Earley-Parser para reconhecer a linguagem minipascal

Ruan Pablo Medeiros

[1] Departamento de Ciência da Computação
Universidade Do Estado de Santa Catarina (UDESC) – Joinville – SC – Brazil

pm.ruan@gmail.com

1. Introdução

Este trabalho tem como objetivo utilizar o algoritmo de Earley para reconhecer a sintaxe basica da linguagem minipascal.

2. Algoritmo de Earley

Os algoritmo de Earley é interessante porque pode analisar todas as linguagens livre de contexto. Ele possui tempo de execução O(n) (onde n é o comprimento da cadeia de caracteres analisada), no caso geral, tempo quadrático O(n) para gramáticas não ambíguas, e tempo linear para quase todas as gramáticas LR(k). Seu desempenho é relativamente bom quando as regras são escritas de forma recursiva.

Neste trabalho ele foi implementado com base em [Menezes 2000], onde são adotados os seguintes passos:

Passo 1: Construção do conjunto D_0 Figura 1.

$$
\begin{aligned}
&D_0 = \emptyset \\
&para\ toda\ S \rightarrow \alpha \in P \\
&faça\ D_0 = D_0 \cup \{S \rightarrow .\ \alpha/0\ \} \\
&repita \\
&\quad para\ toda\ A \rightarrow .B\beta/0 \in D_0 \\
&\quad faça\ para\ toda\ B \rightarrow \phi \in P \\
&\quad\quad faça\ D_0 = D_0 \cup \{B \rightarrow .\phi/0\} \\
&até\ que\ o\ cardinal\ de\ D_0\ não\ aumente
\end{aligned}
$$

Figura 1. Construção do conjunto D0

Passo 2: Construção dos demais conjuntos D_r Figura 2.

- Os demais conjuntos Dr partem de D_0.
- Ao gerar o simbolo a_r de w, o algoritmo constrói o conjunto D_r, contendo as produções que podem gerar a_{r+1}.
- w = fita/palavra a ser reconhecida.
- a_r = simbolo lido de w.

3. Linguagem minipascal

A Linguagem minipascal a ser reconhecida é exemplificada na Figura 3

4. Gramática reconhecedora

A gramática reconhecedora, é o conjunto das produções necessárias para que a linguagem seja reconhecida.

As produções são as seguitnes:

$P \rightarrow S$

$S \rightarrow DEC \quad S$

$S \rightarrow ATRIB\ S$

$S \rightarrow PROC\ S$

$S \rightarrow IF\ S$

$S \rightarrow WHILE\ S$

$S \rightarrow FUNC\ S$

$S \rightarrow DEC$

$S \rightarrow ATRIB$

$S \rightarrow PROC$

$S \rightarrow IF$

$S \rightarrow WHILE$

$S \rightarrow FUNC$

$DEC \rightarrow TID \quad TTYDESIG \quad TTYPE \quad TENDINSTRUCTION \quad DEC$

$DEC \rightarrow TID \quad TTYDESIG \quad TTYPE \quad TENDINSTRUCTION$

$DECPF \rightarrow TID \quad TTYDESIG \quad TTYPE \quad DECPF$

$DECPF \rightarrow TENDINSTRUCTION \quad DECPF$

$DECPF \rightarrow TID \quad TTYDESIG \quad TTYPE$

$DECPF \rightarrow TENDINSTRUCTION$

$ATRIB \rightarrow TID \quad TATRIB \quad VALORES \quad TENDINSTRUCTION \quad ATRIB$

$ATRIB \rightarrow TID \quad TATRIB \quad VALORES \quad TENDINSTRUCTION$

$VALORES \rightarrow TID$

$VALORES \rightarrow TINTEGER$

$VALORES \rightarrow TFLOAT$

$VALORES \rightarrow OPERACAO$

$VALORES \rightarrow PARENT$

$OPERACAO \rightarrow VALORES \quad TBOPERATOR \quad VALORES$

$PARENT \rightarrow TLPARENT \quad VALORES \quad TRPARENT$

$COND \rightarrow VALORES \quad TCOMP \quad VALORES \quad LOGICA$

$COND \rightarrow VALORES \;\; TCOMP \;\; VALORES$

$COND \rightarrow VALORES \;\; LOGICA$

$COND \rightarrow VALORES$

$COND \rightarrow VALORES \;\; TCOMP \;\; VALORES$

$COND \rightarrow VALORES$

$L \rightarrow TAND$

$L \rightarrow TOR$

$LOGICA \rightarrow L \;\; COND$

$BEGEND \rightarrow TBEGIN \;\; S \;\; TEND \;\; TENDINSTRUCTION$

$BEGEND \rightarrow TBEGIN \;\; TEND \;\; TENDINSTRUCTION$

$BEGENDRES \rightarrow TBEGIN \;\; S \;\; TRESULT \;\; TATRIB$
$VALORES \;\; TENDINSTRUCTION \;\; TEND \;\; TENDINSTRUCTION$

$BEGENDRES \rightarrow TBEGIN \;\; TRESULT \;\; TATRIB \;\; VALORES$
$TENDINSTRUCTION \;\; TEND \;\; TENDINSTRUCTION$

$VAR \rightarrow TVAR \;\; DEC$

$VAR \rightarrow TVAR$

$IF \rightarrow TIF \;\; COND \;\; TTHEN \;\; BEGEND$

$WHILE \rightarrow TWHILE \;\; COND \;\; TDO \;\; BEGEND$

$PROC \rightarrow TPROCEDURE \;\; TID$
$TLPARENT \;\; DECPF \;\; TRPARENT$
$TENDINSTRUCTION \;\; VAR \;\; BEGEND$

$PROC \rightarrow TPROCEDURE \;\; TID$
$TLPARENT \;\; DECPF \;\; TRPARENT$
$TENDINSTRUCTION \;\; BEGEND$

$PROC \rightarrow TPROCEDURE \;\; TID$
$TLPARENT \;\; TRPARENT \;\; TENDINSTRUCTION \;\; VAR \;\; BEGEND$

$PROC \rightarrow TPROCEDURE \;\; TID$
$TLPARENT \;\; TRPARENT \;\; TENDINSTRUCTION \;\; BEGEND$

$FUNC \rightarrow TFUNCTION \;\; TID$
$TLPARENT \;\; DECPF$
$TRPARENT \;\; TTYDESIG \;\; TTYPE$
$TENDINSTRUCTION \;\; BEGENDRES$

$FUNC \rightarrow TFUNCTION \;\; TID$

$TLPARENT \quad TRPARENT \quad TTYDESIG$

$TTYPE \quad TENDINSTRUCTION \quad BEGENDRES$

5. Implementação

A implementação do trabalho está disponível em: `https://github.com/ruanpablom/earley_parser`. O trabalho foi implementado utilizando as linguagens C e Python.

O código do arquivo pascal.l é necessario para que o programa flex crie os tokens que serão escritos na fita de entrada através do programa main.c.

O programa main.c escrito em C é quem vai gerar a fita de entrada para o algoritmo de Earley. Será necessário na hora de executar o main, passar por paramêtro o código escrito em mini pascal.

Por fim, o programa earley.py escrito em Python é o algoritmo de Earley que irá fazer a analize da fita de entrada e dizer se o código escrito em mini pascal é ou não válido.

6. Execução

Para usuarios Linux a execução do algoritmo se da de forma simples, basta executar qualquer um dos exemplos ".sh"(teste.sh,teste1.sh, teste2.sh), cada executável possui um código em mini pascal como entrada(teste.pas, teste.pas1, teste.pas2), podendo estes serem editados.

Para os demais sistemas operacionais, deve ser primeiro compilado o arquivo main.c, em seguida executado o arquivo main com uma entrada(teste.pas, teste.pas1, teste.pas2) e por fim, executar o programa earley.py (para executar o earley.py necessita ter instalado no computador o python).

Para r variando de 1 até n
Faça $D_r = \emptyset$
  para toda $A \rightarrow \alpha.a_r\beta/s \in D_{r-1}$
  faça $D_r = D_r \cup \{A \rightarrow \alpha a_r\,.\beta/s\}$
  repita
    para toda $A \rightarrow \alpha\,.B\beta/s \in D_r$
    faça para toda $B \rightarrow \phi \in P$
        faça $D_r = D_r \cup \{B \rightarrow .\phi/r\}$
    para toda $A \rightarrow \alpha\,./s \in D_r$
    faça para toda $B \rightarrow \beta.A\phi/k \in D_s$
        faça $D_r = D_r \cup \{B \rightarrow \beta A\,.\phi/k\}$
  até que o cardinal de D_r não aumente

Figura 2. Construção dos conjuntos D_r

7. Conclusão

Neste trabalho concluímos que o algoritmo de Early serve como um ótimo reconhecedor de linguagens.

```
myglobalvar: real;

procedure minhafuncao(a:integer;
b: integer);
var
  v1: integer;
begin
  if a>10 then begin

    myglobalvar := 50;
    if b  > 20 then
      begin

      end;
  end;
end;
```

```
function minhafuncao(a:integer; b:
integer): integer;
begin
  if a>10 then begin
    if b  > 20 then
      begin

      end;
  end;
  Result:=10;
end;
```

Figura 3. Exemplos mini pascal

Referências

Menezes, P. F. B. (2000). In *Linguagens Formais e Autômatos, 3 a ed*. Porto Alegre, Brasil : Instituto de Informática da UFRGS.