Editor de Gramáticas Livre de Contexto

Projeto feito por Gabriel Müller e Juliana Pinheiro para disciplina de Linguagens Formais e Compiladores na UFSC. Feito na linguagem Python3 com o framework gráfico PyQt5.

Instalação

É necessário instalar PyQt5 para Python 3. Em sistemas Ubuntu / Debian:

```
sudo apt install python3-pyqt5
```

Para executar, na pasta do projeto:

```
python3 main.py
```

Funcionamento

Gramática Livre de Contexto (`class Grammar`)

A classe `Grammar` é definida por um símbolo inicial e dicionário de conjuntos de tuplas representando as produções. Segue um exemplo de como as produções de uma gramática G são representadas:

```
G: P = { S -> aA | ε
         A -> aA | a }

productions = {
    'S': {('a', 'A'), ('&')},
    'A': {('a', 'A'), ('a')}
}
```

O programa permite criação e edição de gramática, salvando-as para exibição e uso em tempo de execução. A edição de gramáticas na interface gráfica se dá apenas pela definição de suas produções, sem necessidade de definir alfabeto, Vn e Vt.

Verificações

O programa verifica e exibe no momento da criação da gramática se ela é finita/infinita, vazia e fatorada.

Transformações

Foram implementadas as seguintes transformações sobre G.L.C.s:

1. Eliminação de Símbolos Inférteis
2. Eliminação de Símbolos Inalcançáveis
3. Eliminação de ε-produções
4. Eliminação de Produções Simples
5. Fatoração de G.L.C.
6. Eliminação de Recursão à Esquerda

Pela interface estão disponíveis as seguintes transformações:

- G.L.C. Própria:
  - Exibe gramáticas intermediárias para as transformações 1, 2, 3 e 4.
  - Exibe conjuntos NF, Vi, Ne e NA .
  - Salva a gramática final (em tempo de execução).
- Fatoração de G.L.C.:
  - O usuário define o número de passos de fatoração.
  - Exibe se G inicial é fatorável no número de passos escolhido.
  - Se G for fatorável, salva a gramática final (em tempo de execução).
- Eliminação de Recursão à Esquerda:
  - Exibe os não-terminais recursivos a esquerda e o tipo das recursões (direta/indireta).
  - Salva a gramática final (em tempo de execução).

Os algoritmos e processos implementados para as transformações são os vistos em aula.

Outras operações

Também foram implementado o cálculo dos seguintes conjuntos para todo $A \in Vn$:

- First(A)
- Follow(A)
- First-NT(A)

Análise (`reader.py`)

A leitura da entrada textual para gramáticas livres de contexto é feita por `read_cfg(string)`. O método espera alguns padrões:

- Espaço em branco entre os símbolos do lado direito;
- Símbolos não-terminais representados por letras maiúsculas seguidas de 0 ou mais dígitos;
- Símbolos terminais representados por um ou mais caracteres contíguos (exceto letras maiúsculas);
- Épsilon(ε) representado por &.