Engenharia Gramatical TP2 - Analisador de Código Fonte

Inês Oliveira Anes Vicente
pg50436

Jorge Miguel Silva Melo
pg50507

Miguel Ângelo Machado Martins
pg50655

May 1, 2023

1 Elaboração de gramática - PtyC

A linguagem descrita pela nossa gramática propõe-se a aproveitar o melhor de dois mundos das linguagens ***Python*** e ***C***, em português.

```
program: statement*
statement: declaracao|atribuicao|selecao|repeticao|chamadafuncao|deffuncao|importar|comentario
```

Um programa é um conjunto de *statements* que podem ser dos seguintes tipos:

- **declaração -** tem de especificar o tipo da variável a ser declarada no início. Pode ou não atribuir imediatamente um valor a essa variável.
- **atribuição -** quando se atribui um valor a uma variável. Estes valores, denominados na gramática por "objeto", podem ter várias formas diferentes.
- **seleção -** temos duas estruturas de seleção:
  - **SE -** pode ou não incluir um **SENAO**.
  - **ESCOLHE -** contém um número indefinido de casos, tendo obrigatoriamente de terminar com o caso *default* - **CASO ()**.
- **repetição -** temos três estruturas com ciclos:
  - **ENQ -** repete um bloco de código enquanto uma condição lógica se verificar.
  - **REPETIR -** repete um bloco de código até uma condição lógica se verificar.
  - **PARA -** repete um bloco de código ao longo de uma lista que é percorrida.
- **chamada de função -** sendo que a função pode ser definida pelo utilizador ou uma das pré-definidas pela linguagem (cons, snoc, head, tail).
- **definição de função -** tem de conter o tipo e pode ou não retornar algo. Não pode ser feito no *body* das estruturas.
- **importar -** especificação dos pacotes a importar. Não pode ser feito no *body* das estruturas.
- **comentário -** qualquer bloco de texto entre **:- -:**. Os comentários podem ter várias linhas, sem ser necessária notação específica para tal.

2 *Infos*

Para auxílio daa informações adicionais, criamos uma estrutura complementar (*JSON*) com os campos necessários para a tabela, sendo elas:

- **Variáveis**: campo referente à informação das variáveis, com os seguintes campos:
  - **foi_declarada**: *bool* a dizer se a variável foi declarada ou não;
  - **foi_inicializada**: *bool* a dizer se a variável foi incicializada ou não;
  - **foi_utilizada**: *bool* a dizer se a variável foi utilizada ou não;
  - **foi_redeclarada**: *bool* a dizer se a variável foi redeclarada ou não;
  - **tipo_de_variavel**: Caso a variável tenha sido declarada, este campo diz o tipo da variável;
  - **valores**: Todos os valores que essa variável já teve.
- **instrucoes**: campo relativo ao número de vários tipos de instruções, sendo elas:
  - **atribuicoes**
  - **leitura e escrita**
  - **condicionais**
  - **ciclos**
- **aninhamentos**: campo que conta os aninhamentos que há, sendo eles:
  - **ciclos_dentro_de_ciclos**
  - **condicionais_dentro_de_condicionais**
  - **ciclos_dentro_de_condicionais**
  - **condicionais_dentro_de_ciclos**
- **imports**: lista com os *imports* dados

```
[arkimedez@Arkimedez ptyC]$ python3 src/ptyC.py programas/fibonacci.ptyC fibonacci
Instruções:
{
  "atribuicoes": 9,
  "ciclos": 1,
  "condicionais": 1,
  "leitura e escrita": 0
}


Imports:
[]


Aninhamentos:
{
  "ciclos_dentro_de_ciclos": 0,
  "ciclos_dentro_de_condicionais": 0,
  "condicionais_dentro_de_ciclos": 0,
  "condicionais_dentro_de_condicionais": 0
}
```

Figure 1: Caption

3 Otimizações - aninhamento de SEs

Cada uma das estruturas do programa é percorrida, à procura de SEs que possam ser aninhados. Considera-se que podemos aninhar dois SEs quando nenhum deles tem um SENAO e o *body* do SE de fora é composto apenas pelo SE de dentro. Quando é detetada uma estrutura passível de ser aninhada, é colocada a conjunção das condições de cada SE como condição so SE de fora e o *body* do SE de dentro passa a ser o do SE de fora, eliminando, desta forma, o SE de dentro, mas mantendo todas as ações do programa.

4 Geração de HTML

Para cada símbolo da gramática possível, há uma função que gera um pedaço de código HTML. A função *selector* verifica qual é o tipo do símbolo e encaminha para a função que trata desse tipo especificamente. Em cada uma dessas funções trata do respetivo símbolo e reencaminha cada uma das partes de volta para o selector. Desta forma, a árvore é percorrida recursivamente.

O *HTML* é gerado com diferentes cores para diferentes constituintes do programa, que podem ser configuradas na pasta *configs*. Ele tem também especial cuidado com a identação do programa e tem *hoovers* com indicações das variáveis não declaradas ou utilizadas mas não inicializadas. O programa gera também, no terminal, algumas estatísticas sobre o código *ptyC* que foi introduzido.

Analisador de Código Fonte

```
:- Program to display the Fibonacci sequence up to nth term -:
Int nterms = 6 ;
:- first two terms -:
Int n1 = 0 ;
Int n2 = 1 ;
Int count = 0 ;
Int nth ;
Lista fibo = [ ];
SE (nterms == 1 ) {
   cons ( n1 , fibo ) ;
}SENAO {
   ENQ (count < nterms ) {
      cons ( n1 , fibo ) ;
      nth = n1 + n2 ;
      n1 = n2 ;
      n2 = nth ;
      count = count + 1 ;
   }
}
```

Figure 2: Exemplo de um *HTML* gerado a partir do nosso código

5 Trabalho futuro e conclusão

Para trabalho futuro, pretendemos melhorar a nossa página *HTML*, colocando tabelas com mais informações e melhorar os erros e avisos gerados, para além de melhorarmos a nossa estrutura *infos*, que ainda apresenta alguns *bugs*. Em suma, achamos que conseguimos completar os objetivos propostos pela equipa docente, tendo criado uma gramática completa e tendo gerado uma página *HTML* embelezada com o código identado e colorido e ainda com avisos de erros e problemas do código criado.