

Context-free Grammars

CSCI 3130 Formal Languages and Automata Theory

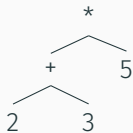
Siu On CHAN

Fall 2021

Chinese University of Hong Kong

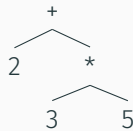
Precedence in Arithmetic Expressions

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



= 25

or



= 17

Grammars describe meaning

$\text{EXPR} \rightarrow \text{EXPR} + \text{TERM}$

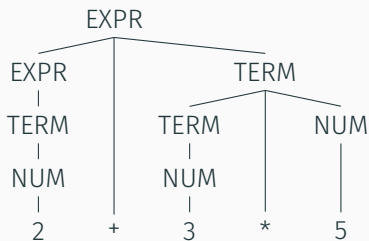
$\text{EXPR} \rightarrow \text{TERM}$

$\text{TERM} \rightarrow \text{TERM} * \text{NUM}$

$\text{TERM} \rightarrow \text{NUM}$

$\text{NUM} \rightarrow 0-9$

rules for valid (simple)
arithmetic expressions



Rules always yield the correct meaning

SENTENCE → NOUN-PHRASE VERB-PHRASE

a girl likes the boy
NOUN-PHRASE VERB-PHRASE

NOUN-PHRASE → A-NOUN

or → A-NOUN PREP-PHRASE

a girl
A-NOUN

a girl with a flower
A-NOUN PREP-PHRASE

NOUN-PHRASE \rightarrow A-NOUN

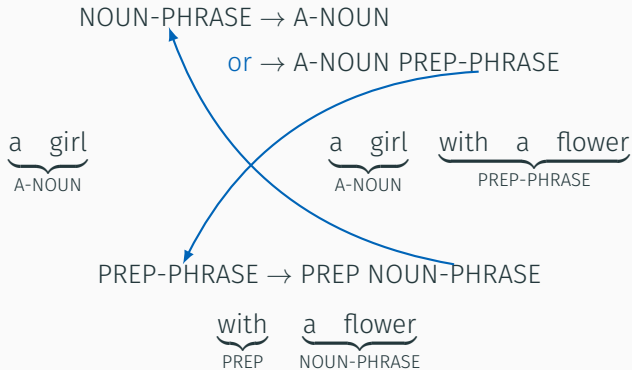
or \rightarrow A-NOUN PREP-PHRASE

a girl
A-NOUN

a girl with a flower
A-NOUN PREP-PHRASE

PREP-PHRASE \rightarrow PREP NOUN-PHRASE

with a flower
PREP NOUN-PHRASE



Recursive structure

Grammar of (parts of) English

SENTENCE → NOUN-PHRASE VERB-PHRASE

NOUN-PHRASE → A-NOUN

NOUN-PHRASE → A-NOUN PREP-PHRASE

VERB-PHRASE → CMLPX-VERB

VERB-PHRASE → CMLPX-VERB PREP-PHRASE

PREP-PHRASE → PREP A-NOUN

A-NOUN → ARTICLE NOUN

CMLPX-VERB → VERB NOUN-PHRASE

CMLPX-VERB → VERB

ARTICLE → a

ARTICLE → the

NOUN → boy

NOUN → girl

NOUN → flower

VERB → likes

VERB → touches

VERB → sees

PREP → with

The meaning of sentences

ARTICLE NOUN

a girl

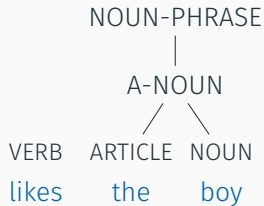
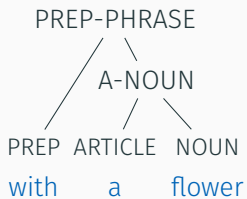
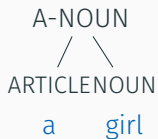
PREP ARTICLE NOUN

with a flower

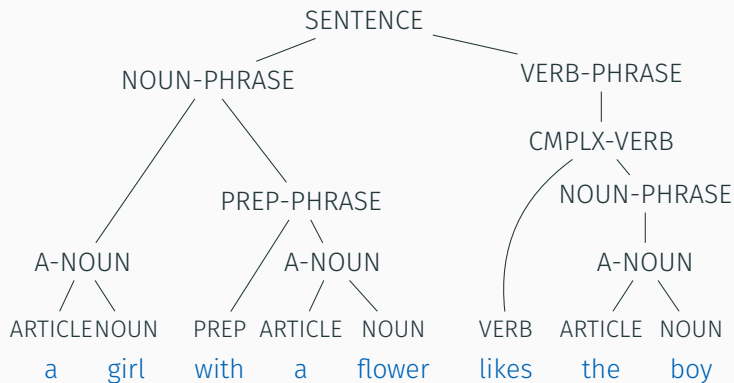
VERB ARTICLE NOUN

likes the boy

The meaning of sentences



The meaning of sentences



Context-free grammar

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

A, B are variables

0, 1 are terminals

$A \rightarrow 0A1$ 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 finite set of **productions** or **substitution rules** of the form

$$A \rightarrow \alpha$$

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

$$E \rightarrow E+E$$

$$E \rightarrow (E)$$

$$E \rightarrow N$$

$$N \rightarrow 0N$$

$$N \rightarrow 1N$$

$$N \rightarrow 0$$

$$N \rightarrow 1$$

Variables: E, N

Terminals: $+, (,), 0, 1$

Start variable: E

shorthand:

$$E \rightarrow E+E \mid (E) \mid N$$

$$N \rightarrow 0N \mid 1N \mid 0 \mid 1$$

conventions:

variables in UPPERCASE

start variable comes first

Derivation

derivation: a sequential application of productions

$$E \Rightarrow E + E$$

$$\Rightarrow (E) + E$$

$$\Rightarrow (E) + N$$

$$\Rightarrow (E) + 1$$

$$\Rightarrow (E + E) + 1$$

$$\Rightarrow (N + E) + 1$$

$$\Rightarrow (N + N) + 1$$

$$\Rightarrow (N + 1N) + 1$$

$$\Rightarrow (N + 10) + 1$$

$$\Rightarrow (1 + 10) + 1$$

derivation

$$E \rightarrow E + E \mid (E) \mid N$$

$$N \rightarrow 0N \mid 1N \mid 0 \mid 1$$

$$\alpha \Rightarrow \beta$$

application of one
production

Derivation

derivation: a sequential application of productions

$$E \Rightarrow E + E$$

$$\Rightarrow (E) + E$$

$$\Rightarrow (E) + N$$

$$\Rightarrow (E) + 1$$

$$\Rightarrow (E + E) + 1$$

$$\Rightarrow (N + E) + 1$$

$$\Rightarrow (N + N) + 1$$

$$\Rightarrow (N + 1N) + 1$$

$$\Rightarrow (N + 10) + 1$$

$$\Rightarrow (1 + 10) + 1$$

$$E \xRightarrow{*} (1 + 10) + 1$$

derivation

$$E \rightarrow E + E \mid (E) \mid N$$

$$N \rightarrow 0N \mid 1N \mid 0 \mid 1$$

$$\alpha \Rightarrow \beta$$

application of one
production

$$\alpha \xRightarrow{*} \beta \quad \text{derivation}$$

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

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

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

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

Can you derive:

00#11

#

00#111

00##11

Analysis example 1

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

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

$A \rightarrow 0A1 \mid B$

$B \rightarrow \#$

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

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

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

Can you derive

()

((()))

Analysis example 2

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

Can you derive

()

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

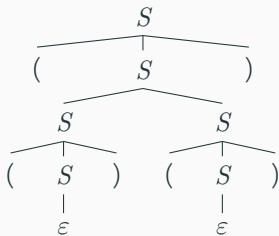
((()))

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

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

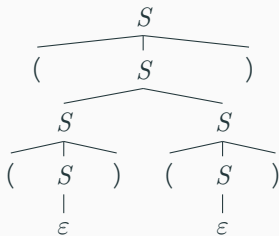
A [parse tree](#) gives a more compact representation

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



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

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

Can you derive

$(())$

$()()$

Analysis example 2

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

Can you derive

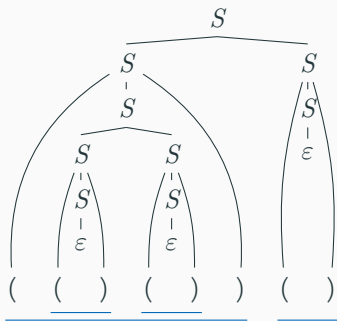
$(())$ No: **uneven** number of (and)

$())(())$ No: some **prefix** has too many)

Analysis example 2

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

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

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

These strings have a [recursive structure](#)

00001111

000111

0011

01

ϵ

Design example 1

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

These strings have a [recursive structure](#)

00001111

000111

0011

01

ϵ

$$S \rightarrow 0S1 \mid \epsilon$$

Design example 2

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

Examples:

010011

00110011

000111

Design example 2

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

Examples:

010011

00110011

000111

These strings have **two parts**:

$$L = L_1 L_2$$

$$L_1 = \{0^n 1^n \mid n \geq 0\}$$

$$L_2 = \{0^m 1^m \mid m \geq 0\}$$

rules for L_1 : $S_1 \rightarrow 0S_11 \mid \varepsilon$

L_2 is the same as L_1

$$S \rightarrow S_1 S_1$$

$$S_1 \rightarrow 0S_11 \mid \varepsilon$$

Design example 3

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

Examples:

011001

0011

1100

00110011

Design example 3

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

Examples:

011001

0011

1100

00110011

These strings have a nested structure:

outer part: $0^n 1^n$

inner part: $1^m 0^m$

$$S \rightarrow 0S1 \mid I$$

$$I \rightarrow 1I0 \mid \varepsilon$$

Design example 4

$L = \{x \mid x \text{ has (at least) two nonempty 0-blocks}$
with the same number of 0s}

01011, 001011001, 10010101000
allowed

11001000, 01111
not allowed

Design example 4

$L = \{x \mid x \text{ has (at least) two nonempty 0-blocks}$
with the same number of 0s}

01011, 001011001, 10010101000
allowed

11001000, 01111
not allowed

1 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0
initial part middle part final part
A B C

A: cannot end in 0

C: cannot begin with 0

Design example 4

1 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0

$\underbrace{\hspace{1.5cm}}_A$ $\underbrace{\hspace{2.5cm}}_B$ $\underbrace{\hspace{1.5cm}}_C$

$S \rightarrow ABC$

$A \rightarrow \varepsilon \mid U1$

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

$C \rightarrow \varepsilon \mid 1U$

A : ε , or ends in 1

C : ε , or begins with 1

U : any string

Design example 4

1 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0

$\underbrace{\hspace{1.5cm}}_A \quad \underbrace{\hspace{3.5cm}}_B \quad \underbrace{\hspace{1.5cm}}_C$

$S \rightarrow ABC$

$A \rightarrow \varepsilon \mid U1$

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

$C \rightarrow \varepsilon \mid 1U$

$B \rightarrow 0D0 \mid 0B0$

$D \rightarrow 1U1 \mid 1$

A : ε , or ends in 1

C : ε , or begins with 1

U : any string

B has a recursive structure

0 0 1 1 0 1 0 0

$\underbrace{\hspace{1.5cm}}_D$

same number of 0s
at least one 0

D : begins and ends in 1