

Sistema de Linguagens Regulares

Relatório de Implementação e Utilização

1. Implementação

O sistema foi desenvolvido utilizando a linguagem de programação Java com o paradigma de orientação a objetos. A implementação segue o padrão de projeto MVC, como mostra o diagrama de classes a seguir:

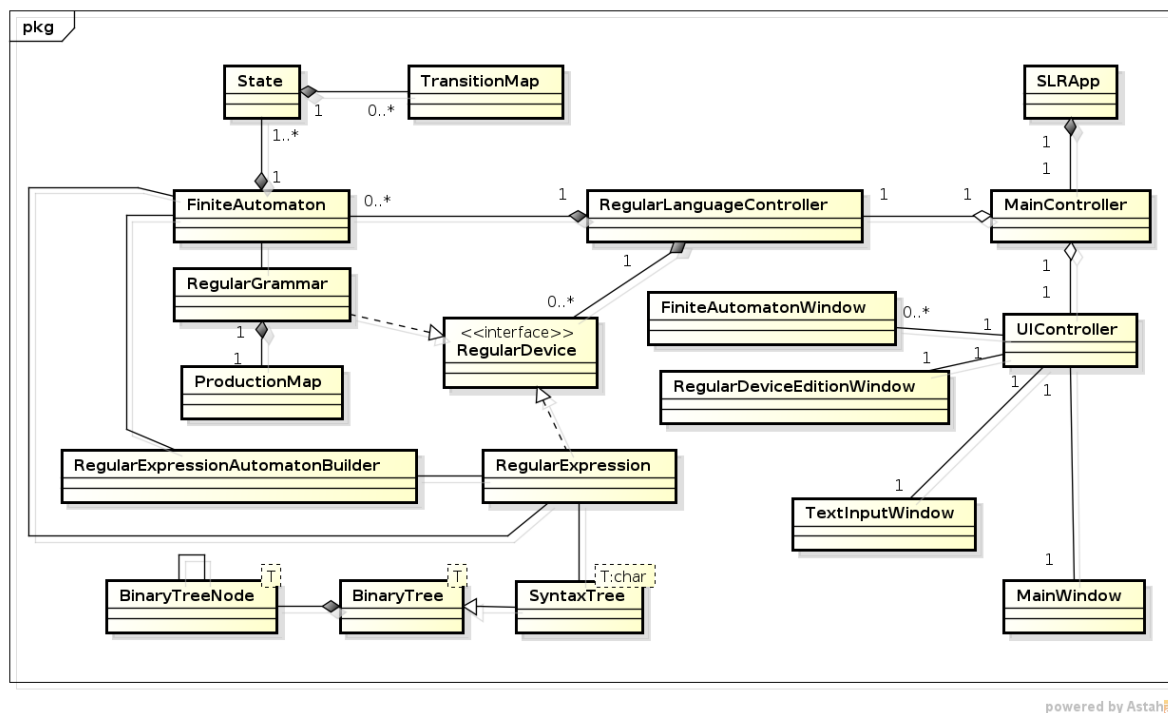


Figura 1: Diagrama de classes simplificado.

Algumas dificuldades foram encontradas na implementação dos algoritmos mais complexos. A correção de eventuais bugs ocorreu de maneira rápida, uma vez que testes unitários foram criados para validação dos algoritmos, através do *framework* JUnit.

2. Utilização do Sistema

A utilização do sistema é fácil e intuitiva. Inicialmente, a janela principal é exibida:

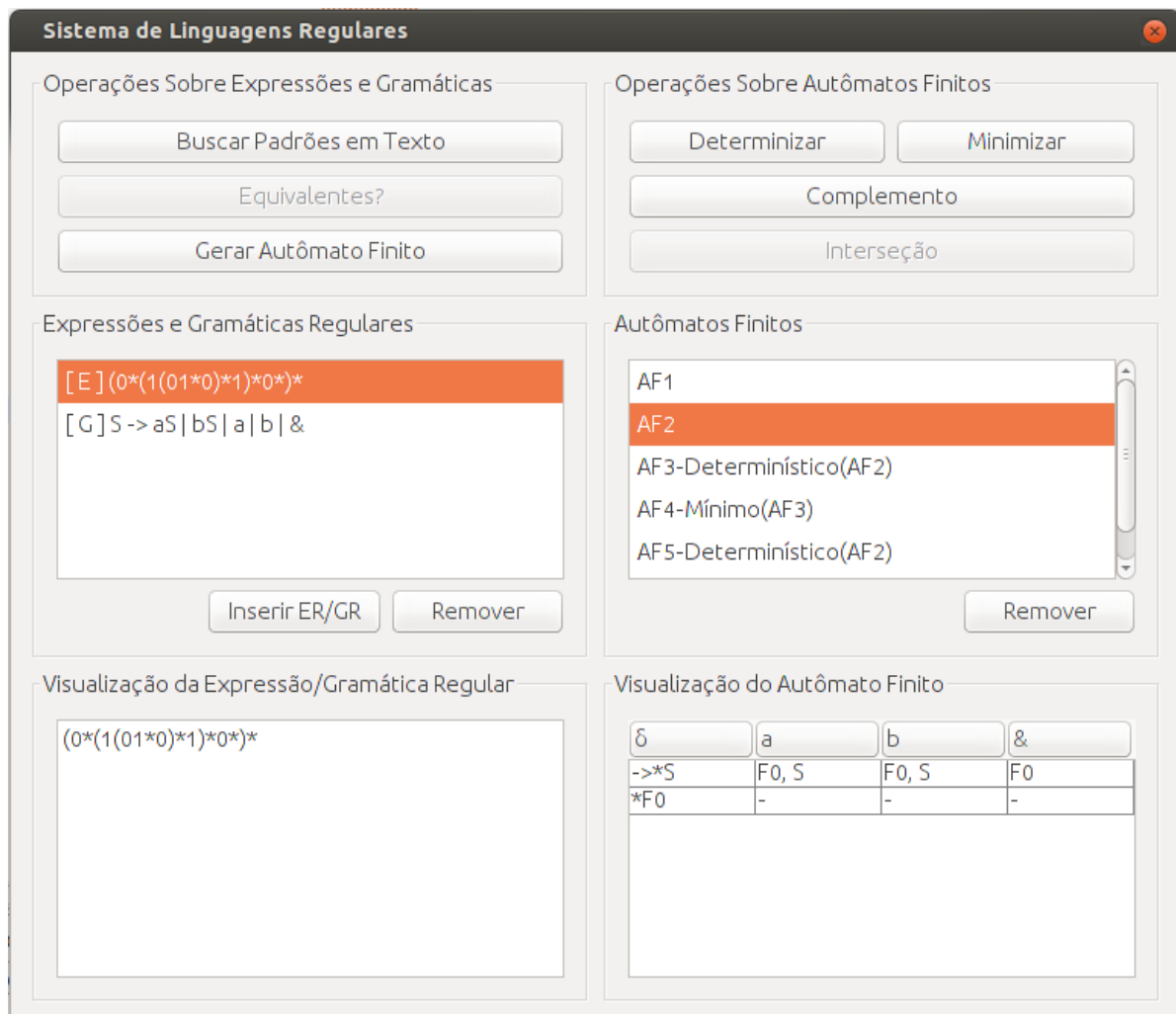


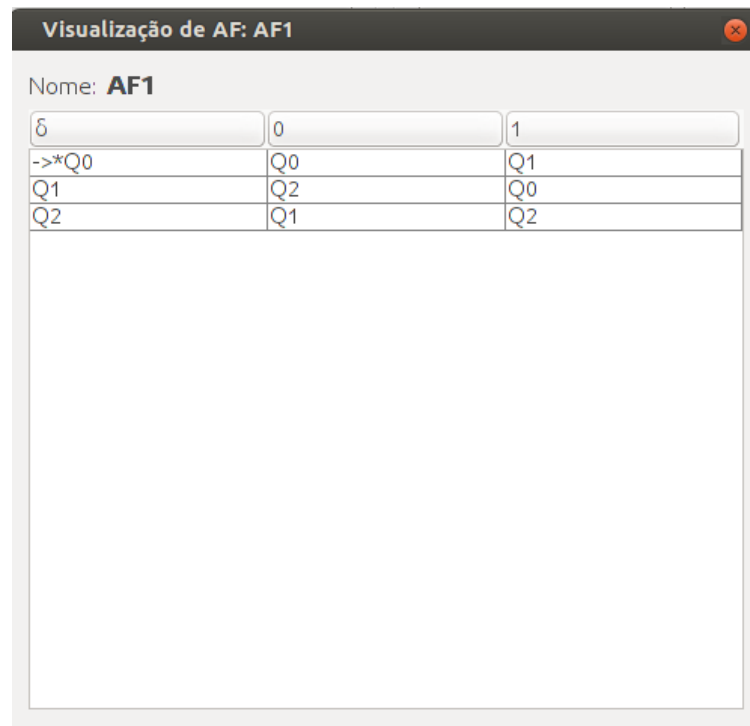
Figura 2: Janela principal do sistema.

Na parte esquerda da janela, é possível ao usuário inserir expressões e gramáticas regulares, tantas quanto forem necessárias, bem como removê-las e executar operações sobre elas. Os botões de operações são habilitados/desabilitados automaticamente, de acordo com o(s) dispositivo(s) selecionado(s). Ainda é possível editar uma expressão ou gramática através de um clique duplo sobre o dispositivo desejado.

Os autômatos finitos gerados nas operações sobre os dispositivos regulares são adicionados à lista de autômatos finitos na parte direita da janela. Sobre eles é possível a execução de diversas operações, as quais são habilitadas automaticamente na área de operações sobre autômatos finitos.

Na parte inferior da janela é possível visualizar a expressão/gramática regular e autômato selecionado. Para a melhor visualização de um autômato, basta dar um clique duplo

sobre o autômato desejado que uma nova janela exibirá a tabela de transições do autômato (Figura 3).



δ	0	1
->*Q0	Q0	Q1
Q1	Q2	Q0
Q2	Q1	Q2

Figura 3: Janela de visualização de um autômato finito.

Para inserir novos dispositivos regulares, basta clicar em Inserir ER/GR na janela principal e, em seguida, uma nova janela será exibida (Figura 4). Nela você pode escolher entre Expressão Regular e Gramática Regular e, então, descrever o dispositivo no campo apropriado (a expressão regular ou as produções da gramática regular). Na parte inferior da janela uma dica é exibida para inserir corretamente o dispositivo desejado.

Ao selecionar uma expressão regular na janela principal, é possível realizar uma busca de padrões denotados pela expressão em um texto qualquer. Para isso basta clicar em Buscar Padrões em Texto na janela principal. Uma nova janela será exibida para a inserção do texto a ser analisado (Figura 5). Após inserir o texto, pressione Ok para efetuar a busca e aguarde o retorno do sistema.

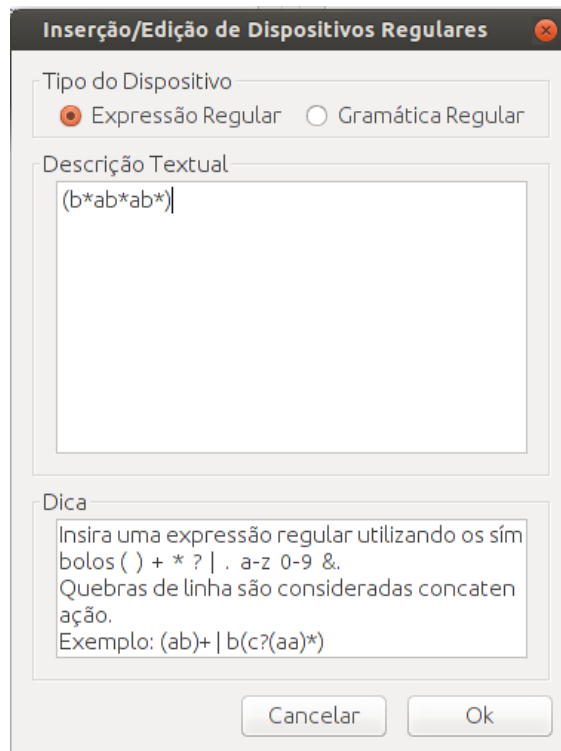


Figura 4: Janela de inserção/edição de dispositivos regulares.

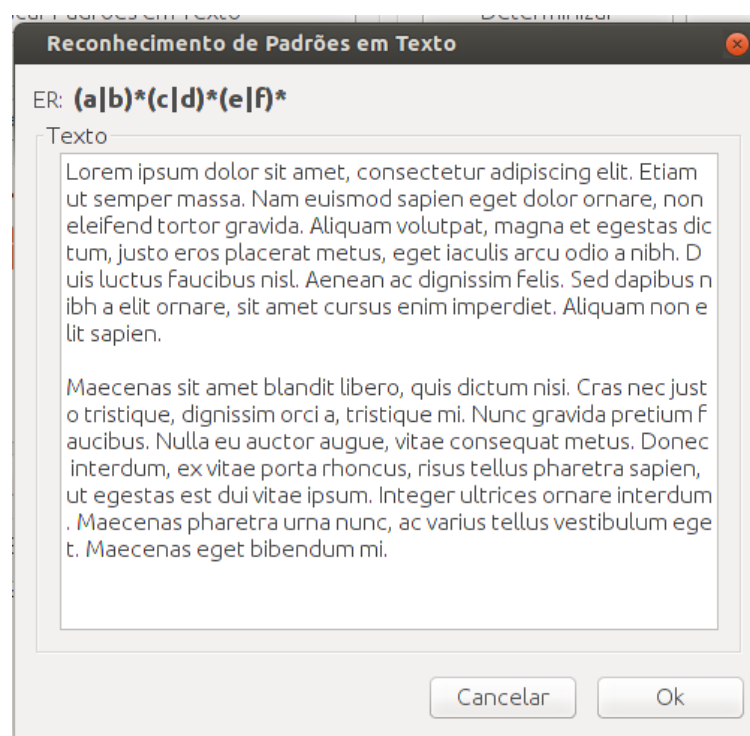


Figura 5: Janela de busca de ocorrência de padrões em texto.

3. Execução dos Testes

Project: LF-SLR

Tests: 50

Started: 50

Failures: 0

Errors: 0

Ignored: 0

Test Suit Name: slr.test.StateTest **Time:** 0.001s

Test Case Name: testTransitInvalid *Class:* slr.test.StateTest *Time:* 0.0s

Test Case Name: testGetName *Class:* slr.test.StateTest *Time:* 0.0s

Test Case Name: testEqualsObject *Class:* slr.test.StateTest *Time:* 0.0s

Test Case Name: testGetReachableStates *Class:* slr.test.StateTest *Time:* 0.001s

Test Case Name: testTransitSingle *Class:* slr.test.StateTest *Time:* 0.0s

Test Case Name: testTransitMultiple *Class:* slr.test.StateTest *Time:* 0.0s

Test Case Name: testIsFinal *Class:* slr.test.StateTest *Time:* 0.0s

Test Suit Name: slr.test.RegularGrammarTest **Time:** 0.019s

Test Case Name: testToString *Class:* slr.test.RegularGrammarTest *Time:* 0.007s

Test Case Name: testConstructor2 *Class:* slr.test.RegularGrammarTest *Time:* 0.001s

Test Case Name: testConstructor3 *Class:* slr.test.RegularGrammarTest *Time:* 0.002s

Test Case Name: testConstructor4 *Class:* slr.test.RegularGrammarTest *Time:* 0.001s

Test Case Name: testConstructor5 *Class:* slr.test.RegularGrammarTest *Time:* 0.001s

Test Case Name: testConstructor6 *Class:* slr.test.RegularGrammarTest *Time:* 0.001s

Test Case Name: testToFiniteAutomaton2 *Class:* slr.test.RegularGrammarTest *Time:* 0.002s

Test Case Name: testToFiniteAutomaton *Class:* slr.test.RegularGrammarTest *Time:* 0.002s

Test Case Name: testConstructor *Class:* slr.test.RegularGrammarTest *Time:* 0.002s

Test Suit Name: slr.test.FiniteAutomatonTest **Time:** 0.082s

Test Case Name: testIsComplete *Class:* slr.test.FiniteAutomatonTest *Time:* 0.0s

Test Case Name: testRecognizeNondeterministicEpsilon2 *Class:* slr.test.FiniteAutomatonTest
Time: 0.001s

Test Case Name: testDeterminize2 *Class:* slr.test.FiniteAutomatonTest *Time:* 0.001s

Test Case Name: testDeterminize3 *Class:* slr.test.FiniteAutomatonTest *Time:* 0.0s

Test Case Name: testDeterminize4 *Class:* slr.test.FiniteAutomatonTest *Time:* 0.001s

Test Case Name: testComplement *Class:* slr.test.FiniteAutomatonTest *Time:* 0.001s

Test Case Name: testUnion *Class:* slr.test.FiniteAutomatonTest *Time:* 0.001s

Test Case Name: testRecognizeDeterministic *Class:* slr.test.FiniteAutomatonTest *Time:* 0.0s

Test Case Name: testGetAlphabet *Class:* slr.test.FiniteAutomatonTest *Time:* 0.001s

Test Case Name: testMinimize *Class:* slr.test.FiniteAutomatonTest *Time:* 0.0s

Test Case Name: testDeterminize *Class:* slr.test.FiniteAutomatonTest *Time:* 0.001s
Test Case Name: testRecognizeNondeterministicEpsilon *Class:* slr.test.FiniteAutomatonTest
Time: 0.001s
Test Case Name: testIsDeterministic2 *Class:* slr.test.FiniteAutomatonTest *Time:* 0.0s
Test Case Name: testIsDeterministic3 *Class:* slr.test.FiniteAutomatonTest *Time:* 0.0s
Test Case Name: testComplete *Class:* slr.test.FiniteAutomatonTest *Time:* 0.001s
Test Case Name: testContains *Class:* slr.test.FiniteAutomatonTest *Time:* 0.014s
Test Case Name: testRecognizeNondeterministic *Class:* slr.test.FiniteAutomatonTest *Time:*
0.001s
Test Case Name: testContains2 *Class:* slr.test.FiniteAutomatonTest *Time:* 0.024s
Test Case Name: testContains3 *Class:* slr.test.FiniteAutomatonTest *Time:* 0.024s
Test Case Name: testIsDeterministic *Class:* slr.test.FiniteAutomatonTest *Time:* 0.0s
Test Case Name: testIsMinimal *Class:* slr.test.FiniteAutomatonTest *Time:* 0.0s
Test Case Name: testIntersection *Class:* slr.test.FiniteAutomatonTest *Time:* 0.003s
Test Case Name: testIsEmpty *Class:* slr.test.FiniteAutomatonTest *Time:* 0.0s
Test Case Name: testIsEquivalentTo *Class:* slr.test.FiniteAutomatonTest *Time:* 0.007s

Test Suit Name: slr.test.RegularExpressionTest **Time:** 0.007s

Test Case Name: testStandardize *Class:* slr.test.RegularExpressionTest *Time:* 0.002s
Test Case Name: testConstructor2 *Class:* slr.test.RegularExpressionTest *Time:* 0.0s
Test Case Name: testConstructor3 *Class:* slr.test.RegularExpressionTest *Time:* 0.0s
Test Case Name: testToFiniteAutomaton2 *Class:* slr.test.RegularExpressionTest *Time:*
0.004s
Test Case Name: testToFiniteAutomaton3 *Class:* slr.test.RegularExpressionTest *Time:* 0.0s
Test Case Name: testToFiniteAutomaton4 *Class:* slr.test.RegularExpressionTest *Time:* 0.0s
Test Case Name: testToFiniteAutomaton5 *Class:* slr.test.RegularExpressionTest *Time:*
0.001s
Test Case Name: testToFiniteAutomaton *Class:* slr.test.RegularExpressionTest *Time:* 0.0s
Test Case Name: testGetSyntaxTree *Class:* slr.test.RegularExpressionTest *Time:* 0.0s
Test Case Name: testConstructor *Class:* slr.test.RegularExpressionTest *Time:* 0.0s