Context-free Grammars

CSCI 3130 Formal Languages and Automata Theory

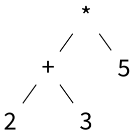
Siu On CHAN

Chinese University of Hong Kong

Fall 2015

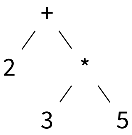
Precedence in Arithmetic Expressions

```
bash$ python
Python 2.7.9 (default, Apr  2 2015, 15:33:21)
>>> 2+3*5
17
```



$= 25$

or



$= 17$

Grammars describe meaning

EXPR → EXPR + TERM

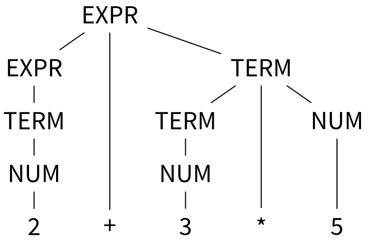
EXPR → TERM

TERM → TERM * NUM

TERM → NUM

NUM → 0-9

rules for valid (simple) arithmetic expressions



Rules always yield the correct meaning

Grammar of English

SENTENCE $\rightarrow$ NOUN-PHRASE VERB-PHRASE

$$\underbrace{\text{a girl}}_{\text{NOUN-PHRASE}}\ \underbrace{\text{likes the boy}}_{\text{VERB-PHRASE}}$$

NOUN-PHRASE $\rightarrow$ A-NOUN

or $\rightarrow$ A-NOUN PREP-PHRASE

$$\underbrace{\text{a girl}}_{\text{A-NOUN}} \qquad\qquad \underbrace{\text{a girl}}_{\text{A-NOUN}}\ \underbrace{\text{with a flower}}_{\text{PREP-PHRASE}}$$

Grammar of English

NOUN-PHRASE $\rightarrow$ A-NOUN

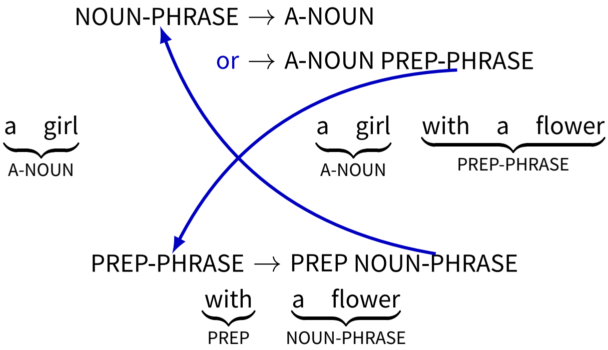
or $\rightarrow$ A-NOUN PREP-PHRASE

$\underbrace{\text{a girl}}_{\text{A-NOUN}}$ $\qquad$ $\underbrace{\text{a girl}}_{\text{A-NOUN}}\ \underbrace{\text{with a flower}}_{\text{PREP-PHRASE}}$

PREP-PHRASE $\rightarrow$ PREP NOUN-PHRASE

$\underbrace{\text{with}}_{\text{PREP}}\ \underbrace{\text{a flower}}_{\text{NOUN-PHRASE}}$

Grammar of English



Recursive structure

Grammar of (parts of) English

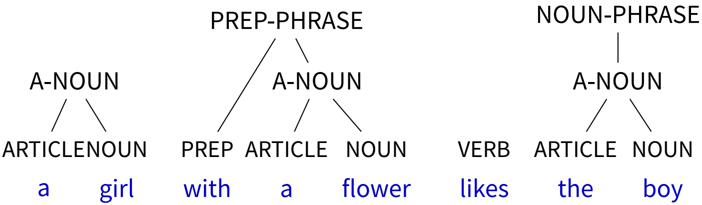
SENTENCE $\longrightarrow$ NOUN-PHRASE VERB-PHRASE
NOUN-PHRASE $\longrightarrow$ A-NOUN
NOUN-PHRASE $\longrightarrow$ A-NOUN PREP-PHRASE
VERB-PHRASE $\longrightarrow$ CMPLX-VERB
VERB-PHRASE $\longrightarrow$ CMPLX-VERB PREP-PHRASE
PREP-PHRASE $\longrightarrow$ PREP A-NOUN
A-NOUN $\longrightarrow$ ARTICLE NOUN
CMPLX-VERB $\longrightarrow$ VERB NOUN-PHRASE
CMPLX-VERB $\longrightarrow$ VERB

ARTICLE $\longrightarrow$ a
ARTICLE $\longrightarrow$ the
NOUN $\longrightarrow$ boy
NOUN $\longrightarrow$ girl
NOUN $\longrightarrow$ flower
VERB $\longrightarrow$ likes
VERB $\longrightarrow$ touches
VERB $\longrightarrow$ sees
PREP $\longrightarrow$ with

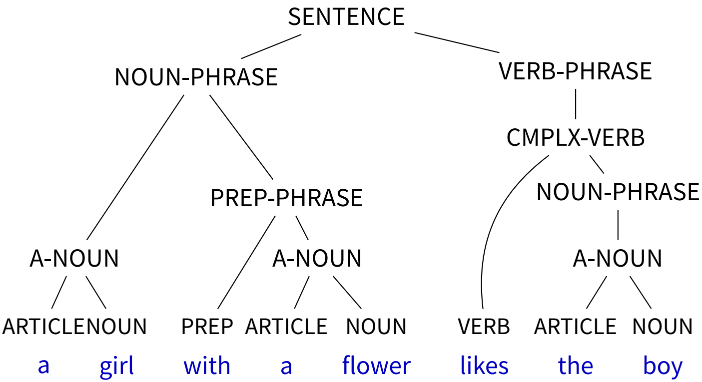
The meaning of sentences

ARTICLE	NOUN	PREP	ARTICLE	NOUN	VERB	ARTICLE	NOUN
a	girl	with	a	flower	likes	the	boy

The meaning of sentences



The meaning of sentences



Context-free grammar

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

A, B are variables
0, 1 are terminals
$A \to 0A1$ is a production
A is the start variable

Context-free grammar

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

A, B are variables
0, 1 are terminals
$A \to 0A1$ is a production
A is the start variable

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$$

derivation

Context-free grammar

A context-free grammar is given by (V, Σ, R, S) where

- V is a finite set of variables or non-terminals
- Σ is a finite set of terminals
- R is a set of productions or substitution rules of the form

  $$A \to \alpha$$

  A is a variable and α is a string of variables and terminals
- $S \in V$ is a variable called the start variable

Notation and conventions

$E \to E\texttt{+}E$

$E \to \texttt{(}E\texttt{)}$

$E \to N$

$N \to \texttt{0}N$

$N \to \texttt{1}N$

$N \to \texttt{0}$

$N \to \texttt{1}$

Variables: E, N

Terminals: +, (,), 0, 1

Start variable: E

shorthand:

$E \to E\texttt{+}E \mid \texttt{(}E\texttt{)} \mid N$

$N \to \texttt{0}N \mid \texttt{1}N \mid \texttt{0} \mid \texttt{1}$

conventions:

variables in UPPERCASE

start variable comes first

Derivation

derivation: a sequential application of productions

$$
\begin{aligned}
E &\Rightarrow E\text{+}E \\
&\Rightarrow (E)\text{+}E \\
&\Rightarrow (E)\text{+}N \\
&\Rightarrow (E)\text{+}1 \\
&\Rightarrow (E\text{+}E)\text{+}1 \\
&\Rightarrow (N\text{+}E)\text{+}1 \\
&\Rightarrow (N\text{+}N)\text{+}1 \\
&\Rightarrow (N\text{+}1N)\text{+}1 \\
&\Rightarrow (N\text{+}10)\text{+}1 \\
&\Rightarrow (1\text{+}10)\text{+}1
\end{aligned}
$$

derivation ↓

$$
\begin{aligned}
E &\rightarrow E\text{+}E \mid (E) \mid N \\
N &\rightarrow 0N \mid 1N \mid 0 \mid 1
\end{aligned}
$$

$\alpha \Rightarrow \beta$

application of one production

Derivation

derivation: a sequential application of productions

$$
\begin{aligned}
E &\Rightarrow E\text{+}E \\
&\Rightarrow (E)\text{+}E \\
&\Rightarrow (E)\text{+}N \\
&\Rightarrow (E)\text{+1} \\
&\Rightarrow (E\text{+}E)\text{+1} \\
&\Rightarrow (N\text{+}E)\text{+1} \\
&\Rightarrow (N\text{+}N)\text{+1} \\
&\Rightarrow (N\text{+1}N)\text{+1} \\
&\Rightarrow (N\text{+10})\text{+1} \\
&\Rightarrow (\text{1+10})\text{+1}
\end{aligned}
$$

derivation (↓)

$$
\begin{aligned}
E &\to E\text{+}E \mid (E) \mid N \\
N &\to 0N \mid 1N \mid 0 \mid 1
\end{aligned}
$$

$\alpha \Rightarrow \beta$
application of one production

$E \overset{*}{\Rightarrow} (\text{1+10})\text{+1}$

$\alpha \overset{*}{\Rightarrow} \beta$ derivation

Context-free languages

The language of a CFG is the set of all strings at the end of a derivation

$$L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$$

Questions we will ask:
I give you a CFG, what is the language?
I give you a language, write a CFG for it

Analysis example 1

$$A \to 0A1 \mid B$$
$$B \to \#$$

$$L(G) = \{0^n\#1^n \mid n \geqslant 0\}$$

Can you derive:

00#11

#

00#111

00##11

Analysis example 1

$$A \to \mathtt{0}A\mathtt{1} \mid B$$
$$B \to \#$$

$$L(G) = \{\mathtt{0}^n\#\mathtt{1}^n \mid n \geqslant 0\}$$

Can you derive:

00#11 $\quad A \Rightarrow \mathtt{0}A\mathtt{1} \Rightarrow \mathtt{00}A\mathtt{11} \Rightarrow \mathtt{00}B\mathtt{11} \Rightarrow \mathtt{00\#11}$

#

00#111

00##11

Analysis example 1

$$A \to 0A1 \mid B$$
$$B \to \#$$

$$L(G) = \{0^n\#1^n \mid n \geqslant 0\}$$

Can you derive:

00#11	$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$
#	$A \Rightarrow B \Rightarrow \#$
00#111	
00##11	

Analysis example 1

$$A \to 0A1 \mid B$$
$$B \to \#$$

$$L(G) = \{0^n\#1^n \mid n \geqslant 0\}$$

Can you derive:

00#11	$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$
#	$A \Rightarrow B \Rightarrow \#$
00#111	No: uneven number of 0s and 1s
00##11	No: too many #

Analysis example 2

$$S \to SS \mid (S) \mid \varepsilon$$

Can you derive

() (()())

$$S \Rightarrow (S)$$
$$\Rightarrow ()$$

Analysis example 2

$$S \to SS \mid (S) \mid \varepsilon$$

Can you derive

$()$ $\qquad$ $(()())$

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow () \end{aligned}$$

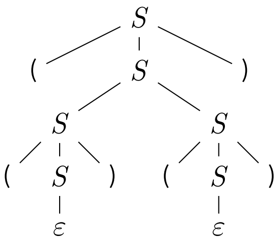
$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow ((S)S) \\ &\Rightarrow ((S)(S)) \\ &\Rightarrow (()(S)) \\ &\Rightarrow (()()) \end{aligned}$$

Parse trees

$$S \to SS \mid (S) \mid \varepsilon$$

A parse tree gives a more compact representation

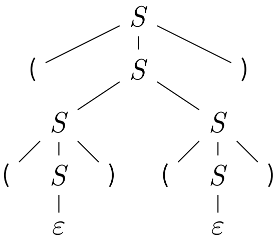
$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow ((S)S) \\ &\Rightarrow ((S)(S)) \\ &\Rightarrow (()(S)) \\ &\Rightarrow (()()) \end{aligned}$$



Parse trees

$$S \Rightarrow (S) \Rightarrow (SS) \Rightarrow ((S)S) \Rightarrow ((S)(S)) \Rightarrow (()(S)) \Rightarrow (()())$$

$$S \Rightarrow (S) \Rightarrow (SS) \Rightarrow ((S)S) \Rightarrow (()S) \Rightarrow (()(S)) \Rightarrow (()())$$



$$S \Rightarrow (S) \Rightarrow (SS) \Rightarrow (S(S)) \Rightarrow ((S)(S)) \Rightarrow (()(S)) \Rightarrow (()())$$

$$S \Rightarrow (S) \Rightarrow (SS) \Rightarrow (S(S)) \Rightarrow (S()) \Rightarrow ((S)()) \Rightarrow (()())$$

One parse tree can represent many derivations

Analysis example 2

$$S \to SS \mid (S) \mid \varepsilon$$

Can you derive

(()()

())(()

Analysis example 2

$$S \to SS \mid (S) \mid \varepsilon$$

Can you derive

(()() No: uneven number of (and)

())(()

Analysis example 2

$$S \to SS \mid (S) \mid \varepsilon$$

Can you derive

(()()	No: uneven number of (and)
())(()	No: some prefix has too many)

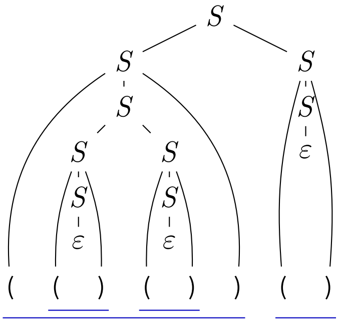
Analysis example 2

$$S \to SS \mid (S) \mid \varepsilon$$

Analysis example 2

$$S \to SS \mid (S) \mid \varepsilon$$

$$L(G) = \{w \mid w \text{ has the same number of (and)}$$
$$\text{no prefix of } w \text{ has more) than (}\}$$



Parsing rules:

Divide w into blocks with same number of (and)

Each block is in $L(G)$

Parse each block recursively

Design example 1

$$L = \{0^n 1^n \mid n \geqslant 0\}$$

These strings have recursive structure

00001111

000111

0011

01

ε

Design example 1

$$L = \{0^n 1^n \mid n \geqslant 0\}$$

These strings have recursive structure

00001111

000111

0011

01

ε

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

Design example 2

$$L = \{\mathtt{0}^n\mathtt{1}^n\mathtt{0}^m\mathtt{1}^m \mid n \geqslant 0, m \geqslant 0\}$$

Examples:

010011

00110011

000111

Design example 2

$$L = \{0^n 1^n 0^m 1^m \mid n \geqslant 0, m \geqslant 0\}$$

Examples:
010011
00110011
000111

These strings have two parts:

$$L = L_1 L_2$$
$$L_1 = \{0^n 1^n \mid n \geqslant 0\}$$
$$L_2 = \{0^m 1^m \mid m \geqslant 0\}$$

rules for L_1 : $S_1 \to 0 S_1 1 \mid \varepsilon$
L_2 is the same as L_1

$$S \to S_1 S_1$$
$$S_1 \to 0 S_1 1 \mid \varepsilon$$

Design example 3

$$L = \{\mathtt{0}^n \mathtt{1}^m \mathtt{0}^m \mathtt{1}^n \mid n \geqslant 0, m \geqslant 0\}$$

Examples:

011001

0011

1100

00110011

Design example 3

$$L = \{\mathtt{0}^n\mathtt{1}^m\mathtt{0}^m\mathtt{1}^n \mid n \geqslant 0, m \geqslant 0\}$$

Examples:
011001
0011
1100
00110011

These strings have a nested structure:

outer part: $\mathtt{0}^n\mathtt{1}^n$
inner part: $\mathtt{1}^m\mathtt{0}^m$

$$S \to \mathtt{0}S\mathtt{1} \mid I$$
$$I \to \mathtt{1}I\mathtt{0} \mid \varepsilon$$

Design example 4

$$L = \{x \mid x \text{ has two 0-blocks with the same number 0s}\}$$

01011, 001011001, 10010101000 — allowed

11001000, 01111 — not allowed

Design example 4

$$L = \{x \mid x \text{ has two 0-blocks with the same number 0s}\}$$

01011, 001011001, 10010101000 — allowed

11001000, 01111 — not allowed

$$\underbrace{1\,0\,0\,1}_{\substack{\text{initial part} \\ A}}\;\underbrace{0\,0\,1\,1\,0\,1\,0\,0}_{\substack{\text{middle part} \\ B}}\;\underbrace{1\,0\,1\,1\,0}_{\substack{\text{final part} \\ C}}$$

A: cannot end in 0

C: cannot begin with 0

Design example 4

$$\underbrace{1001}_{A}\underbrace{00110100}_{B}\underbrace{10110}_{C}$$

A: ε, or ends in 1

C: ε, or begins with 1

U: any string

$$S \to ABC$$
$$A \to \varepsilon \mid U1$$
$$U \to 0U \mid 1U \mid \varepsilon$$
$$C \to \varepsilon \mid 1U$$

Design example 4

$$\underbrace{1\,0\,0\,1}_{A}\underbrace{\color{blue}{0\,0}\,1\,1\,0\,1\,\color{blue}{0\,0}}_{B}\underbrace{1\,0\,1\,1\,0}_{C}$$

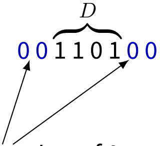
$$S \rightarrow ABC$$
$$A \rightarrow \varepsilon \mid U1$$
$$U \rightarrow 0U \mid 1U \mid \varepsilon$$
$$C \rightarrow \varepsilon \mid 1U$$
$$B \rightarrow 0D0 \mid 0B0$$
$$D \rightarrow 1U1 \mid 1$$

A: ε, or ends in 1

C: ε, or begins with 1

U: any string

B has recursive structure



same number of 0s

at least one 0

D: begins and ends in 1

Context-free versus regular

Write a CFG for the language $(0 + 1)^*111$

Context-free versus regular

Write a CFG for the language $(0+1)^* 111$

$$S \to U111$$
$$U \to 0U \mid 1U \mid \varepsilon$$

Can you do so for every regular language?

Context-free versus regular

Write a CFG for the language $(0+1)^*111$

$$S \rightarrow U111$$
$$U \rightarrow 0U \mid 1U \mid \varepsilon$$

Can you do so for every regular language?

Every regular language is context-free

From regular to context-free

regular expression	$\Rightarrow$ CFG
$\varnothing$	grammar with no rules
ε	$S \rightarrow \varepsilon$
a (alphabet symbol)	$S \rightarrow$ a
$E_1 + E_2$	$S \rightarrow S_1 \mid S_2$
$E_1 E_2$	$S \rightarrow S_1 S_2$
E_1^*	$S \rightarrow S S_1 \mid \varepsilon$

S becomes the new start variable

Context-free versus regular

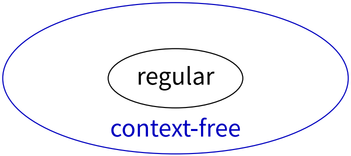
Is every context-free language regular?

Context-free versus regular

Is every context-free language regular?

$S \to 0S1 \qquad L = \{0^n 1^n \mid n \geqslant 0\}$

Is context-free but not regular

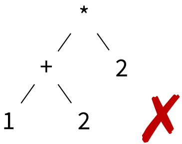


Ambiguity

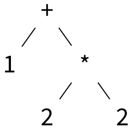
Ambiguity

$$E \to E+E \mid E^*E \mid (E) \mid N$$
$$N \to 1N \mid 2N \mid 1 \mid 2$$

1+2*2



$= 6$



$= 5$

A CFG is ambiguous if some string has more than one parse tree

Example

Is $S \rightarrow SS|\text{x}$ ambiguous?

Example

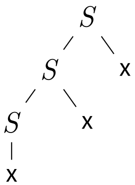
Is $S \rightarrow SS|\text{x}$ ambiguous?

Yes, because

Two ways to derive xxx

Disambiguation

$S \to SS|\mathsf{x}$ $\quad\Rightarrow\quad$ $S \to S\mathsf{x}|\mathsf{x}$



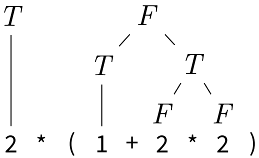
Sometimes we can rewrite the grammar to remove ambiguity

Disambiguation

$$E \to E\text{+}E \mid E\text{*}E \mid (E) \mid N$$
$$N \to \mathsf{1}N \mid \mathsf{2}N \mid \mathsf{1} \mid \mathsf{2}$$

+ and * have the same precedence!

Dived expression into terms and factors



Disambiguation

$$E \rightarrow E\texttt{+}E \mid E\texttt{*}E \mid \texttt{(}E\texttt{)} \mid N$$
$$N \rightarrow \texttt{1}N \mid \texttt{2}N \mid \texttt{1} \mid \texttt{2}$$

An expression is a sum of one or more terms $\quad E \rightarrow T \mid E\texttt{+}T$

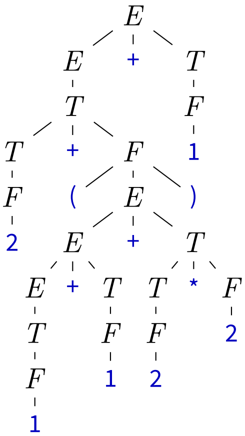
Each term is a product of one or more factors $\quad T \rightarrow F \mid T\texttt{*}F$

Each factor is a parenthesized expression or a number $\quad F \rightarrow \texttt{(}E\texttt{)} \mid \texttt{1} \mid \texttt{2}$

Parsing example

$$E \to T \mid E\texttt{+}T$$
$$T \to F \mid T\texttt{*}F$$
$$F \to (E) \mid \texttt{1} \mid \texttt{2}$$

Parse tree for
2+(1+1+2*2)+1



Disambiguation

Disambiguation is not always possible because

There exists inherently ambiguous languages

There is no general procedure for disambiguation

Disambiguation

Disambiguation is not always possible because
There exists inherently ambiguous languages
There is no general procedure for disambiguation

In programming languages, ambiguity comes from the precedence rules,
and we can resolve like in the example

In English, ambiguity is sometimes a problem:

$\overbrace{\text{I look at}}\overbrace{\text{the dog with one eye}}$

$\underbrace{\text{I look at the dog}}\underbrace{\text{with one eye}}$