DEPARTAMENTO DE INFORMATICA E ESTATÍSCA
UNIVERSIDADE FEDERAL DE SANTA CATARINA

Trabalho I Linguagens Formais e Compiladores

Relatório de Implementação

Otto Menegasso Pires, Thales A. Zirbel Hubner

Sumário

1 Introdução **1**

2 Implementação **1**
2.1 União 1
2.2 Complemento 1
2.3 Intersecção 1
2.4 Diferença 1
2.5 Minimização 1
2.6 Determinização 2
2.6.1 Epsilon-Transição 2
2.7 Conversão de Expressões Regulares para Autômatos Finitos 2
2.8 Equivalência de Expressões Regulares 3

3 Considerações Finais **3**
3.1 Interface Gráfica 3
3.2 Instruções de Uso 3

---

20 de junho de 2017

1 Introdução

Esse trabalho foi feito para a disciplina INE5421-Linguagens Formais e Compiladores. O Objetivo desse trabalho era criar um programa capaz de representar e editar Expressões Regulares e Autômatos Finitos. Além disso o programa também deveria possuir as seguintes funcionalidades:

- Converter Expressões Regulares em Autômatos Finitos
- Determinizar e Minimizar Autômatos Finitos
- Operações de União, Complemento, Intersecção e Diferença de Linguagens Regulares via Autômatos Finitos
- Verificar a equivalência de Expressões Regulares

2 Implementação

A Linguagem escolhida pelo grupo foi Python, e para criar a interface gráfica foi usado a API PyQT5. Os algoritmos implementados nas operações foram apresentados em sala de aula.

2.1 União

A União entre Autômatos Finitos é feita a partir da criação de um novo estado inicial que possui transições não-determinísticas para os estados iniciais dos autômatos individuais. Após a geração desse AFND é efetuada a operação de determinização (explicada mais adiante).

2.2 Complemento

A operação de Complemento gera um novo autômato cujos estados finais sejam iguais aos estados não-finais do autômato de entrada e os estados não-finais sejam iguais aos estados finais do autômato de entrada. Transições e o estado inicial são preservados.

2.3 Intersecção

A Intersecção foi implementada chamando as operações de União e Complemento, de maneira que dado dois autômatos de entrada, o novo autômato gerado é o complemento da união dos complementos dos autômatos de entrada.

2.4 Diferença

A Diferença foi implementada chamando as operações de Intersecção e Complemento, de maneira que dado dois autômatos de entrada, o novo autômato gerado é a intersecção entre o primeiro autômato e o complemento do segundo.

2.5 Minimização

Baseando-se no algoritmo demonstrado em sala, o programa minimiza um autômato de entrada em três etapas. Primeiro, os estados inacessíveis são removidos. Segundo, os estados mortos são removidos. Terceiro, os estados equivalentes são unificados.

1. Os estados acessíveis são registrados. Partindo do estado inicial, todos os estados alcansáveis a partir das transições do autômato, direta ou indiretamente, são marcados. Aqueles que não foram encontrados são removidos.
2. Partindo agora dos estados finais, seguindo todas as transições que atingem um estado final, diretamente ou indiretamente, os estados que possuem essas transições são marcados. Aqueles que não foram marcados são estados mortos e por isso são removidos.

3. Para se encontrar estados equivalentes primeiro são criados dois conjuntos de estados, um contendo os finais e um contendo os não-finais. Em seguida as transições de cada estado são comparadas, os estados cujas transições levam a um mesmo conjunto são agrupados em um conjunto de equivalência, que é criado caso não exista previamente. Após a divisão, é criado um novo autômato cujos estados são os conjuntos de equivalência encontrados no passo anterior.

2.6 Determinização

Para determinizar o autômato, foi usado um algoritmo comentado em sala de aula, onde se analisa as transições não-determinísticas existentes. Para cada conjunto de estados que uma transição leva é criado um novo estado equivalente à união dos estados contidos no conjunto.

As transições desse novo estado são criadas a partir da junção das transições dos estados que o compõem. Se a partir dessa operação surgirem novas transições não determinísticas, novos estados são criados e repete-se esse passo.

Feito isso, os estados que ficaram inutilizados são removidos.

2.6.1 Epsilon-Transição

Inicialmente verifica-se se o autômato possui epsilon-transições. Caso não possua ele eh determinizado usando o processo descrito anteriormente.

Caso ele possua, primeiro se calcula o epsilon-fecho de cada estado e cria-se estados equivalentes a cada fecho. Após isso todas as transições que levavam a um estado individualmente são substituídas por transições que levam para o epsilon-fecho daquele estado.

As transições de cada epsilon-fecho são criadas a partir da união das transições de cada estado que o compõe. Caso sejam criadas transições não determinísticas, o autômato é determinizado usando a operação descrita acima.

2.7 Conversão de Expressões Regulares para Autômatos Finitos

A conversão de uma expressão regular para um autômato finito é feito atráves do algoritmo de De Simone, ligeiramente modificado. Segue os passos do algoritmo a seguir:

1. Antes de iniciar o processo de reconhecimento da expressão é adicionado pontos para facilitar o encontro de concatenações, o alfabeto da maquina gerada também é determinada nesta etapa.

2. Se cria a árvore recursivamente atráves do nodo raiz que divide a expressão regular para cada operador encontrada, procurando os operadores de maior prioridade primeiro.

3. Para cada operador é definido uma operação quando se sobe e encontra este operador e para quando se desce e encontra este operador.

4. Se custura a árvore em pré-ordem, o operador "Ou"/"tem sua costura apontada para seu pai, caso exista.

5. O estado inicial é criado, tendo sua composição sendo os nodos folhas que se chega atráves do nodo raiz. Para cada letra o alfabeto que se têm nos nodos de composição, se é criado um a transição sem destino. Caso se alcance o fim da árvore, o estado é declarado final. Se é adicionado o estado inicial a um conjunto de estados novos.

6. Enquanto há estados novos, realize os seguintes passos:

   (a) Para cada transição sem destino de cada estado novo, se vê quais nodos se alcança pelos nodos que tem o mesmo simbolo da transição, montando a composição de nodos alcançaveis.

   (b) É verificado se há algum estado que já tenha esta composição. Caso sim, o destino da transição é este estado. Caso contrário se é criado um estado com esta composição e este é adicionado aos novos estados, se lambda (fim da árvore) faz parte da composição este é um estado final.

   (c) Quando não há mais transições sem destino em um estado novo, este é removido da lista e se passa ao próximo estado novo.

7. Ao final, quando todos os estados tem composições diferentes e transições com destinos definidos, se monta um autômato finito com estes estados.

2.8 Equivalência de Expressões Regulares

A operação de equivalência foi implementada em três passos:

1. É feita a conversão das Expressões Regulares para Autômatos Finitos usando o algoritmo descrito anteriormente
2. Um terceiro autômato é gerado a partir da diferença entre os dois autômatos obtidos anteriormente.
3. Se a operação de diferença gerar um autômato sem estados finais, então as duas expressões regulares eram equivalentes.

3 Considerações Finais

3.1 Interface Gráfica

O programa é composto por uma janela principal onde é possível acessar todas as funções implementadas. Na barra principal existem dois menus, File e Edit.

- File:Acessando o menu File é possível criar um novo Autômato Finito ou Expressão Regular. Existe também a opção de fechar a aplicação
- Edit:Esse Menu contêm todas as operações que se pode aplicar nos Autômatos Finitos ou Expressões Regulares.

Os Autômatos são representados em tabelas e divididos em abas. O autômato cuja aba esteja selecionada será mostrado na janela. As operações entre dois autômatos sempre receberão de entrada o autômato atualmente selecionado e um outro autômato escolhido pelo usuário.

3.2 Instruções de Uso

- Ao realizar uma operação entre dois autômatos, certifique-se que eles não possuem estados com mesmo nome
- Ao Inserir o alfabeto de um novo autômato digite todas as letras não separadas por espaços ou qualquer outro símbolo. Exemplo: Para inserir o alfabeto a, b, c digite "abc"no campo especificado.