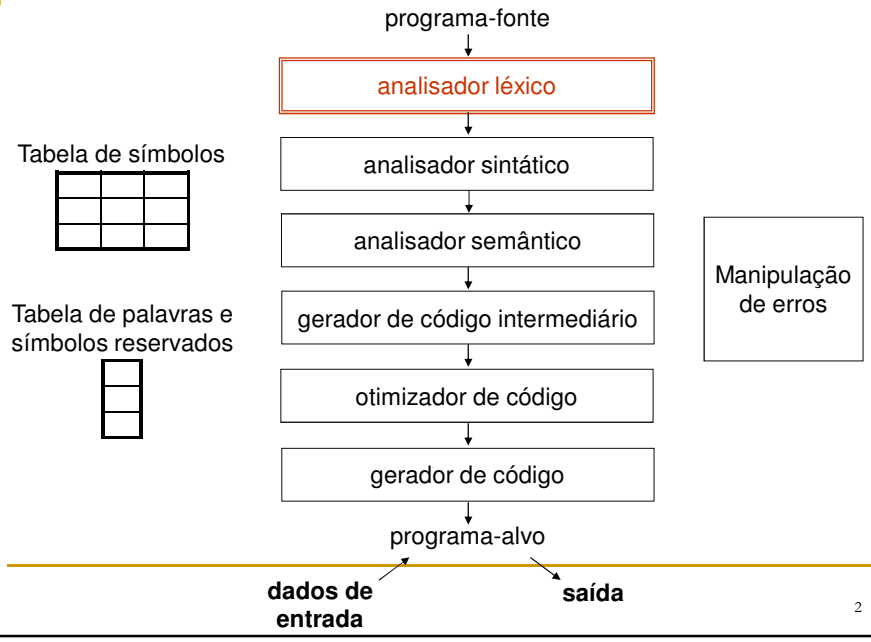
Análise léxica

Função, interação com o compilador
Especificação e reconhecimento de tokens
Implementação
Tratamento de erros

Prof. Thiago A. S. Pardo

1

Estrutura geral de um compilador



2

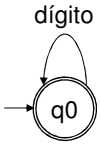
Analisador léxico

- Primeira etapa de um compilador
- Função
  - Ler o arquivo com o programa-fonte
  - Identificar os tokens correspondentes
  - Relatar erros (de forma muito limitada)
- Exemplos de tokens
  - Identificadores
  - Palavras reservadas e símbolos especiais
  - Números

3

Tokens de um programa

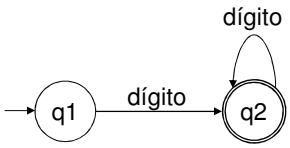
- Autômato para números inteiros sem signal
  - É apropriado? Por quê?



4

Tokens de um programa

- Autômato para números inteiros sem sinal
  - É apropriado? Por quê?



5

Tokens de um programa

- Exercício
  - Construir autômato para consumir comentários
    - {essa função serve para...}
    - /*essa função serve para...*/

6

Analisador léxico

- Conjunto de procedimentos que reconhecem cadeias válidas e lhes associam tokens
- Cada vez que é chamado, retorna par cadeia-token para o analisador sintático
- Consome caracteres irrelevantes: espaços em branco, tabulações, códigos de nova linha, comentários
- Produz mensagens de erro apropriadas quando uma cadeia não é reconhecida por algum autômato
  - Não se atinge o estado final do autômato
  - Tratamento apropriado na implementação do analisador léxico

Tratamento de erros

- Símbolo não pertencente ao conjunto de símbolos terminais da linguagem / identificador mal formado: j@
  - Não há autômato para reconhecer esses símbolos
- Número mal formado: 2.a3
  - Estado final do autômato de números reais não é atingido
- Tamanho do identificador / tamanho excessivo de número: minha_variável_para_..., 5555555555555555
  - Dependência da especificação da linguagem
  - Verificável por ação associada ao estado final dos autômatos
- Fim de arquivo inesperado (comentário não fechado): {...
  - Dependência da especificação da linguagem: comentários de várias linhas?
  - Estado final do autômato de comentários não é atingido
- Char ou string mal formados: ‘a, “hello world
  - Dependência da especificação da linguagem: um único token ou conjunto de tokens?

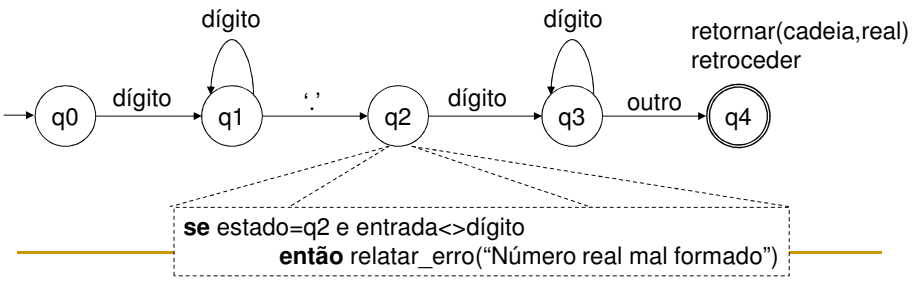
Tratamento de erros

- Compilação não pode parar
  - Erros devem ser sempre relatados
    - <#,erro_léxico> ou <#,nada> ou <#,caractere_inválido>
- Opções para recuperação de erro: **beg#in**
  - Retornar par <beg#,ERRO> e, na próxima chamada, <in,id>
  - Separar o caractere ilegal em um outro pacote
    - Para chamadas sucessivas, retorna pares <beg,id>, <#,ERRO> e <in,id>
- Classificação dos erros
  - Não distinguir erros léxicos, i.e., associa-se um token 'nada' (ou qualquer outro que indique um erro) ao erro e deixa-se a identificação do erro para uma próxima etapa
    - Retorna pares <beg,id>, <#,nada> e <in,id>
  - Análise mais informada do erro léxico (slide anterior)

9

Tratamento de erros

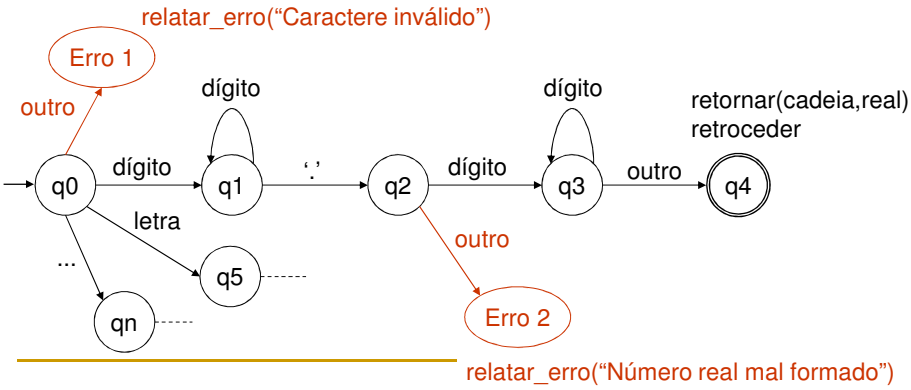
- Algumas opções
  - Associar tratamento de erros individuais a cada estado do autômato, de forma que haja uma relação unívoca entre o estado e o erro possível
    - Vantagem: autômato mais compacto
  - Exemplo: autômato para números reais



10

Tratamento de erros

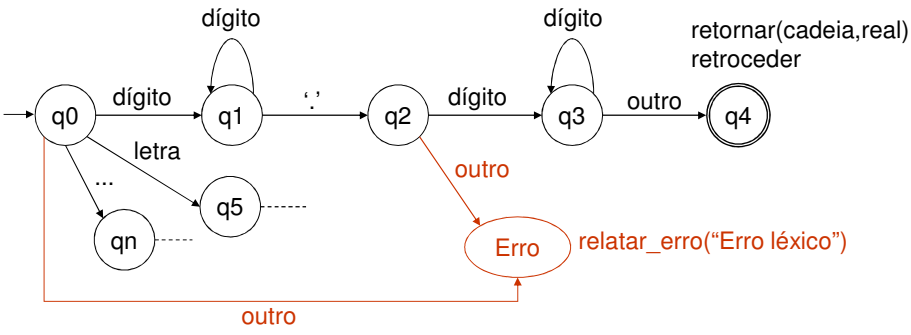
- Algumas opções
  - Criar estados extras no autômato: estados de erro
    - Individuais (um para cada tipo de erro): tratamento mais informado
      - Vantagem: maior clareza



11

Tratamento de erros

- Algumas opções
  - Criar estados extras no autômato: estados de erro
    - Genéricos: erros podem não ser diferenciados
      - Vantagem: maior clareza



12

Tratamento de erros

- Exercício: adicione o tratamento de erros no autômato de identificadores
- Como os seguintes erros seriam reconhecidos?
  - @minha_variavel
  - minha@_variavel
  - minha_variavel@

13

Questões de implementação

- Tabela de palavras reservadas
  - Carregada no início da execução do compilador
  - Busca deve ser eficiente
    - *Hashing*, sem colisões
- Reconhecimento de tokens
  - Criação e manutenção de um buffer
    - Facilidade de leitura e devolução de caracteres
  - Ter sempre um caractere lido no início de um processamento (símbolo *lookahead*)
    - Uniformidade e consistência na análise léxica

14

Analisador léxico

- Idealmente, deveria ser representado por apenas um autômato com vários estados finais, possivelmente
- Cada estado final deve conter ações semânticas adequadas relativas à manipulação das cadeias e tokens identificados
  - Retroceder
  - Retornar
  - Outras?
- Erros devem ser tratados devidamente, com mensagens de erros precisas e significativas para o usuário

15

Questões de implementação

- O analisador léxico é uma das fases que mais consome tempo na compilação
  - Entre 20 e 30% do tempo de compilação
- Bom planejamento é necessário

16