Trabalho I: Algoritmos para Manipulação de Linguagens Regulares

Bruno Gilmar Honnef[1], **Pedro Alexandre Barradas da Côrte**[1], **Ricardo do Nascimento Boing**[1]

[1] Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Campus Reitor João David Ferreira Lima, 88.040-900 – Florianópolis – SC – Brasil

brunohonnef@gmail.com pedroabcorte@gmail.com ricardoboing.ufsc@gmail.com

11 de Outubro de 2018

1. Informações sobre os fontes

O trabalho foi desenvolvido na linguagem de programação Java, com o auxilio da IDE Eclipse. Ao todo, existem 29 classes de interface, 5 classes de suporte geral, 7 classes para tratamento de conjuntos, e 12 classes de implementação dos algoritmos. As 12 classes de implementação são divididas em: 4 classes de gramáticas, 4 classes de expressões e 4 classes de autômatos.

Em relação ao tratamento de conjuntos, o objetivo é facilitar a criação de Estados, Alfabetos, Produções, Símbolos Terminais e Não Terminais, etc, pois com base no método equals de cada classe é feita uma verificação se determinado objeto já existe no conjunto, e o realiza (ou não) uma tarefa pré-definida para casos de equivalência.

2. Guia de usuário

2.1. Expressão Regular

2.1.1. Criar Expressão Regular (Expressão $>$ Novo)

Uma tela será carregada com dois campos: um campo (preenchido e desabilitado) contendo o nome da nova expressão regular, e outro para digitar a expressão regular desejada.

- **Salvar**: Salva a expressão regular na memória;
- **Limpar**: Remove todo conteúdo digitado;
- **Selecionar arquivo**: Lê uma expressão regular salva em disco.

2.1.2. Editar Expressão Regular (Expressão $>$ Editar)

Uma tela será carregada contendo um menu com todas as expressões regulares salvas em memória. Ao selecionar uma das expressões regulares são abertos um campo de nome e outro contendo a expressão, ambos desabilitados.

- **Gerar AF**: Cria um autômato finito a partir da expressão regular;
- **Salvar em Disco**: Salva a expressão regular em um arquivo $.er$;
- **Editar**: Habilita o campo para edição da expressão regular. Os botões Editar, Salvar em Disco e Remover são removidos da tela, e são adicionados os botões Salvar e Cancelar;

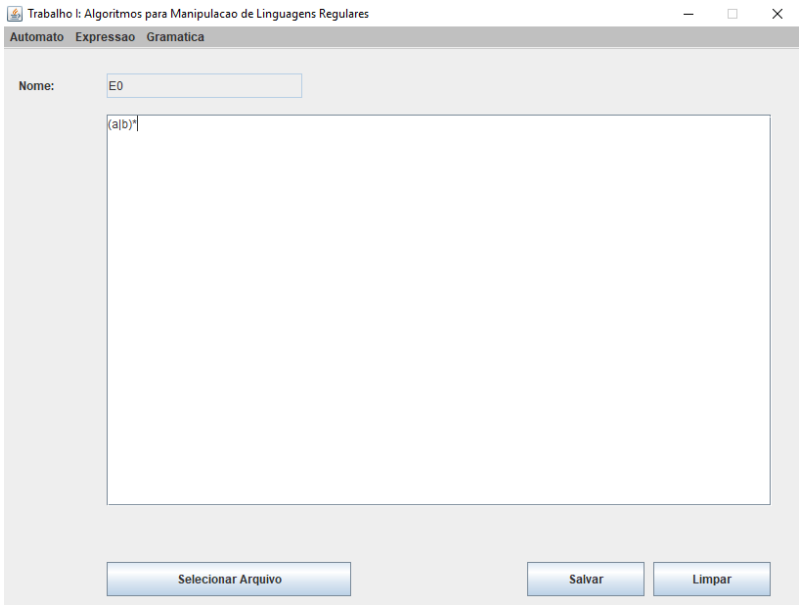

Figura 1. Tela de criação de Expressões Regulares

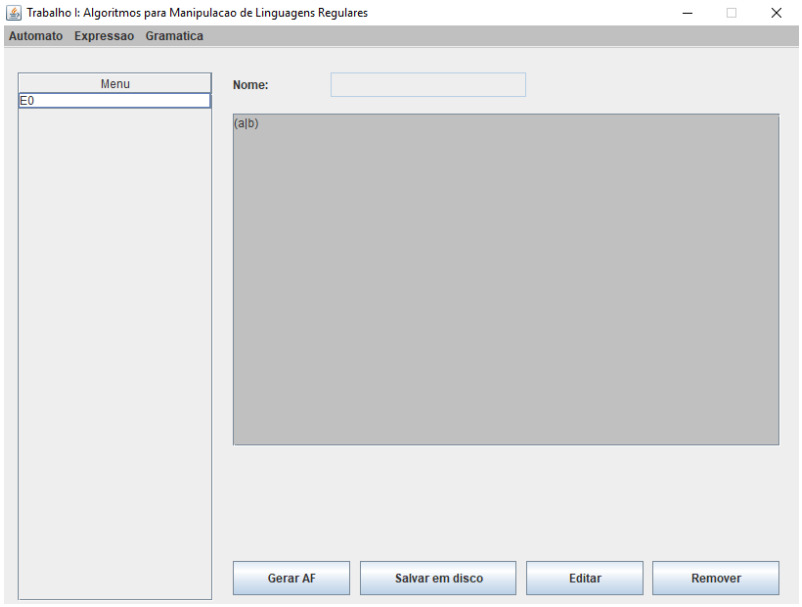

Figura 2. Tela de edição de Expressões Regulares

- **Salvar**: Salva a expressão na memória, porém não atualiza arquivos em disco. Para salvar em disco, deve-se clicar no botão correspondente, e um novo arquivo será gerado. Remove os botões Salvar e Cancelar da tela e adiciona os botões Editar, Salvar em Disco e Remover.

- **Remover**: Remove a expressão regular da memória do programa.

2.2. Gramática Regular

2.2.1. Criar Gramática Regular (Gramática > Novo)

Uma tela será carregada com dois campos: um campo (preenchido e desabilitado) contendo o nome da nova gramática regular, e outro para digitar a gramática regular desejada.

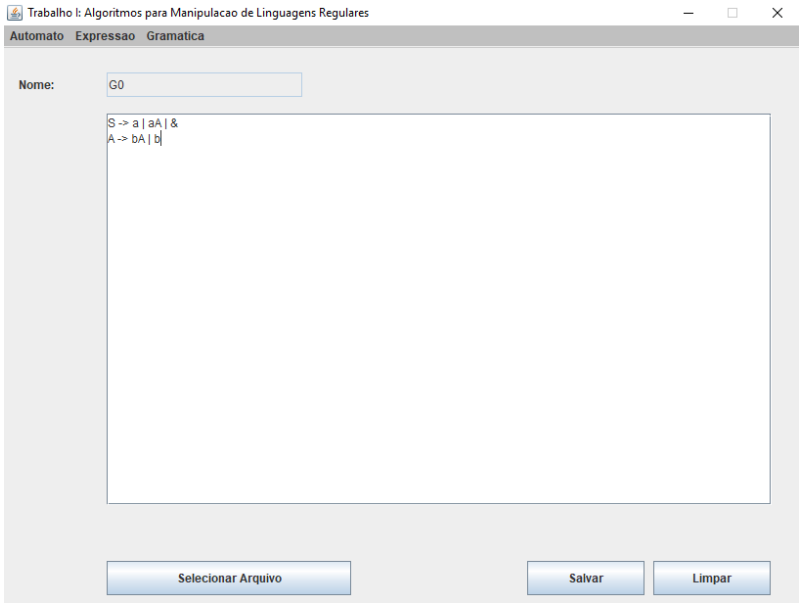

Figura 3. Tela de criação de Gramáticas Regulares

- **Salvar**: Salva a gramática regular na memória;
- **Limpar**: Remove todo conteúdo digitado;
- **Selecionar arquivo**: Lê uma gramática regular salva em disco.

2.2.2. Editar Gramática Regular (Gramática > Editar)

Uma tela será carregada contendo um menu com todas as gramáticas regulares salvas em memória. Ao selecionar uma das gramáticas regulares são abertos um campo de nome e outro contendo a gramática, ambos desabilitados.

- **Gerar AF**: Cria um autômato finito;
- **Salvar em Disco**: Salva a gramática regular em um arquivo $.gr$;
- **Editar**: Habilita o campo para edição da gramática regular. Os botões Editar, Salvar em Disco e Remover são removidos da tela, e são adicionados os botões Salvar e Cancelar;
- **Salvar**: Salva a gramática na memória, porém não atualiza arquivos em disco. Para salvar em disco, deve-se clicar no botão correspondente, e um novo arquivo será gerado. Remove os botões Salvar e Cancelar da tela e adiciona os botões Editar, Salvar em Disco e Remover.
- **Remover**: Remove a gramática regular da memória do programa.

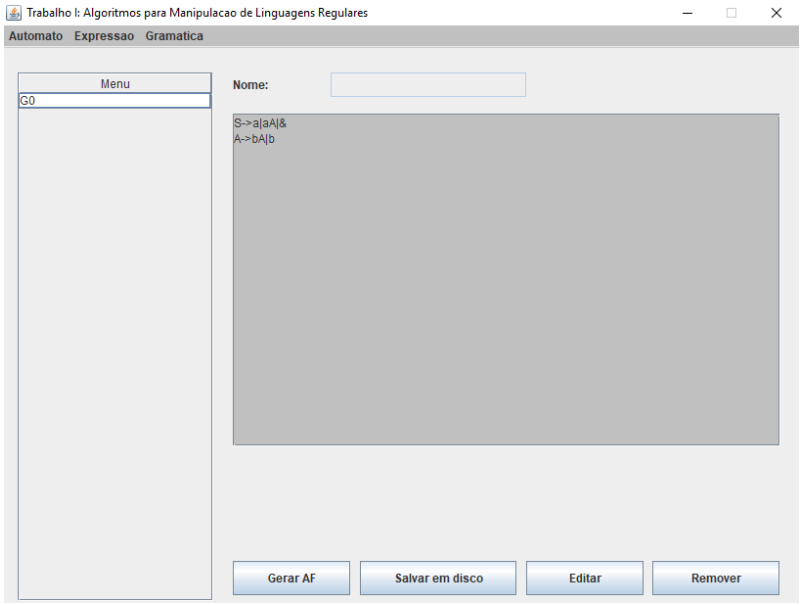

Figura 4. Tela de edição de Gramáticas Regulares

2.3. Automato Finito

2.3.1. Criar Autômato Finito (Autômato $>$ Novo)

Uma tela será carregada com um campo e uma tabela: o campo é preenchido, desabilitado, e contem o nome do novo autômato finito. Já a tabela é inicializada contendo duas linhas e três colunas. As colunas ”Inicial”e ”Final”simbolizam, respectivamente, se o estado é inicial ou final. Por padrão o primeiro estado da tabela é o inicial, portanto a coluna ”Inicial”sempre estará desabilitada. Já a coluna ”Final”deve ficar vazia em caso do estado não ser final, ou preenchido com $*$ caso ele seja final. A coluna ”Estado”, por sua vez, conterá o nome do estado. Por fim, a coluna ”Transição”conterá dois tipos de valores. Na primeira linha ela deve ser preenchida com os símbolos de entrada, e nas seguintes deverá ser preenchida com o nome do estado destino da transição. Ex: Para $\delta(q0, a) = q1$ deverá ser preenchida a coluna ”Estado”com o valor $q0$, na mesma linha, na coluna ”Transição”deverá ser preenchido com $q1$, e na primeira linha, na coluna ”Transição”deverá ser preenchida com a. Um exemplo pode ser visto na Fig. 5. Após ser criado, não se garante que os estados continuem com os mesmos símbolos inseridos durante a sua criação. Além disso, antes de clicar no botão salvar, caso uma célula tenha acabado de ser escrita (ou seja, mais nada foi feito a partir do mouse ou teclado) deve-se pressionar a tela ”Enter”, pois caso contrário, por bug’s da linguagem de programação, a célula não seria atualizada.

- $+$ **Estado**: Adiciona uma linha adicional na tabela para criação de um novo estado;
- $-$ **Estado**: Remove a última linha da tabela, ou seja, remove o último estado;
- $+$ **Símbolo**: Cria uma coluna adicional para criação de novas transições, com uma determinada entrada;
- $-$ **Símbolo**: Remove a última coluna da tabela;
- **Selecionar arquivo**: Preenche a tabela com valores de um autômato salvo em arquivo $.af$;

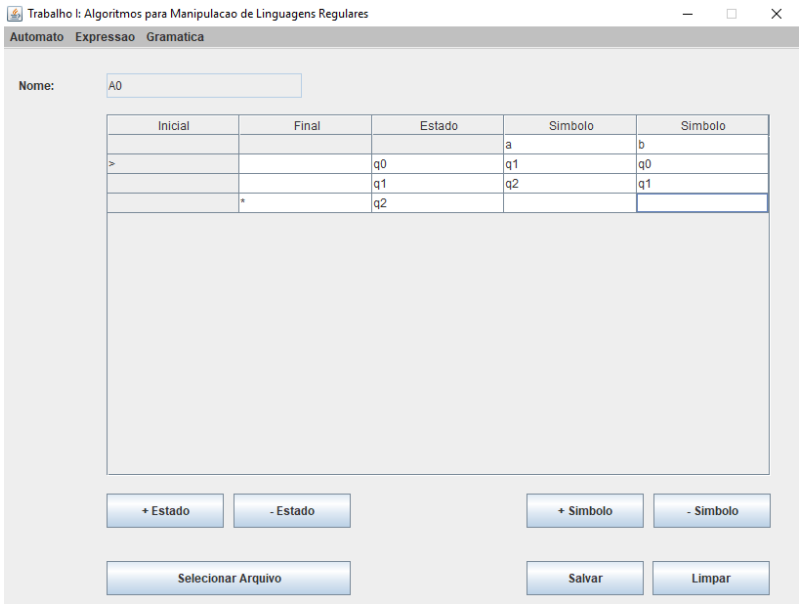

Figura 5. Tela de criação de autômatos finitos.

- **Salvar**: Salva o novo autômato na memória do programa;
- **Limpar**: Remove todas alterações na tabela;

2.3.2. Editar Autômato Finito (Autômato > Editar)

Possui grande parte das características da janela de criação de autômatos, exceto por não possuir os botões de seleção de arquivo e limpar.

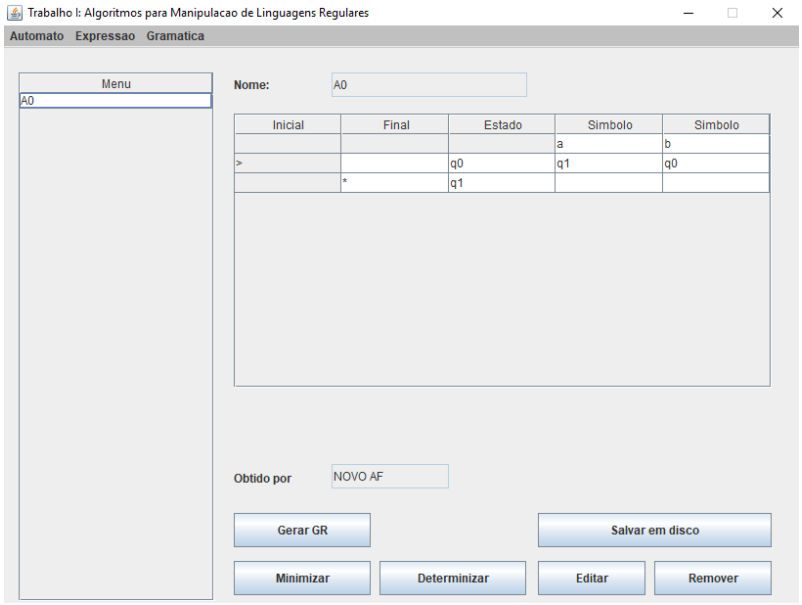

Figura 6. Tela de edição de Autômatos Finitos.

- **Gerar GR**: Cria uma gramática regular a partir do autômato;

- **Minimizar**: Cria um novo autômato minimizado e seus intermediários;
- **Determinizar**: Cria um novo autômato determinizado;
- **Editar**: Habilita a edição do autômato e desabilita os botões Gerar GR, Minimizar, Determinizar e Salvar em disco. No lugar dos botões Editar e Cancelar surgem Salvar e Cancelar. Os botões + Estado, − Estado, + Símbolo, − Símbolo também ficam visiveis na tela;
- **Remover**: Remove o autômato da memória, mas não de arquivos;
- **Salvar em disco**: Salva o autômato em um arquivo;
- **Cancelar**: Desabilita a edição do autômato e restaura os botões;
- **Salvar**: Salva o autômato em memória (mas não em arquivo) e realiza as todas operações do botão Cancelar;
- + **Estado**: Adiciona uma linha adicional na tabela para criação de um novo estado;
- − **Estado**: Remove a última linha da tabela, ou seja, remove o último estado;
- + **Símbolo**: Cria uma coluna adicional para criação de novas transições, com uma determinada entrada;
- − **Símbolo**: Remove a última coluna da tabela;

2.3.3. Operações com Autômatos Finitos (Autômato > Operações)

Uma tela será carregada contendo dois combox e três botões. Ao selecionar um autômato em um dos combox será mostrado uma tabela contendo o autômato correspondente. Para realizar as operações de união e intersecção é preciso que os dois combox contenham algum autômato. Em alguns casos os combox não se atualizam sozinhos, por algum eventual bug que não deu tempo de concertar, portanto é preciso, em alguns casos, que algum autômato seja selecionado para que a lista de opções seja atualizada.

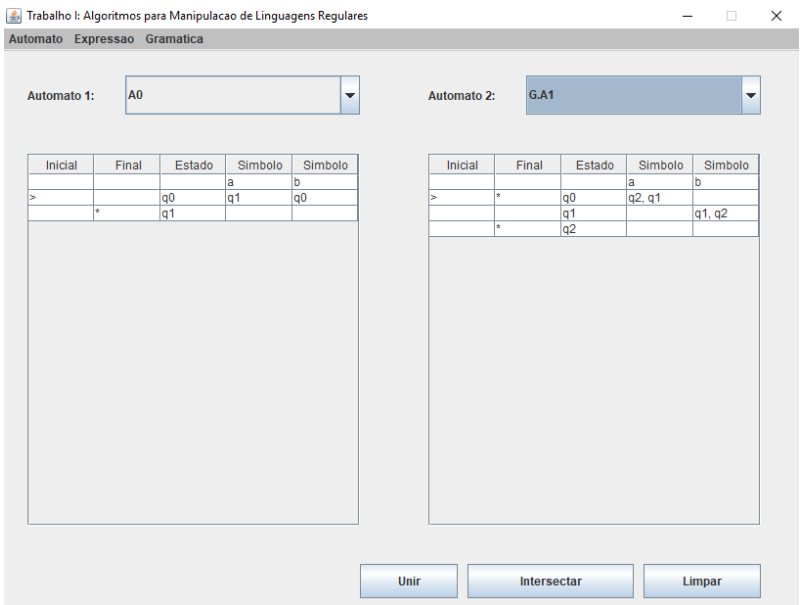

Figura 7. Tela de edição de Gramáticas Regulares.

- **Unir**: Cria um novo autômato que é a união dos dois autômatos selecionados;

- **Intersecção**: Cria um novo autômato que é a intersecção dos dois autômatos selecionados e adiciona-os na lista de autômatos. Autômatos intermediários também são salvos na lista;
- **Limpar**: Remove as seleções dos combox e esconde as tabelas.

3. Algoritmo DeSimone

3.1. Árvore Sintática

O algoritmo se divide em duas etapas: criar a árvore sintática e gerar os estados. Para gerar a árvore sintática devem ser considerados:

1. A árvore sintática é uma arvore binária com costura;
2. Os nós folhas devem obrigatoriamente corresponder a um símbolo de entrada;
3. Todos símbolos de entrada devem ser um nó folha.
4. Todo nó folha possui uma costura;
5. Existem 4 símbolos reservados: fechamento ($*$), interrogação ($?$), "ou"($|$) e concatenação ($.$);
6. Os nós $*$ e $?$ devem conter um filho esquerdo e uma costura pelo filho direito;
7. Os nós $|$ e $.$ possuem dois filhos, portanto não possuem costura;
8. A ordem de precedência, para criação da árvore, corresponde ao nível da árvore em que um determinado símbolo (dentre os quatro reservados) deve ficar. A ordem de precedência, do menor para o maior, é mostrada abaixo:
   (a) ou ($|$)
   (b) concatenação ($.$)
   (c) fechamento ($*$) ou interrogação ($?$)
9. Caso λ: a última costura da árvore é feita para um quinto nó reservado: o nó λ.

3.2. Construção dos estados

O primeiro passo corresponde a nomeação dos nós folhas, com nome único. A forma mais intuitiva é nomear o nó com um número inteiro (único) seguido do seu símbolo, como mostrado na Fig. 9.

O segundo passo é criar o estado inicial, e para isso é preciso seguir algumas regras (ou rotinas), representadas pelas Fig. 10 (rotina de subida) e Fig. 11 (rotina de descida) e descritas abaixo:

- **Rotina de descida**
  - **Fechamento ($*$)**: desce ao filho esquerdo e no retorno da recursão realiza a rotina de subida ao nó costura;
  - **Interrogação ($?$)**: possui a mesma rotina do fechamento;
  - **"Ou"($|$)**: desce ao filho esquerdo e no retorno da recursão desce ao filho direito;
  - **Concatenação ($.$)**: desce somente ao filho esquerdo.
- **Rotina de subida**
  - **Fechamento ($*$)**: desce ao filho esquerdo e no retorno da recursão realiza a rotina de subida ao nó costura;
  - **Interrogação ($?$)**: sobe ao nó costura;

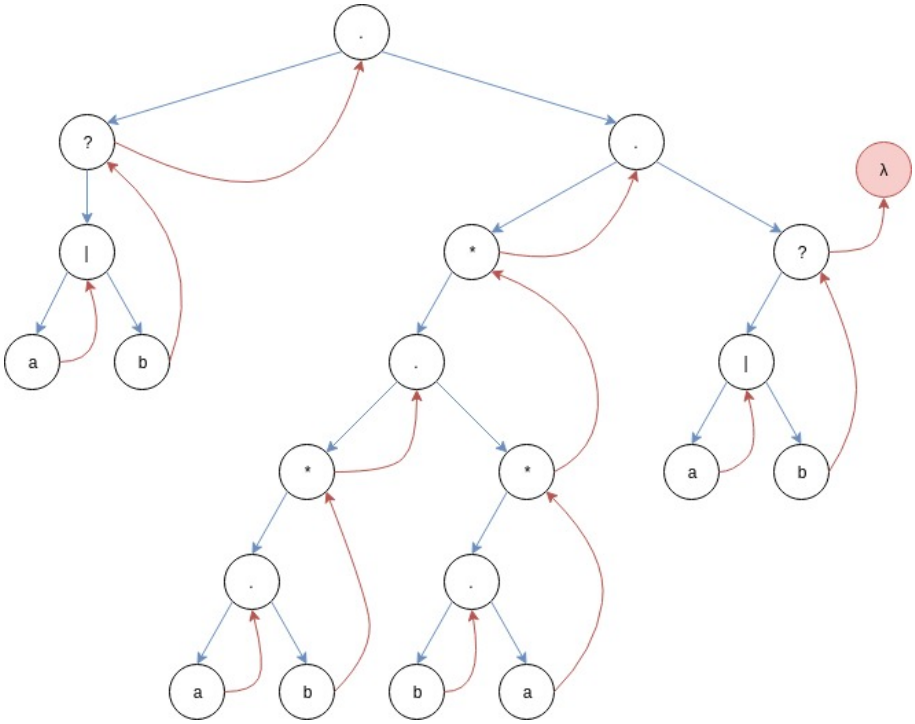

Figura 8. Árvore binária costurada gerada pela expressão $(a|b)^?((ab)^*(ba)^*)^*(a|b)^?$.

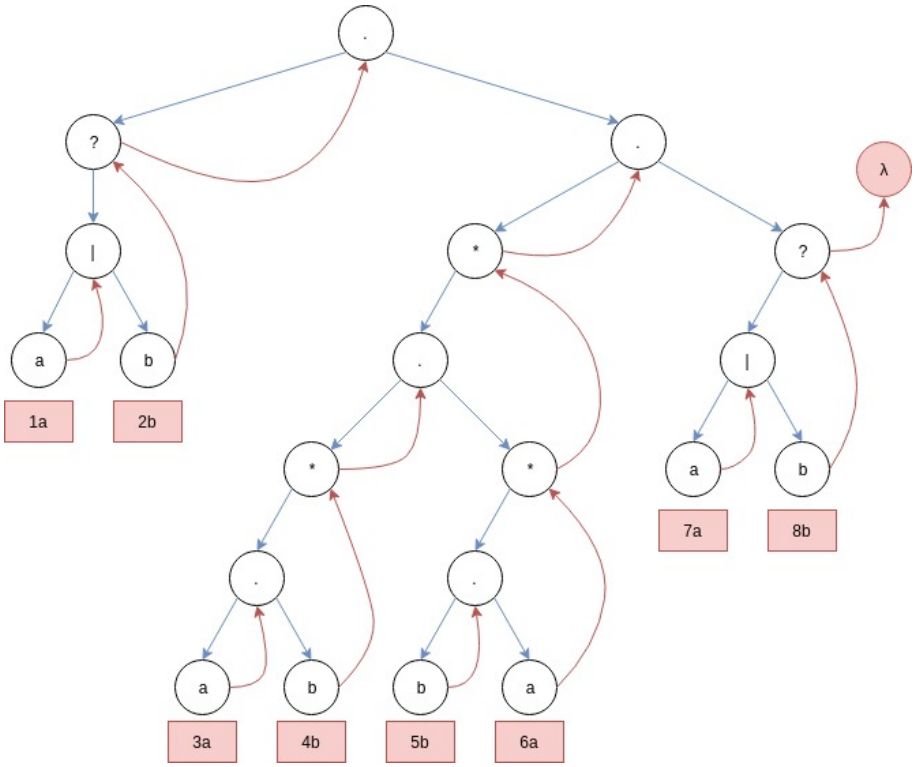

Figura 9. Árvore binária costurada gerada pela expressão $(a|b)^?((ab)^*(ba)^*)^*(a|b)^?$ e com representação dos nós folhas.

– **"Ou"**(|): desce até o último filho direito e sobe pela costura. Na árvore da Fig. 9, o nó |, pai de $1a$ e $2b$, desceria até o nó $2b$, que é o último, e

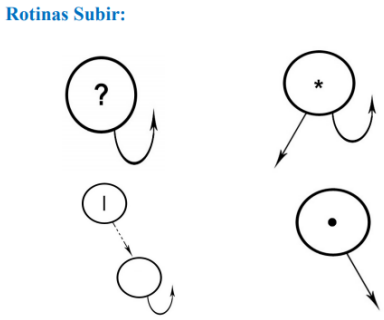

Figura 10. Representação das rotinas subir.

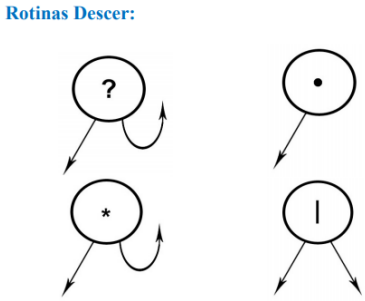

Figura 11. Representação das rotinas descer.

realizaria a subida ao nó ?;
- **Concatenação** (.): desce ao filho direito;
- **Nó folha (não representado na figura)**: sobe ao nó costura;

Um nó visitado através de uma rotina de descida terá de realizar sua rotina de descida, e um nó alcançado pela rotina de subida deverá realizar sua rotina de subida. Ao alcançar um nó folha ou nó λ, esse nó deve ser adicionado em uma lista de composições do estado em questão. Por exemplo, se o no | do lado esquerdo da Fig. 9 for visitado pela descida do nó ?, então o nó | realizará sua rotina de descida, que é descer ao nó a e ao nó b. Porém se | for alcançado pela subida do nó a, então ele realizará sua rotina de descida, que será descer até o último filho (b) e subir pela costura desse filho, que no caso será visitar o nó ?.

Um nó que recursivamente alcançar a si próprio deverá interromper a recursão por aquele caminho, pois todos os nós folhas possíveis já foram adicionados na lista de composições. Se o λ estiver contido entre as composições do estado então o estado será de aceitação.

A formação do estado inicial q_0, portanto, será realizada descendo o nó raiz. A Tab. 1 mostra o resultado para a árvore da Fig. 9.

O terceiro passo é a criação dos demais estados. Com base nas combinações e transições de todos os estados já criados, são gerados novos estados a partir de cada transição. Para a entrada a, no estado inicial (q_0), por exemplo, será criado o estado q_1, e para entrada b será criado o estado q_2. Deve-se, portanto, realizar a subida de todos os nós a: $1a$, $3a$ e $7a$ para criar a lista de composições do estado q_1. Para a entrada b,

Tabela 1. Composições do estado q_0.

	δ	a	b	composições
$> *$	q_0			1a, 2b, 3a, 5b, 7a, 8b, λ

deve-se realizar a subida de todos os nós b: $2b$, $5b$, $8b$ para criar a lista de composições do estado q_2. Se no estado q_0 não houvesse nenhum nó b (não existisse $2b$, $5b$, $8b$ na lista de composições) então não existiria nenhuma transição pela entrada b, e assim vale para qualquer outro símbolo de entrada. O resultado é mostrado na Tab. 2.

Tabela 2. Estado q_1 e q_2 gerados pelo estado q_0.

	δ	a	b	composições
$> *$	q_0	q_1	q_2	1a, 2b, 3a, 5b, 7a, 8b, λ
$*$	q_1			
$*$	q_2			

Um estado é equivalente a outro se ambos forem gerados pelas mesmas composições ou se possuírem a mesma lista de composições. Quando isso ocorre é realizado o descarte de um dos dois estados, como mostrado na Tab. 3. As Tab. 4 e Tab. 5 mostram o automato final.

Tabela 3. Estado q_4 equivalente ao estado q_2.

	δ	a	b	composições
$> *$	q_0	q_1	q_2	1a, 2b, 3a, 5b, 7a, 8b, λ
$*$	q_1	q_3	~~q_4~~ q_2	3a, 5b, 7a, 8b, λ , 4b
$*$	q_2			
$*$	q_3			
$*$	~~q_4~~			~~6a, λ , 3a, 5b, 7a, 8b~~

Tabela 4. Autômato Finito final (sujo).

	δ	a	b	composições
$> *$	q_0	q_1	q_2	1a, 2b, 3a, 5b, 7a, 8b, λ
$*$	q_1	q_3	~~q_4~~ q_2	3a, 5b, 7a, 8b, λ , 4b
$*$	q_2	~~q_5~~ q_1	q_6	3a, 5b, 7a, 8b, λ , 6a
$*$	q_3	-	q_7	4b, λ
$*$	~~q_4~~			~~6a, λ , 3a, 5b, 7a, 8b~~
$*$	~~q_5~~			~~4b, λ , 5b, 3a, 7a, 8b~~
$*$	q_6	~~q_8~~ q_7	-	6a, λ
$*$	q_7	~~q_9~~ q_3	~~q_{10}~~ q_6	3a, 5b, 7a, 8b, λ
$*$	~~q_8~~			~~5b, 3a, 7a, 8b, λ~~
$*$	~~q_9~~			~~4b, λ~~
$*$	~~q_{10}~~			~~6a, λ~~

Tabela 5. Autômato Finito final (limpo).

	δ	a	b	composições
$> *$	q_0	q_1	q_2	1a, 2b, 3a, 5b, 7a, 8b, λ
$*$	q_1	q_3	q_2	3a, 5b, 7a, 8b, λ , 4b
$*$	q_2	q_1	q_6	3a, 5b, 7a, 8b, λ , 6a
$*$	q_3	-	q_7	4b, λ
$*$	q_6	q_7	-	6a, λ
$*$	q_7	q_3	q_6	3a, 5b, 7a, 8b, λ