Análise Léxica – Parte I

Prof. Magnos Martinello
Março de 2009

Tokens, Lexemas e Padrões

- Um lexema é um conjunto de caracteres no programa-fonte que é reconhecido pelo padrão de algum *token*

Exemplo: const pi = 3.1416;

- A subcadeia pi é um lexema para o token "identificador"
- Tratamos os tokens como símbolos terminais na gramática para a linguagem-fonte.
- São instâncias de uma mesma classe léxica (identificadores, números, etc)

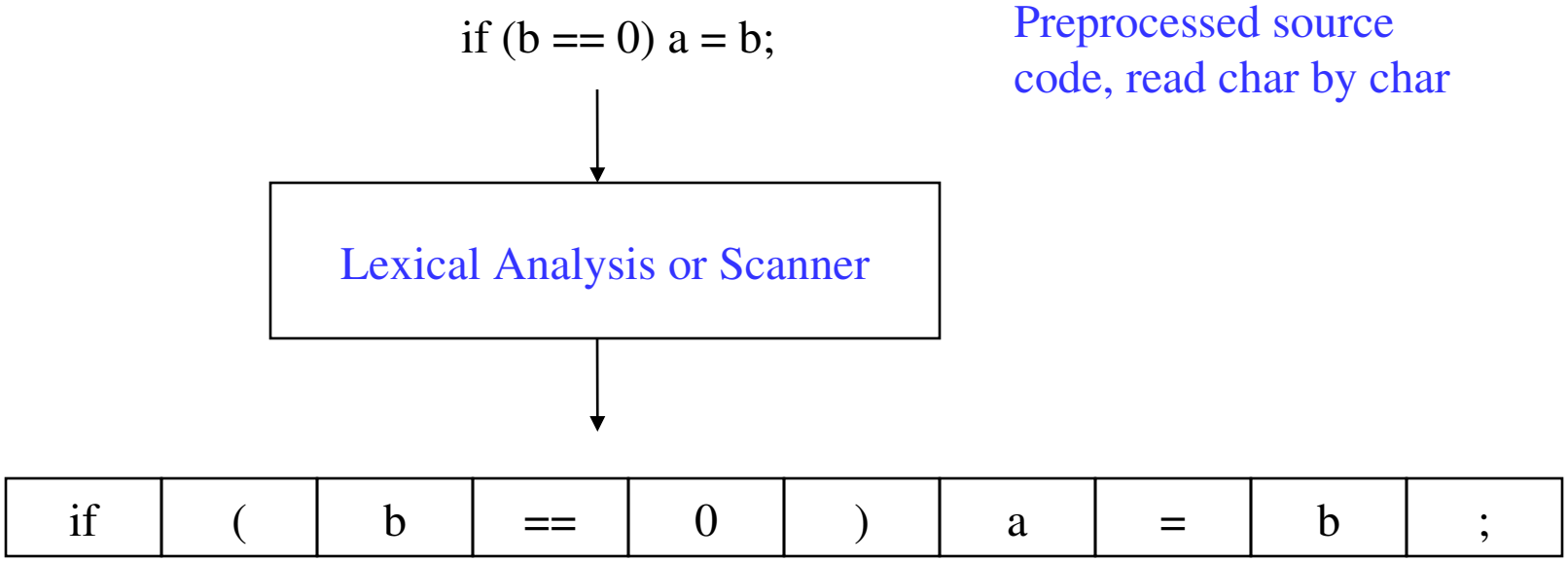
Tokens, Lexemas e Padrões

- Na maioria das linguagens as seguintes *construções* são tratadas como tokens:
  - Palavras-chave, operadores, identificadores, constantes, literais, strings, símbolos de pontuação (parênteses, vírgulas, etc)
  - Quando a sequencia de caracteres pi aparece no programa-fonte, um token representando um identificador é repassado ao parser
  - Na implementação transmitisse um inteiro associado ao token

Exemplo de tokens

Token	**Lexemas exemplo**	**Descrição informal do padrão**
const	const	const
palavras-chave	If, while, break	If, while, break
id	Pi, contador	Letra seguida por letras e/ou dígitos (expressões regulares)
num	3.1416, 0, 6.3E23	Qualquer constante numérica (expressões regulares)
literal	“conteúdo da memória”	Quaisquer caracteres entre aspas
relação	<, <=, =, <>, > , >=	< ou <= ou = ou <> ou > ou >=

Lexical Analysis Process



Preprocessed source code, read char by char

Lexical analysis

- Transform multi-character input stream to token stream
- Reduce length of program representation (remove spaces)
-

Como implementar um analisador léxico ?

- Utilizar uma ferramenta que permita produzir um analisador léxico a partir de uma especificação baseada em expressões regulares
- A partir de uma linguagem de programação convencional
- Escrever em linguagem de montagem

Como descrever Tokens ?

- Expressões regulares são uma notação importante para especificar padrões
- Cada padrão corresponde a um conjunto de cadeias
- Em pascal, um identificador é uma letra seguida por zero ou mais letras ou dígitos
  - » Letra (letra | dígito) *
- Nesta notação, a barra vertical significa ou, os parênteses servem para agrupar sub-expressões, o asterisco significa “zero ou mais instâncias” com o resto da expressão significa concatenação

Como descrever Tokens ?

Uma expressão regular (RE) é formada por um conjunto de regras.

- Cada expressão regular denota uma linguagem L(R)
  - ε é uma expressão regular que denota {ε }, i.e, um conjunto que contém uma cadeia vazia
  - a se a é um símbolo no alfabeto Σ , então a é uma expressão regular que nota {a} conjunto contendo a cadeia a
  - R|S é uma expressão regular denotando L(R) U L(S)
  - RS é um expressão regular denotando concatenação
  - R* concatenação de R 0 ou mais vezes (fechamento Kleene de R)

Exemplos de Linguagem

- Um expressão regular R descreve um conjunto de strings de caracteres denotado por L(R)
- L(R) = é um linguagem definida por R
  - L(abc) = { abc }
  - L(hello|goodbye) = { hello, goodbye }
  - L(1(0|1)*) = todos os números binários que iniciam com 1
  - L((a | b)(a | b)) = {aa, ab, ba, bb}
  - L((a)*) = {ε , a, aa, aaa, …}
  - L(a | a*b) = {?}
- Cada token pode ser definido usando uma expressão regular

Notação de expressões regulares RE

- R+ one or more strings of R: R(R*)
- R? optional R: (R|ε)
- [abcd] one of listed characters: (a|b|c|d)
- [a-z] one character from this range: (a|b|c|d...|z)
- [^ab] anything but one of the listed chars
- [^a-z] one character not from this range

Exemplos de Expressões Regulares

- Regular Expression, R
  - a
  - ab
  - a|b
  - (ab)*
  - (a| ε)b
  - digit = [0-9]
  - posint = digit+
  - int = -? posint
  - real = int (ε | (. posint))
    = -?[0-9]+ (ε |(.[0-9]+))
- Strings in L(R)
  - “a”
  - “ab”
  - “a”, “b”
  - “”, “ab”, “abab”, ...
  - “ab”, “b”
  - “0”, “1”, “2”, ...
  - “8”, “412”, ...
  - “-23”, “34”, ...
  - “-1.56”, “12”, “1.056”, ...
  - Note, “.45” is not allowed in this definition of real