Context-free Grammars

CSCI 3130 Formal Languages and Automata Theory

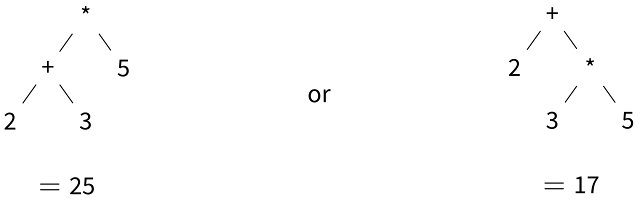
Siu On CHAN

Chinese University of Hong Kong

Fall 2016

Precedence in Arithmetic Expressions

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



Grammars describe meaning

EXPR → EXPR + TERM

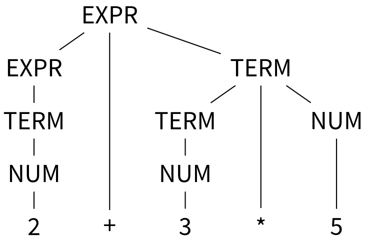
EXPR → TERM

TERM → TERM * NUM

TERM → NUM

NUM → 0-9

rules for valid (simple)
arithmetic expressions



Rules always yield the correct meaning

Grammar of English

$$\text{SENTENCE} \longrightarrow \text{NOUN-PHRASE VERB-PHRASE}$$

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

$$\text{NOUN-PHRASE} \longrightarrow \text{A-NOUN}$$

$$\text{or} \longrightarrow \text{A-NOUN PREP-PHRASE}$$

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

Grammar of English

$$\text{NOUN-PHRASE} \rightarrow \text{A-NOUN}$$

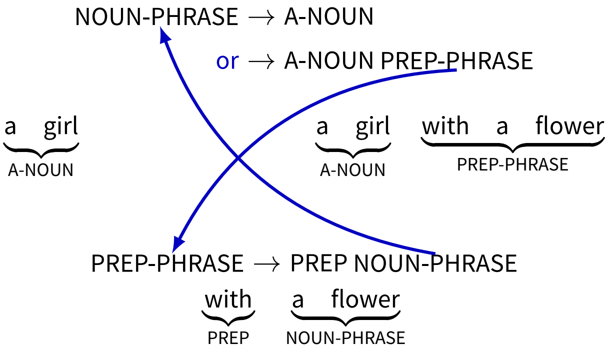
$$\text{or} \rightarrow \text{A-NOUN PREP-PHRASE}$$

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

$$\text{PREP-PHRASE} \rightarrow \text{PREP NOUN-PHRASE}$$

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

Grammar of English



Recursive structure

Grammar of (parts of) English

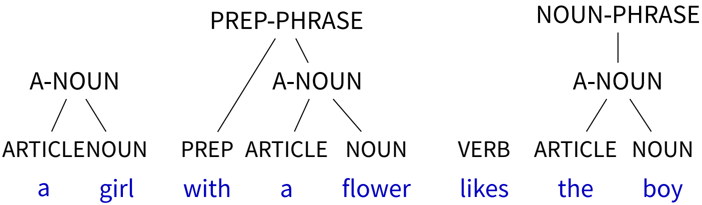
SENTENCE $\longrightarrow$ NOUN-PHRASE VERB-PHRASE
NOUN-PHRASE $\longrightarrow$ A-NOUN
NOUN-PHRASE $\longrightarrow$ A-NOUN PREP-PHRASE
VERB-PHRASE $\longrightarrow$ CMPLX-VERB
VERB-PHRASE $\longrightarrow$ CMPLX-VERB PREP-PHRASE
PREP-PHRASE $\longrightarrow$ PREP A-NOUN
A-NOUN $\longrightarrow$ ARTICLE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

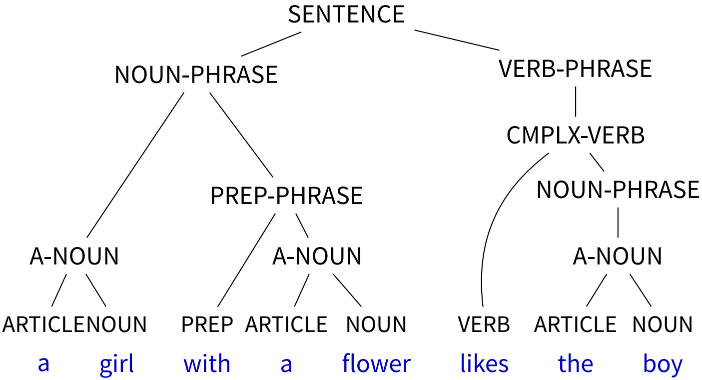
The meaning of sentences

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

$$E \to E\text{+}E$$
$$E \to (E)$$
$$E \to N$$

$$N \to 0N$$
$$N \to 1N$$
$$N \to 0$$
$$N \to 1$$

Variables: E, N
Terminals: +, (,), 0, 1
Start variable: E

shorthand:

$$E \to E\text{+}E \mid (E) \mid N$$
$$N \to 0N \mid 1N \mid 0 \mid 1$$

conventions:

variables in UPPERCASE
start variable comes first

Derivation

derivation: a sequential application of productions

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

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

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

derivation

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

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

$E \stackrel{*}{\Rightarrow} (\texttt{1+10})\texttt{+1}$

$\alpha \stackrel{*}{\Rightarrow} \beta$ 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

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

Can you derive:

00#11

#

00#111

00##11

Analysis example 1

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

Can you derive:

00#11 $\quad A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$

#

00#111

00##11

Analysis example 1

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

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

() ${}$ (()())

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

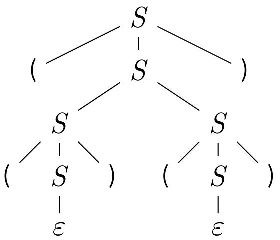
$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow ((S)S) \\ &\Rightarrow ((S)(S)) \\ &\Rightarrow (()(S)) \\ &\Rightarrow (()()) \end{aligned}$$

Parse trees

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

A parse tree gives a more compact representation

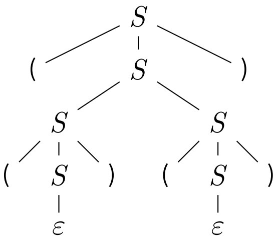
$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow ((S)S) \\ &\Rightarrow ((S)(S)) \\ &\Rightarrow (()(S)) \\ &\Rightarrow (()()) \end{aligned}$$



Parse trees

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

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



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

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

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

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

Can you derive

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

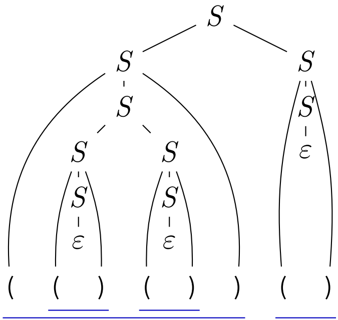
Analysis example 2

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

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

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

Examples:

010011

00110011

000111

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

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 : $S_1 \to \mathtt{0}S_1\mathtt{1} \mid \varepsilon$
L_2 is the same as L_1

$$S \to S_1 S_1$$
$$S_1 \to \mathtt{0}S_1\mathtt{1} \mid \varepsilon$$

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

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

Examples:
011001
0011
1100
00110011

These strings have a nested structure:

outer part: $0^n 1^n$
inner part: $1^m 0^m$

$$S \to 0S1 \mid I$$
$$I \to 1I0 \mid \varepsilon$$

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

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

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

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

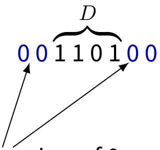
$$\begin{aligned}
S &\to ABC \\
A &\to \varepsilon \mid U\mathtt{1} \\
U &\to \mathtt{0}U \mid \mathtt{1}U \mid \varepsilon \\
C &\to \varepsilon \mid \mathtt{1}U \\
B &\to \mathtt{0}D\mathtt{0} \mid \mathtt{0}B\mathtt{0} \\
D &\to \mathtt{1}U\mathtt{1} \mid \mathtt{1}
\end{aligned}$$

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