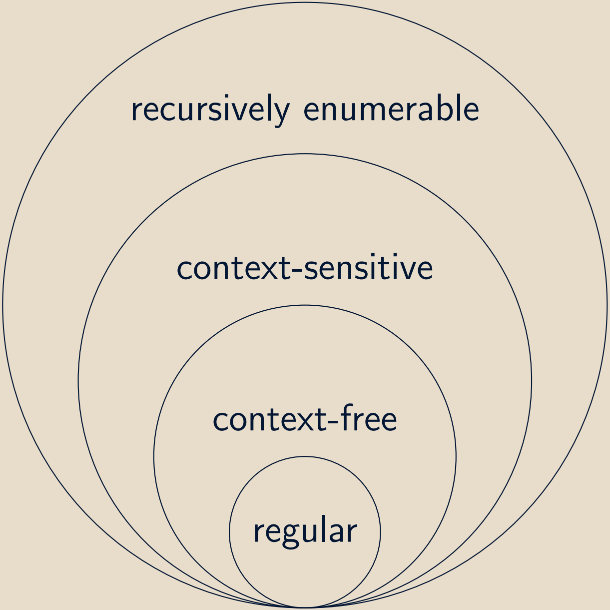
CMSC 141 Automata and Language Theory

Context-Free Languages

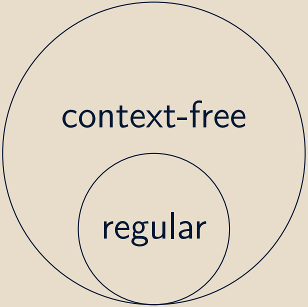
Mark Froilan B. Tandoc

October 8, 2014

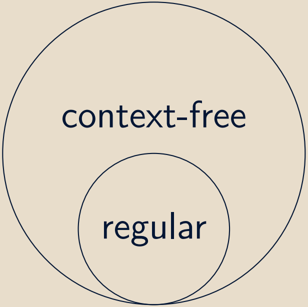
Chomsky Hierarchy



Chomsky Hierarchy

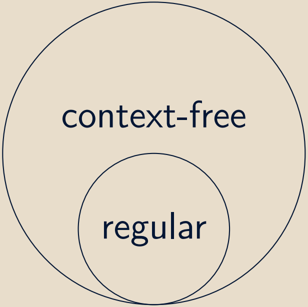


Chomsky Hierarchy



- Regular Languages $V \rightarrow T^*(V + \varepsilon)$
  - $S \rightarrow abS \mid a \mid \varepsilon$
  - $S \rightarrow 0S \mid 1S \mid 11T$
    $T \rightarrow 0T \mid 1T \mid \varepsilon$

Chomsky Hierarchy



- Regular Languages $V \rightarrow T^*(V+\varepsilon)$
  - $S \rightarrow abS \mid a \mid \varepsilon$
  - $S \rightarrow 0S \mid 1S \mid 11T$
    $T \rightarrow 0T \mid 1T \mid \varepsilon$
- Context-Free Languages $V \rightarrow (V+T)^*$
  - $S \rightarrow ab \mid aSb$
  - $S \rightarrow () \mid SS \mid (S) \mid \varepsilon$

Ambiguous Grammars

Ambiguous Grammars

- A grammar is *ambiguous* if there is more than one parse tree for any string x in the language

Ambiguous Grammars

- A grammar is *ambiguous* if there is more than one parse tree for any string x in the language
- A grammar is *non-ambiguous* if every string in a language has a unique parse tree

Ambiguous Grammars

- A grammar is *ambiguous* if there is more than one parse tree for any string x in the language
- A grammar is *non-ambiguous* if every string in a language has a unique parse tree
- $S \rightarrow ab \mid aSb$ is non-ambiguous

Ambiguous Grammars

- A grammar is *ambiguous* if there is more than one parse tree for any string x in the language
- A grammar is *non-ambiguous* if every string in a language has a unique parse tree
- $S \rightarrow ab \mid aSb$ is non-ambiguous
- $S \rightarrow a \mid S + S$ is ambiguous

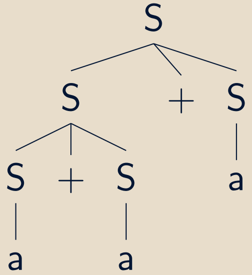
Ambiguous Grammars

Grammar: $S \rightarrow a \mid S + S$
Derive: `a + a + a`

Ambiguous Grammars

Grammar: $S \rightarrow a \mid S + S$

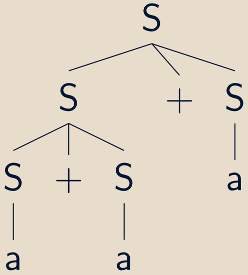
Derive: `a + a + a`

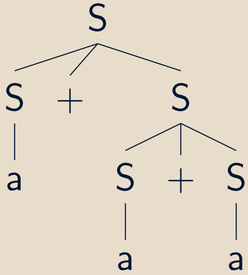


Ambiguous Grammars

Grammar: $S \rightarrow a \mid S + S$

Derive: `a + a + a`





Ambiguity in Natural Languages

- The man on the hill saw the boy with a telescope.

Ambiguity in Natural Languages

- The man on the hill saw the boy with a telescope.
- Look at the dog with one eye.

Removing Ambiguity

Removing Ambiguity

- $G_1 : S \to a \mid S + S$ is ambiguous

Removing Ambiguity

- $G_1 : S \to a \mid S + S$ is ambiguous
- $G_2 : S \to a \mid S + a$ is non-ambiguous

Removing Ambiguity

- $G_1 : S \rightarrow a \mid S + S$ is ambiguous
- $G_2 : S \rightarrow a \mid S + a$ is non-ambiguous
- The two grammars generate the same language $L(G_1) = L(G_2)$

Removing Ambiguity

- $G_1 : S \rightarrow a \mid S + S$ is ambiguous
- $G_2 : S \rightarrow a \mid S + a$ is non-ambiguous
- The two grammars generate the same language $L(G_1) = L(G_2)$
- G_2 is better because it is non-ambiguous by forcing the rule on left-associativity

Removing Ambiguity

Removing Ambiguity

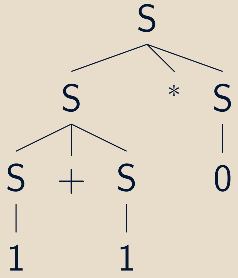
- Consider: $S \to 0 \mid 1 \mid S + S \mid S^{*}S \mid (S)$

Removing Ambiguity

- Consider: $S \to 0 \mid 1 \mid S + S \mid S^*S \mid (S)$
- How many ways can we parse $1 + 1^*0$

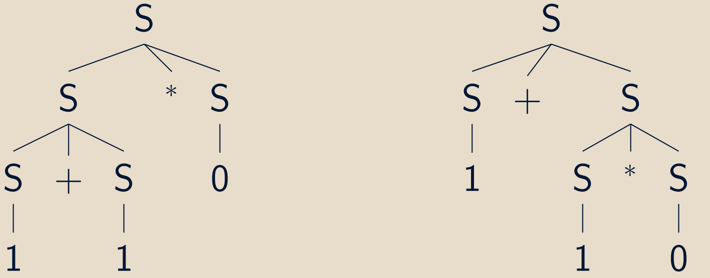
Removing Ambiguity

- Consider: $S \rightarrow 0 \mid 1 \mid S + S \mid S^*S \mid (S)$
- How many ways can we parse $1 + 1^*0$



Removing Ambiguity

- Consider: $S \to 0 \mid 1 \mid S + S \mid S^* S \mid (S)$
- How many ways can we parse $1 + 1^* 0$



Removing Ambiguity

A non-ambiguous grammar

Removing Ambiguity

A non-ambiguous grammar
(enforcing operator precedence)

Removing Ambiguity

A non-ambiguous grammar

(enforcing operator precedence)

$$
\begin{aligned}
S &\rightarrow E \\
E &\rightarrow E + T \mid T && \text{(expressions)} \\
T &\rightarrow T^{*} F \mid F && \text{(terms)} \\
F &\rightarrow 0 \mid 1 \mid (E) && \text{(factors)}
\end{aligned}
$$

Removing Ambiguity

A non-ambiguous grammar

(enforcing operator precedence)

$$
\begin{aligned}
S &\to E \\
E &\to E + T \mid T && \text{(expressions)} \\
T &\to T{*}F \mid F && \text{(terms)} \\
F &\to 0 \mid 1 \mid (E) && \text{(factors)}
\end{aligned}
$$

Exercise: Try parsing strings like $1 + 1{*}0$ and $(1 + 1){*}0$

Dangling-Else Problem

Dangling-Else Problem

Consider:

$S \rightarrow B$ | if C then S | if C then S else S

$B \rightarrow \textit{block}$

$C \rightarrow (\textit{cond})$

Dangling-Else Problem

Consider:

$S \rightarrow B \mid \text{if C then S} \mid \text{if C then S else S}$

$B \rightarrow \mathit{block}$

$C \rightarrow (\mathit{cond})$

```
if (cond1) then
  if (cond2) then block1
else block2
```

Dangling-Else Problem

Consider:

$S \rightarrow B \mid \text{if C then S} \mid \text{if C then S else S}$

$B \rightarrow \textit{block}$

$C \rightarrow (\textit{cond})$

```
if (cond1) then
  if (cond2) then block1
else block2
```

```
if (cond1) then
  if (cond2) then block1
  else block2
```

Dangling-Else Problem

Consider:

$S \rightarrow B$ | if C then S | if C then S else S

$B \rightarrow block$

$C \rightarrow (cond)$

```
if (cond1) then
  if (cond2) then block1
else block2
```

```
if (cond1) then
  if (cond2) then block1
  else block2
```

Exercise: Design/Create a non-ambiguous version of the grammar that associates an else-clause to the nearest if

Inherently-Ambiguous Languages

Inherently-Ambiguous Languages

- A language L is inherently ambiguous if every grammar for L is ambiguous

Inherently-Ambiguous Languages

- A language L is inherently ambiguous if every grammar for L is ambiguous
- Example: $L = \{a^m b^m c^n\} \bigcup \{a^m b^n c^n\}; m, n > 0$

Inherently-Ambiguous Languages

- A language L is inherently ambiguous if every grammar for L is ambiguous
- Example: $L = \{a^m b^m c^n\} \bigcup \{a^m b^n c^n\}; m, n > 0$
- Sample strings are aabbcccc, abbbccc, abc

Inherently-Ambiguous Languages

- A language L is inherently ambiguous if every grammar for L is ambiguous
- Example: $L = \{a^m b^m c^n\} \bigcup \{a^m b^n c^n\}; m, n > 0$
- Sample strings are aabbcccc, abbbccc, abc
- A CFG for L is given by

$$
\begin{aligned}
S &\rightarrow XC \mid AY \\
X &\rightarrow ab \mid aXb \\
A &\rightarrow a \mid aA \\
C &\rightarrow c \mid cC \\
Y &\rightarrow bc \mid bYc
\end{aligned}
$$

Inherently-Ambiguous Languages

- A language L is inherently ambiguous if every grammar for L is ambiguous
- Example: $L = \{a^m b^m c^n\} \bigcup \{a^m b^n c^n\}; m, n > 0$
- Sample strings are aabbcccc, abbbccc, abc
- A CFG for L is given by

$$
\begin{aligned}
S &\rightarrow XC \mid AY \\
X &\rightarrow ab \mid aXb \\
A &\rightarrow a \mid aA \\
C &\rightarrow c \mid cC \\
Y &\rightarrow bc \mid bYc
\end{aligned}
$$

- Show that aabbcc has 2 different parse trees

Inherently-Ambiguous Languages

- A language L is inherently ambiguous if every grammar for L is ambiguous
- Example: $L = \{a^m b^m c^n\} \bigcup \{a^m b^n c^n\}; m, n > 0$
- Sample strings are aabbcccc, abbbccc, abc
- A CFG for L is given by

$$\begin{aligned} S &\rightarrow XC \mid AY \\ X &\rightarrow ab \mid aXb \\ A &\rightarrow a \mid aA \\ C &\rightarrow c \mid cC \\ Y &\rightarrow bc \mid bYc \end{aligned}$$

- Show that aabbcc has 2 different parse trees
- Why?

References

- Previous slides on CMSC 141
- M. Sipser. Introduction to the Theory of Computation. Thomson, 2007.
- J.E. Hopcroft, R. Motwani and J.D. Ullman. Introduction to Automata Theory, Languages and Computation. 2nd ed, Addison-Wesley, 2001.
- E.A. Albacea. Automata, Formal Languages and Computations, UPLB Foundation, Inc. 2005
- JFLAP, www.jflap.org
- Various online LaTeX and Beamer tutorials