FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

RELATÓRIO FINAL

Compiladores 2021/22

Amanda Menezes	2017124788
Pedro Meira	2019223208

Gramática Re-Escrita

A gramática foi gerada tendo por base a informação dada no enunciado, com ajustes e certas "divisões" para evitar erros.

Os parênteses retos foram substituídos, em forma de código, por uma ou nenhuma ocorrência, já os parênteses curvos geram nenhuma ou várias ocorrências.

Esta substituição consiste, maioritariamente, em sub-estados que permitem o número certo de repetições para cada um dos casos, sendo, no caso dos parênteses curvos, recursivo. Ambas as situações aceitam parâmetro nulo, contudo, para evitar erros, quando o parâmetro seria nulo não há chamada para o sub-estado, mas sim um tratamento da situação.

Exemplo:

VarSpec −→ ID {COMMA ID} Type

```
VarSpec: auxId auxVarSpec Type
       | auxId Type
       ;

auxVarSpec: COMMA auxId
       | COMMA auxId auxVarSpec
       ;
```

É possível notar que foi utilizada uma recursão à direita, Métodos descendentes de análise sintática não podem manipular gramáticas recursivas à esquerda. Assim uma transformação que elimine a recursão à esquerda é necessária

No que toca as precedências, a tabela abaixo reúne a estrutura presente no analisador sintático:

Precedência	Token
%right	ASSIGN
%left	OR
%left	AND
%left	EQ NOT LE GE GT LT NE
%left	PLUS MINUS
%left	STAR DIV MOD
%left	UNARY
%nonassoc	LPAR RPAR

As palavras reservadas são guardadas como tokens

Foram criados vários estados auxiliares, quer para facilitar a interpretação do código, quer para melhorar o programa. Um exemplo disso é o estado auxId, cuja função é tratar do ID sempre que um é gerado, evitando a criação de dois nós na mesma produção.

Algoritmos e estruturas de dados da AST e Tabela de símbolos

No que toca a estruturas de dados da AST, cada nó *node* da árvore guarda:

- a expressão (*char* token*);
- o tipo da expressão (*char* type*);
- a linha e a coluna (*int*);
- a anotação (*char)*, para quando necessária;
- um filho (o primeiro a ser gerado, *node *son*);
- um irmão (o primeiro a ser gerado, *node *sibling*).

Como cada nó guarda apenas um filho, caso o mesmo tenha vários filhos, a lista dos mesmos é percorrida de forma recursiva a partir do atributo *sibling.*

No caso de produções que contenham erros, é gerado um nó do tipo *error,* que não será impresso.

No caso de produções vazias, é gerado um nó do tipo *NULL*, que não será impresso.

Nas situações em que foi necessário gerar nós auxiliares, foram gerados nós do tipo *faketype*, que não serão impressos.

O *print* da AST é recursivo, utilizando uma var auxiliar *depth* para saber o número de pontos que devem ser colocados e verifica o tipo de nó, para não imprimir os nós dos tipos acima referidos.

Para a tabela de símbolos foram necessárias quatro funções para criação e impressão da tabela dos símbolos globais (Global Symbol Table) e locais (Function Symbol Table). Primeiramente é feita uma condição para verificar se trata-se de uma função declarada ou uma variável. Caso seja uma função é preciso identificar se o nó é válido, ou seja, se ele não é do tipo "*NULL*", "*faketype*" ou "*error*" como referido anteriormente. Se for válido, é chamada a função responsável pela análise e impressão de erros semânticos. Caso ele seja uma declaração de variável, pode ser global ou local. Para isso é preciso identificar se o "neto" mais à direita do nó atual é uma variável declarada e se isso for verificado é adicionado como variável local. Se o próprio nó atual é identificado como uma variável declarada, este será uma variável global.