Perfil de EL - Engenharia de Linguagens
(1º ano do MEI)

Trabalho Prático 3 (TP3) de
EG – Engenharia Gramatical

Ano Letivo 2021/22

1 Grafos na análise e interpretação de código fonte

Este último Trabalho Prático (TP3) surge na sequência do TP2 em que se pediu para desenvolver, usando o módulo Lark do Python, um Analisador de Código para uma evolução da Linguagem de Programação Imperativa Simples (`LPIS2`), a qual deveria permitir declarar variáveis atómicas e estruturadas (incluindo as estruturas: *conjunto, lista, tuplo, dicionario*), instruções condicionais e pelo menos 3 variantes de ciclos.

Pretende-se agora enriquecer o já poderoso Analisador Estático criado, permitindo também estudar o *Comportamento* dos programas-fonte com base na construção dos vários DAG (Directed Acyclic Graph) que se usam para estudar o fluxo da execução (controlo), CFG, e dos dados (em função das dependências entre as variáveis), DDG, bem como o SDG qu permite perceber o fluxo de execução e de dados ao longo das chamadas a funções.

Como é habitual, o TP será entregue na forma de um relatório desenvolvido em LaTeX, utilizando para isso o template de relatório que se encontra no Material de Apoio à disciplina da Blackboard.

Em suma, deve escrever em Python, usando o Parser e os Visitors do módulo para geração de processadores de linguagens Lark.Interpreter, uma ferramenta que analise programas escritos na sua linguagem LPIS2 e gere em formato dot [1].

1.1 Construção de grafos para análise de código

1. **CFG (Control Flow Graph)** - Crie e represente o *CFG* para as seguintes instruções, suportadas pela sua linguagem:
   (a) Para as estruturas cíclicas (caso não tenha tempo ou esteja com grande dificuldade considere **apenas um** dos **ciclos** - Exemplo : for, while , do-while, etc...);
   (b) Para a estrutura **condicional** *if-else*;
   (c) Para as instruções de **declaração**, **atribuição** e **input/output**.
2. **SDG (System Dependecy Graph)** "lite", que apenas tem em consideração o controlo de fluxo (ignorando o fluxo dos dados).

1.2 Análise de código utilizando grafos (opcional)

Apenas com o auxílio do ***SDG*** gerado na **alínea 2**, **questão 1.1**, identifique e represente as seguintes informações num formato à sua escolha:

1. Zonas de **código inalcansável** (grafos de ilha);
2. A **complexidade de McCabe's** para um determinado excerto de código.

[1] https://www.graphviz.org/doc/info/lang.html

1.3 Sugestões de output

1. Adicione o novo output, em formato *html*, à ferramenta de análise de código desenvolvida no trabalho prático 2.
2. Crie um documento de texto onde escreva as informações pedidas.

A Exemplo de CFG para instrução IF

Link para um bom Visualizador DOT online: `https://bit.ly/3kK7vLj`

```
frase = """
 if x { z = 2; }
 z = z+1;
"""

digraph G {
  inicio -> "if x"
  "if x" -> "z=2"
  "z=2" -> "z=z+1"
  "if x" -> "z=z+1"
  "z=z+1" -> "fim"
  "if x" [shape=diamond];
}
```

Lista-se abaixo um exemplo mais sofisticado, indicativo do CFG a gerar para um dado programa-fonte

```
frase = """
a=0;
 x = 10;
 if x { a = 15; if a {x=5; z=9;} print 12; }
 print a;
 y = 5;
 x=y;
 if y {  x = 9; print x; }
 print y;
"""

digraph G{
 "inicio" -> "set a = 0"
"set a = 0" -> "set x = 10"
"set x = 10" -> "if x"
"if x" -> "set a = 15"
"set a = 15" -> "if a"
"if a" -> "set x = 5"
"set x = 5" -> "set z = 9"
"set z = 9" -> "fi2"
"if a" -> "fi2"

"fi2" -> "print 12"
"print 12" -> "fi1"
"if x" -> "fi1"

"fi1" -> "print a"
"print a" -> "set y = 5"
"set y = 5" -> "set x = y"
"set x = y" -> "if y"
"if y" -> "set x = 9"
"set x = 9" -> "print x"
"print x" -> "fi3"
"if y" -> "fi3"

"fi3" -> "print y"
"print y" -> "fim";
}
```