Compiladores

Análise Sintática

Bruno Lopes

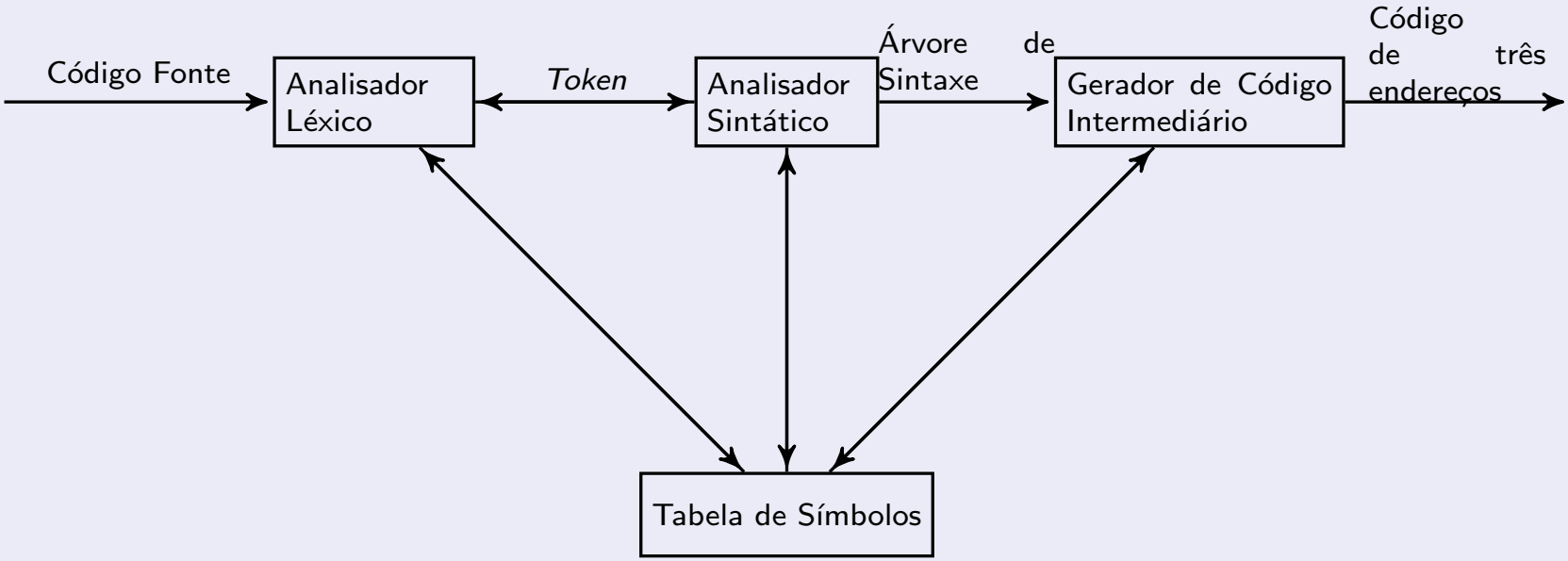
Instituto de Computação
uff

Front-end

- Lida com a linguagem de entrada
- Teste de pertinência: código fonte $\in$ linguagem fonte?
- Programa está bem formado?
  - Sintaticamente?
  - Semanticamente?
- Cria um código intermediário

Instituto de Computação

Front-end



Instituto de Computação

Parser

Construção

Converter uma especificação de linguagem em código.

1. Gramática Livre de Contexto
2. Autômato de Pilha
3. Transformar em código

Instituto de Computação

Análise Sintática

Determina a estrutura sintática

Verifica se a entrada está bem formada

Entrada

Sequência de *tokens*

Árvore sintática

Representação intermediária

Instituto de Computação

Análise Sintática

Determina a estrutura sintática

Verifica se a entrada está bem formada

Entrada

Sequência de *tokens*

Árvore sintática

Representação intermediária

Instituto de Computação

Análise Sintática

Determina a estrutura sintática

Verifica se a entrada está bem formada

Entrada

Sequência de *tokens*

Árvore sintática

Representação intermediária

Instituto de Computação

Recuperação de erros

- Tentativa de inferência do programa correto
- Determinar a ocorrência de um erro *asap*
- Após a ocorrência, escolher um ponto para encerrar o processo
- Evitar cascata de erros
- Evitar laços infinitos de erros

Instituto de Computação

Recuperação de erros

- Tentativa de inferência do programa correto
- Determinar a ocorrência de um erro *asap*
- Após a ocorrência, escolher um ponto para encerrar o processo
- Evitar cascata de erros
- Evitar laços infinitos de erros

Instituto de Computação

Recuperação de erros

- Tentativa de inferência do programa correto
- Determinar a ocorrência de um erro *asap*
- Após a ocorrência, escolher um ponto para encerrar o processo
- Evitar cascata de erros
- Evitar laços infinitos de erros

Instituto de Computação

Recuperação de erros

- Tentativa de inferência do programa correto
- Determinar a ocorrência de um erro *asap*
- Após a ocorrência, escolher um ponto para encerrar o processo
- Evitar cascata de erros
- Evitar laços infinitos de erros

Instituto de Computação

Recuperação de erros

- Tentativa de inferência do programa correto
- Determinar a ocorrência de um erro *asap*
- Após a ocorrência, escolher um ponto para encerrar o processo
- Evitar cascata de erros
- Evitar laços infinitos de erros

Instituto de Computação

Recuperação de erros

Reportar o problema e parar

- não perde tempo tentando traduzir um programa incorreto
- garantia de encontrar pelo menos um erro de sintaxe

Encontrar o máximo possível de erros

Após encontrar um erro, deve mover para um estado em que possa continuar

- descarta símbolos de entrada até encontrar uma palavra de sincronização (e.g. ponto e vírgula)
- produções de erro

Computação

Recuperação de erros

Reportar o problema e parar

- não perde tempo tentando traduzir um programa incorreto
- garantia de encontrar pelo menos um erro de sintaxe

Encontrar o máximo possível de erros

Após encontrar um erro, deve mover para um estado em que possa continuar

- descarta símbolos de entrada até encontrar uma palavra de sincronização (e.g. ponto e vírgula)
- produções de erro

Computação

Recuperação de erros

Reportar o problema e parar

- não perde tempo tentando traduzir um programa incorreto
- garantia de encontrar pelo menos um erro de sintaxe

Encontrar o máximo possível de erros

Após encontrar um erro, deve mover para um estado em que possa continuar

- descarta símbolos de entrada até encontrar uma palavra de sincronização (e.g. ponto e vírgula)
- produções de erro

Computação

Preenchendo o resto da tabela

- Células não preenchidas de uma linha A: desempilha o não terminal: se a entrada é $ ou se a entrada está em Follow(A).
- Avança na entrada: desempilha *tokens* da entrada até encontrar algum que permita o reinício do parser (*token* diferente de $ e não está em First(A) $\cup$ Follow(A).

Instituto de Computação

1. <stmt> ::= <if-stmt>
2. <stmt> ::= other
3. <if-stmt> ::= if (<exp>) <stmt> <else-stmt>
4. <else-stmt> ::= else <stmt>
5. <else-stmt> ::= ϵ
6. <exp> ::= 0
7. <exp> ::= 1

	if	other	else	0	1	$

Instituto de Computação

1. <stmt> ::= <if-stmt>
2. <stmt> ::= other
3. <if-stmt> ::= if (<exp>) <stmt> <else-stmt>
4. <else-stmt> ::= else <stmt>
5. <else-stmt> ::= ϵ
6. <exp> ::= 0
7. <exp> ::= 1

	if	**other**	**else**	**0**	**1**	**$**
<stmt>	1	2				

Instituto de Computação

1. <stmt> ::= <if-stmt>
2. <stmt> ::= other
3. <if-stmt> ::= if (<exp>) <stmt> <else-stmt>
4. <else-stmt> ::= else <stmt>
5. <else-stmt> ::= ϵ
6. <exp> ::= 0
7. <exp> ::= 1

	if	**other**	**else**	**0**	**1**	**$**
<stmt>	1	2				
<if-stmt>	3					

Instituto de Computação

1. <stmt> ::= <if-stmt>
2. <stmt> ::= other
3. <if-stmt> ::= if (<exp>) <stmt> <else-stmt>
4. <else-stmt> ::= else <stmt>
5. <else-stmt> ::= ϵ
6. <exp> ::= 0
7. <exp> ::= 1

	if	**other**	**else**	**0**	**1**	**$**
<stmt>	1	2				
<if-stmt>	3					
<else-stmt>			4, 5			5

Instituto de Computação

1. <stmt> ::= <if-stmt>
2. <stmt> ::= other
3. <if-stmt> ::= if (<exp>) <stmt> <else-stmt>
4. <else-stmt> ::= else <stmt>
5. <else-stmt> ::= ϵ
6. <exp> ::= 0
7. <exp> ::= 1

	if	**other**	**else**	**0**	**1**	**$**
<stmt>	1	2				
<if-stmt>	3					
<else-stmt>			4, 5			5
<exp>				6	7	

Instituto de Computação

1. <stmt> ::= <if-stmt>
2. <stmt> ::= other
3. <if-stmt> ::= if (<exp>) <stmt> <else-stmt>
4. <else-stmt> ::= else <stmt>
5. <else-stmt> ::= ϵ
6. <exp> ::= 0
7. <exp> ::= 1

	if	**other**	**else**	**0**	**1**	**$**
<stmt>	1	2				
<if-stmt>	3					
<else-stmt>			4, 5			5
<exp>				6	7	

if (0) if (1) other other

Instituto de Computação

```
<declaration> ::= <type> <var-list>
<type> ::= int | float
<var-list> ::= <id>, <var-list> | <id>
```

- int x,y,z
- int x, float y, z

Instituto de Computação

Parser *top-down* (descendente)

- Começam da raiz da árvore sintática e crescem em direção às folhas
- Escolhem uma produção e tentam correspondê-la com a entrada
- Uma escolha ruim fará com que aconteça um retrocesso (embora algumas gramáticas sejam livres de *backtracking*)

Instituto de Computação

Parser *bottom-up* (ascendente)

- Partem das folhas e fazem a árvore crescer em direção à raiz
- Conforme uma entrada é consumida, codifica as possibilidades em um estado interno
- Reduzem uma string para o símbolo inicial, por inversão de produções
- Manipulam uma classe maior de gramáticas

Instituto de Computação