha \Rightarrow$

$\alpha \Rightarrow$

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

Nota
$\alpha \Rightarrow$
$\alpha \Rightarrow$

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

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

Nota

$\alpha \Rightarrow$

$\alpha \Rightarrow$

The *language* of a grammar $G = (V,$

$L(G) = \{s \mid s \in T^\star$ 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 its productions
-

Derivation of a string

- Begin with the start symbol
- Repeatedly replace one variable with the right hand side of one of its productions
-

Exa

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

Derivation of a string

- Begin with the start symbol
- Repeatedly replace one variable with the right hand side of one of its productions
-

Exa

$S \to 0S1 \mid \varepsilon$

$\Rightarrow$

Derivation of a string

- Begin with the start symbol
- Repeatedly replace one variable with the right hand side of one of its productions
-

Exa

$S \rightarrow 0S1 \mid \varepsilon$

$$\Rightarrow 00S11$$

$$\Rightarrow 000S111$$

$$\Rightarrow$$

Derivation of a string

- Begin with the start symbol
- Repeatedly replace one variable with the right hand side of one of its productions
-

Exa

$S \to 0S1 \mid \varepsilon$

$$\Rightarrow 00S11$$

$$\Rightarrow 000S111$$

$$\Rightarrow 000111$$

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"

$S \Rightarrow N$

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"

$S \Rightarrow N$

$\Rightarrow$ th $\quad Noun\ VerbPhrase$

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"

$S \Rightarrow N$

$\Rightarrow$ th $Noun\ VerbPhrase$

$\Rightarrow$ the girl $VerbPhrase$

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”

$S \Rightarrow NounPhrase\ VerbPhrase$

$\Rightarrow$ the $Noun\ VerbPhrase$

$\Rightarrow$ the girl $VerbPhrase$

$\Rightarrow$ the girl $Verb\ NounPhrase$

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”

$S \Rightarrow N$

$\Rightarrow$ th $Noun\ VerbPhrase$

$\Rightarrow$ 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”

$S \Rightarrow NounPhrase\ VerbPhrase$

$\Rightarrow$ the $Noun\ VerbPhrase$

$\Rightarrow$ the girl $VerbPhrase$

$\Rightarrow$ the girl $Verb\ NounPhrase$

$\Rightarrow$ the girl sees $NounPhrase$

$\Rightarrow$ the girl sees the $Noun$

$\Rightarrow$ the girl sees the ball

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”

$S \Rightarrow N$ … $rase$

$\Rightarrow$ the $Noun\ VerbPhrase$ $\qquad \Rightarrow NounPhrase\ Verb\ NounPhrase$

$\Rightarrow$ the girl $VerbPhrase$

$\Rightarrow$ the girl $Verb\ NounPhrase$

$\Rightarrow$ the girl sees $NounPhrase$

$\Rightarrow$ the girl sees the $Noun$

$\Rightarrow$ the girl sees the ball

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”

$S \Rightarrow N$ … $rase$

$\Rightarrow$ th $Noun\ VerbPhrase$ $\qquad \Rightarrow NounPhrase\ Verb\ NounPhrase$

$\Rightarrow$ the girl $VerbPhrase$ $\qquad \Rightarrow$ … oun

$\Rightarrow$ the girl $Verb\ NounPhrase$

$\Rightarrow$ the girl sees $NounPhrase$

$\Rightarrow$ the girl sees the $Noun$

$\Rightarrow$ the girl sees the ball

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"

$S \Rightarrow N$ … $rase$

$\Rightarrow$ th $Noun\ \ VerbPhrase$ $\qquad \Rightarrow NounPhrase\ \ Verb\ \ NounPhrase$

$\Rightarrow$ the girl $VerbPhrase$ $\qquad \Rightarrow$ … oun

$\Rightarrow$ the girl $Verb\ \ NounPhrase$ $\qquad \Rightarrow$ …

$\Rightarrow$ the girl sees $NounPhrase$

$\Rightarrow$ the girl sees the $Noun$

$\Rightarrow$ the girl sees the ball

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"

$S \Rightarrow N$ ... $rase$

$\Rightarrow$ th $Noun\ VerbPhrase$ $\quad\quad \Rightarrow NounPhrase\ Verb\ NounPhrase$

$\Rightarrow$ the girl $VerbPhrase$ $\quad\quad \Rightarrow$ ... oun

$\Rightarrow$ the girl $Verb\ NounPhrase$ $\quad\quad \Rightarrow$ ...

$\Rightarrow$ the girl sees $NounPhrase$ $\quad\quad \Rightarrow NounPhrase$

$\Rightarrow$ the girl sees the $Noun$

$\Rightarrow$ the girl sees the ball

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"

$S \Rightarrow N$ ... $rase$

$\Rightarrow$ th $Noun\ \ VerbPhrase$ $\qquad \Rightarrow NounPhrase\ \ Verb\ \ NounPhrase$

$\Rightarrow$ the girl $VerbPhrase$ $\qquad \Rightarrow$ ... oun

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

$\Rightarrow$ the girl sees $NounPhrase$ $\qquad \Rightarrow NounPhrase$

$\Rightarrow$ the girl sees the $Noun$ $\qquad \Rightarrow$ the $Noun$ sees the ball

$\Rightarrow$ the girl sees the ball

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"

$S \Rightarrow N$[illegible]

$\Rightarrow$ th[illegible] $Noun\ VerbPhrase$

$\Rightarrow$ the girl $VerbPhrase$

$\Rightarrow$ the girl $Verb\ NounPhrase$

$\Rightarrow$ the girl sees $NounPhrase$

$\Rightarrow$ the girl sees the $Noun$

$\Rightarrow$ the girl sees the ball

[illegible]$rase$

$\Rightarrow NounPhrase\ Verb\ NounPhrase$

$\Rightarrow$ [illegible] oun

$\Rightarrow$ [illegible]

$\Rightarrow NounPhrase$ [illegible]

$\Rightarrow$ the $Noun$ sees the ball

$\Rightarrow$ the girl sees the ball

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.

$$S \to \{SL\}$$

$$L \to ;SL \mid \varepsilon$$

$E \to$ ... (description of an exp

$I \to$ ... (description of an identifier)

Context-Free Languages

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

{Reg

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

Example

Consider the grammar G:

$S \quad AB \qquad\qquad S \quad AB$

What is $L(G)$?

Example

Consider the grammar G:

$S \quad AB \qquad\qquad S \quad AB$

$aaaaB$

What is $L(G)$?

Example

Consider the grammar G:

$S \quad AB$ $\qquad$ $S \quad AB$

$aaaaB$

What is $L(G)$?

$\Rightarrow aaaabbbbbb$

Example

Consider the grammar G:

$S \quad AB \qquad\qquad S \quad AB$

$aaaaB$

What is $L(G)$?

$\Rightarrow aaaabbbbbb$

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. $S \rightarrow aSa \mid bSb \mid \varepsilon$

2.

3. $S \rightarrow SS \mid bS \mid a$

4. $S \rightarrow aT \mid bT \mid \varepsilon$
   $T \rightarrow aS \mid bS$

5. $S \rightarrow aSa \mid bSb \mid a \mid b$

More examples

Give grammars generating these languages

1. $ba^{n+1}b \quad n \quad 0$

2.

3. Even-length strings in $\{a, b\}^\star$ wit

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.

The

lang

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.

The
lang

- Union: the grammar for $M \cup N$ s

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.

The
lang

- Union: the grammar for $M \cup N$ s
- Concatenation: the grammar for 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.

The
lang

- Union: the grammar for $M \cup N$ s
- Concatenation: the grammar for S_N
- Star closure: the grammar for M ε

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

Using the union rule

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

The n n $\geq 0\}$

So a gr

and a g G_N N

Using the union rule

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

The n n $\geq 0\}$

So a gr

and a g G_N N

Using the union rule

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

The n n $\geq 0\}$

So a gr

and a g G_N N $S_N \to \varepsilon \mid bS_N$

Using the union rule

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

The [illegible] n [illegible] n [illegible] $\geq 0\}$

So a gr[illegible]
and a g[illegible] G_N [illegible] N [illegible] $S_N \to \varepsilon \mid bS_N$

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

Using the concatenation rule

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

The m n $0\}$

So a gr
and a g G_N N $S_N \to \varepsilon \mid bS_N$

Using the concatenation rule

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

The m n $0\}$

So a gr

and a g G_N N $S_N \to \varepsilon \mid bS_N$

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. $(aa \mid bb)^*$

The

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. $(aa \mid bb)^*$

The

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

Using the star closure rule

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

The

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

Using the star closure rule we get:

$S \to S_M S \mid \varepsilon$

$S_M \to aa \mid bb$

Parsing

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

i.e. To

correct parse tree

Parse Tree

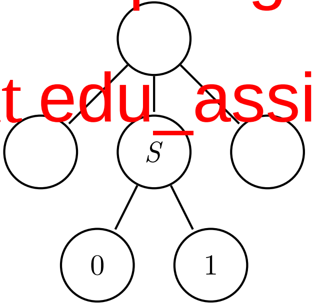
A *parse tree* is a tree labelled by symbols from the CFG

- root $=$ the start symbol
- interior node $=$ a variable
-
-

Parse Tree

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

- root = the start symbol
- interior node = a variable
-
-



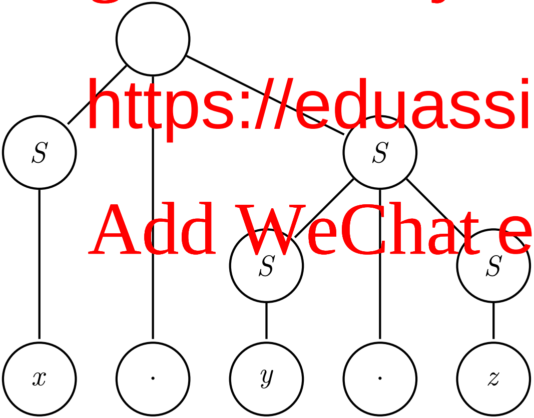
Example parse tree for "0011" in
$S \to 0S1 \mid 01$
An in-order traversal of the leaf nodes retrieves the string

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

$y \mid z$

Natural Language Processing (NLP) example

$$S \rightarrow NounPhrase\ \ VerbPhrase$$

$$NounPhrase \quad ComplexNoun \quad ComplexNoun\ \ PrepPhrase$$

$$ase$$

$$ComplexNoun \rightarrow Article\ \ Noun$$

$$ComplexVerb \rightarrow Verb \mid Verb\ \ Noun$$

$$Article \rightarrow \text{a} \mid \text{the}$$

$$Noun \rightarrow \text{girl} \mid \text{dog} \mid \text{stick} \mid$$

$$Verb \rightarrow \text{chases} \mid \text{sees}$$

$$Prep \rightarrow \text{with}$$

Simple example

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

Simple example

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

We ha

$S \Rightarrow$

Simple example

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

We ha

$S \Rightarrow$

All the production rules for $VerbPhra$ rb,
which in turn must produce a $Verb$.

Simple example

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

We ha

$S \Rightarrow$

All the production rules for $VerbPhra$ rb, 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*

$\mathit{Sentence} \Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase}$
$\Rightarrow \mathit{ComplexNoun}\ \mathit{VerbPhrase}$
$\Rightarrow$
$\Rightarrow$
$\Rightarrow$
$\Rightarrow$
$\Rightarrow$ the girl $\mathit{Verb}\ \mathit{NounPhrase}$
$\Rightarrow$ the girl chases $\mathit{NounPhrase}$
$\Rightarrow$ the girl chases $\mathit{ComplexNoun}\ \mathit{PrepPhrase}$
$\Rightarrow$ the girl chases $\mathit{Article}\ \mathit{Noun}\ \mathit{PrepPhrase}$
$\Rightarrow$ the girl chases the $\mathit{Noun}\ \mathit{PrepPhrase}$
$\Rightarrow$ the girl chases the dog $\mathit{PrepPhrase}$
$\Rightarrow$ the girl chases the dog $\mathit{Prep}\ \mathit{ComplexNoun}$
$\Rightarrow$ the girl chases the dog with $\mathit{ComplexNoun}$
$\Rightarrow$ the girl chases the dog with $\mathit{Article}\ \mathit{Noun}$
$\Rightarrow$ the girl chases the dog with a Noun
$\Rightarrow$ the girl chases the dog with a stick

$\mathit{Sentence} \Rightarrow \mathit{NounPhrase}\ \mathit{VerbPhrase}$
$\mathit{ComplexNoun}\ \mathit{VerbPhrase}$
e
$\mathit{epPhrase}$
the girl $\mathit{Verb}\ \mathit{NounPhrase}\ \mathit{PrepPhrase}$
$\mathit{pPhrase}$
Phrase
rase
ase
$\Rightarrow$ the girl chases the dog Prep $\mathit{ComplexNoun}$
$\Rightarrow$ the girl chases the dog with $\mathit{ComplexNoun}$
$\Rightarrow$ the girl chases the dog with $\mathit{Article}\ \mathit{Noun}$
$\Rightarrow$ the girl chases the dog with a Noun
$\Rightarrow$ the girl chases the dog with a stick

Ambiguity: example

Ambiguity: several meanings for the same sentence.

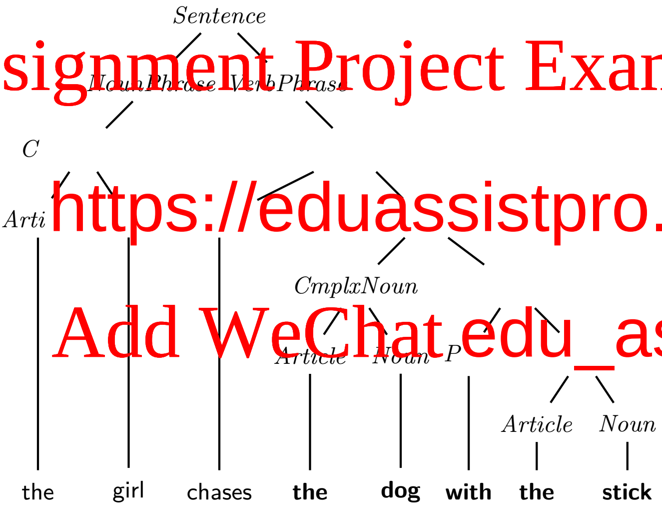
"The girl chases the dog with a stick" has *two leftmost derivations*

$\Rightarrow$

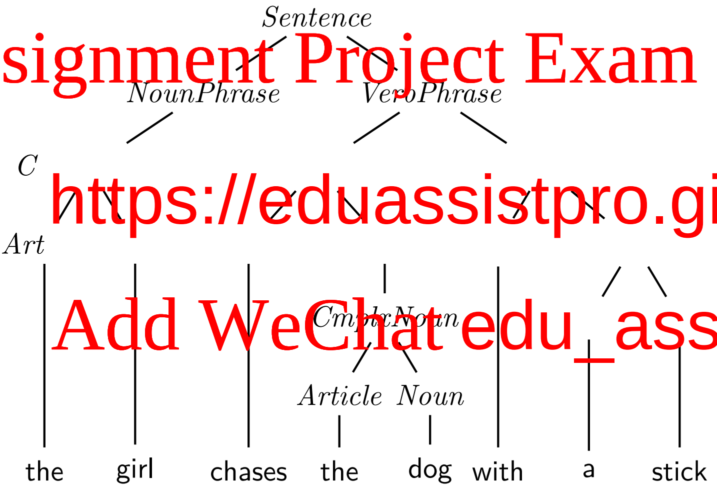
$$
\begin{aligned}
\mathit{Sentence} &\Rightarrow^{+} \text{the girl } \mathit{VerbP} \\
&\Rightarrow \text{the girl } \mathit{ComplexVerb}\ \mathit{PrepPhrase} \\
&\Rightarrow^{+} \text{the girl chases } \underline{\text{the dog}} \text{ with a stick}
\end{aligned}
$$

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.

Defi
A gra

Ambiguous Grammars

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

Defi
A gra

Each parse tree has only one leftmost derivation, s
equivalent to saying that the string has two distinc
derivations.

Ambiguous Grammars

Definition:

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

Defi

A gra

Each parse tree has only one leftmost derivation, s

equivalent to saying that the string has two distinc

derivations.

Similarly for rightmost derivations.

Is this grammar ambiguous?

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

Righ

Is this grammar ambiguous?

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

Righ

$$\Rightarrow E - c$$

$$\Rightarrow E - E - c$$
$$\Rightarrow E - b - c$$
$$\Rightarrow a - b - c$$

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

Is this grammar ambiguous?

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

Righ

$$\Rightarrow E - c$$

$$\Rightarrow E - E - c$$
$$\Rightarrow E - b - c$$
$$\Rightarrow a - b - c$$

$$\Rightarrow a - b - c$$

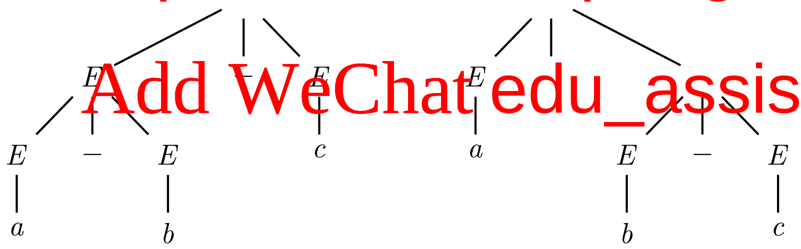
i.e. $(a - b) - c$

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

Is this grammar ambiguous?

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

Righ



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:

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

$$T \quad a \quad b \quad c$$

Removing ambiguity (example)

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

Introduce a new nonterminal symbol T:

$$E \rightarrow E - T \mid T$$

$$T \quad a \quad b \quad c$$

No

$$E \Rightarrow E - T$$

Removing ambiguity (example)

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

Introduce a new nonterminal symbol T:

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

$$T \quad a \quad b \quad c$$

No

$$E \Rightarrow E - T$$

$$\Rightarrow E - c$$

Removing ambiguity (example)

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

Introduce a new nonterminal symbol T:

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

$$T \quad a \quad b \quad c$$

No

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

Removing ambiguity (example)

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

Introduce a new nonterminal symbol T:

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

$$T \quad a \quad b \quad c$$

No

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

Removing ambiguity (example)

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

Introduce a new nonterminal symbol T:

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

No

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

Removing ambiguity (example)

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

Introduce a new nonterminal symbol T:

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

$$T \quad a \quad b \quad c$$

No

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

Removing ambiguity (example)

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

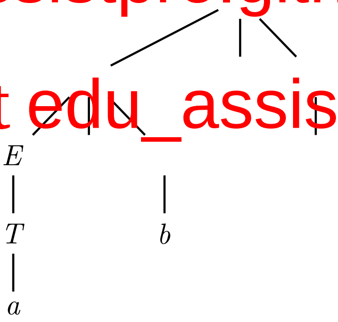
Introduce a new nonterminal symbol T:

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

$$T \quad a \quad b \quad c$$

No

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



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

- $^{+}\ X\beta$

Recursion

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

- $^{+}\ X\beta$
- $^{+}\ \alpha X$

Recursion

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

- $^{+} X\beta$
- $^{+} \alpha X$
-

Recursion

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

- $^{+}\ X\beta$
- $^{+}\ \alpha X$
-

A grammar is recursive if any of its variables is recursi

Recursion

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

- $^{+}\ X\beta$
- $^{+}\ \alpha X$
-

A grammar is recursive if any of its variables is recursi

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

Balanced Parentheses

This grammar generates the language of balanced parentheses:

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 p

Re		
$()()()$	B	
$()()()$	$(B)B$	
$)()()$	$B)B$	
$)()()$	$)B$	$B \to \varepsilon$
$()()$	B	matching terminals
...	...	...

Clean grammars

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

Clean grammars

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

- A variable is useless if it cannot appear in the de
  string. i.e. there is no derivation σ
  is a string of terminals. Useless variables can b
  without affecting the language generate

Clean grammars

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

- A variable is useless if it cannot appear in the de
  string. i.e. there is no derivation σ
  is a string of terminals. Useless variables can b
  without affecting the language generate

- 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) χ α
Ty
Ty
Ty

χ	arbitrary string of one or more symbols
α	arbitrary string of symbols, possibly n
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 $A \quad \alpha$
-
-
-
- Clean grammars

Next lecture:

- Push-Down Automata
- Parsing