CENG 2010 - Programming Language Concepts

Week 4: Context-Free Grammars and Syntactical Analysis

Burak Ekici

March 27, 2023

Table of Contents

Syntax

- keywords, formatting and grammatical structure of the language

Syntax

- keywords, formatting and grammatical structure of the language
- usually superficial differences in between languages:

```
/* in C */
if (x == y) then {...} else {...}

(* in OCaml *)
if x = y then begin ... end else begin ... end
```

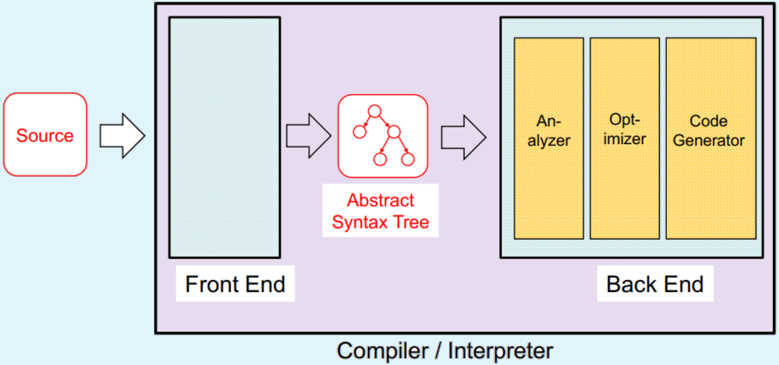
Syntax

- keywords, formatting and grammatical structure of the language
- usually superficial differences in between languages:

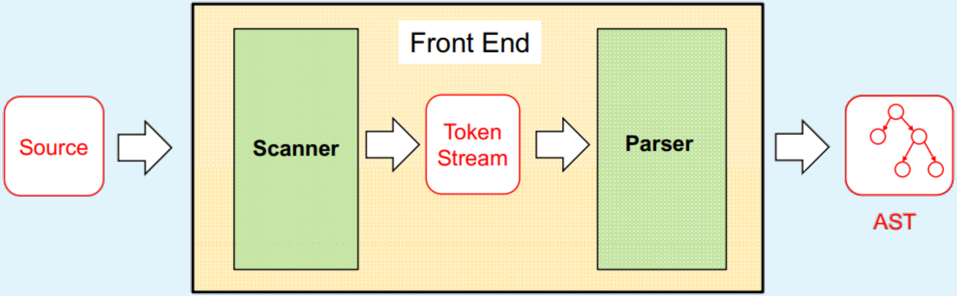
```
/* in C */
if (x == y) then {...} else {...}

(* in OCaml *)
if x = y then begin ... end else begin ... end
```

- regular expressions, context-free grammars, and parsing constitute the syntax of a language

Architecture of Compilers and Interpreters

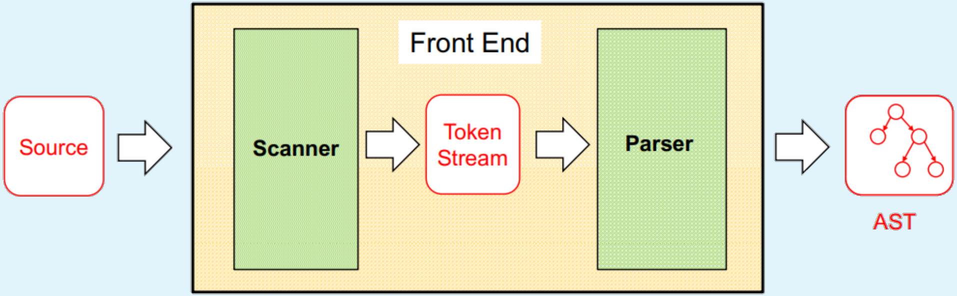


Front-End: Lexer and Parser



scanner/lexer : source code $\mapsto$ tokens (keywords, variables, numbers, etc.) regular expressions

Front-End: Lexer and Parser



scanner/lexer	:	source code	$\mapsto$	tokens (keywords, variables, numbers, etc.)	regular expressions
parser	:	tokens	$\mapsto$	abstract syntax trees/parse trees	context free grammars

Implementation (Front-End)

- goal: map program texts into PTs/ASTs

Implementation (Front-End)

- goal: map program texts into PTs/ASTs
- PTs and ASTs are easier to work with – analyze, optimize, execute the program

Implementation (Front-End)

- goal: map program texts into PTs/ASTs
- PTs and ASTs are easier to work with – analyze, optimize, execute the program
- front end use regular expressions at scanning/lexing

Implementation (Front-End)

- goal: map program texts into PTs/ASTs
- PTs and ASTs are easier to work with – analyze, optimize, execute the program
- front end use regular expressions at scanning/lexing
- regular expressions cannot reliably parse paired braces $\{\{\ldots\}\}$ and parentheses $(((\ldots)))$, etc.

Implementation (Front-End)

- goal: map program texts into PTs/ASTs
- PTs and ASTs are easier to work with – analyze, optimize, execute the program
- front end use regular expressions at scanning/lexing
- regular expressions cannot reliably parse paired braces $\{\{\ldots\}\}$ and parentheses $(((\ldots)))$, etc.
- regular expressions for tokenizing (scanning/lexing), and context free grammars for parsing tokens

Table of Contents

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
  4. $S \in N$: start symbol

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \rightarrow \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
  4. $S \in N$: start symbol
- one step derivation relation $\xrightarrow[G]{1}$ on $(N \cup \Sigma)^*$: $\beta A \gamma \xrightarrow[G]{1} \beta \alpha \gamma$ if $A \rightarrow \alpha \in P$ and $\beta, \gamma \in (N \cup \Sigma)^*$

Example

CFG $G = (N, \Sigma, P, S)$

1. $N = \{S\}$
2. $\Sigma = \{a, b\}$
3. $P = \{S \rightarrow aSb,\ S \rightarrow \varepsilon\}$

Example

CFG $G = (N, \Sigma, P, S)$

1. $N = \{S\}$
2. $\Sigma = \{\mathrm{a}, \mathrm{b}\}$
3. $P = \{S \to aSb,\ S \to \varepsilon\}$

two derivations:

$$S \xrightarrow[G]{1} aSb$$

Example

CFG $G = (N, \Sigma, P, S)$

1. $N = \{S\}$
2. $\Sigma = \{a, b\}$
3. $P = \{S \to aSb,\ S \to \varepsilon\}$

two derivations:

$$S \xrightarrow[G]{1} aSb \xrightarrow[G]{1} ab$$

Example

CFG $G = (N, \Sigma, P, S)$

1. $N = \{S\}$
2. $\Sigma = \{\mathrm{a}, \mathrm{b}\}$
3. $P = \{S \rightarrow aSb, S \rightarrow \varepsilon\}$

two derivations:

$$S \xrightarrow[G]{1} aSb \xrightarrow[G]{1} ab \qquad S \xrightarrow[G]{1} \mathrm{a}S\mathrm{b} \xrightarrow[G]{1} aaSbb$$

Example

CFG $G = (N, \Sigma, P, S)$

1. $N = \{S\}$
2. $\Sigma = \{a, b\}$
3. $P = \{S \rightarrow aSb,\ S \rightarrow \varepsilon\}$

two derivations:

$$S \xrightarrow[G]{1} aSb \xrightarrow[G]{1} ab \qquad\qquad S \xrightarrow[G]{1} aSb \xrightarrow[G]{1} aaSbb \xrightarrow[G]{1} aabb$$

Example

CFG $G = (N, \Sigma, P, S)$

1. $N = \{S\}$
2. $\Sigma = \{\mathrm{a}, \mathrm{b}\}$
3. $P = \{S \rightarrow aSb,\ S \rightarrow \varepsilon\} = \{S \rightarrow \mathrm{a}S\mathrm{b} \mid \varepsilon\}$

two derivations:

$$S \xrightarrow[G]{1} aSb \xrightarrow[G]{1} ab \qquad S \xrightarrow[G]{1} \mathrm{a}S\mathrm{b} \xrightarrow[G]{1} aaSbb \xrightarrow[G]{1} aabb$$

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
  4. $S \in N$: start symbol
- one step derivation relation $\xrightarrow[G]{1}$ on $(N \cup \Sigma)^*$: $\beta A \gamma \xrightarrow[G]{1} \beta\alpha\gamma$ if $A \to \alpha \in P$ and $\beta, \gamma \in (N \cup \Sigma)^*$
- $\xrightarrow[G]{n} = (\xrightarrow[G]{1})^n \quad \forall n \geqslant 0$

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
  4. $S \in N$: start symbol
- one step derivation relation $\xrightarrow[G]{1}$ on $(N \cup \Sigma)^*$: $\quad \beta A \gamma \xrightarrow[G]{1} \beta\alpha\gamma \quad$ if $A \to \alpha \in P$ and $\beta, \gamma \in (N \cup \Sigma)^*$
- $\xrightarrow[G]{n} = (\xrightarrow[G]{1})^n \quad \forall\, n \geqslant 0 \qquad\qquad \xrightarrow[G]{*} = \bigcup_{n \geqslant 0} \xrightarrow[G]{n}$

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
  4. $S \in N$: start symbol
- one step derivation relation $\xrightarrow[G]{1}$ on $(N \cup \Sigma)^*$: $\beta A \gamma \xrightarrow[G]{1} \beta \alpha \gamma$ if $A \to \alpha \in P$ and $\beta, \gamma \in (N \cup \Sigma)^*$
- $\xrightarrow[G]{n} = (\xrightarrow[G]{1})^n \quad \forall n \geqslant 0$ $\qquad \xrightarrow[G]{*} = \bigcup_{n \geqslant 0} \xrightarrow[G]{n}$
- members of the set $(N \cup \Sigma)^*$ are called strings

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
  4. $S \in N$: start symbol
- one step derivation relation $\xrightarrow[G]{1}$ on $(N \cup \Sigma)^*$: $\beta A \gamma \xrightarrow[G]{1} \beta \alpha \gamma$ if $A \to \alpha \in P$ and $\beta, \gamma \in (N \cup \Sigma)^*$
- $\xrightarrow[G]{n} = (\xrightarrow[G]{1})^n \quad \forall n \geqslant 0$ $\qquad \xrightarrow[G]{*} = \bigcup_{n \geqslant 0} \xrightarrow[G]{n}$
- members of the set $(N \cup \Sigma)^*$ are called strings
- string s is called a sentential form if $S \xrightarrow[G]{*} s$ (derivable from the start symbol S)

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
  4. $S \in N$: start symbol
- one step derivation relation $\xrightarrow[G]{1}$ on $(N \cup \Sigma)^*$: $\quad \beta A \gamma \xrightarrow[G]{1} \beta\alpha\gamma \quad$ if $A \to \alpha \in P$ and $\beta, \gamma \in (N \cup \Sigma)^*$
- $\xrightarrow[G]{n} = (\xrightarrow[G]{1})^n \quad \forall\, n \geqslant 0 \qquad\qquad \xrightarrow[G]{*} = \bigcup_{n \geqslant 0} \xrightarrow[G]{n}$
- members of the set $(N \cup \Sigma)^*$ are called strings
- string s is called a sentential form if $S \xrightarrow[G]{*} s$ (derivable from the start symbol S)
- sentential form x is called a sentence if $x \in \Sigma^*$ (consisting terminal symbols only)

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
  4. $S \in N$: start symbol
- one step derivation relation $\xrightarrow[G]{1}$ on $(N \cup \Sigma)^*$: $\beta A \gamma \xrightarrow[G]{1} \beta \alpha \gamma$ if $A \to \alpha \in P$ and $\beta, \gamma \in (N \cup \Sigma)^*$
- $\xrightarrow[G]{n} = (\xrightarrow[G]{1})^n \quad \forall n \geqslant 0 \qquad \xrightarrow[G]{*} = \bigcup_{n \geqslant 0} \xrightarrow[G]{n}$
- members of the set $(N \cup \Sigma)^*$ are called strings
- string s is called a sentential form if $S \xrightarrow[G]{*} s$ (derivable from the start symbol S)
- sentential form x is called a sentence if $x \in \Sigma^*$ (consisting terminal symbols only)
- language generated by G: $L(G) = \{x \in \Sigma^* \mid S \xrightarrow[G]{*} x\}$

Example

CFG $G = (N, \Sigma, P, S)$

1. $N = \{S\}$
2. $\Sigma = \{\mathrm{a}, \mathrm{b}\}$
3. $P = \{S \to aSb,\ S \to \varepsilon\} = \{S \to \mathrm{a}S\mathrm{b} \mid \varepsilon\}$

two derivations:

$$S \xrightarrow[G]{1} aSb \xrightarrow[G]{1} ab \qquad\qquad S \xrightarrow[G]{1} \mathrm{a}S\mathrm{b} \xrightarrow[G]{1} aaSbb \xrightarrow[G]{1} aabb$$

Lemma

$L(G) = \{\mathrm{a}^n\mathrm{b}^n \mid n \geqslant 0\}$

Definitions

- context-free grammar (CFG) is quadruple $G = (N, \Sigma, P, S)$ with
  1. N : finite set of nonterminals
  2. Σ : finite set of terminals, disjoint from N
  3. P : finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$
  4. $S \in N$: start symbol
- one step derivation relation $\xrightarrow[G]{1}$ on $(N \cup \Sigma)^*$: $\quad \beta A \gamma \xrightarrow[G]{1} \beta \alpha \gamma \quad$ if $A \to \alpha \in P$ and $\beta, \gamma \in (N \cup \Sigma)^*$
- $\xrightarrow[G]{n} = (\xrightarrow[G]{1})^n \quad \forall n \geqslant 0 \qquad \xrightarrow[G]{*} = \bigcup_{n \geqslant 0} \xrightarrow[G]{n}$
- members of the set $(N \cup \Sigma)^*$ are called strings
- string s is called a sentential form if $S \xrightarrow[G]{*} s$ (derivable from the start symbol S)
- sentential form x is called a sentence if $x \in \Sigma^*$ (consisting terminal symbols only)
- language generated by G: $\quad L(G) = \{x \in \Sigma^* \mid S \xrightarrow[G]{*} x\}$
- set $B \subseteq \Sigma^*$ is context-free if $B = L(G)$ for some CFG G

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$\begin{array}{lcl} S & \rightarrow & aS \mid T \\ T & \rightarrow & bT \mid U \\ U & \rightarrow & cU \mid \varepsilon \end{array}$$

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$
\begin{aligned}
S &\rightarrow aS \mid T \\
T &\rightarrow bT \mid U \\
U &\rightarrow cU \mid \varepsilon
\end{aligned}
$$

- provide derivations for the following strings:

“b”

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$
\begin{aligned}
S &\rightarrow aS \mid T \\
T &\rightarrow bT \mid U \\
U &\rightarrow cU \mid \varepsilon
\end{aligned}
$$

- provide derivations for the following strings:

"b" $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bU \xrightarrow[G]{1} b\varepsilon = b$

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$
\begin{aligned}
S &\rightarrow aS \mid T \\
T &\rightarrow bT \mid U \\
U &\rightarrow cU \mid \varepsilon
\end{aligned}
$$

- provide derivations for the following strings:

"b" $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bU \xrightarrow[G]{1} b\varepsilon = b$

"ac"

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$\begin{aligned} S &\rightarrow aS \mid T \\ T &\rightarrow bT \mid U \\ U &\rightarrow cU \mid \varepsilon \end{aligned}$$

- provide derivations for the following strings:

"b" $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bU \xrightarrow[G]{1} b\varepsilon = b$

"ac" $\quad S \xrightarrow[G]{1} aS \xrightarrow[G]{1} aT \xrightarrow[G]{1} aU \xrightarrow[G]{1} acU \xrightarrow[G]{1} ac\varepsilon = ac$

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$
\begin{aligned}
S &\rightarrow aS \mid T \\
T &\rightarrow bT \mid U \\
U &\rightarrow cU \mid \varepsilon
\end{aligned}
$$

- provide derivations for the following strings:

"b" $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bU \xrightarrow[G]{1} b\varepsilon = b$

"ac" $\quad S \xrightarrow[G]{1} aS \xrightarrow[G]{1} aT \xrightarrow[G]{1} aU \xrightarrow[G]{1} acU \xrightarrow[G]{1} ac\varepsilon = ac$

"bbc"

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$\begin{aligned} S &\rightarrow aS \mid T \\ T &\rightarrow bT \mid U \\ U &\rightarrow cU \mid \varepsilon \end{aligned}$$

- provide derivations for the following strings:

“b” $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bU \xrightarrow[G]{1} b\varepsilon = b$

“ac” $\quad S \xrightarrow[G]{1} aS \xrightarrow[G]{1} aT \xrightarrow[G]{1} aU \xrightarrow[G]{1} acU \xrightarrow[G]{1} ac\varepsilon = ac$

“bbc” $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bbT \xrightarrow[G]{1} bbU \xrightarrow[G]{1} bbcU \xrightarrow[G]{1} bbc\varepsilon = bbc$

Example

- grammar $G = \left(\{S, T, U\}, \{a, b, c\}, P, S\right)$ with P :

$$\begin{array}{lcl} S & \rightarrow & aS \mid T \\ T & \rightarrow & bT \mid U \\ U & \rightarrow & cU \mid \varepsilon \end{array}$$

- provide derivations for the following strings:

“b” $\qquad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bU \xrightarrow[G]{1} b\varepsilon = b$

“ac” $\qquad S \xrightarrow[G]{1} aS \xrightarrow[G]{1} aT \xrightarrow[G]{1} aU \xrightarrow[G]{1} acU \xrightarrow[G]{1} ac\varepsilon = ac$

“bbc” $\qquad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bbT \xrightarrow[G]{1} bbU \xrightarrow[G]{1} bbcU \xrightarrow[G]{1} bbc\varepsilon = bbc$

- does G generate following strings?

$$S \xrightarrow[G]{+} \text{“}ccc\text{”}$$

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$\begin{aligned} S &\rightarrow aS \mid T \\ T &\rightarrow bT \mid U \\ U &\rightarrow cU \mid \varepsilon \end{aligned}$$

- provide derivations for the following strings:

"b" $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bU \xrightarrow[G]{1} b\varepsilon = b$

"ac" $\quad S \xrightarrow[G]{1} aS \xrightarrow[G]{1} aT \xrightarrow[G]{1} aU \xrightarrow[G]{1} acU \xrightarrow[G]{1} ac\varepsilon = ac$

"bbc" $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bbT \xrightarrow[G]{1} bbU \xrightarrow[G]{1} bbcU \xrightarrow[G]{1} bbc\varepsilon = bbc$

- does G generate following strings?

$S \xrightarrow[G]{+}$ "ccc" yes

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$\begin{aligned} S &\rightarrow aS \mid T \\ T &\rightarrow bT \mid U \\ U &\rightarrow cU \mid \varepsilon \end{aligned}$$

- provide derivations for the following strings:

"b" $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bU \xrightarrow[G]{1} b\varepsilon = b$

"ac" $\quad S \xrightarrow[G]{1} aS \xrightarrow[G]{1} aT \xrightarrow[G]{1} aU \xrightarrow[G]{1} acU \xrightarrow[G]{1} ac\varepsilon = ac$

"bbc" $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bbT \xrightarrow[G]{1} bbU \xrightarrow[G]{1} bbcU \xrightarrow[G]{1} bbc\varepsilon = bbc$

- does G generate following strings?

$S \xrightarrow[G]{+}$ "ccc" yes

$S \xrightarrow[G]{+}$ "bab"

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$
\begin{aligned}
S &\rightarrow aS \mid T \\
T &\rightarrow bT \mid U \\
U &\rightarrow cU \mid \varepsilon
\end{aligned}
$$

- provide derivations for the following strings:

“b” $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bU \xrightarrow[G]{1} b\varepsilon = b$

“ac” $\quad S \xrightarrow[G]{1} aS \xrightarrow[G]{1} aT \xrightarrow[G]{1} aU \xrightarrow[G]{1} acU \xrightarrow[G]{1} ac\varepsilon = ac$

“bbc” $\quad S \xrightarrow[G]{1} T \xrightarrow[G]{1} bT \xrightarrow[G]{1} bbT \xrightarrow[G]{1} bbU \xrightarrow[G]{1} bbcU \xrightarrow[G]{1} bbc\varepsilon = bbc$

- does G generate following strings?

$S \xrightarrow[G]{+}$ “ccc” yes

$S \xrightarrow[G]{+}$ “bab” no

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$\begin{array}{lcl} S & \rightarrow & aS \mid T \\ T & \rightarrow & bT \mid U \\ U & \rightarrow & cU \mid \varepsilon \end{array}$$

  constructs $\mathcal{L} = \left\{a^i b^j c^k \mid i, j, k \geqslant 0\right\}$

Example

- grammar $G = \big(\{S, T, U\}, \{a, b, c\}, P, S\big)$ with P :

$$\begin{aligned} S &\rightarrow aS \mid T \\ T &\rightarrow bT \mid U \\ U &\rightarrow cU \mid \varepsilon \end{aligned}$$

  constructs $\mathcal{L} = \big\{a^i b^j c^k \mid i, j, k \geqslant 0\big\}$

- grammar $G' = \big(\{S, A, B, C\}, \{a, b, c\}, P, S\big)$ with P :

$$\begin{aligned} S &\rightarrow ABC \\ A &\rightarrow aA \mid \varepsilon \\ B &\rightarrow bB \mid \varepsilon \\ C &\rightarrow cC \mid \varepsilon \end{aligned}$$

  constructs the same language $\mathcal{L}$

Some Useful Tricks to Design Grammars

1. arbitrary number of symbols:

$S \to aS \mid \varepsilon$ zero or more a's

$S \to bS \mid b$ one or more b's

Some Useful Tricks to Design Grammars

1. arbitrary number of symbols:

   $S \to aS \mid \varepsilon$ zero or more a's

   $S \to bS \mid b$ one or more b's

2. disjoint parts of a language:

   $A \to aA \mid \varepsilon$ zero or more a's

   $B \to bB \mid \varepsilon$ zero or more b's

   $S \to AB$ combining above generations

   generates $\mathcal{L} = \{a^i b^j \mid i, j \geqslant 0\}$

Some Useful Tricks to Design Grammars

1. arbitrary number of symbols:

   $S \rightarrow aS \mid \varepsilon$ zero or more a's

   $S \rightarrow bS \mid b$ one or more b's

2. disjoint parts of a language:

   $A \rightarrow aA \mid \varepsilon$ zero or more a's

   $B \rightarrow bB \mid \varepsilon$ zero or more b's

   $S \rightarrow AB$ combining above generations

   generates $\mathcal{L} = \{a^i b^j \mid i, j \geqslant 0\}$

3. matching (balanced) symbols:

   $S \rightarrow aSb \mid \varepsilon$

   generates $\mathcal{L} = \{a^n b^n \mid n \geqslant 0\}$

Some Useful Tricks to Design Grammars

1. arbitrary number of symbols:

   $S \to aS \mid \varepsilon$ zero or more a's

   $S \to bS \mid b$ one or more b's

2. disjoint parts of a language:

   $A \to aA \mid \varepsilon$ zero or more a's

   $B \to bB \mid \varepsilon$ zero or more b's

   $S \to AB$ combining above generations

   generates $\mathcal{L} = \{a^i b^j \mid i, j \geqslant 0\}$

3. matching (balanced) symbols:

   $S \to aSb \mid \varepsilon$

   generates $\mathcal{L} = \{a^n b^n \mid n \geqslant 0\}$

4. matching *linearly* related symbols:

   $S \to aSbb \mid \varepsilon$

   generates $\mathcal{L} = \{a^n b^{2n} \mid n \geqslant 0\}$

Example

- $\{0^n1^m \mid m \leqslant n\}$ $G = (\{S\}, \{0, 1\}, P, S)$

Example

- $\{0^n 1^m \mid m \leqslant n\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0S1 \mid 0S \mid \varepsilon\}$

Example

- $\{0^n1^m \mid m \leqslant n\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0S1 \mid 0S \mid \varepsilon\}$

- $\{1^n0 \mid n \geqslant 0\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

Example

- $\{0^n 1^m \mid m \leqslant n\}$ $\quad G = (\{S\}, \{0, 1\}, P, S)$

  where $P = \{S \rightarrow 0S1 \mid 0S \mid \varepsilon\}$

- $\{1^n 0 \mid n \geqslant 0\}$ $\quad G = (\{S\}, \{0, 1\}, P, S)$

  where $P = \{S \rightarrow 0 \mid 1S\}$

Example

- $\{0^n 1^m \mid m \leqslant n\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0S1 \mid 0S \mid \varepsilon\}$

- $\{1^n 0 \mid n \geqslant 0\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0 \mid 1S\}$

- $\{a^n (b^m \mid c^k) \mid n \geqslant 0, m, k > n\}$

Example

- $\{0^n 1^m \mid m \leqslant n\}$ $\quad G = \big(\{S\}, \{0,1\}, P, S\big)$

  where $\quad P = \quad \{S \quad \to \quad 0S1 \mid 0S \mid \varepsilon\}$

- $\{1^n 0 \mid n \geqslant 0\}$ $\quad G = \big(\{S\}, \{0,1\}, P, S\big)$

  where $\quad P = \quad \{S \quad \to \quad 0 \mid 1S\}$

- $\{a^n (b^m \mid c^k) \mid n \geqslant 0, m, k > n\} = \{a^n b^m \mid m > n \geqslant 0\} \cup \{a^n c^k \mid k > n \geqslant 0\}$ $\quad G = \big(\{S, T, U, V, Y\}, \{a, b, c\}, P, S\big)$

Example

- $\{0^n1^m \mid m \leqslant n\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0S1 \mid 0S \mid \varepsilon\}$

- $\{1^n0 \mid n \geqslant 0\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0 \mid 1S\}$

- $\{a^n(b^m \mid c^k) \mid n \geqslant 0, m, k > n\} = \{a^nb^m \mid m > n \geqslant 0\} \cup \{a^nc^k \mid k > n \geqslant 0\}$ $\quad G = (\{S, T, U, V, Y\}, \{a, b, c\}, P, S)$

  where $P = \{$
  $$\begin{aligned} T &\rightarrow aTb \mid U \\ U &\rightarrow Ub \mid b \end{aligned}$$

Example

- $\{0^n 1^m \mid m \leqslant n\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0S1 \mid 0S \mid \varepsilon\}$

- $\{1^n 0 \mid n \geqslant 0\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0 \mid 1S\}$

- $\{a^n(b^m \mid c^k) \mid n \geqslant 0, m, k > n\} = \{a^n b^m \mid m > n \geqslant 0\} \cup \{a^n c^k \mid k > n \geqslant 0\}$ $\quad G = (\{S, T, U, V, Y\}, \{a, b, c\}, P, S)$

$$\text{where} \quad P = \{ \begin{array}{lcl} T & \rightarrow & aTb \mid U \\ U & \rightarrow & Ub \mid b \\ V & \rightarrow & aVc \mid Y \\ Y & \rightarrow & Yc \mid c \end{array}$$

Example

- $\{0^n 1^m \mid m \leqslant n\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0S1 \mid 0S \mid \varepsilon\}$

- $\{1^n 0 \mid n \geqslant 0\}$ $\quad G = (\{S\}, \{0,1\}, P, S)$

  where $P = \{S \rightarrow 0 \mid 1S\}$

- $\{a^n (b^m \mid c^k) \mid n \geqslant 0, m, k > n\} = \{a^n b^m \mid m > n \geqslant 0\} \cup \{a^n c^k \mid k > n \geqslant 0\}$ $\quad G = (\{S, T, U, V, Y\}, \{a, b, c\}, P, S)$

$$\text{where} \quad P = \left\{ \begin{array}{lcl} T & \rightarrow & aTb \mid U \\ U & \rightarrow & Ub \mid b \\ V & \rightarrow & aVc \mid Y \\ Y & \rightarrow & Yc \mid c \\ S & \rightarrow & T \mid V \end{array} \right\}$$

Table of Contents

Example

CFG G: $S \rightarrow [S] \mid SS \mid \varepsilon$

Example

CFG G: $S \to [S] \mid SS \mid \varepsilon$ three derivations of $[[]]$:

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$

Example

CFG G: $S \to [S] \mid SS \mid \varepsilon$ three derivations of [[]]:

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$

Example

CFG G: $S \to [S] \mid SS \mid \varepsilon$ three derivations of $[[\,]]$:

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[\,]]$
2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[\,]]$
3. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S[S]] \xrightarrow[G]{1} [S[\,]] \xrightarrow[G]{1} [[\,]]$

Example

CFG G: $S \to [S] \mid SS \mid \varepsilon$ three derivations of $[[\,]]$:

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[\,]]$
2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[\,]]$
3. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S[S]] \xrightarrow[G]{1} [S[\,]] \xrightarrow[G]{1} [[\,]]$

Definition

- in leftmost derivation always leftmost nonterminal is replaced ① ②

Example

CFG G: $S \rightarrow [S] \mid SS \mid \varepsilon$ three derivations of $[[]]$:

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
3. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S[S]] \xrightarrow[G]{1} [S[]] \xrightarrow[G]{1} [[]]$

Definition

- in leftmost derivation always leftmost nonterminal is replaced 1 2
- in rightmost derivation always rightmost nonterminal is replaced 1 3

Example

CFG G: $S \to [S] \mid SS \mid \varepsilon$ three derivations of $[[]]$:

① $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$

② $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$

③ $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S[S]] \xrightarrow[G]{1} [S[]] \xrightarrow[G]{1} [[]]$

Definition

- in leftmost derivation always leftmost nonterminal is replaced ① ②
- in rightmost derivation always rightmost nonterminal is replaced ① ③
- parse tree is representation of derivation in which replacement order is ignored

Example

CFG G: $S \to [S] \mid SS \mid \varepsilon$ three derivations of $[[]]$:

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
3. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S[S]] \xrightarrow[G]{1} [S[]] \xrightarrow[G]{1} [[]]$

Definition

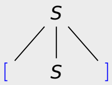
- in leftmost derivation always leftmost nonterminal is replaced 1 2
- in rightmost derivation always rightmost nonterminal is replaced 1 3
- parse tree is representation of derivation in which replacement order is ignored
- CFG is ambiguous if some string has different parse trees

Example

CFG G: $S \to [\,S\,] \mid S\,S \mid \varepsilon$

1. $S \xrightarrow[G]{1} [\,S\,]$

parse tree

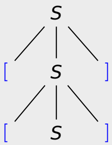


Example

CFG G: $S \to [\,S\,] \mid S\,S \mid \varepsilon$

1. $S \xrightarrow[G]{1} [\,S\,] \xrightarrow[G]{1} [\,[\,S\,]\,]$

parse tree

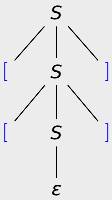


Example

CFG G: $S \to [\, S\,] \mid S\, S \mid \varepsilon$

➊ $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[\,]]$

parse tree



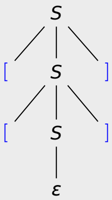
Example

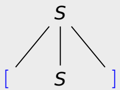
CFG G: $S \rightarrow [\,S\,] \mid S\,S \mid \varepsilon$

1. $S \xrightarrow[G]{1} [\,S\,] \xrightarrow[G]{1} [[\,S\,]] \xrightarrow[G]{1} [[\,]]$

2. $S \xrightarrow[G]{1} [\,S\,]$

parse trees





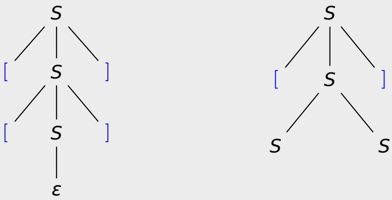
Example

CFG G: $S \to [\,S\,] \mid S\,S \mid \varepsilon$

1. $S \xrightarrow[G]{1} [\,S\,] \xrightarrow[G]{1} [[\,S\,]] \xrightarrow[G]{1} [[\,]]$

2. $S \xrightarrow[G]{1} [\,S\,] \xrightarrow[G]{1} [\,S\,S\,]$

parse trees

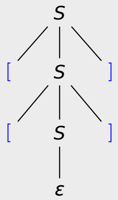


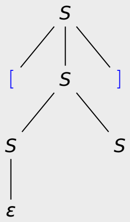
Example

CFG G: $S \to [\,S\,] \mid S\,S \mid \varepsilon$

1. $S \xrightarrow[G]{1} [\,S\,] \xrightarrow[G]{1} [[\,S\,]] \xrightarrow[G]{1} [[\,]]$
2. $S \xrightarrow[G]{1} [\,S\,] \xrightarrow[G]{1} [\,S\,S\,] \xrightarrow[G]{1} [\,S\,]$

parse trees





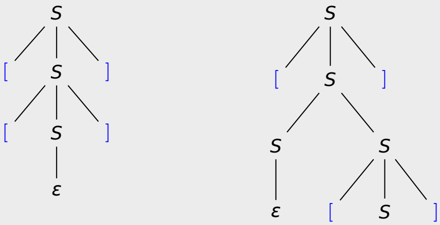
Example

CFG G: $S \rightarrow [\,S\,] \mid S\,S \mid \varepsilon$

➊ $S \xrightarrow[G]{1} [\,S\,] \xrightarrow[G]{1} [[\,S\,]] \xrightarrow[G]{1} [[\,]]$

➋ $S \xrightarrow[G]{1} [\,S\,] \xrightarrow[G]{1} [\,S\,S\,] \xrightarrow[G]{1} [\,S\,] \xrightarrow[G]{1} [[\,S\,]]$

parse trees

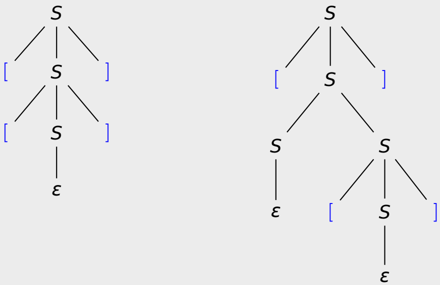


Example

CFG G: $S \rightarrow [S] \mid S\,S \mid \varepsilon$

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [S\,S] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$

parse trees

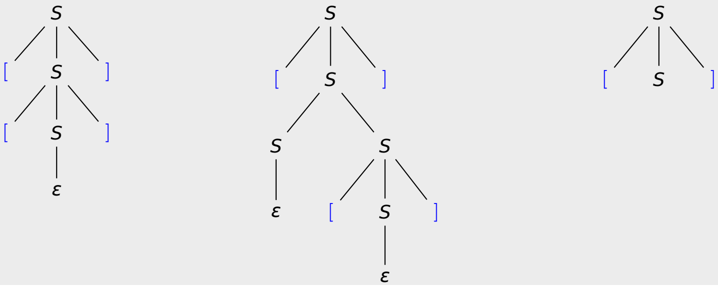


Example

CFG G: $S \to [S] \mid S\,S \mid \varepsilon$

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [S\,S] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
3. $S \xrightarrow[G]{1} [S]$

parse trees



Example

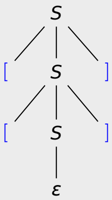
CFG G: $S \to [\,S\,] \mid S\,S \mid \varepsilon$

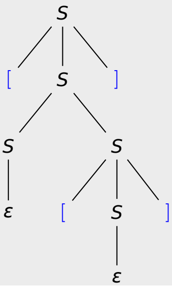
① $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$

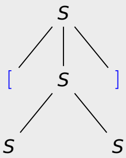
② $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [S\,S] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$

③ $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [S\,S]$

parse trees





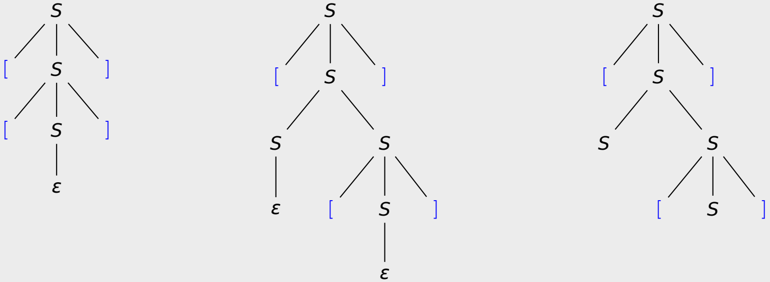


Example

CFG G: $S \to [\,S\,] \mid S\,S \mid \varepsilon$

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[\,]]$
2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [S\,S] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[\,]]$
3. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [S\,S] \xrightarrow[G]{1} [S\,[S]]$

parse trees



Example

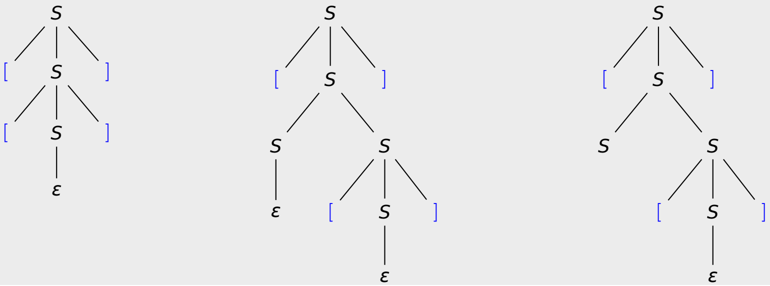
CFG G: $S \to [\,S\,] \mid S\,S \mid \varepsilon$

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[\,]]$

2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [S\,S] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[\,]]$

3. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [S\,S] \xrightarrow[G]{1} [S[S]] \xrightarrow[G]{1} [S[\,]]$

parse trees

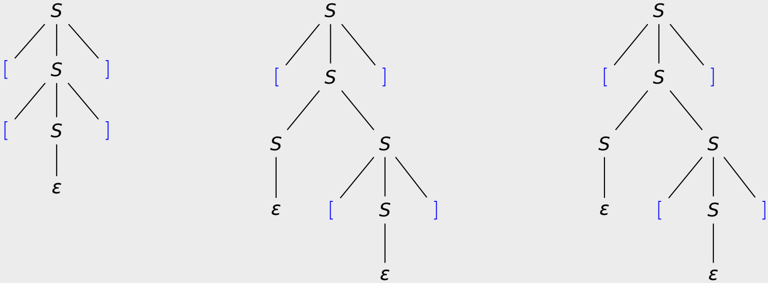


Example

CFG G: $S \to [S] \mid SS \mid \varepsilon$

1. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
2. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [[S]] \xrightarrow[G]{1} [[]]$
3. $S \xrightarrow[G]{1} [S] \xrightarrow[G]{1} [SS] \xrightarrow[G]{1} [S[S]] \xrightarrow[G]{1} [S[]] \xrightarrow[G]{1} [[]]$

parse trees



Example

- CFG G : $S \rightarrow [S] \mid S\,S \mid \varepsilon$

- G is ambiguous

Example

- CFG G : $S \rightarrow [S] \mid S\,S \mid \varepsilon$
- CFG G': $S \rightarrow \varepsilon \mid T$

  $T \rightarrow T\,U \mid U$

  $U \rightarrow [\,] \mid [T]$
- G is ambiguous

Example

- CFG G : $S \rightarrow [S] \mid S\,S \mid \varepsilon$
- CFG G': $S \rightarrow \varepsilon \mid T$
  $T \rightarrow T\,U \mid U$
  $U \rightarrow [\,] \mid [T]$
- G is ambiguous G' is unambiguous $L(G) = L(G')$

Example

- CFG G : $\quad S \to [S] \mid S\,S \mid \varepsilon$
- CFG G': $\quad S \to \varepsilon \mid T$
  $\quad T \to T\,U \mid U$
  $\quad U \to [\,] \mid [T]$
- G is ambiguous $\qquad G'$ is unambiguous $\qquad L(G) = L(G')$

$$\begin{array}{c} S \\ | \\ T \\ | \\ U \\ | \\ [T] \\ | \\ [U] \\ | \\ [[\,]] \end{array}$$

Example

- CFG G : $S \rightarrow S - S \mid int$
- G is ambiguous

  with G $7-5-2$ could be parsed as $(7-5)-2$ and $7-(5-2)$

Example

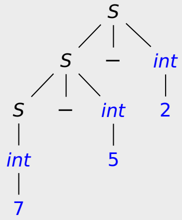
- CFG G : $S \rightarrow S - S \mid int$
- CFG G': $S \rightarrow S - int \mid int$
- G is ambiguous

with G $7 - 5 - 2$ could be parsed as $(7 - 5) - 2$ and $7 - (5 - 2)$

Example

- CFG G : $S \rightarrow S - S \mid \textcolor{blue}{int}$
- CFG G': $S \rightarrow S - \textcolor{blue}{int} \mid \textcolor{blue}{int}$
- G is ambiguous $\quad G'$ is unambiguous $\quad L(G) = L(G')$

with G $\quad 7 - 5 - 2$ could be parsed as $\quad (7-5)-2$ and $7-(5-2)$
with G' $\quad 7 - 5 - 2$ could only be parsed as $\quad (7-5)-2$

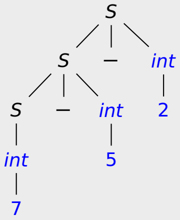


Example

- CFG G : $S \rightarrow S - S \mid int$
- CFG G': $S \rightarrow S - int \mid int$
- G is ambiguous G' is unambiguous $L(G) = L(G')$

  with G $7-5-2$ could be parsed as $(7-5)-2$ and $7-(5-2)$
  with G' $7-5-2$ could only be parsed as $(7-5)-2$

- (if applicable) one way to remove ambiguity is to benefit from associativity of binary operators

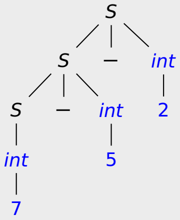


Example

- CFG G : $S \rightarrow S - S \mid int$
- CFG G': $S \rightarrow S - int \mid int$
- G is ambiguous G' is unambiguous $L(G) = L(G')$

  with G $7-5-2$ could be parsed as $(7-5)-2$ and $7-(5-2)$
  with G' $7-5-2$ could only be parsed as $(7-5)-2$

- (if applicable) one way to remove ambiguity is to benefit from associativity of binary operators
- G' enforces the "-" operator to be left associative



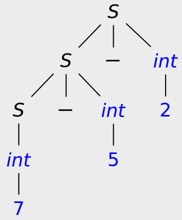
Example

- CFG G : $S \to S - S \mid int$
- CFG G': $S \to S - int \mid int$
- G is ambiguous G' is unambiguous $L(G) = L(G')$

  with G $7-5-2$ could be parsed as $(7-5)-2$ and $7-(5-2)$
  with G' $7-5-2$ could only be parsed as $(7-5)-2$

- (if applicable) one way to remove ambiguity is to benefit from associativity of binary operators
- G' enforces the "-" operator to be left associative
- CFG G'': $S \to int - S \mid int$

  turns the "-" operator into a right associative one



Example

- CFG G : $S \rightarrow S \times S \mid S + S \mid \mathit{int}$

- G is ambiguous

  with G $7 + 5 \times 2$ could be parsed as $7 + (5 \times 2)$ and $(7 + 5) \times 2$

Example

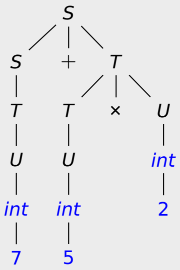
- CFG G : $S \to S \times S \mid S + S \mid int$
- CFG G': $S \to S + T \mid T$
  $T \to T \times U \mid U$
  $U \to int \mid (S)$
- G is ambiguous

  with G $7 + 5 \times 2$ could be parsed as $7 + (5 \times 2)$ and $(7 + 5) \times 2$

Example

- CFG G : $S \rightarrow S \times S \mid S + S \mid int$
- CFG G': $S \rightarrow S + T \mid T$
  $T \rightarrow T \times U \mid U$
  $U \rightarrow int \mid (S)$
- G is ambiguous G' is unambiguous $L(G) = L(G')$

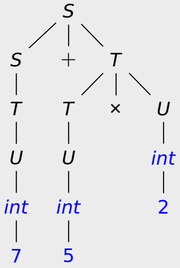
  with G $7 + 5 \times 2$ could be parsed as $7 + (5 \times 2)$ and $(7 + 5) \times 2$
  with G' $7 + 5 \times 2$ could only be parsed as $7 + (5 \times 2)$



Example

- CFG G : $S \to S \times S \mid S + S \mid int$
- CFG G': $S \to S + T \mid T$
  $T \to T \times U \mid U$
  $U \to int \mid (S)$
- G is ambiguous G' is unambiguous $L(G) = L(G')$

  with G $7 + 5 \times 2$ could be parsed as $7 + (5 \times 2)$ and $(7 + 5) \times 2$
  with G' $7 + 5 \times 2$ could only be parsed as $7 + (5 \times 2)$
- (if applicable) one way to remove ambiguity is to benefit from precedence of operators

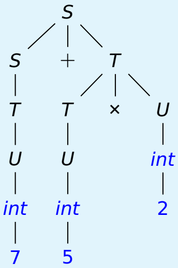


Parse Trees vs Abstract Syntax Trees

- parse trees (PT) are concrete syntax trees containing all derivation details

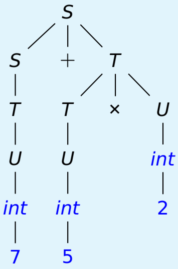
Parse Trees vs Abstract Syntax Trees

- parse trees (PT) are concrete syntax trees containing all derivation details



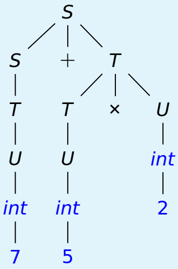
Parse Trees vs Abstract Syntax Trees

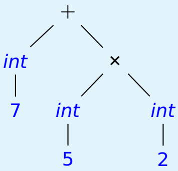
- parse trees (PT) are concrete syntax trees containing all derivation details
- abstract syntax trees (AST) are reduced PTs eliminating irrelevant information for the evaluation step



Parse Trees vs Abstract Syntax Trees

- parse trees (PT) are concrete syntax trees containing all derivation details
- abstract syntax trees (AST) are reduced PTs eliminating irrelevant information for the evaluation step





Remark

there exist context-free sets without unambiguous grammars

Remark

there exist context-free sets without unambiguous grammars (inherently ambiguous)

Remark

there exist context-free sets without unambiguous grammars (inherently ambiguous)

Example

$A = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ is context-free and inherently ambiguous

Remark

there exist context-free sets without unambiguous grammars (inherently ambiguous)

Example

$A = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ is context-free and inherently ambiguous

$A = \{a^i b^i c^k\} \cup \{a^i b^j c^j\}$

Remark

there exist context-free sets without unambiguous grammars (inherently ambiguous)

Example

$A = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ is context-free and inherently ambiguous

$A = \{a^i b^i c^k\} \cup \{a^i b^j c^j\}$

let $A = L(G)$ such that G:

Remark

there exist context-free sets without unambiguous grammars (inherently ambiguous)

Example

$A = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ is context-free and inherently ambiguous

$A = \{a^i b^i c^k\} \cup \{a^i b^j c^j\}$

let $A = L(G)$ such that G:

$$S \to T \mid W$$

$$\begin{array}{ll} T \to UV & W \to XY \\ U \to aUb \mid \varepsilon & X \to aX \mid \varepsilon \\ V \to cV \mid \varepsilon & Y \to bYc \mid \varepsilon \end{array}$$

Remark

there exist context-free sets without unambiguous grammars (inherently ambiguous)

Example

$A = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ is context-free and inherently ambiguous

$A = \{a^i b^i c^k\} \cup \{a^i b^j c^j\}$

let $A = L(G)$ such that G:

$$S \to T \mid W$$

$$\begin{array}{ll} T \to UV & W \to XY \\ U \to aUb \mid \varepsilon & X \to aX \mid \varepsilon \\ V \to cV \mid \varepsilon & Y \to bYc \mid \varepsilon \end{array}$$

the union we used has a non-empty intersection, where letters a, b and c all are in equal number

Remark

there exist context-free sets without unambiguous grammars (inherently ambiguous)

Example

$A = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ is context-free and inherently ambiguous

$A = \{a^i b^i c^k\} \cup \{a^i b^j c^j\}$

let $A = L(G)$ such that G:

$$S \rightarrow T \mid W$$

$$\begin{array}{ll} T \rightarrow UV & W \rightarrow XY \\ U \rightarrow aUb \mid \varepsilon & X \rightarrow aX \mid \varepsilon \\ V \rightarrow cV \mid \varepsilon & Y \rightarrow bYc \mid \varepsilon \end{array}$$

the union we used has a non-empty intersection, where letters a, b and c all are in equal number

Lemma

there is no CFG G' such that $L(G')$ is unambiguous with $L(G') = A$

Remark

1. given an ambiguous CFG G, the language $L(G)$ may or may not be ambiguous

Remark

1. given an ambiguous CFG G, the language $L(G)$ may or may not be ambiguous
   - one can find an unambiguous CFG G' such that $L(G') = L(G)$

Remark

1. given an ambiguous CFG G, the language $L(G)$ may or may not be ambiguous
   - one can find an unambiguous CFG G' such that $L(G') = L(G)$
2. there is no algorithm to convert ambiguous CFG to unambiguous CFG

Remark

1. given an ambiguous CFG G, the language $L(G)$ may or may not be ambiguous
   - one can find an unambiguous CFG G' such that $L(G') = L(G)$
2. there is no algorithm to convert ambiguous CFG to unambiguous CFG
3. unambiguous context free languages can be parsed by deterministic push down automata

Table of Contents

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states
  2. Σ: input alphabet

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states
  2. Σ: input alphabet
  3. Γ: stack alphabet

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states
  2. Σ: input alphabet
  3. Γ: stack alphabet
  4. δ: finite subset of $(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with

  1. Q: finite set of states
  2. Σ: input alphabet
  3. Γ: stack alphabet
  4. δ: finite subset of $(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
  5. $s \in Q$: start state

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states
  2. Σ: input alphabet
  3. Γ: stack alphabet
  4. δ: finite subset of $(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
  5. $s \in Q$: start state
  6. $\perp \in \Gamma$: initial stack symbol

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states
  2. Σ: input alphabet
  3. Γ: stack alphabet
  4. δ: finite subset of $(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
  5. $s \in Q$: start state
  6. $\perp \in \Gamma$: initial stack symbol
  7. $F \subseteq Q$: final states

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), \; ((1,], [), (1, \varepsilon)), \; ((1, [, [), \; (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states
  2. Σ: input alphabet
  3. Γ: stack alphabet
  4. δ: finite subset of $(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
  5. $s \in Q$: start state
  6. $\perp \in \Gamma$: initial stack symbol
  7. $F \subseteq Q$: final states
- configuration: element of $Q \times \Sigma^* \times \Gamma^*$

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states
  2. Σ: input alphabet
  3. Γ: stack alphabet
  4. δ: finite subset of $(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
  5. $s \in Q$: start state
  6. $\perp \in \Gamma$: initial stack symbol
  7. $F \subseteq Q$: final states
- configuration: element of $Q \times \Sigma^* \times \Gamma^*$ (current state, remaining input, stack content)

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states
  2. Σ: input alphabet
  3. Γ: stack alphabet
  4. δ: finite subset of $(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
  5. $s \in Q$: start state
  6. $\perp \in \Gamma$: initial stack symbol
  7. $F \subseteq Q$: final states
- configuration: element of $Q \times \Sigma^* \times \Gamma^*$ (current state, remaining input, stack content)
- start configuration on input x: $(s, x, \perp)$

Definitions

- NPDA is septuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ with
  1. Q: finite set of states
  2. Σ: input alphabet
  3. Γ: stack alphabet
  4. δ: finite subset of $(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
  5. $s \in Q$: start state
  6. $\perp \in \Gamma$: initial stack symbol
  7. $F \subseteq Q$: final states
- configuration: element of $Q \times \Sigma^* \times \Gamma^*$ (current state, remaining input, stack content)
- start configuration on input x: $(s, x, \perp)$
- next configuration relation is binary relation $\xrightarrow[M]{1}$ defined as: $(p, ay, A\beta) \xrightarrow[M]{1} (q, y, \gamma\beta)$
  for all $((p, a, A), (q, \gamma)) \in \delta$ with $a \in \Sigma \cup \{\varepsilon\}$ and $y \in \Sigma^*$, $\beta \in \Gamma^*$

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \bot, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\bot, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \bot), (1, [\bot)),\ ((1,], [), (1, \varepsilon)),\ ((1, [, [), (1, [[)), ((1, \varepsilon, \bot), (2, \varepsilon))\}$

input: [[ ] [[ ]]]

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \bot, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\bot, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \bot), (1, [\bot)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \bot), (2, \varepsilon))\}$

input: [[ ] [[ ]]]
state: 1
stack: $\bot$

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input: [[ ] [[ ]]]
state: 1
stack: $\perp$

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:		[	[	]	[	[	]	]	]
state:	1	1							
stack:	$\perp$	$[\perp$							

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]
state:	1	1						
stack:	$\perp$	$\perp$						
		[						

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)),\ ((1,], [), (1, \varepsilon)),\ ((1, [, [), (1, [[)),\ ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]
state:	1	1	1					
stack:	$\perp$	$\perp$ [	$\perp$ [[					

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]
state:	1	1	1					
stack:	$\perp$	$\perp$	$\perp$					
		[	[					
			[					

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]
state:	1	1	1	1				
stack:	$\perp$	$\perp$	$\perp$	$\perp$				
		[	[	[				
			[					

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]
state:	1	1	1	1				
stack:	$\perp$	$\perp$	$\perp$	$\perp$				
		[	[	[				
			[					

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)),\ ((1,], [), (1, \varepsilon)),\ ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:		[		[		]		[		[		]		]		]	
state:	1		1		1		1		1								
stack:	$\perp$		$\perp$		$\perp$		$\perp$		$\perp$								
			[		[		[		[								
					[				[								

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]
state:	1	1	1	1	1			
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$			
		[	[	[	[			
			[		[			

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]
state:	1	1	1	1	1	1		
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$		
		[	[	[	[	[		
			[		[	[		
						[		

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input: $[\ [\ ]\ [\ [\ ]\ ]\ ]$

state: 1 1 1 1 1 1

stack:

$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
	[	[	[	[	[
		[		[	[
					[

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]
state:	1	1	1	1	1	1	1	
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	
		[	[	[	[	[	[	
			[		[	[	[	
						[		

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:		[		[		]		[		[		]		]	]
state:	1		1		1		1		1		1		1		
stack:	$\perp$		$\perp$		$\perp$		$\perp$		$\perp$		$\perp$		$\perp$		
			[		[		[		[		[		[		
					[				[		[		[		
											[				

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]
state:	1	1	1	1	1	1	1	1
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
		[	[	[	[	[	[	[
			[		[	[	[	
						[		

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:		[	[	]	[	[	]	]
								]
state:	1	1	1	1	1	1	1	1
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
		[	[	[	[	[	[	[
			[		[	[	[	
						[		

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)),\ ((1,], [), (1, \varepsilon)),\ ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]	
state:	1	1	1	1	1	1	1	1	1
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
		[	[	[	[	[	[	[	
			[		[	[	[		
						[			

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]	ε
state:	1	1	1	1	1	1	1	1	1
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
		[	[	[	[	[	[	[	
			[		[	[	[		
						[			

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)),\ ((1,], [), (1, \varepsilon)),\ ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:		[	[	]	[	[	]	]	]	ε
state:	1	1	1	1	1	1	1	1	1	2
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	
		[	[	[	[	[	[	[		
			[		[	[	[			
						[				

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]	ε		[[ ]]]
state:	1	1	1	1	1	1	1	1	1	2	
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$		
		[	[	[	[	[	[	[			
			[		[	[	[				
						[					

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)),\ ((1,], [), (1, \varepsilon)),\ ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]	ε			[	[	]	]	]
state:	1	1	1	1	1	1	1	1	1	2		1	1	1	1	1
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$			$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
		[	[	[	[	[	[	[					[	[	[	
			[		[	[	[							[		
						[										

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)),\ ((1,], [), (1, \varepsilon)),\ ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]	ε	
state:	1	1	1	1	1	1	1	1	1	2
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	
		[	[	[	[	[	[	[		
			[		[	[	[			
						[				

input:	[	[	]	]	]
state:	1	1	1	1	1
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
		[	[	[	
			[		

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)), ((1,], [), (1, \varepsilon)), ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]	ε	
state:	1	1	1	1	1	1	1	1	1	2
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	
		[	[	[	[	[	[	[		
			[		[	[	[			
						[				

input:	[	[	]	]	]
state:	1	1	1	1	2
stack:	$\perp$	$\perp$	$\perp$	$\perp$	ε
		[	[	[	
			[		

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by NPDA $M = (Q, \Sigma, \Gamma, \delta, 1, \perp, F)$ with

1. $Q = \{1, 2\}$
2. $\Sigma = \{[,]\}$
3. $\Gamma = \{\perp, [\}$
4. $F = \{2\}$
5. $s = 1$
6. $\delta = \{((1, [, \perp), (1, [\perp)),\ ((1,], [), (1, \varepsilon)),\ ((1, [, [), (1, [[)), ((1, \varepsilon, \perp), (2, \varepsilon))\}$

input:	[	[	]	[	[	]	]	]	ε	
state:	1	1	1	1	1	1	1	1	1	2
stack:	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	
		[	[	[	[	[	[	[		
			[		[	[	[			
						[				

input:	[	[	]	]	]
state:	1	1	1	1	2
stack:	$\perp$	$\perp$	$\perp$	$\perp$	ε
		[	[	[	
			[		

Theorem

CFGs and NPDAs are equivalent:

1. $A = L(G)$ for some CFG G $\iff$
2. $A = L(M)$ for some NPDA M

Theorem

CFGs and NPDAs are equivalent:

1. $A = L(G)$ for some CFG G $\iff$
2. $A = L(M)$ for some NPDA M

Theorem

CFGs and NPDAs are equivalent:

1. $A = L(G)$ for some CFG G $\iff$
2. $A = L(M)$ for some NPDA M

Determinism in PDAs (Informally)

ability to perform at most one transition (move)

Determinism in PDAs (Informally)

ability to perform at most one transition (move)

- from the same state

Determinism in PDAs (Informally)

ability to perform at most one transition (move)

- from the same state
- popping the same symbol off the stack

Determinism in PDAs (Informally)

ability to perform at most one transition (move)

- from the same state
- popping the same symbol off the stack
- $\begin{cases}\text{consuming the same input character} \\ \text{consuming an input character and the empty string } \varepsilon\end{cases}$

Determinism in PDAs (Informally)

ability to perform at most one transition (move)

- from the same state
- popping the same symbol off the stack
- $\begin{cases}\text{consuming the same input character}\\ \text{consuming an input character and the empty string } \varepsilon\end{cases}$

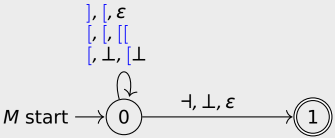
Definition

A deterministic pushdown automaton (DPDA) is an octuple $M = (Q, \Sigma, \Gamma, \delta, \bot, \dashv, s, F)$

1. $\dashv$ is a special symbol not in Σ, called the right end-marker
2. for any $p \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $A \in \Gamma$, the set $\delta \subseteq (Q \times (\Sigma \cup \{\dashv\} \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$ contains
   - at most one element of the form $((p, a, A), (q, \beta))$
   - exactly one transition of the form $((p, a, A), (q, \beta))$ or $((p, \varepsilon, A), (q, \beta))$

Example

$A = \{x \in \{[,]\}^* \mid x \text{ is balanced}\}$ is accepted by DPDA $M = \big(\{0, 1\}, \{[,]\}, \{[, \bot\}, \delta, \bot, \dashv, 0, \{1\}\big)$ with



the final state acceptance criterion.

Table of Contents

Parsers (Syntactical Analyzers)

translator tool : list of tokens (lexer generated) $\rightarrow$ parse trees (or error)

Parsers (Syntactical Analyzers)

translator tool : list of tokens (lexer generated) $\longrightarrow$ parse trees (or error)

- simulate deterministic push down automaton to generate parse trees

Parsers (Syntactical Analyzers)

translator tool : list of tokens (lexer generated) $\longrightarrow$ parse trees (or error)

- simulate deterministic push down automaton to generate parse trees
- parser generators

Parsers (Syntactical Analyzers)

translator tool : list of tokens (lexer generated) $\longrightarrow$ parse trees (or error)

- simulate deterministic push down automaton to generate parse trees
- parser generators
  - Yacc and Bison for C

Parsers (Syntactical Analyzers)

translator tool : list of tokens (lexer generated) $\rightarrow$ parse trees (or error)

- simulate deterministic push down automaton to generate parse trees
- parser generators
  - Yacc and Bison for C
  - *menhir* for OCaml

menhir

- generates parsers in compatible with OCaml programs

  input : list of *parsing rules* in the form of *context free grammars* with corresponding *actions*

  output : a parser as a DPDA accepting the language generated by the input grammar

- recognizes patterns specified by the rules, associates the corresponding actions

menhir (Specifications)

```
%{
        header
%}

%token tkn1
%left  tkn2
%start <ocaml-type> S
%%

S:
  | ...          { action }
  | production1  { action };
T:
  | production2  { action };
...
```

menhir (Specifications)

```
%{
        header
%}

%token tkn1
%left  tkn2
%start <ocaml-type> S
%%

S:
  | ...          { action }
  | production1  { action };
T:
  | production2  { action };
...
```

where

header   useful OCaml code – usually just opens

menhir (Specifications)

```
%{
        header
%}

%token tkn1
%left  tkn2
%start <ocaml-type> S
%%

S:
  | ...          { action }
  | production1  { action };
T:
  | production2  { action };
...
```

where

header	useful OCaml code – usually just opens
token	terminal symbols obtained from the lexer, e.g., `tkn1`

menhir (Specifications)

```
%{
        header
%}

%token tkn1
%left  tkn2
%start <ocaml-type> S
%%

S:
  | ...          { action }
  | production1  { action };
T:
  | production2  { action };
...
```

where

header	useful OCaml code – usually just opens
token	terminal symbols obtained from the lexer, e.g., `tkn1`
left	defines the `tkn1` left associative

menhir (Specifications)

```
%{
       header
%}

%token tkn1
%left  tkn2
%start <ocaml-type> S
%%

S:
  | ...          { action }
  | production1  { action };
T:
  | production2  { action };
...
```

where

header	useful OCaml code – usually just opens
token	terminal symbols obtained from the lexer, e.g., `tkn1`
left	defines the `tkn1` left associative
S, T, etc	CFG non-terminals such that S is the start symbol

menhir (Specifications)

```
%{
        header
%}

%token tkn1
%left  tkn2
%start <ocaml-type> S
%%

S:
  | ...          { action }
  | production1  { action };
T:
  | production2  { action };
...
```

where

header	useful OCaml code – usually just opens
token	terminal symbols obtained from the lexer, e.g., `tkn1`
left	defines the `tkn1` left associative
S, T, etc	CFG non-terminals such that *S* is the start symbol
production	the CFG production rules

menhir (Specifications)

```
%{
       header
%}

%token tkn1
%left  tkn2
%start <ocaml-type> S
%%

S:
  | ...          { action }
  | production1  { action };
T:
  | production2  { action };
...
```

where

header	useful OCaml code – usually just opens
token	terminal symbols obtained from the lexer, e.g., `tkn1`
left	defines the `tkn1` left associative
S, T, etc	CFG non-terminals such that *S* is the start symbol
production	the CFG production rules
actions	are OCaml expressions of the same type

menhir (Specifications)

```
%{
        header
%}

%token tkn1
%left  tkn2
%start <ocaml-type> S
%%

S:
  | ...           { action }
  | production1   { action };
T:
  | production2   { action };
...
```

where

header	useful OCaml code – usually just opens
token	terminal symbols obtained from the lexer, e.g., `tkn1`
left	defines the `tkn1` left associative
S, T, etc	CFG non-terminals such that *S* is the start symbol
production	the CFG production rules
actions	are OCaml expressions of the same type
%%	the obligatory sign separates the header from the production rules

Example (Balanced Parentheses – Abstract Syntax Tree)

```
(* ast.ml *)
open Printf

type bp =
  | Join   : (bp*bp) → bp
  | Single: bp       → bp
  | Const : bp
  | Str    : string  → bp
  | Int    : int     → bp

let rec bp2String (b: bp): string =
  match b with
  | Join(b1, b2) → bp2String b1 ^ "␣" ^ bp2String b2
  | Single b1    → "LPAREN␣" ^ bp2String b1 ^ "␣RPAREN"
  | Const        → "LPAREN␣RPAREN"
  | Str s        → "IDENT=[" ^ s ^ "]"
  | Int i        → "NUM=[" ^ (string_of_int i) ^ "]"

let printBp (b: bp): unit =
  printf "%s\n" (bp2String b)
```

Example (menhir: Balanced Parentheses – Header and Tokens)

```
(* parser.mly *)
%{
    open Ast
%}

%token BLANK LPAREN RPAREN EOL
%token <string> IDENT
%token <int> NUM
%start <bp> start

%%
```

Example (menhir: Balanced Parentheses – Production Rules)

```
(* parser.mly *)
start:
  | a = T; EOL                    { a };

T:
  | a = T; b = U                  { Join(a,b) }
  | b = U                         { b };

U:
  | LPAREN; RPAREN                { Const }
  | LPAREN; a = T; RPAREN         { Single a }
  | s = IDENT                     { Str s }
  | i = NUM                       { Int i }
  | BLANK                         { Str "" }
```

Example (menhir: Balanced Parentheses – Main)

```
(* main.ml *)
open Printf
open Lexing
open Lexer
open Ast
open Parser

let tokenSwitch (t: Lexer.token): Parser.token =
  match t with
  | BLANK    → BLANK
  | LPAREN   → LPAREN
  | RPAREN   → RPAREN
  | EOL      → EOL
  | IDENT s  → IDENT s
  | NUM i    → NUM i

let compose (f: α → β) (g: β → γ): α → γ = fun x → g (f x)
let astOfString (s: string): bp = start (compose tokenize tokenSwitch) (from_string s)
let main: unit = let ast = astOfString "[][][[[[]]]][ekici2023]" in printBp ast;
```

Remark (in "parser.mly")

where

start: (Lexing.lexbuf → Parser.token) → bp   function maps an input stream to parser tokens

Thanks! & Questions?