UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

IGOR RONSONI

CIÊNCIA DA COMPUTAÇÃO
TRABALHO PRÁTICO DE LINGUAGENS FORMAIS E AUTÔMATOS

CHAPECÓ
2021

Universidade Federal da Fronteira Sul

Igor Ronsoni

Autômato Finito Determinístico livre de estados mortos e inalcançáveis

Chapecó
2021

Autômato finito determinístico quase mínimo

Igor Ronsoni

Universidade Federal da Fronteira Sul, Rodovia SC 484 - Curso de Ciência da Computação

igor.ronsoni@estudante.uffs.edu.br

Resumo:

O presente artigo tem como objetivo, detalhar o processo de implementação do trabalho prático da disciplina de linguagens formais e autômatos. O trabalho consiste em implementar um autômato finito quase mínimo a partir de um arquivo de texto com uma relação de tokens e gramáticas regulares de uma linguagem qualquer. O trabalho apresenta como saída 4 (quatro) arquivos em formato csv (comma-separated values), mostrando o AFND, AFD, autômato minimizado e adicionado estado de erro.

1. Introdução

Na teoria dos autômatos, um autômato finito determinístico é uma máquina de estados que aceita ou rejeita uma cadeia de símbolos gerando assim um conjunto mínimo de estados pelos quais pode-se passar para a geração de tokens específicos.

2. Implementação

A aplicação foi desenvolvida em Python pois é uma linguagem que possui uma maior facilidade para se manusear strings vindas de arquivos.

Como primeiro etapa, foi realizado a leitura do arquivo de entrada e gravado os estados referentes a divisão de transições em um dicionário (dict() Python), cada nome de estado gerado era visto como key (chave) para um dicionário de tokens e estados transicionais, cada token era uma key para uma lista (list() Python) de estados de transição. A versão final do dicionário será escrita como sendo o AFND da linguagem.

A segunda etapa realizada foi a determinação, sendo feito a leitura do dicionário criado anteriormente e criando novos estados para onde há uma transição para mais de um estado com um único token a partir do mesmo estado. Juntamente nesta etapa foi retirada os estados inalcançáveis.

A terceira etapa foi a minimização do autômato, sendo retirado os estados mortos da gramática através de uma busca em profundidade.

A quarta e última etapa consiste em adicionar um estado de erro à linguagem, realizando transições para o mesmo quando escolhido tokens inválidos.

3. Estrutura

A gravação dos estados assim como dos tokens e das transições foi feita em um “conjunto de dicionário de estados”. O conjunto de dicionários consiste em um dicionário contendo o nome dos estados da linguagem como keys, chaves para um valor, no nosso caso, o valor será um dicionário de transição. Esse dicionário por fim, tem como keys, os tokens de transição para os estados, e o valor referente a cada key será uma lista de estados para os quais será feita a transição.

Em nossa aplicação, além do salvamento da gramática em um conjunto de dicionário de estados, também é feito o armazenamento dos tokens da linguagem sem repetição em um lista (list() Python) e o salvamento dos nomes dos estados finais em um conjunto (set() Python).

4. Indeterminismo

Um autômato não determinístico é um autômato onde a partir de um estado podemos alcançar dois outros estados com um mesmo token, caracterizando assim uma linguagem computacional irreconhecível.

5. Determininação

Para ser feita a determinização do autômato, a aplicação faz uma busca em profundidade, passando por cada estado a partir do estado inicial, reconhecido pelo token ‘S’.

O algoritmo usado pela aplicação segue a sequência básica de encontrar um determinismo, criar um novo estado, substituir o determinismo pelo novo estado e gravar as informações para realizar a troca posteriormente caso seja necessário.

6. Minimização

Para cada linguagem regular possível de ser determinada existe um AFD com um número mínimo de estados.

Um autômato minimizado deve remover três classes de estados sem mexer no AFD original:

- **Estados mortos**: São estados nos quais não se chega a um final.
- **Estados inalcançáveis**: São estados impossíveis de se chegar a partir de um estado inicial.
- **Estados indistinguíveis**: São estados nos quais com qualquer entrada seguem o mesmo caminho de estados de aceitação. Não era necessário implementar a remoção deste conjunto de estados na aplicação.

Um autômato mínimo não possui nenhum desses três conjuntos. Na presente aplicação, não era necessário a implementação da remoção de indistinguíveis, por

isso, foi declarado o nome do artigo como "Autômato finito determinístico quase mínimo".

7. Remoção de inalcançáveis

Estados inalcançáveis são estados impossíveis de serem alcançados a partir de um estado inicial. A remoção de tais estados está sendo feita com um algoritmo de busca em profundidade, onde, será passado em todos os estados a partir de um estado inicial e realizando o salvamento dos estados visitados em um novo dicionário. A busca está sendo feita juntamente com o processo de determinação do autómato, pois ambas utilizam a busca em profundidade. A geração do novo dicionário será necessária para a retirada de estados mortos.

8. Remoção de mortos

Estados mortos é identificado quando a partir de um estado não se chega a um estado final, denota-se então que a transição inteira é uma transição improdutiva.

Na presente aplicação, a forma pela qual estamos removendo esse conjunto de estados é através de uma busca em profundidade no dicionário gerado sem os estados inalcançáveis. Quando chegar a um estado que não possui transições será verificado se o estado atual é um estado final, em caso de estados que possuem várias transições é criado uma flag indicando se o estado é morto ou não.

9. Estado de Erro

Transições inexistentes é realizada a transição para o estado de erro.

Para indicar um invalidade na geração de um símbolo, é necessário a adição do estado de erro, assim a aplicação se encarrega de rejeitar a sequência de tokens já existente em caso de seleção de um token invalido.

10. Conclusão

A aplicação demorou mais do que o esperado para ser realizada, pois durante a implementação da determinização estava ocorrendo muitos erros, pois o modo como a aplicação estava armazenando a gramática estava ocasionando inconsistências na hora da leitura em profundidade. O método que estava sendo utilizado, ao invés de guardar como key do dicionário os tokens, estava sendo guardado nos estados. O método novo implementado possibilitou realizar os testes com menos linhas de código e assim mais facilidade de verificação dos estados.

O método novo utilizado foi desenvolvido com base em um trabalho já desenvolvido na matéria de Linguagens Formais e Autômatos no ano de 2018 pelos alunos Eduardo Stefanello e Laurivan Sareta.

Referências

Minimização de AFD. Wikipedia, 2020. Disponível em: <https://pt.wikipedia.org/wiki/Minimiza%C3%A7%C3%A3o_de_AFD>. Acesso em: 11 de Janeiro de 2021.

TrabalhoLFA. GitHub, 2018. Disponível em: <https://github.com/laurivansareta/TrabalhoLFA>. Acesso em: 5 de Janeiro de 2021