INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

ECM253 – Linguagens Formais, Autômatos e Compiladores

Atividade

Análise sintática descendente recursiva

Marco Furlan

8 de setembro de 2022

Esta **atividade** consiste em **alterar** o **projeto** de **exemplo** de **analisador sintático descendente recursivo** em **Java apresentado** em sala de **aula** e **disponível** no **OpenLMS** da disciplina.

1 Descrição do projeto

Alterar o **projeto original** para **atender** os seguintes **novos requisitos**:

(i) A **gramática** deve ser **alterada** para **representar** que um **programa** é um **conjunto** de **uma ou mais expressões** (como no projeto original), **separadas** por **ponto e vírgula**. Desse modo, a **entrada de exemplo** (arquivo) apresentada a seguir **deverá ser analisada com sucesso**:

```
a*3+2/(x+y);
(x+2)/7;
```

Dica. Primeiro, **alterar** a **gramática** para que ela possa considerar essas **repetições** de **expressões** terminadas por ponto e vírgula:

```
goal = program;
program = expr,';', {expr,';'} ;
expr = term, eprime;
eprime = { ('+' | '-'), term };
term = factor, tprime;
tprime = { ( '*' | '/' ), factor };
factor = '(', expr, ')' | number | id;
```

Depois, realizar as **alterações** nos códigos:

- **Analisador léxico**: adicionar o reconhecimento de ponto e vírgula (precisa adicionar um novo *tag*, também);

- **Analisador sintático**: adicionar novas funções de reconhecimento. **Cuidado** na implementação da regra `program`: o critério para repetir as expressões terminadas por ponto e vírgula deve ser `token.tag != Tag.EOF`, pois o marcador de fim de arquivo é que seguirá a última expressão do arquivo. Usar uma repetição `do-while` é mais simples.

(ii) **Reconhecer expressões negativas**. Para reconhecer expressões negativas, **não adianta alterar o analisador léxico**, pois haverá uma confusão entre o reconhecimento do símbolo de menos unário e seu símbolo como operador binário. Assim, tal **modificação** deve ser realizada no **analisador sintático**. Primeiramente, deve-se **adicionar** na **gramática** uma regra que tenha a **maior prioridade** possível e que represente o **operador menos unário**. Alterar a gramática dessa forma (tente produzir manualmente uma expressão para confirmar que funciona):

```
goal = program;
program = expr,';', {expr,';'} ;
expr = term, eprime;
eprime = { ('+' | '-'), term };
term = factor, tprime;
tprime = { ( '*' | '/' ), factor };
factor = '(', expr, ')' | '-',expr |number | id;
```

Depois basta alterar a função que trata o reconhecimento de `fator` para incluir o reconhecimento de `'-',expr`.

2 O que é para entregar

Enviar o projeto desenvolvido, compactado em arquivo tipo ZIP para o link indicado no OpenLMS da disciplina.