

CS419 Compiler Construction

A Simple One-Pass Compiler [Chapter 2]

Lecture 4

Edited by Dr. Ismail Hababeh

Adapted from slides by Dr. Mohammad Daoud

German Jordanian University

Originally from slides by Dr. Robert A. Engelen

Token Rules

- In many programming languages, the following classes cover most tokens:
 - One token for each **keyword**. The pattern for a keyword is the same as the keyword itself
 - Tokens for the **operators** can be either individual or in classes, such as comparison **<, <=, >, >=, ...**
 - One token representing **identifier**
 - One or more token representing **constants**, such as **numbers** and literal **strings**
 - One Token for each **punctuation symbol**, such as **(,), ;**

Token Hidden Information

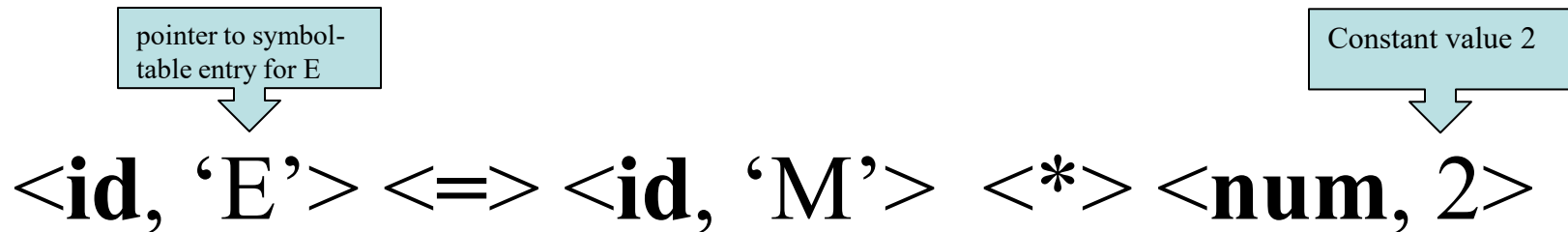
- In the token **id**, **information about the identifier**, such as its lexeme (smallest unit in the programming language) and type, is **kept in the symbol table**. Therefore, the identifier name **is a pointer to the symbol-table entry** for that identifier.

Token Information - Example

Write the tokens stream for the following expression:

$$E = M * 2$$

Solution:



Specification of Token Patterns

- An *alphabet* Σ is a *finite set of symbols* (characters)
 - Examples: $\{1,0\}$ is the binary set,
ASCII is an alphabet used in software systems
- A *string* s is a finite sequence of symbols from Σ
 - $|s|$ denotes the length of string s
 - ε denotes the empty string, thus $|\varepsilon| = 0$
- A *language* is a specific set of strings over some fixed alphabet Σ

Specification of Token Patterns

String Operations

- The *concatenation* of two strings x and y is denoted by xy
- The *exponentiation* of a string s is defined by:

$$s^i = s^{i-1}s \text{ for } i > 0$$

Example: $s^2 = ss$, $s^3 = sss = s^2s$, $s^0 = \varepsilon$
(note that $s\varepsilon = \varepsilon s = s$)

Specification of Token Patterns

Language Operations

- *Union*

$$L \cup M = \{s \mid s \in L \text{ or } s \in M\}$$

- *Concatenation*

$$LM = \{xy \mid x \in L \text{ and } y \in M\}$$

- *Exponentiation*

$$L^i = L^{i-1}L, L^0 = \{\varepsilon\}$$

- *Kleene closure*

$$L^* = \cup_{i=0, \dots, \infty} L^i$$

- *Positive closure*

$$L^+ = \cup_{i=1, \dots, \infty} L^i$$

Language Operations - Examples

Let $L = \{A, B, \dots, Z, a, b, \dots, z\}$ and $D = \{0, 1, \dots, 9\}$

L and D are languages whose strings have a length of 1

- $L \cup D$ is the language with 62 strings of length 1 (52 letters + 10 digits), each of which strings is either one letter or one digit
- LD is the language set of 520 strings of length 2 (52 letters * 10 digits), each consisting of one letter followed by one digit
- L^4 is the language set of all 4-letter strings
- $L(L \cup D)^*$ the language set of all strings of letters and digits beginning with a letter
- D^+ is the language set of all strings of one or more digits

Specification of Token Patterns

Regular Expressions

- Regular Expressions (**REs**) provide a mechanism to **select specific strings from a set of character strings**.
- **Example:** we describe the C language identifiers by:

*letter(letter|_ digit)**

Where | denotes union

- Each regular expression r denotes a language $L(r)$
- The regular expressions are built recursively out of smaller regular expressions (using the rules in the next slide).

Specification of Token Patterns

Regular Expressions

- Basic symbols:
 - ε is a regular expression denoting language $L(\varepsilon)=\{\varepsilon\}$
 - $a \in \Sigma$ is a regular expression denoting language $L(a)=\{a\}$
- If r and s are regular expressions denoting languages $L(r)$ and $M(s)$ respectively, then
 - $r \mid s$ is a regular expression denoting $L(r) \cup M(s)$
 - rs is a regular expression denoting $L(r)M(s)$
 - r^* is a regular expression denoting $L(r)^*$
 - (r) is a regular expression denoting $L(r)$
- A language defined by a regular expression is called a *regular set*

Specification of Token Patterns

Regular Expressions - Example

Let the alphabet $\Sigma = \{a,b\}$

- The regular expression $a|b$ denotes the language $\{a,b\}$
- $(a|b)(a|b)$ denotes the language $\{aa, ab, ba, bb\}$
- a^* denotes the language $\{\epsilon, a, aa, aaa, \dots\}$
- $(a|b)^*$ denotes the language of all strings consisting of zero or more instances of a or b:
 $\{\epsilon, a, b, aa, bb, ab, ba, \dots\}$
- $(a)^*b$ denotes the language $\{b, ab, aab, aaab, \dots\}$

Specification of Token Patterns

Regular Definitions

- Naming convention for regular expressions.
- Example: let Σ be an alphabet, then **regular definition** is a sequence of definitions:

$$d_1 \rightarrow r_1$$

$$d_2 \rightarrow r_2$$

...

$$d_n \rightarrow r_n$$

where r_i is a regular expression over

$$\Sigma \cup \{d_1, d_2, \dots, d_{i-1}\}$$

- Each d_j is a new symbol not in Σ

Specification of Token Patterns

Regular Definitions - Example

- The regular expression for C language **identifiers** are defined as follows:

letter \rightarrow **A** | **B** | ... | **Z** | **a** | **b** | ... | **z**

digit \rightarrow **0** | **1** | ... | **9**

id \rightarrow **letter** (**letter** | **_** | **digit**)^{*}

- Note: recursion is illegal, **why?**
Example:

digits \rightarrow **digit** **digits**

Specification of Token Patterns

Notational Shorthand's

- We frequently use the following shorthand's:

$r^* = r^+ | \varepsilon$ // clean closure

$r^+ = rr^*$ // positive closure

$r? = r | \varepsilon$ // Zero or one instance

$[a-z] = a | b | c | \dots | z$

- Example:

digit $\rightarrow [0-9]$

num $\rightarrow \text{digit}^+ (.\text{digit}^+)? (E (+|-)? \text{digit}^+)?$