Linguagens Formais e Autômatos

Aula 05 - Equivalência DFA x NFA

Prof. Dr. Daniel Lucrédio
Departamento de Computação / UFSCar
Última revisão: ago/2015

Referências bibliográficas

- **Introdução à teoria dos autômatos, linguagens e computação / John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman** ; tradução da 2.ed. original de Vandenberg D. de Souza. - Rio de Janeiro : Elsevier, 2002 (Tradução de: Introduction to automata theory, languages, and computation - ISBN 85-352-1072-5)
  - Capítulo 2 - Seção 2.3
- **Introdução à teoria da computação / Michael Sipser** ; tradução técnica Ruy José Guerra Barretto de Queiroz ; revisão técnica Newton José Vieira. -- São Paulo : Thomson Learning, 2007 (Título original : Introduction to the theory of computation. "Tradução da segunda edição norte-americana" - ISBN 978-85-221-0499-4)
  - Capítulo 1 - Seção 1.2

Equivalência DFA e NFA

- Intuitivamente, NFA é mais poderoso
  - Mas as linguagens aceitas por um NFA são regulares
  - Ou seja, qualquer NFA pode ser convertido em um DFA que reconhece a mesma linguagem
- Teorema:
  - Uma Linguagem L é aceita por algum DFA se e somente se L é aceita por algum NFA
  - Prova por construção dos dois lados:
    - “Se”: um processo que constrói um DFA a partir de um NFA
    - “Somente se”: um processo que constrói um NFA a partir de um DFA

Conversão NFA → DFA

- Na maioria dos casos, um DFA equivalente tem o mesmo número de estados que o NFA, só que mais transições
- No pior caso, tem 2^n estados
  - NFA N = $(Q_N, \Sigma, \delta_N, q0, F_N)$
  - DFA D = $(Q_D, \Sigma, \delta_D, \{q0\}, F_D)$
  - $L(D) = L(N)$
- Q_D é o conjunto de subconjuntos de Q_N
- F_D é o conjunto S de subconjuntos de Q_N,

  tal que $S \cap F_N \neq \varnothing$
- Para cada conjunto $S \subseteq Q_N$
  - e para cada $a \in \Sigma$
  - $\delta_D(S,a) = \cup$ todos os $\delta_N(p,a)$ para $p \in S$

Conversão NFA → DFA

- Consiste em pegar todas as combinações de estados e agregar as transições do NFA
  - Cada combinação de estados do NFA é um estado no DFA
- Consiste basicamente na implementação “em paralelo”
  - Mas pré-calculando as combinações de estados

Conversão NFA → DFA

- Passo a passo com exemplo
- Dado o NFA (cadeias que terminam com 01):

	0	**1**
→ q0	{q0,q1}	{q0}
q1	∅	{q2}
*** q2**	∅	∅

Conversão NFA → DFA

- Passo 1:
  - Faça uma tabela “vazia”, com as mesmas entradas como colunas (a tabela vai crescer para baixo)

	0	1

Conversão NFA → DFA

- Passo 2:
  - Crie um novo estado inicial no DFA, um conjunto que contém somente o estado inicial do NFA

	0	**1**
→ {q0}		

Conversão NFA → DFA

- Passo 3:
  - Para cada entrada, insira no DFA um conjunto que contém a união de todos os resultados da transição NFA daquela entrada para todos os estados do conjunto à esquerda

	0	**1**
→ **{q0}**	{q0,q1}	{q0}

Conversão NFA → DFA

- Passo 4:
  - Para cada novo conjunto de estados que aparecer, insira uma nova linha na tabela do DFA e volte para o passo 3

	0	**1**
→ **{q0}**	{q0,q1}	{q0}
{q0,q1}	{q0,q1}	{q0,q2}

Conversão NFA → DFA

- Passo 4 (novamente):

	0	**1**
→ **{q0}**	{q0,q1}	{q0}
{q0,q1}	{q0,q1}	{q0,q2}
{q0,q2}	{q0,q1}	{q0}

Conversão NFA → DFA

- Passo 5: Quando não houver mais novos estados, marque como estado de aceitação os conjuntos que contém ao menos um estado de aceitação do NFA

	0	**1**
→ **{q0}**	{q0,q1}	{q0}
{q0,q1}	{q0,q1}	{q0,q2}
* **{q0,q2}**	{q0,q1}	{q0}

Conversão NFA → DFA

- Passo 6: “Renomeie” os conjuntos para estados, de forma a facilitar a leitura do DFA

	0	**1**
→ **A**	B	A
B	B	C
* **C**	B	A

Conversão NFA → DFA

- Dado o seguinte NFA:
  - Construa um DFA que aceite a mesma linguagem

	0	**1**
→ **p**	{p,q}	{p}
q	{r}	{r}
r	{s}	∅
* **s**	{s}	{s}

Conversão NFA → DFA

	0	**1**
→ **{p} A**	{p,q} B	{p} A
{p,q} B	{p,q,r} D	{p,r} C
{p,r} C	{p,q,s} E	{p} A
{p,q,r} D	{p,q,r,s} F	{p,r} C
*** {p,q,s} E**	{p,q,r,s} F	{p,r,s} G
*** {p,q,r,s} F**	{p,q,r,s} F	{p,r,s} G
*** {p,r,s} G**	{p,q,s} E	{p,s} H
*** {p,s} H**	{p,q,s} E	{p,s} H

Conversão NFA → DFA

- Dado o seguinte NFA:
  - Construa um DFA que aceite a mesma linguagem

	0	**1**
→ *** q0**	{q1}	{q2}
*** q1**	∅	{q0}
*** q2**	{q0}	∅

Conversão NFA → DFA

	0	**1**
→ *** {q0} A**	{q1} B	{q2} C
*** {q1} B**	{} D	{q0} A
*** {q2} C**	{q0} A	{} D
{} D (morto)	{} D	{} D

Conversão DFA → NFA

- “Resto” da prova
- Parte fácil
  - Construir um NFA a partir de um DFA
  - Basta “copiar” o diagrama (ou tabela), trocando estados por conjuntos de estados
  - Um DFA é um caso específico de NFA
    - NFA permite 0 ou mais transições em cada situação
    - DFA permite sempre 1 transição em cada situação
    - 1 está entre 0 ou mais

Conversão DFA → NFA

	0	**1**
→ **q1**	q1	q2
* **q2**	q1	q2

	0	**1**
→ **q1**	{q1}	{q2}
* **q2**	{q1}	{q2}

Conversão DFA → NFA

- Formalmente:
  - Seja $D = (Q,\Sigma,\delta_D,q_0,F)$ um DFA
  - Defina $N = (Q,\Sigma,\delta_N,q_0,F)$
    - Onde δ_N é definido pela regra:
      - Se $\delta_D(q,a)=p$, então $\delta_N(q,a)=\{p\}$
- Como consequência
  - Se $\hat{\delta}_D(q_0,w)=p$, então $\hat{\delta}_N(q_0,w)=\{p\}$
- Portanto, w é aceito por D se e somente se é aceito por N
  - Isto é: $L(D) = L(N)$

Fim

Aula 05 - Equivalência DFA x NFA