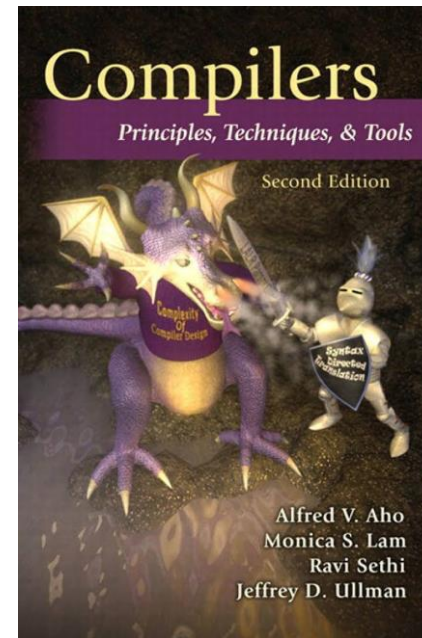


Compiler

Lec 04

Book

Compilers: Principles, Techniques, and Tools is a computer science textbook by Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman about compiler construction.



PowerPoint

<http://www.bu.edu.eg/staff/ahmedaboalatah14-courses/14779>

The screenshot shows a web interface for Benha University. At the top, there is a blue header with the university logo and name on the left, and a welcome message for 'Staff Search: 06' and 'Welcome: Ahmed Hassan Ahmed Abu El Atta (Log out)' on the right. Below the header, a navigation menu on the left lists various university-related links. The main content area displays the user's current location as 'Home/Courses/Compilers' and provides course details for 'Compilers' by 'Ass. Lect. Ahmed Hassan Ahmed Abu El Atta'. A table lists course attributes: Course name (Compilers), Level (Undergraduate), Last year taught (2018), and Course description (Not Uploaded). Below this, there is a section for 'Course password' and another table for course management actions: Course files (add files), Course URLs (add URLs), Course assignments (add assignments), and Course Exams & Model Answers (add exams). A vertical sidebar on the right contains social media icons for Google, Benha University, RG, LinkedIn, Facebook, Twitter, Google+, YouTube, WordPress, and a general social media icon, along with an '(edit)' link at the bottom.

Benha University

Staff Search: 06 Welcome: Ahmed Hassan Ahmed Abu El Atta (Log out)

You are in: [Home/Courses/Compilers](#) [Back To Courses](#)

Ass. Lect. Ahmed Hassan Ahmed Abu El Atta :: Course Details:
Compilers [add course](#) | [edit course](#)

Course name	Compilers
Level	Undergraduate
Last year taught	2018
Course description	Not Uploaded

Course password

Course files	add files
Course URLs	add URLs
Course assignments	add assignments
Course Exams & Model Answers	add exams

(edit)

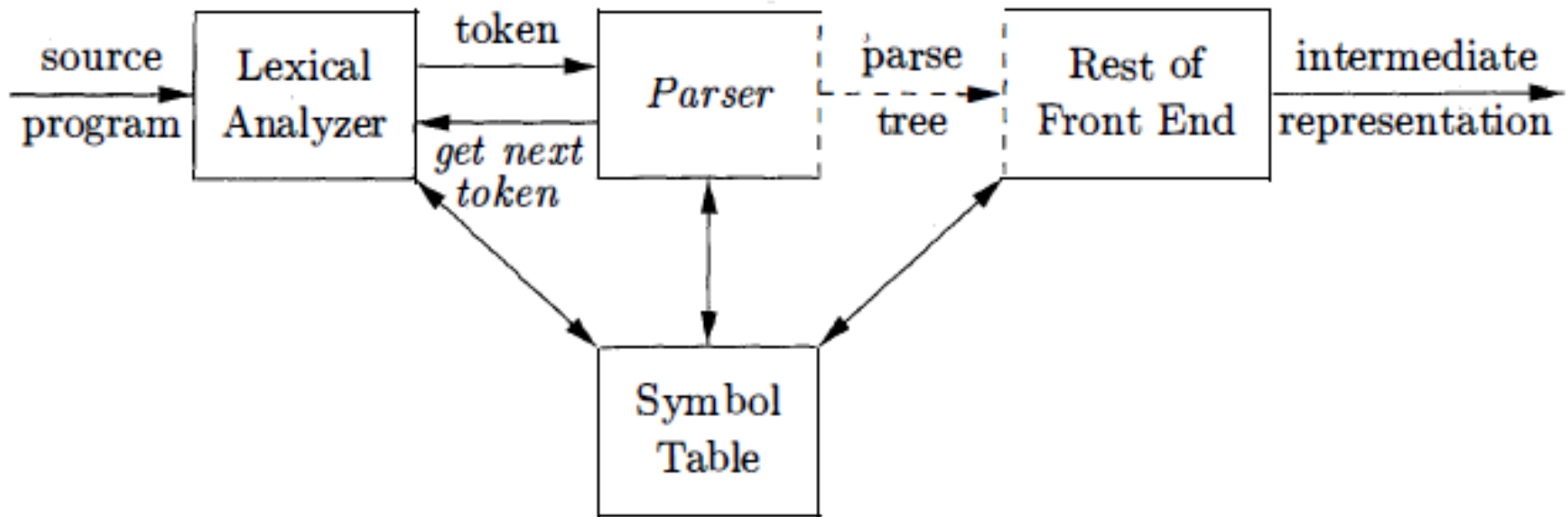
Syntax Analysis

PART I

The role of parser

Construct a parse tree (tree need not be constructed explicitly; the parser and the rest of the front end could well be implemented by a single module).

Report and recover from errors.



Types of parsers for grammars

Universal

- Can parse any grammar (Cocke-Younger-Kasami algorithm and Earley's algorithm).
- Too inefficient to use in production compilers (**Time-Cost**).

Top-down

- Build parse trees from the root to the leaves.
- Scan input from left to right.

Bottom-up

- start from the leaves and work their way up to the root.
- Scan input from left to right.

Grammars

$$G=(T, N, P, S)$$

A set of terminals: basic symbols from which sentences are formed.

A set of nonterminals: syntactic variables denoting sets of strings.

A set of productions: rules specifying how the terminals and nonterminals can be combined to form sentences. ($A \rightarrow \beta$)

The start symbol: a distinguished nonterminal denoting the language.

Example

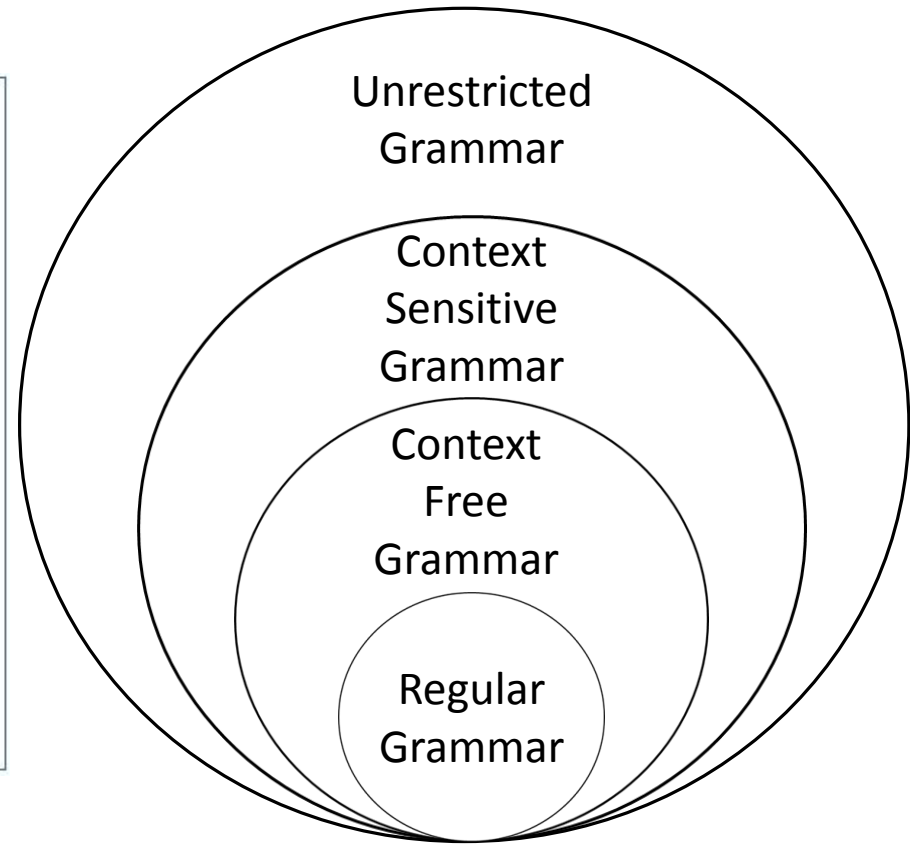
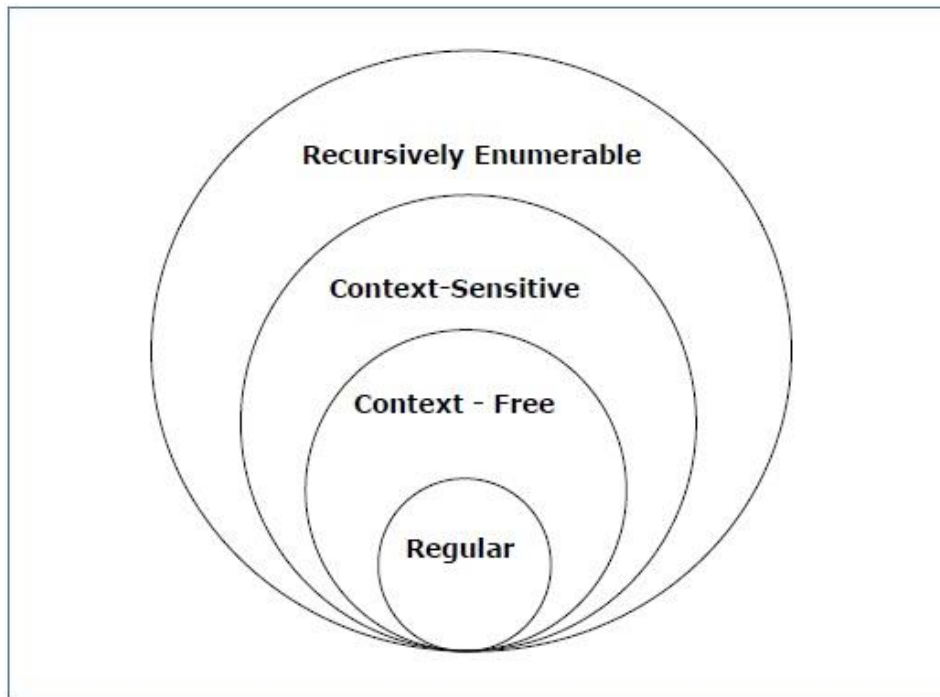
$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (E) \mid \mathbf{id} \end{aligned}$$

$$T = \{ +, *, (,), \mathbf{id} \}$$

$$N = \{ E, T, F \}$$

Start symbol “ E ”

Chomsky hierarchy (classification of grammars)



Regular grammar

A grammar is said to be regular if it is right-linear, where each production in P has the form,

$$A \rightarrow wB \text{ or } A \rightarrow w$$

$A, B \in N$ and $w \in T^*$

Example

$A \rightarrow \varepsilon$

$A \rightarrow a \mid aB$

$B \rightarrow b$

Context-free grammar

A grammar is said to be context-free if each production in P is of the form,

$$A \rightarrow \alpha$$

$$A \in N \text{ and } \alpha \in (NUT)^*$$

Example

$S \rightarrow Aa$

$A \rightarrow B \mid aA$

$B \rightarrow aBc \mid \varepsilon$

Context sensitive grammar

A grammar is said to be context sensitive if each production in P is of the form,

$$\alpha \rightarrow \beta$$

$$|\alpha| \leq |\beta| \text{ and } \alpha, \beta \in (NUT)^*$$

And $S \rightarrow \varepsilon$ is allowed if S does not appear on the right side of any rule.

Example

$AB \rightarrow AbBc$

$A \rightarrow bcA$

$B \rightarrow b$

Unrestricted grammar

A grammar is said to be unrestricted if each production in P is of the form,

$$\alpha \rightarrow \beta$$

$$\alpha \neq \varepsilon \text{ and } \alpha, \beta \in (NUT)^*$$

Example

$S \rightarrow ACaB$

$Bc \rightarrow acB$

$CB \rightarrow DB$

$aD \rightarrow Db$

Derivations

$$\alpha A \beta \implies \alpha \gamma \beta \quad \text{if} \quad A \longrightarrow \gamma$$

$$\alpha \implies^* \alpha$$

$$\alpha \implies^* \beta \quad \text{and} \quad \beta \implies^* \gamma \quad \text{then} \quad \alpha \implies^* \gamma$$

Derivations

$$E \rightarrow E + E \mid E * E \mid -E \mid (E) \mid \text{id}$$

A **leftmost** derivation always chooses the **leftmost nonterminal** to rewrite

$$E \Rightarrow E \Rightarrow (E) \Rightarrow (E + E) \Rightarrow (\text{id} + E) \Rightarrow (\text{id} + \text{id})$$

A **rightmost** derivation always chooses the **rightmost nonterminal** to rewrite

$$E \Rightarrow E \Rightarrow (E) \Rightarrow (E + E) \Rightarrow (E + \text{id}) \Rightarrow (\text{id} + \text{id})$$

Parse trees

BASIS : The tree for $a_1 = A$ is a single node labeled A .

INDUCTION:

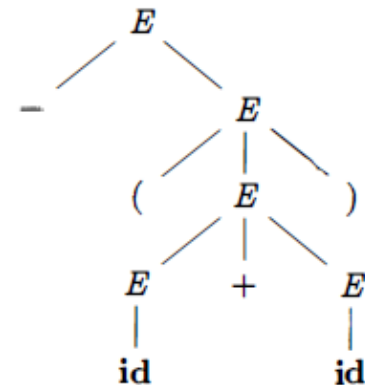
- Suppose α_i is derived from α_{i-1} by replacing X_j , a nonterminal, by $\beta = Y_1 Y_2 \dots Y_m$.
- To model this step of the derivation, find the j^{th} leaf from the left in the current parse tree. This leaf is labeled X_j . Give this leaf “ m ” children, labeled Y_1, Y_2, \dots, Y_m , from the left.

-(id+id)

$E \Rightarrow -E \Rightarrow -(E) \Rightarrow$

$-(E+E) \Rightarrow -(\text{id}+E)$

$\Rightarrow -(\text{id}+\text{id})$



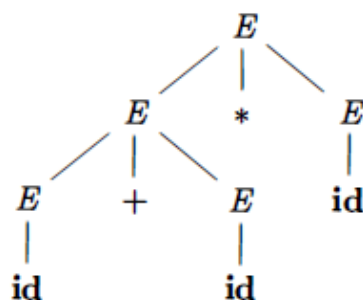
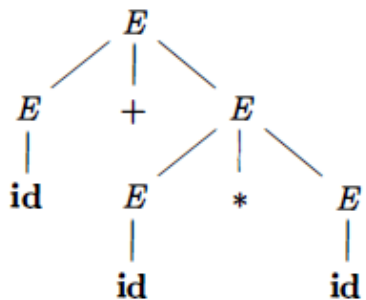
Ambiguity

For some strings there exist more than one parse tree

Or more than one leftmost derivation

Or more than one rightmost derivation

Example: $\text{id}+\text{id}*\text{id}$



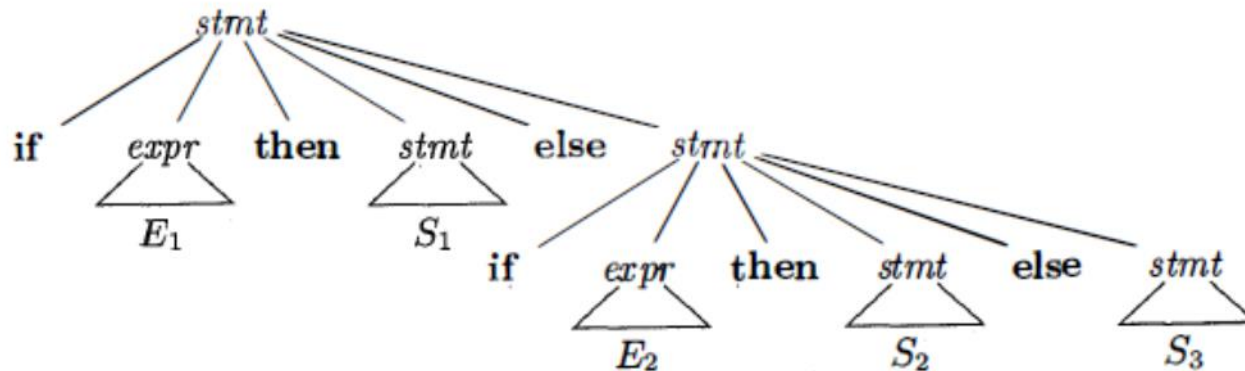
$E \Rightarrow E + E$
 $\Rightarrow \text{id} + E$
 $\Rightarrow \text{id} + E * E$
 $\Rightarrow \text{id} + \text{id} * E$
 $\Rightarrow \text{id} + \text{id} * \text{id}$

$E \Rightarrow E * E$
 $\Rightarrow E + E * E$
 $\Rightarrow \text{id} + E * E$
 $\Rightarrow \text{id} + \text{id} * E$
 $\Rightarrow \text{id} + \text{id} * \text{id}$

Ambiguity

stmt → **if** *expr* **then** *stmt*
 | **if** *expr* **then** *stmt* **else** *stmt*
 | **other**

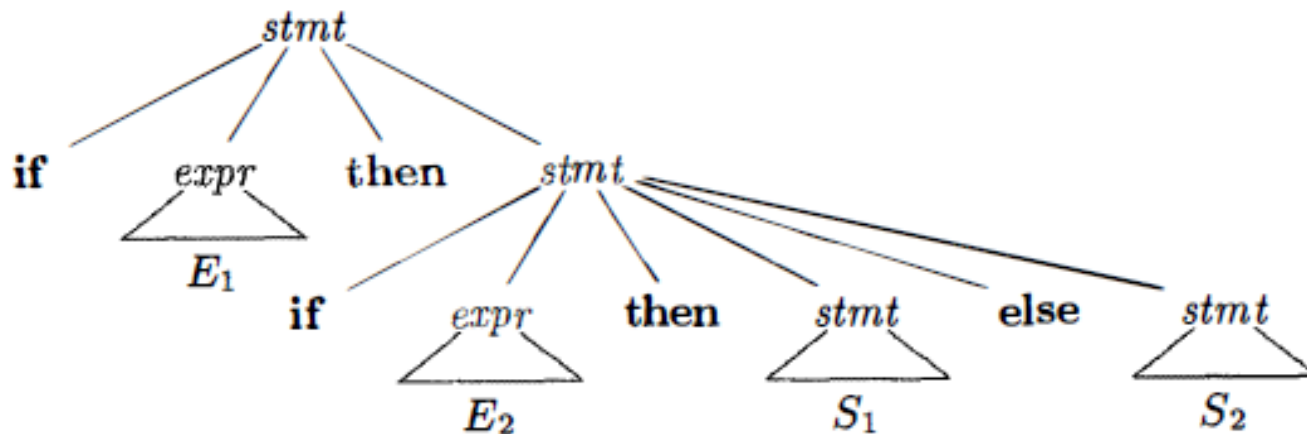
if E_1 **then** S_1 **else** **if** E_2 **then** S_2 **else** S_3



Ambiguity

if E_1 **then** **if** E_2 **then** S_1 **else** S_2

if E_1 **then** **if** E_2 **then** S_1 **else** S_2



Deal with Ambiguity

Rewrite to equivalent unambiguous grammar

- possible, but results in more complex grammar (several similar rules).

Use the ambiguous grammar

- use "rule priority", the parser can select the correct rule.
- works for the dangling else problem, but not for ambiguous grammars in general.
- not all parser generators support it well.

Change the language

- e.g., add a keyword "fi" that closes the "if"-statement.
- restrict the "then" part to be a block: "{ ... }".
- only an option if you are designing the language yourself.

Eliminating Ambiguity

stmt → *matched_stmt*
| *open_stmt*

matched_stmt → **if** *expr* **then** *matched_stmt* **else** *matched_stmt*
| **other**

open_stmt → **if** *expr* **then** *stmt*
| **if** *expr* **then** *matched_stmt* **else** *open_stmt*

Eliminating Ambiguity

$$E \rightarrow E + E \mid E * E \mid -E \mid (E) \mid \text{id}$$

Restrict certain subtrees by introducing new nonterminals.

$$\begin{aligned} E &\rightarrow E + T \mid E - T \mid T \\ T &\rightarrow T * F \mid T / F \mid F \\ F &\rightarrow (E) \mid \mathbf{id} \end{aligned}$$

Examples

Write grammar for each of the following languages:

1. $L = \{a, b\}$.
2. Set of all strings over $\{a, b\}$.
3. Strings that consist of a sequence of a's followed by a sequence of b's
4. Strings that consist of a sequence of a's, where the number of a's is even.
5. The set of strings that begin with ab and end with ba, over alphabet $\{a, b\}$.
6. the language $\{a^n b^n \mid n \geq 0 \text{ and } n \text{ is even}\}$.

Examples

1. The set of strings of parentheses ().
2. $L = \{w \mid w \text{ starts and ends with the same symbol}\}$.
3. The complement language $\{a^n b^n \mid n \geq 0\}$.
4. The palindrome.
5. Is the following grammar ambiguous :
$$S \rightarrow aSbS \mid bSaS \mid \epsilon$$
6. Is the following grammars ambiguous

$$S \rightarrow a \mid aAb \mid abSb$$

$$A \rightarrow aAAb \mid bS$$

