Programação de Linguagens

Instituto Politécnico do Cávado e do Ave
Escola Superior de Tecnologia
Licenciatura Engenharia de Sistemas Informáticos
Ano Letivo: 2023/2024

Relatório
Trabalho Prático 02

Joana Freitas Pimenta – 22999

Diogo Manuel Marques – 23000

Ludgero Miguel Simões – 23135

Junho 2024

Índice

Introdução

O presente relatório aborda o segundo trabalho prático realizado em contexto da unidade curricular de Processamento de Linguagens. Detalha o desenvolvimento e a implementação de uma aplicação para o processamento de uma linguagem funcional especifica, a Linguagem Funcional do Cávado e do Ave (FCA). Este projeto visa fornecer experiência prática na construção de analisadores de linguagens e na tradução de linguagens através de ações semânticas.

O trabalho foi estruturado em várias etapas fundamentais:

- Especificação da gramática concreta da linguagem de entrada.
- Construção de um reconhecedor léxico utilizando a biblioteca 'lex' para identificar os símbolos terminais.
- Desenvolvimento de um reconhecedor sintático com a biblioteca 'yacc' para interpretar a gramática concreta.
- Planejamento de uma árvore de sintaxe abstrata (AST) para representar a linguagem de entrada e associação de ações semânticas às produções da gramática.
- Implementação de um gerador de código que, a partir da AST, produza a resposta solicitada.

Neste relatório vai-se adentrar as diferentes fases do desenvolvimento, especificando abordagens, objetivos atingidos e detalhar as funcionalidades implementadas.

Desenvolvimento

1. Especificação da Gramática

Para dar início a este trabalho, começou-se por utilizar os exemplos fornecidos no enunciado do trabalho como base. A partir destes exemplos, iniciou-se por definir as principais palavras reservadas da linguagem FCA. Nestas categorias principais, incluíu-se uma variedade de elementos. Esses componentes consistiram em atribuições, declarações de funções, chamadas de funções, operações matemáticas, junção de textos, comandos de decisão e repetições, além de manipulação de arrays.

Para cada uma dessas palavras reservadas da linguagem, especificamos termos de regras de produção. Estas regras de produção representam como os símbolos terminais e não-terminais se combinam para formar construções válidas na linguagem FCA.

Posteriormente, procuramos estruturar a gramática em formato JSON. Este formato foi escolhido porque é claro e fácil de interpretar, mesmo para aqueles que não estão familiarizados com a linguagem FCA.

2. Construção do Reconhecedor Léxico

Com o comportamento da linguagem FCA definida, seguiu-se para a construção do reconhecedor léxico. Nesta etapa, com a utilização da biblioteca 'lex´ do 'PLY' criou-se um analisador léxico capaz de identificar e categorizar os símbolos terminais da linguagem.

Passou-se por definir a lista de tokens que a linguagem FCA suporta. Cada token corresponde a termos reconhecidos pela FCA. Na lista podemos encontrar tokens para operadores aritméticos, palavras-chave (como FUNCAO, ESCREVER, SE, SENAO, etc.), identificadores de variáveis, literais inteiros e strings, operadores de comparação, e símbolos especiais (como parênteses, vírgulas, e ponto-e-vírgula).

Para cada token, escreveram-se regras léxicas utilizando expressões regulares. Estas expressões foram implementadas no ficheiro 'arith_lexer.py'. As expressões regulares permitiram identificar padrões específicos no texto de entrada, categorizando cada palavra conforme a classe de token correspondente.

Ademais, desenvolveram-se funções específicas para lidar com tokens mais complexos e exceções. Foi também definido quais caracteres o lexer deveria ignorar, como o espaço, tabuladores(\t) e o newline(\n). Também foi definida a função `t_error' que é responsável por tratar caracteres ilegais. Passando à configuração do analisador léxico, foi criado o objeto lexer conectado às regras estabelecidas anteriormente.

Para garantir o funcionamento adequado e eficiente do reconhecedor léxico, foram implementados testes unitários, localizados no ficheiro `arith_lexer_test.py'. Criou-se cenários de teste que abrangiam uma ampla gama de tokens possíveis na linguagem FCA. Por exemplo, a identificação apropriada de palavras-chave, operadores, identificadores e literais.

Adotou-se, também, um processo iterativo para a construção do reconhecedor léxico. Este processo envolveu ajustes e refinamentos das expressões regulares e funções de manipulação de tokens conforme necessário após cada rodada de testes. Isso foi feito para resolver problemas que surgiram e para aprimorar a precisão do reconhecedor.

3. Construção do Reconhecedor Sintático

Para cada construção sintática da linguagem FCA, foram estabelecidas regras de produção que descrevem a combinação de tokens e não-terminais. Estas regras foram aplicadas no ficheiro `arith_grammar.py'. Cada regra ira gerar uma operação especifica para facilitar a execução da linguagem FCA.

Cada função de produção foi elaborada para atender a uma regra específica na gramática. Estas funções encarregam-se de montar a estrutura hierárquica do programa, ou seja, a árvore de derivação ou árvore de sintaxe concreta.

A robustez do analisador sintático foi garantida pela implementação de uma função de tratamento de erros sintáticos. Esta função foi encarregada de identificar e gerenciar erros de sintaxe, fornecendo mensagens de erro úteis para a depuração.

4. Árvore de Sintaxe Abstrata

Para representar a estrutura lógica do programa, cada regra de produção foi adaptada para construir a Árvore de Sintaxe Abstrata (AST). Para assegurar a precisão do analisador sintático, uma série de testes foram implementados no

ficheiro `arith_grammar_test.py'. Estes testes envolveram a análise de diversas expressões e comandos da linguagem FCA, verificando se o analisador sintático produzia a AST correta e identificava corretamente quaisquer erros sintáticos.

Finalmente, o analisador sintático foi integrado com o reconhecedor léxico. Utilizando o lexer definido anteriormente, o parser processou a sequência de tokens gerada pelo lexer, construindo a AST de acordo com as regras de produção estabelecidas. Esta integração foi essencial para permitir a análise completa de programas escritos na linguagem FCA.

Foram criados nós para representar operações aritméticas, armazenando o operador e os operandos envolvidos. Os nós para as declarações de variáveis guardam o nome da variável e a expressão que lhe é atribuída. Nós para as chamadas de funções guardam o nome da função e os argumentos que lhe são passados.

Ao finalizar esta etapa, foi possível transformar qualquer programa válido escrito em FCA numa AST bem definida. A construção do analisador sintático não só assegurou a conformidade sintática dos programas.

5. Gerador de código

Depois de montar e analisar a Árvore de Sintaxe Abstrata (AST), o próximo passo foi criar o gerador de código. Esta etapa transformou a AST em código que pode ser executado, de acordo com as regras da Linguagem.

O método principal `evaluate', procede ao processamento de cada nó da árvore AST e retorna o resultado. Este método foi utilizado para analisar a AST e executar comandos da FCA de forma organizada. A análise começou no nó raiz da AST e passou por todos os nós. O gerador de código foi implementado no ficheiro `arith_eval.py'. Este script percorreu a AST e gerou a saída correspondente. Para comandos de saída, como `ESCREVER', a função `evaluate' mostrou o resultado no terminal. Para outras operações, como atribuições e chamadas de função, os valores foram guardados para uso futuro.

O gerador de código foi integrado ao sistema completo de análise e execução da linguagem FCA. O script principal (`main.py') organizou todas as etapas, desde a análise léxica e sintática até a análise da AST.

Resultados

A. Atribuição a uma variável

Ao executar o 'exemplo-A-01.fca' pode-se verificar os seguintes resultados:

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-A-01.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'atribuicao',
           'args': ['tmp_01',
                    {'op': '+',
                     'args': [{'op': '*',
                               'args': [{'op': 'int', 'args': ['2']},
                                        {'op': 'int', 'args': ['3']}]},
                              {'op': 'int', 'args': ['4']}]}]},
          {'op': 'atribuicao',
           'args': ['a1_',
                    {'op': '-',
                     'args': [{'op': 'int', 'args': ['12345']},
                              {'op': '*',
                               'args': [{'op': 'int', 'args': ['5191']},
                                        {'op': 'int', 'args': ['15']}]}]}]},
          {'op': 'atribuicao',
           'args': ['idade_valida?', {'op': 'int', 'args': ['1']}]},
          {'op': 'atribuicao',
           'args': ['mult_3!',
                    {'op': '*',
                     'args': [{'var': 'a1_'}, {'op': 'int', 'args': ['3']}]}]}]}
<< -196560
```

FIGURA 1 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-A-01.FCA'

B. Instruções de I/O e strings

Ao executar os exemplos pode-se verificar os seguintes resultados:

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-B-01.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'atribuicao',
           'args': ['valor', {'op': 'string', 'args': ['teste']}]},
          {'op': 'escrever', 'args': [{'var': 'valor'}]},
          {'op': 'comentario', 'args': ['conteÃºdo de valor Ã© apresentado']},
          {'op': 'escrever',
           'args': [{'op': '*',
                     'args': [{'op': 'int', 'args': ['365']},
                              {'op': 'int', 'args': ['2']}]}]},
          {'op': 'comentario', 'args': ['730']},
          {'op': 'escrever', 'args': [{'op': 'string', 'args': ['Ola Mundo']}]},
          {'op': 'comentario', 'args': ['OlÃ¡, Mundo!']},
          {'op': 'atribuicao',
           'args': ['curso', {'op': 'string', 'args': ['ESI']}]},
          {'op': 'escrever',
           'args': [{'op': 'concat',
                     'args': [{'op': 'string', 'args': ['OlÃ¡, ']},
                              {'var': 'curso'}]}]},
          {'op': 'comentario', 'args': ['OlÃ¡, ESI']}]}
teste
730
Ola Mundo
OlÃ¡, ESI
<< None
```

FIGURA 2 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-B-01.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-B-02.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'comentario',
           'args': ['exemplo interpolaÃ§Ã£o de strings\n   OlÃ¡, EST IPCA!']},
          {'op': 'atribuicao',
           'args': ['escola', {'op': 'string', 'args': ['EST']}]},
          {'op': 'atribuicao',
           'args': ['inst', {'op': 'string', 'args': ['IPCA']}]},
          {'op': 'escrever',
           'args': [{'op': 'interpolacao',
                     'args': [{'op': 'string', 'args': ['OlÃ¡, ']},
                              {'var': 'escola'},
                              {'op': 'string', 'args': [' ']},
                              {'var': 'inst'},
                              {'op': 'string', 'args': ['!']}]}]}]}
OlÃ¡, EST IPCA!
<< None
```

FIGURA 3 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-B-02.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-B-03.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'atribuicao',
           'args': ['valor', {'op': 'entrada', 'args': []}]},
          {'op': 'atribuicao',
           'args': ['ate10', {'op': 'aleatorio', 'args': ['10']}]}]}
Função entrada, introduza o valor: Entrada
<< 4
```

FIGURA 4 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-B-03.FCA'

C. Declaração de Funções

Ao executar os exemplos pode-se verificar os seguintes resultados:

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-C-01.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'funcao',
           'args': ['soma',
                    [{'var': 'a'}, {'var': 'b'}],
                    {'op': '+', 'args': [{'var': 'a'}, {'var': 'b'}]}]},
          {'op': 'funcao',
           'args': ['soma2',
                    [{'var': 'c'}],
                    {'op': 'seq',
                     'args': [{'op': 'atribuicao',
                               'args': ['c',
                                        {'op': '+',
                                         'args': [{'var': 'c'},
                                                  {'op': 'int',
                                                   'args': ['1']}]}]},
                              {'op': '+',
                               'args': [{'var': 'c'},
                                        {'op': 'int', 'args': ['1']}]}]}]},
          {'op': 'atribuicao',
           'args': ['seis',
                    {'op': 'call_func',
                     'args': ['soma',
                              [{'op': 'int', 'args': ['4']},
                               {'op': 'int', 'args': ['2']}]]}]},
          {'op': 'atribuicao',
           'args': ['oito',
                    {'op': 'call_func',
                     'args': ['soma2', [{'var': 'seis'}]]}]}]}
<< 8
```

FIGURA 5 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-C-01.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-C-02.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'funcao',
           'args': ['area_retangulo',
                    [{'var': 'a'}, {'var': 'b'}],
                    {'op': 'seq',
                     'args': [{'op': '*',
                               'args': [{'var': 'a'}, {'var': 'b'}]}]}]},
          {'op': 'funcao',
           'args': ['area_quadrado',
                    [{'var': 'a'}],
                    {'op': 'seq',
                     'args': [{'op': 'call_func',
                               'args': ['area_retangulo',
                                        [{'var': 'a'}, {'var': 'a'}]]}]}]},
          {'op': 'atribuicao',
           'args': ['a',
                    {'op': 'call_func',
                     'args': ['area_retangulo',
                              [{'op': 'int', 'args': ['10']},
                               {'op': 'int', 'args': ['20']}]]}]},
          {'op': 'atribuicao',
           'args': ['b',
                    {'op': 'call_func',
                     'args': ['area_quadrado',
                              [{'op': 'int', 'args': ['30']}]]}]}]}
<< 900
```

FIGURA 6 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-C-02.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-C-03.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'funcao',
           'args': ['area',
                    [{'var': 'a'}, {'var': 'b'}],
                    {'op': '*', 'args': [{'var': 'a'}, {'var': 'b'}]}]},
          {'op': 'funcao',
           'args': ['area',
                    [{'var': 'c'}],
                    {'op': 'call_func',
                     'args': ['area', [{'var': 'c'}, {'var': 'c'}]]}]},
          {'op': 'atribuicao',
           'args': ['d',
                    {'op': 'call_func',
                     'args': ['area',
                              [{'op': 'int', 'args': ['10']},
                               {'op': 'int', 'args': ['20']}]]}]},
          {'op': 'atribuicao',
           'args': ['e',
                    {'op': 'call_func',
                     'args': ['area', [{'op': 'int', 'args': ['30']}]]}]}]}
<< 900
```

FIGURA 7 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-C-03.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-C-04.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'funcao',
           'args': ['fib',
                    [{'op': 'int', 'args': ['0']}],
                    {'op': 'int', 'args': ['0']}]},
          {'op': 'funcao',
           'args': ['fib',
                    [{'op': 'int', 'args': ['1']}],
                    {'op': 'int', 'args': ['1']}]},
          {'op': 'funcao',
           'args': ['fib',
                    [{'var': 'n'}],
                    {'op': 'seq',
                     'args': [{'op': 'atribuicao',
                               'args': ['a',
                                        {'op': 'call_func',
                                         'args': ['fib',
                                                  [{'op': '-',
                                                    'args': [{'var': 'n'},
                                                             {'op': 'int',
                                                              'args': ['1']}]}]]}]},
                              {'op': 'atribuicao',
                               'args': ['b',
                                        {'op': 'call_func',
                                         'args': ['fib',
                                                  [{'op': '-',
                                                    'args': [{'var': 'n'},
                                                             {'op': 'int',
                                                              'args': ['2']}]}]]}]},
                              {'op': '+',
                               'args': [{'var': 'a'}, {'var': 'b'}]}]}]},
          {'op': 'atribuicao',
           'args': ['fib5',
                    {'op': 'call_func',
                     'args': ['fib', [{'op': 'int', 'args': ['5']}]]}]}]}
<< 5
```

FIGURA 8 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-C-04.FCA'

D. Listas

Ao executar os exemplos pode-se verificar os seguintes resultados:

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-D-01.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'atribuicao',
           'args': ['lista',
                    {'op': 'elementos_array',
                     'args': [{'op': 'int', 'args': ['1']},
                              {'op': 'int', 'args': ['2']},
                              {'op': 'int', 'args': ['3']}]}]},
          {'op': 'escrever', 'args': [{'var': 'lista'}]},
          {'op': 'comentario', 'args': ['[1,2,3]']},
          {'op': 'atribuicao', 'args': ['vazia', []]}]}
[1, 2, 3]
<< []
```

FIGURA 9 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-D-01.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-D-02.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'funcao',
           'args': ['mais2',
                    [{'var': 'x'}],
                    {'op': '+',
                     'args': [{'var': 'x'}, {'op': 'int', 'args': ['2']}]}]},
          {'op': 'funcao',
           'args': ['soma',
                    [{'var': 'a'}, {'var': 'b'}],
                    {'op': '+', 'args': [{'var': 'a'}, {'var': 'b'}]}]},
          {'op': 'atribuicao',
           'args': ['lista1', {'op': 'map', 'args': ['mais2', []]}]},
          {'op': 'comentario', 'args': ['[]']},
          {'op': 'atribuicao',
           'args': ['lista2',
                    {'op': 'map',
                     'args': ['mais2',
                              {'op': 'elementos_array',
                               'args': [{'op': 'int', 'args': ['1']},
                                        {'op': 'int', 'args': ['2']},
                                        {'op': 'int', 'args': ['3']}]}]}]},
          {'op': 'comentario',
           'args': ['= [ mais2(1),mais2(2),mais2(3)] = [3,4,5]']},
          {'op': 'atribuicao',
           'args': ['lista3',
                    {'op': 'fold',
                     'args': ['soma',
                              {'op': 'elementos_array',
                               'args': [{'op': 'int', 'args': ['1']},
                                        {'op': 'int', 'args': ['2']},
                                        {'op': 'int', 'args': ['3']}]},
                              {'op': 'int', 'args': ['0']}]}]},
          {'op': 'comentario', 'args': ['= soma( 1, soma(2, soma ( 3, 0)))']},
          {'op': 'comentario', 'args': ['= soma( 1, soma(2, 3 ))']},
          {'op': 'comentario', 'args': ['= soma( 1, 5)']},
          {'op': 'comentario', 'args': ['= 6']}]}
<< None
```

FIGURA 10 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-D-02.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-D-02b.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'funcao',
           'args': ['mais2',
                    [{'var': 'x'}],
                    {'op': '+',
                     'args': [{'var': 'x'}, {'op': 'int', 'args': ['2']}]}]},
          {'op': 'funcao',
           'args': ['soma',
                    [{'var': 'a'}, {'var': 'b'}],
                    {'op': '+', 'args': [{'var': 'a'}, {'var': 'b'}]}]},
          {'op': 'atribuicao',
           'args': ['lista1', {'op': 'map', 'args': ['mais2', []]}]},
          {'op': 'atribuicao',
           'args': ['lista2',
                    {'op': 'map',
                     'args': ['mais2',
                              {'op': 'elementos_array',
                               'args': [{'op': 'int', 'args': ['1']},
                                        {'op': 'int', 'args': ['2']},
                                        {'op': 'int', 'args': ['3']}]}]}]},
          {'op': 'atribuicao',
           'args': ['lista3',
                    {'op': 'fold',
                     'args': ['soma',
                              {'op': 'elementos_array',
                               'args': [{'op': 'int', 'args': ['1']},
                                        {'op': 'int', 'args': ['2']},
                                        {'op': 'int', 'args': ['3']}]},
                              {'op': 'int', 'args': ['0']}]}]}]}
<< 6
```

FIGURA 11 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-D-02B.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-D-03.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'funcao',
           'args': ['somatorio',
                    [{'op': 'array_vazio', 'args': []}],
                    {'op': 'int', 'args': ['0']}]},
          {'op': 'funcao',
           'args': ['somatorio',
                    [{'op': 'var_id_array', 'args': ['x', 'xs']}],
                    {'op': '+',
                     'args': [{'var': 'x'},
                              {'op': 'call_func',
                               'args': ['somatorio', [{'var': 'xs'}]]}]}]},
          {'op': 'atribuicao',
           'args': ['resultado',
                    {'op': 'call_func',
                     'args': ['somatorio',
                              [{'op': 'elementos_array',
                                'args': [{'op': 'int', 'args': ['1']},
                                         {'op': 'int', 'args': ['2']},
                                         {'op': 'int', 'args': ['3']}]}]}]}]}
<< 6
```

FIGURA 12 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-D-03.FCA'

E. Outras instruções

Procedemos à implementação das instruções opcionais, incluindo o uso do condicional (SE) e os operadores lógicos: /\, \/, NEG. Essas funcionalidades adicionais ampliam a versatilidade do nosso analisador e permitem um maior grau de expressão na linguagem FCA.

Ao executar os exemplos pode-se verificar os seguintes resultados:

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-E-01.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'atribuicao',
           'args': ['valor1', {'op': 'int', 'args': ['10']}]},
          {'op': 'atribuicao',
           'args': ['valor2', {'op': 'int', 'args': ['20']}]},
          {'op': 'se',
           'args': [{'op': '<', 'args': [{'var': 'valor1'}, {'var': 'valor2'}]},
                    {'op': 'seq',
                     'args': [{'op': 'escrever',
                               'args': [{'op': 'string', 'args': ['Menor']}]}]},
                    {'op': 'senao_se',
                     'args': [{'op': '>',
                               'args': [{'var': 'valor1'}, {'var': 'valor2'}]},
                              {'op': 'seq',
                               'args': [{'op': 'escrever',
                                         'args': [{'op': 'string',
                                                   'args': ['Maior']}]}]}]},
                    {'op': 'senao',
                     'args': [{'op': 'seq',
                               'args': [{'op': 'escrever',
                                         'args': [{'op': 'string',
                                                   'args': ['Igual']}]}]}]}]}]}
Menor
<< None
```

FIGURA 13 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-E-01.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-E-02.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'atribuicao',
           'args': ['valor1', {'op': 'int', 'args': ['20']}]},
          {'op': 'atribuicao',
           'args': ['valor2', {'op': 'int', 'args': ['20']}]},
          {'op': 'se',
           'args': [{'op': '<', 'args': [{'var': 'valor1'}, {'var': 'valor2'}]},
                    {'op': 'seq',
                     'args': [{'op': 'escrever',
                               'args': [{'op': 'string', 'args': ['Menor']}]}]},
                    {'op': 'senao_se',
                     'args': [{'op': '<',
                               'args': [{'op': 'neg',
                                         'args': [{'var': 'valor1'}]},
                                        {'var': 'valor2'}]},
                              {'op': 'seq',
                               'args': [{'op': 'escrever',
                                         'args': [{'op': 'string',
                                                   'args': ['Maior ou '
                                                            'igual']}]}]}]}]}]}
Maior ou igual
<< None
```

FIGURA 14 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-E-02.FCA'

```
C:\Users\diogo\Desktop\projects\pl_tp_02>python main.py exemplos_enunciado/exemplo-E-03.fca
Grammar gerado:
{'op': 'seq',
 'args': [{'op': 'atribuicao',
           'args': ['valor1', {'op': 'int', 'args': ['111']}]},
          {'op': 'atribuicao',
           'args': ['valor2', {'op': 'int', 'args': ['20']}]},
          {'op': 'se',
           'args': [{'op': '||',
                     'args': [{'op': '<=',
                               'args': [{'var': 'valor1'}, {'var': 'valor2'}]},
                              {'op': '==',
                               'args': [{'op': 'int', 'args': ['1']},
                                        {'op': 'int', 'args': ['1']}]}]},
                    {'op': 'seq',
                     'args': [{'op': 'escrever',
                               'args': [{'op': 'string', 'args': ['Menor']}]}]},
                    {'op': 'senao',
                     'args': [{'op': 'seq',
                               'args': [{'op': 'escrever',
                                         'args': [{'op': 'string',
                                                   'args': ['Maior ou '
                                                            'igual']}]}]}]}]}]}
Menor
<< None
```

FIGURA 15 – OUTPUT DA EXECUÇÃO DO 'EXEMPLO-E-03.FCA'

Conclusão

Concluindo, abordou-se o desenvolvimento de uma aplicação para a Linguagem Funcional do Cávado e do Ave (FCA), desde a especificação da gramática até a geração de código executável.

A especificação da gramática exigiu um entendimento detalhado das estruturas da linguagem, assim como a construção do reconhecedor léxico e sintático utilizando os módulos `lex' e `yacc' da biblioteca `ply´ para garantir a correta categorização e combinação dos tokens.

O desenvolvimento do gerador de código, capaz de transformar a AST em código executável, foi desafiador. A integração de todas estas etapas alcançou um sistema sólido, capaz de avaliar e executar programas em FCA de forma precisa.

Este projeto não só consolidou o conhecimento teórico, como também foi uma boa experiência prática. O resultado atingiu os objetivos propostos, à parte da opcional de geração de código executável, e foi bastante satisfatório.