Análise sintática

Função, interação com o compilador
Análise descendente e ascendente
Especificação e reconhecimento de cadeias de tokens válidas
Implementação
Tratamento de erros

Prof. Thiago A. S. Pardo
taspardo@icmc.usp.br

1

Tratamento de erros sintáticos

- Funções
  - 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

2

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)

3

Tratamento de erros sintáticos

- O tratamento inadequado de erros pode introduzir uma avalanche de erros espúrios, 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
  - Como analisar um programa Fortran em um compilador Pascal?

4

Tratamento de erros sintáticos

- Muitas técnicas para o tratamento de erros
  - Nenhuma delas se mostrou universalmente aceitável
  - Poucos métodos têm ampla aplicabilidade
  - Artesanal, muitas vezes
- Estratégias de tratamento de erro
  - Modo de pânico
  - Recuperação de frases
  - Produções de erro
  - Correção global

5

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

6

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
  - Planejamento das correções possíveis pelo projetista do compilador
  - Atenção: pode gerar um ciclo infinito!
    - Inserem-se símbolos infinitamente

7

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
  - Idealmente, o compilador deveria fazer tão poucas modificações no programa quanto possível
  - Escolhe-se uma seqüência mínima de modificações no programa que o tornem correto com o menor custo possível
  - Método muito custoso de se implementar; apenas de interesse teórico

8

ASD preditiva: 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 var, de onde recomeçar a análise? Quem é esse símbolo de recomeço?

Próximo id, seguidor do terminal var

9

ASD preditiva: 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 do não terminal de declaração de variáveis

10

ASD preditiva: 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;
...
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

11

ASD preditiva: tratamento de erros sintáticos

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

- Símbolos de sincronização para um símbolo qualquer A sendo consumido
  - Inicialmente, utilizam-se os seguidores(A)
  - Acrescentam-se os símbolos seguidores do pai de A, isto é, de quem o acionou
    - Útil quando tudo dentro de A deu errado
  - Símbolos de sincronização extras, para garantir que muito da entrada não seja consumido
    - Determinados pelo projetista do compilador

12

ASD preditiva: tratamento de erros sintáticos

- Exemplo

<comandos>::=<cmd>;<comandos>|λ
<cmd> ::=
read... |
write... |
while (<condicao>) do <cmd> |
if...

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

13

ASD preditiva: tratamento de erros sintáticos

- Exemplo

<comandos>::=<cmd>;<comandos>|λ
<cmd> ::=
read... |
write... |
while (<condicao>) do <cmd> |
if...

Se acontecer um erro dentro de <condição> e o programador omitiu os seguidores deste, buscam-se os seguidores do pai para continuar a análise: ponto e vírgula (;), por exemplo

14

ASD preditiva: tratamento de erros sintáticos

- Exemplo

<comandos>::=<cmd>;<comandos>|λ
<cmd> ::=
read... |
write... |
while (<condicao>) do <cmd> |
if...

Ao se buscar pelos seguidores, pode-se perder a análise de grande parte do programa nesse caso (a análise de <comandos>, por exemplo). Em pontos críticos do programa, buscam-se, portanto, tokens a partir de onde seja seguro retomar a análise: read, write e if, por exemplo

15

ASD preditiva: tratamento de erros sintáticos

- Nos analisadores sintáticos descendentes preditivos, a busca por tokens de sincronização é feita sempre que um token esperado não é encontrado
  - Análise recursiva: nos pontos de ERRO dos procedimentos recursivos
  - Análise não recursiva: quando não é possível seguir em frente com a análise, por falta de produção na tabela sintática ou por incompatibilidade de terminais na pilha e na cadeia de entrada

16

ASD preditiva recursiva: tratamento de erros sintáticos

- Algoritmo do procedimento ERRO

```
procedimento ERRO(num_erro, conjunto_simb_sincr)
begin
    imprimir mensagem de erro relativa ao erro de número num_erro;
    enquanto (token_corrente não pertence a conjunto_simb_sincr) faça
        obter_símbolo();
end;
```

- Observações
  - Tomar o cuidado de verificar quando se chegou ao fim do programa-fonte
  - Opcionalmente, a mensagem de erro pode ser impressa antes da chamada do procedimento ERRO
  - Verificar quem é o seguidor encontrado

17

ASD preditiva recursiva: tratamento de erros sintáticos

- Exemplo
  - <programa> ::= program id ; <corpo> .

```
procedimento programa(S)
begin
    se (simb=program) então obter_símbolo()
    senão
        imprimir("Erro: program esperado");
        ERRO({id}+S);
    se (simb=id) então obter_símbolo()
    senão
        imprimir("Erro: identificador de programa esperado");
        ERRO({;}+S);
    se (simb=simb_pv) então obter_símbolo()
    senão
        imprimir("Erro: ponto e vírgula esperado");
        ERRO(Primeiro(corpo)+S);
    corpo({.}+S);
    se (simb=ponto) então obter_símbolo()
    senão imprimir("Erro: ponto esperado");
end;
```

Atenção: se id encontrado, continua a análise normalmente; caso contrário, se elemento de S encontrado, deve-se sair desse procedimento

Pode-se passar como um ou mais parâmetros

18

ASD preditiva recursiva: tratamento de erros sintáticos

- Exemplo
  - <corpo> ::= <dc> begin <comandos> end

```
procedimento corpo(S)
begin
   dc({begin}+S);
   se (simb=begin) então obter_símbolo()
   senão
      imprimir("Erro: begin esperado");
      ERRO(Primeiro(comandos)+S);
   comandos({end}+S);
   se (simb=end) então obter_símbolo()
   senão
      imprimir("Erro: end esperado");
      ERRO(S);
end;
```

19

ASD preditiva recursiva: tratamento de erros sintáticos

- Exemplo
  - <cmd> ::= read(<variaveis>) | write(<variaveis>) |...

```
procedimento cmd(S)
begin
...
   se (simb=simb_abre_parenteses) então obter_símbolo()
   senão
      imprimir("Erro: ( esperado");
      ERRO(Primeiro(variaveis)+S+{read,write,while,if,id...});
...
end;
```

20

ASD preditiva recursiva: tratamento de erros sintáticos

- Exercício: faça o procedimento recursivo com tratamento de erro
  - <dc_v> ::= var <variaveis> : <tipo_var> ; <dc_v>

21

ASD preditiva recursiva: tratamento de erros sintáticos

- Exercício: faça o procedimento recursivo com tratamento de erro
  - <dc_v> ::= var <variaveis> : <tipo_var> ; <dc_v>

```
procedimento dc_v(S)
begin
        se (simb=var) então obter_símbolo()
        senão
                imprimir("Erro: var esperado");
                ERRO(Primeiro(variaveis)+S);
        variaveis({:}+S);
        se (simb=simb_dp) então obter_símbolo()
        senão
                imprimir("Erro: ':' esperado");
                ERRO(Primeiro(tipo_var)+S);
        tipo_var({;}+S);
        se (simb=simb_pv) então obter_símbolo()
        senão
                imprimir("Erro: ';' esperado");
                ERRO(Primeiro(dc_v)+S);
        dc_v(S);
end;
```

22

ASD preditiva recursiva: tratamento de erros sintáticos

- Outra estratégia de tratamento de erro: aninhamento de ifs
  - <dc_v> ::= var <variaveis> : <tipo_var> ; <dc_v>

```
procedimento dc_v(S)
begin
    se (simb=var) então
        obter_símbolo();
        variaveis({:}+S);
        se (simb=simb_dp) então
                obter_símbolo();
                tipo_var({;}+S);
                se (simb=simb_pv) então
                        obter_símbolo();
                        dc_v(S);
                senão
                        imprimir("Erro: ';' esperado");
                        ERRO(S);
        senão
                imprimir("Erro: ':' esperado");
                ERRO(S);
    senão
        imprimir("Erro: var esperado");
        ERRO(S);
end;
```

23

ASD preditiva recursiva: tratamento de erros sintáticos

- Outra estratégia de tratamento de erro: verificação de erro ao fim
  - <dc_v> ::= var <variaveis> : <tipo_var> ; <dc_v>

```
procedimento dc_v(S)
begin
    se (simb=var) então obter_símbolo()
    senão imprimir("Erro: var esperado");
    variaveis({:}+S);
    se (simb=simb_dp) então obter_símbolo()
    senão imprimir("Erro: ':' esperado");
    tipo_var({;}+S);
    se (simb=simb_pv) então obter_símbolo()
    senão imprimir("Erro: ';' esperado");
    dc_v(S);
    ERRO(S);
end;
```

24

ASD preditiva recursiva: tratamento de erros sintáticos

- Outra opção
  - No início de cada procedimento, verificação da presença do símbolo esperado
    - Detecção mais cedo de erros
- Decisão sobre implementação
  - Quais e quantos erros serão detectados e relatados
  - Lógica de programação
  - Controle

25