UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

COMPILADORES

Tratamento de Erros Sintáticos
Autómato de Pilha

Docentes: Ruben Moisés Manhiça
Cristiliano Maculuve

Maputo, 28 de abril de 2023

Conteúdo da Aula

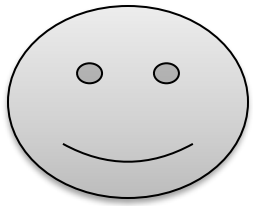
1. Tratamentos de erros sintáticos;
2. Autómato de Pilha

Tratamento de erros sintáticos

- Funções do tratamento de erros
  - Deve relatar a presença de erros de forma clara e precisa
  - Deve se recuperar de cada erro para continuar a análise do programa
  - Pode reparar alguns erros
- Erro sintático
  - Programa não condiz com a gramática da linguagem: símbolo esperado não encontrado
- A realização efetiva do tratamento de erros pode ser uma tarefa difícil

Tratamento de erros sintáticos

- Felizmente, a maioria dos erros são simples
  - Pesquisa com estudantes de Pascal
    - 80% dos enunciados contém apenas um erro; 13% tem dois
    - 90% são erros em um único token
    - 60% são erros de pontuação: p.ex., uso do ponto e vírgula (;)
    - 20% são erros de operadores e operandos: p.ex., omissão de : no símbolo :=
    - 15% são erros de palavras-chave: p. ex., erros ortográficos (wrteln)



Tratamento de erros sintáticos

- O tratamento inadequado de erros pode introduzir uma quantidade enorme de erros, que não foram cometidos pelo programador, mas pelo tratamento de erros realizado

- Tratamento de erros deve ser cauteloso e selecionar os tipos de erros que podem ser tratados para se obter um processamento eficiente

Tratamento de erros sintáticos

- Muitas técnicas para o tratamento de erros
  - Nenhuma delas se mostrou universalmente aceitável
  - Poucos métodos têm ampla aplicabilidade
  - Muitas vezes é artesanal

- Estratégias de tratamento de erro
  - Modo de pânico
  - Recuperação de frases
  - Produções de erro
  - Correção global

Tratamento de erros sintáticos

- Modo de pânico
  - Método mais simples e fácil de implementar; usado pela maioria dos analisadores sintáticos
  - Ao encontrar um erro
    1. Relata-se o erro
    2. Pulam-se tokens até que um token de sincronização seja encontrado
       - **Tokens de sincronização**: pontos do programa (palavras-chave, delimitadores, etc.) em que é possível se retomar a análise sintática com certa segurança;
       - esses tokens precisam ser determinados pelo projetista do compilador
- Exemplo

```
while (x<2 do
    read(y)...
```

Ao notar a falta do parênteses, relata-se a falta do mesmo e se consome tudo até que o token *read* seja encontrado, a partir de onde se recomeça a análise

tratamento de erros sintáticos

- Exemplo

<comandos>::=<comando><mais_comandos>
<mais_comandos>::=;<comandos>| ε
<comando> ::=
read... |
write... |
while (<condição>) do <comandos> $ |
if...

Se acontecer um erro dentro de <condição>, buscam-se seus seguidores para se retomar a análise:)

tratamento de erros sintáticos

- Modo de pânico: Ignoram-se tokens até que se encontre um a partir do qual se possa retomar a análise

```
program P
var x: integer
begin
...
end.
```

Diante da ausência/erro de var, de onde recomeçar a análise? Quem é esse símbolo de recomeço?

Próximo id, seguidor de var

tratamento de erros sintáticos

- Modo de pânico: pulam-se tokens até que se encontre um a partir do qual se possa retomar a análise

```
program P
var x: integer
begin
...
end.
```

Diante da ausência/erro de um trecho grande de código, de onde recomeçar a análise? Quem é esse símbolo de recomeço?

Símbolo de begin, seguidor de declaração de variáveis

tratamento de erros sintáticos

- Modo de pânico: ignoram-se tokens até que se encontre um a partir do qual se possa retomar a análise

```
program P
...
begin
read(x);
write(x+2);
...
end.
```

Diante da ausência/erro do ponto e vírgula (;), de onde recomeçar a análise? Quem é esse símbolo de recomeço?

Símbolo de write, primeiro de comando

Tratamento de erros sintáticos

- Recuperação de frases (correção local)
  - Ao se detectar um erro, realizam-se correções locais na entrada
    - Por exemplo, substituição de vírgula por ponto e vírgula, remover um ponto e vírgula estranho ou inserir um
  - Planeamento das correções possíveis pelo projetista do compilador

Tratamento de erros sintáticos

- Produções de erro
  - Com um bom conhecimento dos erros que podem ser cometidos, pode-se aumentar a gramática da linguagem com produções para reconhecer as produções ilegais e tratá-las adequadamente
- Correção global
  - Método muito custoso de se implementar; apenas de interesse teórico
  - Idealmente, o compilador deveria fazer tão poucas modificações no programa quanto possível
  - Escolhe-se uma sequência mínima de modificações no programa que o tornem correto com o menor custo possível

Autômato com Pilha

- São os formalismos (máquinas) capazes de reconhecer as Linguagens Livres de Contexto;
- Maior poder que os Autômatos Finitos, pois possuem um “espaço de armazenamento” extra que é utilizado durante o processamento de uma cadeia;
- Possui uma pilha que caracteriza uma memória auxiliar onde pode-se inserir e remover informações;
- Mesmo poder de reconhecimento das GLC’s;

Autômato com Pilha

- Exemplo de LLC: $\{a^n b^n \mid n \geq 0\}$
- Um AF não é capaz de reconhecer este tipo de linguagem devido à sua incapacidade de “recordar” (memorizar) informação sobre a cadeia analisada;
- Autômatos com Pilha (AP) possuem uma pilha para armazenar informação, adicionando poder aos AF’s.

Autômato com Pilha

- Definição:
  - AP é uma sextupla $<\Sigma,\Gamma,S,S_0,\delta,B>$, onde:
    - Σ é o alfabeto de entrada do AP;
    - Γ é o alfabeto da pilha;
    - S é o conjunto finito não vazio de estados do AP;
    - S_0 é o estado inicial, $S_0 \in S$;
    - δ é a função de transição de estados,
      δ: $S \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow$ conjunto de subconjuntos finitos de $S \times \Gamma^*$
    - B é o símbolo da base da pilha, $B \in \Gamma$.

Autômato com Pilha

- Ao contrário da fita de entrada, a pilha pode ser lida e alterada durante um processamento;
- O autômato verifica o conteúdo do topo da pilha, retira-o e substitui por uma cadeia $\alpha \in \Gamma^*$.
  - Se α = A, e A $\in \Gamma$, então o símbolo do topo é substituído por A e a cabeça de leitura escrita continua posicionada no mesmo lugar;
  - Se α = A1A2...Na, n>1 então o símbolo do topo da pilha é retirado, sendo An colocado em seu lugar, An-1 na posição seguinte, e assim por diante. A cabeça é deslocada para a posição ocupada por A1 que é então o novo topo da pilha;
  - Se $\alpha = \lambda$ então o símbolo do topo da pilha é retirado, fazendo a pilha decrescer.

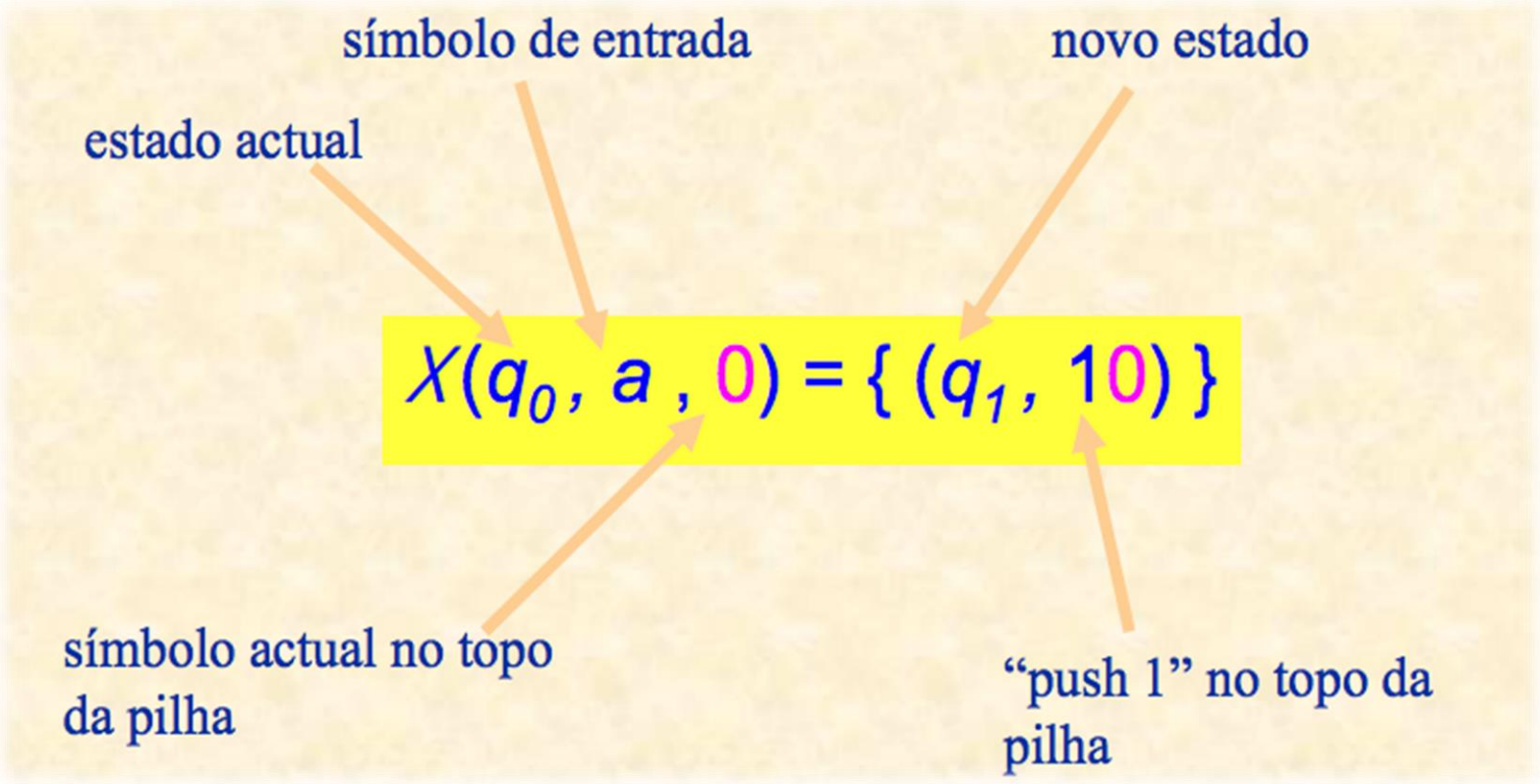
Autômato com Pilha

- A função de transição δ, é função do estado corrente, da letra corrente na fita de entrada e do símbolo no topo da pilha;
- Além disso, esta função determina não só o próximo estado que o AP assume, mas também como o topo da pilha deve ser substituído;
- O AP inicia sua operação num estado inicial especial denotado por S_0 e com um único símbolo na pilha, denotado por B.

Operações sobre a pilha

- "**push**"– introduzir um carácter adicional;
- "**pop**" – apagar um carácter da pilha;
- **Substituição** de um carácter por outro;
- **Inicialização** com qualquer símbolo de; normalmente com o carácter especial #

Transição


símbolo de entrada
novo estado
estado actual
$X(q_0, a, 0) = \{ (q_1, 10) \}$
símbolo actual no topo da pilha
"push 1" no topo da pilha

Transições	Operações sobre a a pilha	Significado
1. $X(q_0, a, \#) = \{ (q_1, 0\#) \}$	push	acrescenta 0 à pilha com #
2. $X(q_0, b, 1) = \{ (q_1, `) \}$	pop	apaga 1
3. $X(q_1, b, 0) = \{ (q_1, 1) \}$	substituição	substitui 0 por 1
4. $X(q_0, b, 1) = \{ (q_1, 1) \}$	nenhuma	não altera
5. $X(q_1, `, 0) = \{ (q_2, `) \}$	pop	apaga 0 sem consumir entrada
6. $X(q_1, `, 0) = \{ (q_2, 10) \}$	push	acrescenta 1 sem consumir entrada

TO BE CONTINUED…

FIM!!!

Duvidas e Questões?