Análise Léxica II

Eduardo Ferreira dos Santos

Ciência da Computação
Centro Universitário de Brasília – UniCEUB

Março, 2017

Sumário

Fases

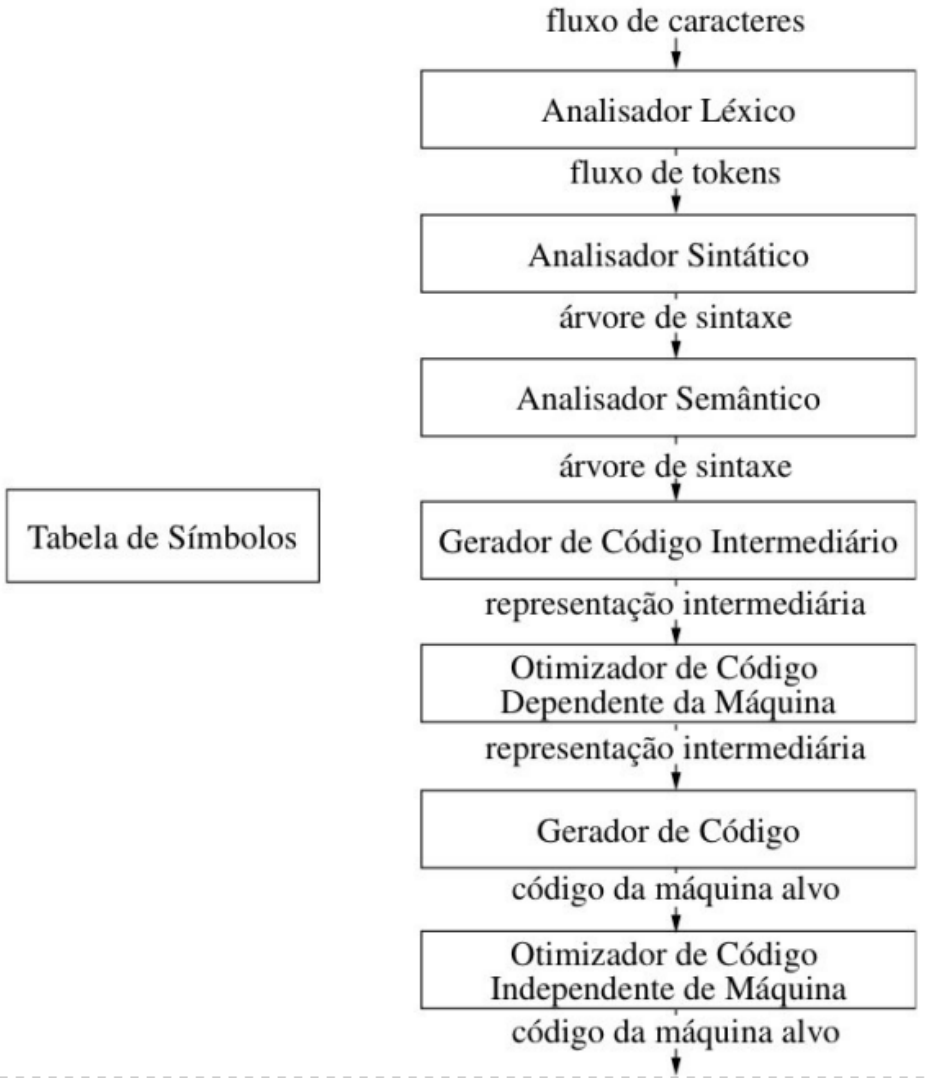

Figura 1.1: Fases do compilador [Aho et al., 2007]

Tokens, padrões e lexemas

Token Dupla formada pelo nome e valor de atributo (opcional)

- Nome do token: símbolo abstrato que representa um tipo de unidade léxica [Aho et al., 2007];
- Também é um símbolo de entrada para o analisador sintático.

Padrões Descrição da forma que os lexemas de um token podem assumir:

- Palavra-chave como token;
- Casamento de sequências de caracteres.

Lexemas Sequência de caracteres no programa fonte que casa com o padrão para um token.

- Identificado como instância desse token.

Objetivos

- Objetivos da análise léxica:
  1. Converter a descrição física do programa numa sequência de tokens;
  2. Cada token é associado a um lexema;
  3. Cada token pode ter atributos (opcional);
  4. A sequência de tokens é enviada ao parser para recuperar a estrutura do programa.
- Desafios:
  - Como particionar o programa em lexemas?
  - Como utilizar o nome correto para cada lexema?

Notação formal

- As expressões regulares são uma forma muito eficiente de representar tokens;
- Definições (relembrando):

  Símbolo Entidade abstrata sem definição formal;
  Alfabeto Conjunto finito de símbolos;
  Palavra Sequência finita de símbolos que pertence a um alfabeto.
- Uma palavra também pode ser definida como uma cadeia ou string.

Exemplo C

- Para os identificadores da linguagem C temos:
  - *letra_* significa qualquer letra ou o sublinhado;
  - *digito* significa qualquer dígito.

$$letra\_(letra\_|digito)^*$$

Problemas 1

- FORTRAN: problema do espaço em branco.

```
DO 5 I = 1,25
DO 5 I = 1.25
```

Problemas 1

- FORTRAN: problema do espaço em branco.

$$\text{DO } 5 \text{ I} = 1{,}25$$
$$\text{DO5I} = 1.25$$

Problemas 1

- FORTRAN: problema do espaço em branco.

$$\text{DO } 5 \text{ I} = 1,25$$
$$\text{DO5I} = 1.25$$

- Pode ser difícil determinar a posição do particionamento.

Como definir

- Defina um conjunto de tokens;
- Define o conjunto de lexemas associados com cada token;
- Defina um algoritmo para resolver os conflitos que surgem nos lexemas.

Casamento de padrões

- Utilizamos as expressões regulares para reconhecer padrões;
- Para completar a análise léxica é necessário:
  1. Construir um trecho de código que examina a cadeia de entrada;
  2. Encontrar um prefixo que seja um lexema.

Exemplo – Comandos

- A gramática da figura descreve comandos de desvio e expressões condicionais;
- Para *relop* utilizamos os operadores de comparação do Pascal ($=, <>$)
- Os termos *if* , *then*, *else*, *id* e *number* são os nomes dos tokens.

stmt	→	**if** *expr* **then** *stmt*
	|	**if** *expr* **then** *stmt* **else** *stmt*
	|	ϵ
expr	→	*term* **relop** *term*
	|	*term*
term	→	**id**
	|	**number**

Figura 2.1: Gramática para comandos de desvio [Aho et al., 2007]

Exemplo – Padrões

digit	→	`[0-9]`
digits	→	*digit*$^+$
number	→	*digits* (. *digits*)? (E [+-]? *digits*)?
lettter	→	`[A-Za-z]`
id	→	*letter* (*letter* | *digit*)*
if	→	`if`
then	→	`then`
else	→	`else`
relop	→	`<	>	<=	>=	=	<>`

Figura 2.2: Padrões para os tokens [Aho et al., 2007]

Palavras-chave

- Para a linguagem, o analisador vai reconhecer as palavras-chave *if* , *then* e *else*;
- Os lexemas são *relop*, *id* e *number*. Também serão reconhecidos;
- Palavras reservadas: não são identificadores, embora os lexemas casem com o padrão para identificadores.
- Remoção de espaços em branco:

$$ws \rightarrow (blank|tab|newline)^{+}$$

- Ao reconhecer o token *ws* não o retornamos ao analisador sintático, mas reiniciamos a análise léxica a partir do próximo caractere.

Tokens e lexemas

LEXEMAS	NOME DO TOKEN	VALOR DO ATRIBUTO
Qualquer *ws*	–	–
`if`	**if**	–
`then`	**then**	–
`else`	**else**	–
Qualquer *id*	**id**	Apontador para entrada de tabela
Qualquer *number*	**number**	Apontador para entrada de tabela
<	**relop**	LT
<=	**relop**	LE
=	**relop**	EQ
<>	**relop**	NE
>	**relop**	GT
>=	**relop**	GE

Figura 2.3: Tokens, padrões e valores de atributo [Aho et al., 2007]

Palavras reservadas

- Usualmente as palavras-chave *if* , *then* ou *else* são reservadas;
- Não são identificadores, mas se parecem com eles;
- É preciso utilizar alguma forma de tratamento para as palavras reservadas:
  1. Instale as palavras reservadas na tabela de símbolos;
  2. Crie diagramas de transição separados para cada palavra-chave.

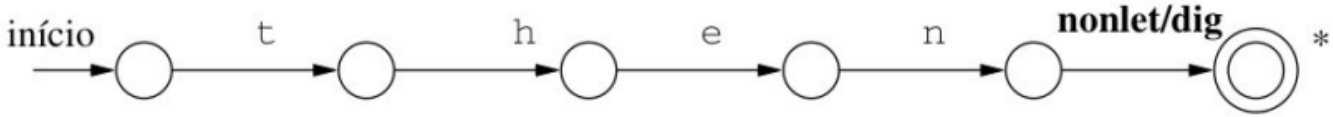

Figura 2.4: Diagrama de transição para a palavra *then* [Aho et al., 2007]

Escolhendo tokens

- Depende da linguagem, mas algumas sugestões podem ser implementadas:
  - Associe tokens às palavras-chave;
  - Associe tokens aos símbolos de pontuação;
  - Descarte informações irrelevantes.

Problema da ambiguidade

- Problema: mais de uma possibilidade para fazer o casamento dos tokens.
- Necessidade de solução de conflitos [Schwarz et al., 2016]:
  - Assuma que todos os tokens estão especificados como expressões regulares;
  - Algoritmo: leitura da esquerda para a direita (*left-to-right scan*);
  - Regra de desambiguação:
    1. Maior cadeia de caracteres no padrão (*maximal munch*);
    2. Sistema de prioridades simples.

Maximal munch

- Converta as expressões regulares em autômatos;
- Rode todos os autômatos em paralelo, mantendo registro do último padrão encontrado;
- Quando todos os autômatos pararem, armazene o último padrão encontrado e continue a busca a partir desse ponto.

Resumo das soluções de conflito

- Construa um autômato para cada expressão regular;
- Junte-os em um único autômato pela regra da união;
- Faça a leitura da entrada, armazenando o último padrão encontrado;
- Escolha o padrão de maior precedência quando houver dúvida;
- Tenha uma regra que capture todos os erros e/ou cadeias que não foram casadas.

Exercício 1

Listing 1: Exemplo do exercício 01 [Aho et al., 2007]

```
float limitedSquare(x) float x {
  /* retorna x ao quadrado, mas nunca mais do que 100 */
  return (x <= -10.0 || x >= 10.0) ? 100 : x*x;
}
```

1. Divida o programa escrito em C++ nos lexemas apropriados;
2. Quais lexemas devem ter valores léxicos associados?
3. Quais são esses valores?

Exercício 2

Listing 2: Exemplo do exercício 02 [Aho et al., 2007]

```
Here is a photo of <b>my house</b>:
<p><img src = "house.gif"><br>
Veja <a href = "morePix.html">More Pictures</a> if you
liked that one</p>
```

1. Divida o documento HTML nos lexemas apropriados;
2. Quais lexemas devem ter valores léxicos associados?
3. Quais são esses valores?

Exercício 3

- Considere a linguagem SQL, formada por quatro instruções: SELECT, INSERT, UPDATE e DELETE.

1. Escreva uma gramática para tratar essa linguagem;
2. Descreva os tokens constituintes da linguagem;
3. Implemente um analisador léxico reduzido para a linguagem SQL, considerando somente as quatro instruções básicas.

OBRIGADO!!!
PERGUNTAS???


Aho, A., Lam, M., Sethi, R., and Ullman, J. (2007).
Compiladores–Princıpios Técnicas e Ferramentas.
Pearson, 2a. edition.


Schwarz, K., Papadakis, H., and Mittal, R. (2016).
Compilers.
Disponível em `http://web.stanford.edu/class/cs143/` Acessado em 30/09/2016.