UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS DO Ó SANTOS TIBURCIO

Uma prova do algoritmo de Brzozowski via álgebras de Kleene

RIO DE JANEIRO
2026

MATHEUS DO Ó SANTOS TIBURCIO

Uma prova do algoritmo de Brzozowski via álgebras de Kleene

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Hugo Musso Gualandi
Co-orientador: João Antonio Recio da Paixão

RIO DE JANEIRO
2026

CIP - Catalogação na Publicação

T554p

Tiburcio, Matheus do Ó Santos
Uma prova do algoritmo de Brzozowski via álgebras de Kleene / Matheus do Ó Santos Tiburcio. - Rio de Janeiro, 2026.
99 f.

Orientador: Hugo Musso Gualandi.
Coorientador: João Antonio Recio da Paixão.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2026.

1. Linguagens regulares. 2. Álgebra de Kleene. 3. Autômatos finitos. 4. Minimização de autômatos finitos. 5. Algoritmo de Brzozowski. I. Gualandi, Hugo Musso, orient. II. Paixão, João Antonio Recio da, coorient. III. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

MATHEUS DO Ó SANTOS TIBURCIO

Uma prova do algoritmo de Brzozowski via álgebras de Kleene

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 11 de fevereiro de 2026.

BANCA EXAMINADORA:

Documento assinado digitalmente
gov.br
HUGO MUSSO GUALANDI
Data: 16/03/2026 16:11:44-0300
Verifique em https://validar.iti.gov.br

Hugo Musso Gualandi
D.Sc. (IC/UFRJ)

Documento assinado digitalmente
gov.br
SEVERINO COLLIER COUTINHO
Data: 17/03/2026 12:37:14-0300
Verifique em https://validar.iti.gov.br

Severino Collier Coutinho
D.Sc. (IC/UFRJ)

Documento assinado digitalmente
gov.br
MATHEUS NUNES ADAUTO
Data: 17/03/2026 10:25:45-0300
Verifique em https://validar.iti.gov.br

Matheus Nunes Adauto
D.Sc. (IC/UFRJ)

Dedico este trabalho à minha mãe, Yngrid Brito do Ó Santos.

AGRADECIMENTOS

Agradeço à UFRJ e ao IC pela infraestrutura e fomento à educação. Agradeço também ao CNPq pela bolsa de iniciação científica PIBIC, que viabilizou financeiramente minha pesquisa nestes últimos dois anos passados. O ensino público se torna fundamental nestes casos, que jamais seja sucateado.

Agradeço a meus professores, de toda a vida, que me impulsionaram e permitiram que chegasse aqui onde me encontro. Em especial, agradeço a Celso Omar, Francisco Viana e Marcelo. Agradeço a William e Carlos Henrique. Agradeço ao João Paixão pelas infinitas conversas sobre a vida e outros assuntos, Hugo Musso Gualandi pelos anos me orientando, Hugo Nobrega, Vinícius Gusmão, Maria Luiza, Silvana Rossetto e a Marcello Goulart.

Agradeço às pessoas que conheci ao decorrer do curso, cada uma agregando de seu modo. Agradeço a Nicholas Araújo, Mariana Fernandes com quem tive tantos momentos, Manoel Marcelo, ao cinéfilo e querido Vinícius. Agradeço também ao pessoal do laboratório LC3. Agradeço à Mariana Gonçalves e Luiza Barroso. Agradeço à todos os alunos que pude ser monitor. Por fim, agradeço aos meus "primos", Andressa Pereira e Pedro Jorge, certamente uma das melhores surpresas que tive recentemente.

Agradeço meus amigos anteriores à UFRJ, que me viram embrionário e que seguem ao meu lado até hoje. Obrigado Anny Rose pelos sermões, Gabriel Torres, Igor Blezer, Magno Ramos, Sharah Noel pelos sermões também. Obrigado também a meus amigos mais antigos e que passei tanta coisa junto: Juan Waltz, Gustavo Vasques e Melissa Ximenes, me sinto sortudo os tendo e espero continuar por mais dez anos com vocês.

Agradeço ao meu pai José Luiz Tiburcio, que trabalhou incessantemente durante toda a sua vida para nos proporcionar, eu e meus irmãos, uma condição digna de vida e que me possibilitou focar nos estudos da maneira que fiz e sigo fazendo. Agradeço a meu irmãozinho, Daniel do Ó Santos Tiburcio, o caçula, que com suas perguntas curiosas e comentários criativos me tirou inúmeras risadas, que me impulsiona também a continuar tentando dar o meu melhor. E não posso deixar de agradecer também o meu irmão do meio, Lucas do Ó Santos Tiburcio, com certeza uma das pessoas que mais importantes para mim, sou muito grato por ter você. Agradeço meu primo Frank Brito e à minha Tia Catia Maria, minha "segunda mãe".

E por último, e mais importante, à minha falecida mãe, Yngrid Brito do Ó Santos, certamente a principal responsável por tudo isso. Foi quem me ensinou a valorizar o estudo. A percepção de que ela não poderá ver este momento me dói, traz consigo também uma sensação de dívida que não pode ser paga. Ainda sigo, porque penso que é, justamente, este comportamento que ela esperaria. Você se foi, mas os pequenos cientistas ainda estão aqui. Dedico este trabalho inteiramente à esta mulher incrível que ela foi. Os meus maiores agradecimentos, Mãe. Te amo.

"As a possible explanation, I suggest that automata theory is the linear algebra of computer science [....] I am more interested, however, in the figurative sense: automata theory as a basic, fundamental subject, known and used by everyone, which has formed part of the intellectual landscape for so long that it is no longer noticed. And yet, there it is, structuring it, organising it: and knowing it allows us to orient ourselves."

Jacques Sakarovitch

RESUMO

Autômatos finitos são objetos recursivamente definidos, o que torna natural provarmos suas propriedades através de indução. Em especial, muitos problemas de teoria dos autômatos necessitam de identificar se dois autômatos diferentes têm a mesma linguagem, isto é, se são equivalentes. Tais provas geralmente são feitas através de indução no comprimento dos caminhos, além de serem feitas em duas direções, visto que precisamos mostrar que todo caminho válido em um autômato tem um equivalente noutro e vice-versa. Estas provas apresentam dois problemas principais: uma escolha ruim de caso base e hipótese indutiva pode tornar provas por indução desnecessariamente complicadas ou até incorretas; provas de ida-e-volta podem ser desafiadoras e, neste contexto, cada caminho do autômato precisa ter seu equivalente encontrado em um autômato diferente. Uma alternativa para ambos problemas é o *teorema da fusão*, que torna provas indutivas em provas por igualdade, simplificando-as e removendo preocupações herdadas de provas indutivas. Neste trabalho, buscamos demonstrar como álgebras de Kleene e o teorema da fusão podem conversar bem com teoria dos autômatos finitos e fazemos isto focando em um problema em específico: minimização de autômatos finitos. Trazemos a primeira prova da corretude do algoritmo de Brzozowski à luz do teorema da fusão e via formalismo matricial.

Palavras-chave: linguagens regulares; álgebra de Kleene; autômatos finitos; minimização de autômatos finitos; algoritmo de Brzozowski.

ABSTRACT

Finite automata are recursively defined objects, which makes natural to prove their properties through induction. Specifically, many problems from automata theory need to identify if two different automata recognize the same language, i.e., if they are equivalent. Such proofs generally are made through induction in the length of the paths, in addition to be mutual inclusion, since we need to show that each valid path in an automaton has an equivalent in a different one and vice-versa. These proofs have two main problems: bad choices for the base case and inductive hypothesis might lead to more complicated proofs and sometimes wrong ones; Mutual implication proofs might be challenging and, in this context, each path from the automaton must have its equivalent found in another automaton. An alternative for both problems is the *fusion theorem*, which turns inductive proofs into equality proofs, simplifying them and removing inherited concerns from inductive proofs. In this work, we seek to demonstrate how Kleene algebras and the fusion theorem can interact well with automata theory and we do it focusing in a specific problem: finite automata minimization. We bring the first proof of correctness of the Brzozowski's algorithm in light of the fusion theorem and using a matrix representation.

Keywords: regular languages; Kleene algebra; finite automata; finite automata minimization; Brzozowski's algorithm.

LISTA DE DEFINIÇÕES

LISTA DE TEOREMAS

LISTA DE ABREVIATURAS E SIGLAS

AFND	Autômato finito não-determinístico
AFD	Autômato finito determinístico
AFCD	Autômato finito codeterminístico

LISTA DE SÍMBOLOS

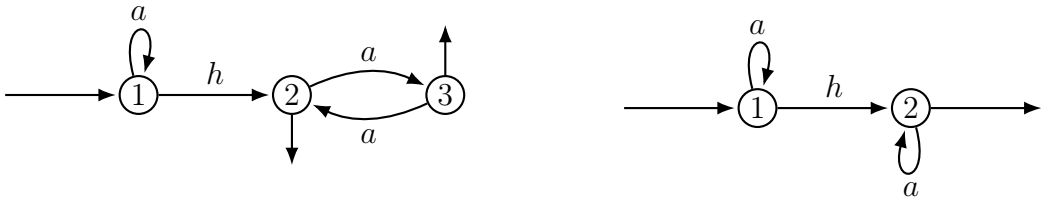
$\xrightarrow{a}$	Aresta de rótulo a ou caminho de rótulo a
$\lvert\cdot\rvert$	Comprimento de uma palavra
ε	Palavra vazia
$\mathcal{P}$	Conjunto das partes, conjunto de todos os subconjuntos
*	Operador estrela de Kleene
$\in$	Pertence
$\subseteq$	Subconjunto de
$\langle\cdot\rangle$	Definição de semianeis e autômatos finitos
$\sum$	Somatório
$\wedge$	Operador de conjunção
$\mathtt{Aut}$	Conjunto de todos os autômatos finitos
$\mathcal{L}$	Operador linguagem
$\mathcal{L}_{\mathtt{e}}$	Operador linguagem à esquerda
$\mathcal{L}_{\mathtt{d}}$	Operador linguagem à direita
$\mathtt{det}$	Operador de determinização
$\mathtt{pod}$	Operador poda
$\mathtt{rev}_w$	Operador reversão em palavras
$\mathtt{rev}_{\mathcal{L}}$	Operador reversão em linguagens e matrizes
$\mathtt{rev}_{\mathtt{Aut}}$	Operador reversão em autômatos
^{R}	Operador reversão genérico
$\mathtt{codet}$	Operador de codeterminização
$\mathtt{copod}$	Operador de copoda
$\mathtt{disj}$	Operador de disjunção
$\mathtt{Min}$	Operador de minimização

SUMÁRIO

1 INTRODUÇÃO

Muitos dos principais problemas em teoria dos autômatos finitos necessitam de identificar se dois autômatos diferentes têm o mesmo comportamento, isto é, se reconhecem a mesma linguagem. Tais provas de equivalência costumam ser feitas através de indução no comprimento de caminhos nos autômatos, além de serem feitas no formato ida-e-volta, visto que precisamos mostrar que toda palavra reconhecida por um autômato será reconhecida pelo outro e vice-versa, o que é um processo difícil.

Por exemplo, os autômatos da Figura 1 são considerados equivalentes e reconhecem a linguagem das palavras com zero ou mais a e exatamente um h, em qualquer ordem. A linguagem do autômato da Figura 1a é o conjunto de caminhos que começam no estado 1 e terminam no estado 2 ou 3 do primeiro autômato, por outro lado, a linguagem o autômato da Figura 1b é o conjunto de caminhos que começam no estado 1 e terminam no estado 2 do segundo autômato. Exemplos de caminhos válidos seriam *aaaaha*, *ahaaaa* e h.



(a) Autômato de exemplo com três estados. (b) Autômato de exemplo com dois estados.

Figura 1 – Dois autômatos finitos diferentes que reconhecem a mesma linguagem: Palavras com zero ou mais a e exatamente um h, em qualquer ordem.

No exemplo apresentado, provar a equivalência é a mostrar que todo caminho em um autômato possui um equivalente no outro. Por exemplo, o caminho $1 \xrightarrow{a} 1 \xrightarrow{h} 2 \xrightarrow{a} 3$ no autômato da Figura 1a tem como equivalente o caminho $1 \xrightarrow{a} 1 \xrightarrow{h} 2 \xrightarrow{a} 2$ no autômato da Figura 1b.

1.1 TEOREMA DA FUSÃO

Como a linguagem é definida em cima de caminhos, as provas de equivalência convidam indução no comprimento deles. De modo geral, provar desta maneira não costuma ser direto, justamente por termos de provar equivalências entre caminhos de diferentes autômatos. A prova tende a ser dividida em duas partes, além do cuidado na escolha das hipóteses e caso base, presente em toda prova indutiva. Temos um outro modo de lidar com este caráter recursivo. Representando autômatos através de matrizes e vetores (CONWAY, 1971; SAKAROVITCH, 2009), é possível tornar esta indução no compri-

mento de caminhos em uma prova por igualdade (KOZEN, 1994; BACKHOUSE, 2002). Para os autômatos da Figura 1a e Figura 1b temos, respectivamente, os seguintes vetores e matrizes de transição:

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad E = \begin{bmatrix} a & h & 0 \\ 0 & 0 & a \\ 0 & a & 0 \end{bmatrix} \qquad v = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}; \tag{1.1.1}$$

$$\hat{u} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \widehat{E} = \begin{bmatrix} a & h \\ 0 & a \end{bmatrix} \qquad \hat{v} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{1.1.2}$$

Neste formalismo, a entrada E_{ij} indica o rótulo de transição do estado i para o estado j. Os vetores u e v representam, respectivamente, os *estados iniciais* e os *estados finais* do autômato. Igualmente, toda a análise feita pode ser replicada para o autômato da Figura 1b, com $\hat{u}$, $\widehat{E}$ e $\hat{v}$. No trabalho de Kozen ele apresenta técnicas puramente algébricas que eliminam por completo a necessidade de possuirmos duas provas (uma para ida, uma para a volta). Para Kozen, mostrar que tais autômatos são equivalentes se reduziria a encontrar uma matriz F que *normalize* ambas as matrizes de transição dos autômatos que estamos observando, além de conseguir descrever os vetores de estados iniciais e finais através de F. Mais especificamente, queremos F de modo que

$$\hat{u}^T = u^T F, \tag{1.1.3}$$

$$F\widehat{E} = EF, \tag{1.1.4}$$

$$F\hat{v} = v. \tag{1.1.5}$$

Kozen demonstra que estas são condições suficientes para a prova de equivalência entre autômatos. A segunda condição é a normalização e a primeira e terceira condição servem para descrever como exatamente os estados iniciais e finais serão transformados. Não provaremos tais resultados neste momento, o leitor pode verificar isto nos trabalhos de Kozen (KOZEN, 1994) ou ao decorrer desta monografia através de nossas provas. Para o exemplo da Figura 1, a matriz F que se adequa é

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}. \tag{1.1.6}$$

Decorre que

$$F\widehat{E} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a & h \\ 0 & a \end{bmatrix} = \begin{bmatrix} a & h \\ 0 & a \\ 0 & a \end{bmatrix} = \begin{bmatrix} a & h & 0 \\ 0 & 0 & a \\ 0 & a & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} = EF; \tag{1.1.7}$$

$$\hat{u}^T = \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} = u^T F; \tag{1.1.8}$$

$$v = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = F\hat{v}. \tag{1.1.9}$$

Desta forma, F traduz entre dois mundos, referentes aos autômatos da Figura 1a e 1b. Este teorema é um caso particular do chamado *teorema da fusão* (BACKHOUSE, 2002). A matriz de transição de um autômato contém toda a informação relativa às suas transições e a normalização que vimos pode ser, a grosso modo, comparada com o caso indutivo. O caso base pode ser comparado com $v = F\hat{v}$, que altera os estados finais para a versão transformada.

Neste trabalho estamos interessados em provar resultados de teoria dos autômatos de maneira equacional, provendo provas por igualdade com os passos devidamente definidos. O teorema da fusão permite com que façamos isso, contornando a indução que usualmente tomaria o lugar. Para explorar o uso do teorema da fusão, escolhemos especificamente o problema de minimização de autômatos finitos, devido ao fato deste algoritmo possuir vários passos e todos precisarem de provas de equivalência.

Um algoritmo para solucionar o problema de equivalências entre autômatos é, na verdade, aplicar uma transformação de *minimização* em ambos os autômatos no qual queremos analisar. Temos como garantia que toda linguagem possui um único autômato *mínimo*, portanto se os dois autômatos que estamos analisando resultarem no mesmo autômato mínimo, então eles originalmente reconheciam a mesma linguagem.

Podemos minimizar autômatos de algumas maneiras, neste trabalho falaremos a respeito do algoritmo de Brzozowski e consequentemente sua corretude. Nossa pergunta de pesquisa é:

Será que somos capazes de utilizar o teorema da fusão
para provar a corretude do algoritmo de Brzozowski?

O algoritmo de Brzozowski é uma boa escolha para utilizar o teorema da fusão, porque muitos dos resultados intermediários necessários para a sua prova de corretude podem ser provados através de induções no comprimento de caminhos. Fizemos a primeira prova de corretude deste algoritmo à luz do teorema da fusão e da notação matricial. Para provar a corretude do algoritmo precisamos provar todos estes resultados intermediários, no qual tentamos também provar por fusão.

1.2 O ALGORITMO DE BRZOZOWSKI

O algoritmo de Brzozowski é dividido em três operações, sendo elas a determinização, a remoção de estados inacessíveis (poda) e a reversão. Essencialmente o algoritmo aplica duas vezes a sequência de operações

$$\text{reversão} \rightarrow \text{determinização} \rightarrow \text{remoção de estados inacessíveis.}$$

A Figura 2 traz um exemplo da execução do algoritmo. Note que o autômato inicial é exatamente do autômato da Figura 1a e autômato final é bem semelhante ao da Figura 1b, entretanto possuindo um estado a mais. Isto ocorre devido à definição de determinismo que utilizamos, forçando com que o estado do meio necessariamente tenha uma transição rotulada em h.

A operação de reversão troca o sentido de todas as transições e setas. A operação de determinização pode criar estados inacessíveis, que são subsequentemente removidos na próxima etapa. Descrevemos e provamos que todas estas operações preservam a linguagem do autômato, fazendo parte dos resultados intermediários que citamos. Destes resultados, temos as seguintes provas:

1. **Corretude do determinismo**, provado por Kozen através do teorema da fusão, embora não o chamasse por este nome (KOZEN, 1994). Esta prova é discutida em mais detalhes na Subseção 3.2.3.

2. **Corretude da remoção de estados inacessíveis**, que provamos por fusão ao decorrer da Seção 3.3.

3. **Operação de estrela de Kleene e reversão comutam**, no qual não conseguimos provar por fusão, utilizando indução convencional. Discutimos sobre reversão e operações relacionadas ao decorrer do Capítulo 4.

4. **Determinizar gera linguagens disjuntas**, no qual não conseguimos provar por fusão e provamos por indução. Enunciamos e provamos o teorema no Capítulo 5.

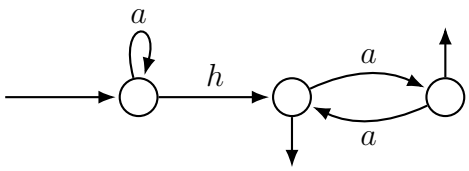
5. **Se as linguagens à direita do autômato são disjuntas e não vazias, se tornam diferentes após a determinização**, que provamos por fusão e também presente no Capítulo 5.

Provamos todos os cinco itens, exceto o primeiro que já possuía uma prova escrita na literatura. Trazemos como contribuição todas estas provas. De maneira mais abrangente, como nossas contribuições, temos provas para

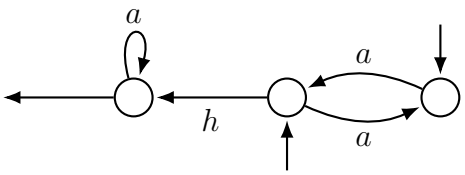
- A forma de E^* de maneira construtiva;
- A corretude do algoritmo de remoção de estados inacessíveis;

- Provas intermediárias necessárias para o algoritmo de Brzozowski;
- A corretude do algoritmo de Brzozowski.

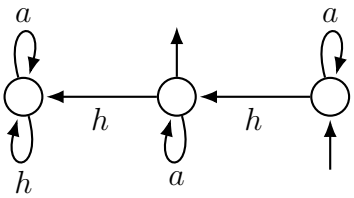
Este trabalho busca demonstrar como o teorema da fusão e álgebras Kleene podem ser utilizados para descrever e derivar resultados de teoria de linguagens formais e teoria dos autômatos. Provas construtivamente a corretude do algoritmo de Brzozowski, incluindo resultados intermediários. Muitos dos resultados puderam ser provados via teorema da fusão, com provas de facetas bem semelhantes.



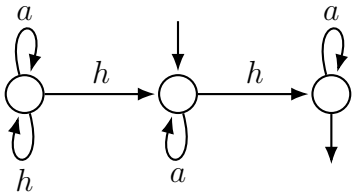
(a) Autômato finito da Figura 1a.



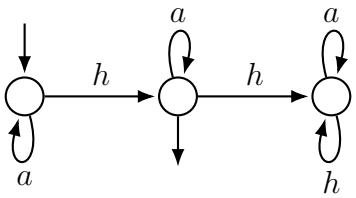
(b) Primeira etapa (reversão).



(c) Segunda etapa (determinização → remoção de estados inacessíveis).



(d) Terceira etapa (reversão).



(e) Quarta etapa (determinização → remoção de estados inacessíveis).

Figura 2 – Exemplo do algoritmo de Brzozowski. Essencialmente ele faz duas vezes o processo de reversão → determinização → remoção de estados inacessíveis.

2 LINGUAGENS E AUTÔMATOS

Neste capítulo apresento conceitos básicos de teoria de linguagens formais e teoria dos autômatos que nos serão úteis no decorrer de toda a monografia. Começo introduzindo as ideias iniciais por meio de grafos, uma forma natural de nos prepararmos para o formalismo matricial dos conceitos aqui apresentados. Após a fundamentação em linguagens formais, defino as estruturas algébricas que nos serão úteis e mostro como interpretar os conceitos previamente vistos neste novo linguajar matricial. Para o leitor, espero uma leve familiaridade com o que é um grafo e como operar matrizes. Para o conhecedor de linguagens formais, peço que se dê a oportunidade de reaprendê-la sobre uma óptica diferente.

2.1 PALAVRAS E LINGUAGENS

Definiremos primeiramente um *alfabeto* e, em seguida, um grafo sobre este. Neste texto, todo grafo será definido sobre um alfabeto, no qual ficará claro pelo contexto.

Definição 1 (Alfabeto)**.** *Um alfabeto é um conjunto finito cujos elementos são chamados de símbolos.*

Definição 2 (Grafo sobre um alfabeto)**.** *Um grafo sobre um alfabeto A é um grafo no qual todos os seus rótulos de arestas são símbolos de A.*

Exemplo 3. *Suponha o grafo de exemplo da Figura 3. Chamamos de rótulo de uma aresta o valor que se encontra logo acima (ou abaixo) dela. Nele, todos os rótulos são elementos vindos de um alfabeto $A = \{a, b\}$. Usualmente representamos as arestas escrevendo uma seta apontando do vértice de origem para o de destino, junto de seu rótulo. Deste modo, $1 \xrightarrow{a} 2$ representa uma das arestas do grafo.*

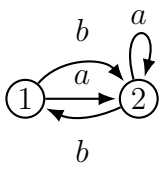


Figura 3 – Grafo com somente dois vértices e quatro arestas direcionadas.

Definição 4 (Caminho em um grafo)**.** *Um caminho em um grafo é uma sequência finita de arestas, na qual o vértice de destino é o vértice de origem da aresta seguinte.*

Exemplo 5. *Dois caminhos possíveis no grafo da Figura 3 seriam*

$$1 \xrightarrow{a} 2 \xrightarrow{b} 1,$$
$$1 \xrightarrow{a} 2 \xrightarrow{a} 2.$$

Denotaremos ainda o *rótulo* do caminho como a junção, na mesma ordem, do rótulo de cada uma das arestas que o compõe. Nos dois caminhos citados, obteríamos como seus rótulos ab e aa, respectivamente.

Exemplo 6. *Um caminho não define unicamente um rótulo. Suponha o grafo da Figura 4, podemos obter os seguintes caminhos, ambos de rótulo aa.*

$$1 \xrightarrow{a} 2 \xrightarrow{a} 1,$$
$$1 \xrightarrow{a} 2 \xrightarrow{a} 2.$$

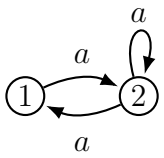


Figura 4 – Grafo de exemplo com somente dois vértices e três arestas direcionadas. Este grafo admite diferentes caminhos de mesmo rótulo, indicando que um rótulo não necessariamente caracteriza unicamente um caminho.

Em linguagens formais, nos aproveitaremos da noção de caminhos para modelarmos *palavras* de uma maneira mais visual e algébrica. De fato, pela definição que aqui apresentaremos, podemos fazer um paralelo direto com o conceito de rótulos de caminhos que acabamos de definir.

Definição 7 (Palavra sobre um alfabeto)**.** *Uma palavra sobre um alfabeto A é uma sequência finita elementos de A.*

Definição 8 (Conjunto de palavras)**.** *A^* é o conjunto de palavras sobre o alfabeto A.*

Exemplo 9. *Considere o alfabeto $B = \{p, i, o\}$. Exemplos de palavras sobre B seriam pipi, popo, oii. B^*, neste contexto, possuiria todas as combinações finitas de elementos de B.*

Dessa forma, todo rótulo de caminho pode ser visto como uma palavra. Uma característica de uma palavra é seu *comprimento*.

Definição 10 (Comprimento de uma palavra)**.** *O comprimento de uma palavra f é o número de símbolos que a compõe. Denotamos seu comprimento como $|f|$.*

Exemplo 11. *As palavras pipi, popo e oii têm comprimentos $|pipi| = 4$, $|popo| = 4$ e $|oii| = 3$, respectivamente.*

Toda palavra necessariamente possui comprimento finito. Considere a palavra *pipipo* sobre o alfabeto B. Podemos considerá-la como a junção, ou *concatenação*, de duas palavras: *pipi* e *po*. Considerar palavras como objetos gerados por outras palavras já existentes é muito útil nesta teoria, vamos definir este processo de maneira mais precisa.

Definição 12 (Concatenação de palavras)**.** *Dadas duas palavras f e g, a concatenação de f por g, denotada $f \cdot g$ ou apenas justapondo ambas na forma fg, é a justaposição dos símbolos de g após os de f.*

Exemplo 13. *Sejam $f = pipi$ e $g = popo$. Então $f \cdot g = pipipopo$.*

Repare que, se temos palavras f e g, então $|fg| = |f| + |g|$. Uma definição extremamente importante e que será melhor motivada futuramente é a de *palavra vazia*.

Definição 14 (Palavra vazia)**.** *A palavra vazia é a palavra que contém zero símbolos. Ela é denotada por ε.*

Recorde que em momento algum proibimos uma sequência vazia de elementos de A como uma palavra válida, portanto ε também pertence à A^*. Em realidade, ε é a única palavra com propriedade $|\varepsilon| = 0$, além de ser o *elemento neutro* da operação de concatenação. Isto é,

$$\varepsilon \cdot f = f \cdot \varepsilon = f. \tag{2.1.1}$$

Ainda falando sobre concatenar, outras propriedades desta operação são que esta é *associativa* e *não-comutativa*. Associatividade atesta que

$$(f \cdot g) \cdot h = f \cdot (g \cdot h). \tag{2.1.2}$$

Não-comutatividade indica que se mudarmos a ordem dos termos no produto, não necessariamente temos o mesmo resultado. Se pensarmos em caminhos em grafos e seus rótulos, essa diferenciação fica mais clara.

Exemplo 15. *Se $f = pipi$ e $g = po$, então $pipipo \neq popipi$.*

2.1.1 Linguagens

Em geral, não estaremos interessados em analisar palavras isoladas, mas sim conjuntos de palavras. A motivação é a de que certos conjuntos podem possuir características específicas em suas formações e a teoria de linguagens formais busca explorar justamente estas construções. Fixado um alfabeto A, uma *linguagem* é um conjunto de palavras quaisquer sobre A. Sendo assim, A^* por si só é uma linguagem. Como A^* contém todas as palavras sobre A, toda linguagem é subconjunto de A^*. Denotamos o *conjunto de subconjuntos* de A^* por $\mathcal{P}(A^*)$, também conhecido como *conjunto das partes*. Portanto, L é uma linguagem sobre A se, somente se, $L \in \mathcal{P}(A^*)$.

Definição 16 (Linguagem sobre um alfabeto)**.** *Uma linguagem L sobre um alfabeto A é um conjunto de palavras sobre A. De outra maneira, L é subconjunto de A^*.*

Exemplo 17. *Considere um alfabeto $C = \{m, u, a, h, i\}$. Duas linguagens possíveis sobre C seriam*

$$L = \{mua, ha\} \text{ e } R = \{ha, hi\}. \tag{2.1.3}$$

Pelo fato de linguagens serem conjuntos, conseguimos definir operações como união, diferença e intersecção de maneira usual.

Exemplo 18. *Para as linguagens do Exemplo 17 obteríamos*

$$L \cup R = \{mua, ha, hi\},\ L - R = \{mua\} \text{ e } L \cap R = \{ha\}. \tag{2.1.4}$$

Em especial, definiremos duas mais. A primeira é a concatenação de linguagens, um modo de generalizarmos concatenação de palavras para conjuntos de palavras.

Definição 19 (Concatenação de linguagens)**.** *Dada duas linguagens L e R, a concatenação de L e R, denotada por $\cdot$ ou apenas por justaposição, é uma nova linguagem tal que*

$$L \cdot R = \{f \cdot g \mid f \in L,\ g \in R\}. \tag{2.1.5}$$

Exemplo 20. *Os produtos das linguagens do Exemplo 17 seriam*

$$LR = \{muaha, muahi, haha, hahi\} \text{ e } RL = \{hamua, himua, haha, hiha\}.$$

Note que $LR \neq RL$.

Assim como a concatenação de palavras, a concatenação de linguagens é associativa e não-comutativa. Baseado na ideia de produto, podemos definir potências. Deste modo, L^n indica a concatenação de L por si própria n vezes, no qual n é um número natural. Além disso, a linguagem que faz o papel de *elemento neutro* neste caso é simplesmente a linguagem cujo único elemento é ε. Logo, $L^0 = \{\varepsilon\}$ para toda linguagem linguagem L.

A segunda operação é a *estrela de Kleene*. Assim como a potenciação, ela descreve repetições de uma linguagem, trazendo consigo a ideia de recursão, iterando sucessivas vezes em uma mesma linguagem.

Definição 21 (Estrela de uma linguagem)**.** *A estrela de Kleene de uma linguagem L, denotada por L^*, é o conjunto de todas as palavras formadas por um número finito de concatenações de elementos de L; isto é,*

$$L^* = \bigcup_{i \in \mathbb{N}} L^i \tag{2.1.6}$$

Exemplo 22. *Aplicando a operação de estrela em* $R = \{ha, hi\}$*, obteríamos*

$$R^* = \{\varepsilon, ha, hi, haha, hahi, hiha, hihi, hahaha, ...\}. \tag{2.1.7}$$

Ao contrário das operações que vimos até agora, a estrela é capaz de gerar uma linguagem infinita a partir de uma finita. Ela será de papel crucial em nosso trabalho e servirá como maneira natural de obtermos muitos de nossos resultados algébricos. Uma propriedade central da estrela é que esta satisfaz as igualdades

$$LL^* \cup \{\varepsilon\} = L^* = L^*L \cup \{\varepsilon\}. \tag{2.1.8}$$

Essa propriedade é um caso particular de um fato mais geral: L^*R é o *menor ponto prefixo* da função $f(X) = L \cdot X \cup R$.

Definição 23 (Ponto prefixo)**.** *x é ponto prefixo de uma função $f : X \to X$ se $f(x) \leq x$.*

Definição 24 (Ponto fixo)**.** *x é ponto fixo de uma função $f : X \to X$ se $f(x) = x$.*

Teorema 25 (L^*R é o menor ponto prefixo)**.** *L^*R é o menor ponto prefixo de $f(X) = L \cdot X \cup R$; isto é,*

$$L \cdot L^*R \cup R \subseteq L^*R; \tag{2.1.9}$$

$$L \cdot X \cup R \subseteq X \implies L^*R \subseteq X. \tag{2.1.10}$$

O Teorema 25 destaca que L^*R é ponto prefixo de $f(x) = L \cdot X \cup R$ (2.1.9) e que é o menor (2.1.10), isto é, que a linguagem L^*R está contida em qualquer outro ponto prefixo de f. Em geral, o menor ponto prefixo de uma função é também o menor ponto fixo dela.

Teorema 26 (Menor ponto prefixo é menor ponto fixo)**.** *Se x é o menor ponto prefixo de $f : X \to X$, então x também é o menor ponto fixo de f.*

Esse é um teorema amplamente conhecido e sua prova pode ser encontrada em diferentes fontes, cito o trabalho de Backhouse (BACKHOUSE, 2002). Portanto, podemos atestar que $L^*R = L \cdot (L^*R) \cup R$. Em particular, quando $R = \{\varepsilon\}$ obtemos a igualdade $L^* = L^*L \cup \{\varepsilon\}$, presente na Equação 2.1.8. Além disso, para qualquer linguagem S tal que

$$S = L \cdot S \cup R, \tag{2.1.11}$$

temos $L^*R \subseteq S$. Como já vimos, concatenação não é comutativa e, portanto, não podemos simplesmente trocar a ordem de L e R, todavia conseguimos um resultado bem semelhante para RL^*.

Teorema 27 (RL^* é o menor ponto prefixo)**.** *RL^* é o menor ponto prefixo de $f(X) = X \cdot L \cup R$; isto é,*

$$RL^* \cdot L \cup R \subseteq RL^*; \tag{2.1.12}$$

$$X \cdot L \cup R \subseteq X \implies RL^* \subseteq X. \tag{2.1.13}$$

2.1.2 Linguagens regulares

União, concatenação e estrela possuem outras propriedades interessantes. Para quaisquer linguagens L, R e S, vale para a operação de união que

$$L \cup (R \cup S) = (L \cup R) \cup S; \qquad \text{(associatividade da união)} \tag{2.1.14}$$

$$L \cup R = R \cup L; \qquad \text{(comutatividade da união)} \tag{2.1.15}$$

$$L \cup \emptyset = L. \qquad \text{(elemento neutro da união)} \tag{2.1.16}$$

Para a concatenação vale que esta é associativa e não-comutativa:

$$(L \cdot R) \cdot S = L \cdot (R \cdot S); \qquad \text{(associatividade da concatenação)} \tag{2.1.17}$$

$$\{\varepsilon\} \cdot L = L = L \cdot \{\varepsilon\}; \qquad \text{(elemento neutro do concatenação)} \tag{2.1.18}$$

$$\emptyset \cdot L = \emptyset = L \cdot \emptyset. \qquad \text{(produto pelo elemento neutro da união)} \tag{2.1.19}$$

A operação de concatenação distribui sobre a operação de união:

$$L \cdot (R \cup S) = L \cdot R \cup L \cdot S; \qquad \text{(distributividade à esquerda do produto)} \tag{2.1.20}$$

$$(L \cup R) \cdot S = L \cdot S \cup R \cdot S. \qquad \text{(distributividade à direita do produto)} \tag{2.1.21}$$

A operação de estrela satisfaz todas as propriedades abaixo:

$$LL^*R \cup R \subseteq L^*R; \qquad (L^*R \text{ é ponto prefixo}) \tag{2.1.22}$$

$$RL^*L \cup R \subseteq RL^*; \qquad (RL^* \text{ é ponto prefixo}) \tag{2.1.23}$$

$$LX \cup R \subseteq X \implies L^*R \subseteq X; \qquad (L^*R \text{ é menor ponto prefixo}) \tag{2.1.24}$$

$$XL \cup R \subseteq X \implies RL^* \subseteq X. \qquad (RL^* \text{ é menor ponto prefixo}) \tag{2.1.25}$$

Além disso, todas as três operações são monótonas. Portanto, se $L \subseteq L'$ e $R \subseteq R'$, então

$$L \cup R \subseteq L' \cup R'; \qquad \text{(monotonicidade da união)} \tag{2.1.26}$$

$$L \cdot R \subseteq L' \cdot R'; \qquad \text{(monotonicidade da concatenação)} \tag{2.1.27}$$

$$L^* \subseteq L'^*. \qquad \text{(monotonicidade da estrela)} \tag{2.1.28}$$

A partir das propriedades listadas acima, podemos derivar todas as abaixo:

$$L^* = LL^* \cup \{\varepsilon\}; \qquad \text{(definição recursiva da estrela)} \tag{2.1.29}$$

$$LL^* = L^*L; \qquad \text{(comutatividade da estrela)} \tag{2.1.30}$$

$$L^* = (L^*)^*; \qquad \text{(idempotência da estrela)} \tag{2.1.31}$$

$$L^* = L^*L^*; \qquad \text{(transitividade da estrela)} \tag{2.1.32}$$

$$\{\varepsilon\} \subseteq L^*; \qquad \text{(reflexividade da estrela)} \tag{2.1.33}$$

$$L \subseteq R^* \implies L^* \subseteq R^*. \qquad \text{(propriedade de fecho da estrela)} \tag{2.1.34}$$

Destacamos também a regra do *shift* e a distributividade da estrela sobre a soma:

$$L(RL)^* = (LR)^*L; \qquad \text{(regra do } \textit{shift} \text{ da estrela)} \qquad (2.1.35)$$

$$(L + R)^* = (L^*R)^*L^*. \qquad \text{(distributividade da estrela)} \qquad (2.1.36)$$

Pelas equações 2.1.14 e 2.1.17, as operações de união e concatenação são associativas e, pelas equações 2.1.26, 2.1.27 e 2.1.28, todas as três operações são monótonas. Ser idempotente, indicado na equação 2.1.31, implica que aplicar a operação estrela duas vezes em uma mesma linguagem surte o mesmo efeito que aplicar apenas uma. A Propriedade 2.1.34 torna a estrela de Kleene um *operador de fecho* e as propriedades 2.1.32 e 2.1.33 lhe dão a alcunha de *fecho transitivo e reflexivo* da linguagem, que encurtaremos para somente *fecho*. Repare que ser monótona e idempotente provam a Propriedade 2.1.34 de fecho. Linguagens são fechadas em todas as três operações que definimos. As provas e demonstrações, assim como tabelas semelhantes, podem ser encontradas em diversos textos introdutórios de linguagens formais, dentre eles os trabalhos de Kozen (KOZEN, 1994) e Sakarovitch (SAKAROVITCH, 2009). Outro trabalho, certamente não-introdutório, mas com provas de algumas propriedades é a tese de doutorado de Alexandra Silva (SILVA, 2010).

Como modelamos nossos problemas em termos de linguagens, uma pergunta natural de se fazer é: dada uma linguagem, como verificamos se uma certa palavra pertence ou não à ela? Esta é uma das principais preocupações em teoria de linguagens formais, entretanto nem sempre é fácil respondê-la. Linguagens podem ser infinitas e, a princípio, não temos razão alguma para crer que haja uma boa construção ou caracterização de todos os seus elementos. Devido à sua já grande importância, neste trabalho forcaremos especificamente em *linguagens regulares*, assim como em um mecanismo verificador para elas.

Definição 28 (Linguagens Regulares)**.** *As linguagens regulares são a menor classe que contém as linguagens finitas e é fechada por união, concatenação e estrela de Kleene.*

Exemplo 29. *Qualquer linguagem finita é regular. Operar linguagens finitas via união, concatenação e estrela de Kleene resulta, novamente, em uma linguagem regular.*

As aplicações de linguagens regulares são vastas. Muitos verificadores de texto utilizam linguagens regulares, assim como certos sistemas de modelagem de programas concorrentes e analisadores léxicos de linguagens de programação. Estudá-las já traz muitas intuições sobre como podemos pensar linguagens formais de maneira algébrica. É este o rumo que tomaremos pelo restante da monografia.

2.2 AUTÔMATOS FINITOS

Encerramos a seção anterior acerca de como podemos reconhecer palavras de uma dada linguagem. Uma das estratégias para solucionar esta questão consiste em utilizar os chamados *autômatos*. Em especial, falaremos de *autômatos finitos*, os reconhecedores das linguagens regulares. Para nós, autômatos finitos serão grafos direcionados cujas arestas terão como rótulos elementos de um alfabeto A. De maneira mais formal:

Definição 30 (Autômatos Finitos Não-Determinísticos (Sakarovitch))**.** *Um Autômato Finito Não-Determinístico (AFND)* $\mathcal{M} = \langle Q, A, E, I, T\rangle$ *é composto por*

- *Um conjunto finito* Q*, nomeado o conjunto de estados de* $\mathcal{M}$*;*
- *Um conjunto finito* A*, chamado de alfabeto;*
- *Um subconjunto* E *de* $Q \times A \times Q$*, representando as transições do autômato;*
- *Um subconjunto* I *de* Q*, chamado de conjunto de estados iniciais;*
- *Um subconjunto* T *de* Q*, chamado de conjunto de estados finais.*

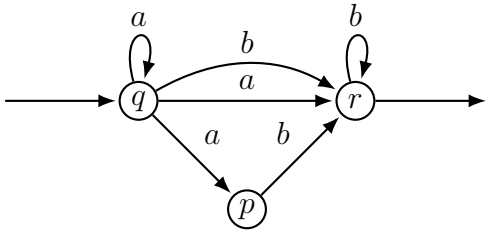


Figura 5 – Exemplo de autômato. A seta sem rótulos apontando para q indica que este é um estado inicial e a seta sem rótulos saindo de r indica que este é um estado final. As arestas representam as transições, de modo que o seu rótulo indica que símbolos devemos ler para que a transição ocorra.

Exemplo 31. *Tomemos o autômato* $\mathcal{M}$ *da Figura 5, este assume a forma*

$$\mathcal{M} = \langle Q, A, E, I, T\rangle,$$

para o qual

$$\begin{aligned} &Q = \{q, p, r\},\ A = \{a, b\},\\ &E = \{(q, a, q), (q, a, r), (q, b, r), (q, a, p), (p, b, r), (r, b, r)\},\\ &I = \{q\}\ e\ T = \{r\}. \end{aligned}$$

O alfabeto A contém os possíveis valores de rótulo para as aresta do grafo. Cada elemento do conjunto de estados Q, dito um *estado*, é um vértice do grafo. Todo estado pertencente ao conjunto de estados iniciais I é representado por uma seta sem rótulos

apontando para ele. De maneira semelhante, todo estado pertencente ao conjunto de estados finais T é indicado por uma seta saindo dele e sem rótulos. Cada elemento do conjunto de transições E representa uma aresta do grafo, no qual a primeira entrada representa a origem da aresta, a segunda contém o rótulo da aresta e a terceira o seu estado de destino. Há a possibilidade de um mesmo estado ser inicial e final.

Exemplo 32. *Suponha o autômato $\mathcal{N}$ da Figura 6. O estado q é tanto um estado inicial quanto um estado final. Casos como esse sempre fazem com que a palavra vazia ε seja uma palavra reconhecida pelo autômato.*

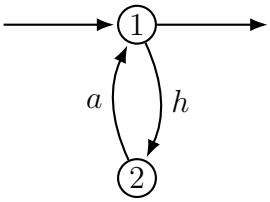


Figura 6 – Autômato de exemplo. A seta sem rótulo apontando para q indica que este é um estado inicial e a seta sem rótulo saindo de q indica que este também é um estado final. Não há problemas em termos um mesmo estado sendo inicial e final.

Como sabemos quais palavras são ou não reconhecidas pelo autômato? Bom, fixamos quais vértices podem ser origem e quais podem ser o destino do caminho. Todo caminho que tenha sua origem e destino dentro destes conjuntos previamente definidos serão caminhos válidos e as palavras aceitas pelo autômato serão seus rótulos.

Definição 33 (Linguagem reconhecida por um autômato)**.** *Seja $\mathcal{M} = \langle Q, A, E, I, T\rangle$ um AFND, se existe um caminho de rótulo f que comece em um estado inicial e termine em um estado final, então f é palavra reconhecida pelo autômato. A linguagem do autômato, denotada por $\mathcal{L}(\mathcal{M})$, é o conjunto de palavras reconhecidas por ele.*

Exemplo 34. *Suponha o autômato $\mathcal{N}$ da Figura 6. A palavra haha é reconhecida por $\mathcal{N}$ e o único caminho válido é $q \xrightarrow{h} p \xrightarrow{a} q \xrightarrow{h} p \xrightarrow{a} q$. A palavra hah não é reconhecida, visto que o único caminho com este rótulo começa em q e termina em p.*

Uma dúvida que se pode ter é: para quais linguagens existe algum autômato finito que a reconheça? Este é um resultado clássico em teoria de linguagens formais e teoria dos autômatos, as linguagens reconhecidas por autômatos finitos são exatamente as linguagens regulares.

Teorema 35 (Teorema de Kleene)**.** *Uma linguagem L é regular se, e somente se, existe um autômato finito não-determinístico $\mathcal{M}$ que a reconheça.*

Deixo a prova de lado, mas o leitor pode verificá-la em diferentes fontes. Como sugestões, temos os trabalhos de Sakarovitch (SAKAROVITCH, 2009) e o livro *Introduction*

to Automata Theory, Language and Computation (HOPCROFT; MOTWANI; ULLMAN, 2001).

2.3 SEMIANÉIS E ÁLGEBRAS DE KLEENE

Devido ao uso extensivo de propriedades algébricas como associatividade e distributividade, optaremos por utilizar uma notação que reforce mais tais ideias. Para tal, utilizaremos símbolos como $+$ e $\cdot$ para uma maior familiaridade. Para raciocinar sobre autômatos com mais de um estado precisaremos de sistemas de equações, no qual lidaremos utilizando matrizes.

Faremos essa representação através de *semianéis*. A principal diferença entre semianéis e *anéis* é que semianéis não exigem a presença de inverso aditivo. As seções seguintes nos servirão como uma forma de generalizar os conceitos de linguagens formais que vimos previamente na Página 26, faremos isso através de estruturas algébricas que sejam suficientemente expressivas para nosso contexto. Penso ser importante ler estas seções seguintes com isto em mente: tudo que estamos vendo já foi visto sob uma óptica menos generalista.

Definição 36 (Semianel)**.** *Um semianel* $\mathcal{S} = \langle S, +, \cdot, 0, 1\rangle$ *é composto por*

- *Um conjunto* S*;*
- *Uma operação de soma* $+$*, de assinatura* $+ : S \times S \to S$*;*
- *Uma operação de produto* $\cdot$*, de assinatura* $\cdot : S \times S \to S$*;*
- *Um elemento neutro da soma denotado por* 0*;*
- *Um elemento neutro do produto denotado por* 1*.*

Exigimos para a soma que $\langle S, +, 0\rangle$ *forme um monoide comutativo. Isto é,*

$$a + (b + c) = (a + b) + c; \qquad (\textit{associatividade da soma}) \tag{2.3.1}$$

$$a + b = b + a; \qquad (\textit{comutatividade da soma}) \tag{2.3.2}$$

$$a + 0 = a. \qquad (\textit{elemento neutro da soma}) \tag{2.3.3}$$

Do produto, exigimos que $\langle S, \cdot, 1\rangle$ *forme um monoide. Em outras palavras,*

$$(a \cdot b) \cdot c = a \cdot (b \cdot c); \qquad (\textit{associatividade do produto}) \tag{2.3.4}$$

$$1 \cdot a = a = a \cdot 1. \qquad (\textit{elemento neutro do produto}) \tag{2.3.5}$$

A operação de produto distribui sobre a soma:

$$a \cdot (b + c) = a \cdot b + a \cdot c; \qquad (\textit{distributividade esquerda do produto}) \tag{2.3.6}$$

$$(a + b) \cdot c = a \cdot c + b \cdot c; \qquad (\textit{distributividade direita do produto}) \tag{2.3.7}$$

$$0 \cdot a = 0 = a \cdot 0. \qquad (\textit{produto pelo elemento neutro da soma}) \tag{2.3.8}$$

Repare que substituindo a operação de soma e produto por, respectivamente, união e concatenação de linguagens satisfazemos todas as propriedades listadas. De fato, se observarmos as propriedades de união e concatenação previamente vistas na Página 26, notamos que todas as propriedades de semianel se encontram ali, sendo o conjunto vazio o elemento neutro da soma (neste caso união) e o conjunto unitário contendo a palavra vazia o elemento neutro do produto (neste caso concatenação).

Definição 37 (Semianel idempotente)**.** *Um semianel é idempotente se, para todo elemento a em S, vale que*

$$a + a = a. \tag{2.3.9}$$

Além desta estrutura de semianel, definiremos sobre seus elementos uma ordem parcial $\leq$ separadamente. A ideia é que queremos tal ordem, entretanto não há necessidade de adicioná-la à estrutura do semianel, visto que podemos a definir utilizando somente a operação de soma. A ordem $\leq$ é definida de modo que

$$a \leq b \quad \equiv \quad a + b = b. \tag{2.3.10}$$

Fixado um alfabeto, as linguagens formais sobre este alfabeto formam um semianel idempotente $\langle \mathcal{P}(A^*), \cup, \cdot, \emptyset, \{\varepsilon\} \rangle$, pois $L \cup L = L$ para qualquer linguagem. Apesar de facilitar trabalharmos com linguagens de maneira algébrica, não conseguimos determinar a operação estrela apenas com a estrutura de semianel idempotente, precisamos de algo mais. Como comentamos, a operação de estrela é um *operador de fecho*, adicionando uma operação desta natureza em um semianel idempotente permite que a estrela seja construída. Tais estruturas são chamadas de *álgebras de Kleene*. Dessa maneira, o leitor poderá observar uma relação de 1-para-1 entre as propriedades de união e concatenação previamente vistas na Página 26 e as listadas na definição de álgebra de Kleene.

Definição 38 (Álgebra de Kleene)**.** *Uma álgebra de Kleene $\mathcal{K} = \langle S, +, \cdot, ^*, 0, 1 \rangle$ é uma estrutura na qual*

- $\langle S, +, \cdot, 0, 1 \rangle$ *forma um semianel idempotente;*
- *A operação unária *, de assinatura $^* : S \to S$, satisfaz todas as propriedades abaixo:*

$$1 + a \cdot a^* \leq a^*; \qquad (\textit{definição de estrela}) \tag{2.3.11}$$

$$1 + a^* \cdot a \leq a^*; \qquad (\textit{definição de estrela}) \tag{2.3.12}$$

$$a \cdot x + b \leq x \implies a^* \cdot b \leq x; \quad (a^*b \textit{ é o menor ponto fixo de } ax + b \leq x) \tag{2.3.13}$$

$$x \cdot a + b \leq x \implies b \cdot a^* \leq x. \quad (ba^* \textit{ é o menor ponto fixo de } xa + b \leq x) \tag{2.3.14}$$

Acerca da relação de ordem parcial $\leq$, no contexto de linguagens formais esta se reduz à relação de inclusão, $\subseteq$. Os quatro axiomas exigidos para a operação de fecho aparecem

ao fim das propriedades presentes na Página 26, tornando o conjunto das linguagens sobre um dado alfabeto uma álgebra de Kleene com a união de linguagens representando a soma e a concatenação de linguagens o produto. Utilizando a notação introduzida acima,

$$\langle \mathcal{P}(A^*), \cup, \cdot, ^*, \emptyset, \{\varepsilon\} \rangle.$$

Denotaremos os elementos de um semianel por letras minúsculas, mas tenha em mente que estes poderiam ser as linguagens que vimos previamente. Do mesmo modo, seguiremos com a notação + e $\cdot$ para união e concatenação, respectivamente. Como exemplo, vamos demonstrar alguns resultados antes citados nas seções anteriores. Recorde que L^*R era ponto prefixo de $f(X) = L \cdot X \cup R$, o axioma 2.3.13 representa esse fato. De maneira similiar, o axioma 2.3.14 atesta que RL^*é ponto prefixo de $f(X) = XL \cup R$.

Lema 39. *(a^*b como ponto prefixo) a^*b é ponto prefixo de $f(x) = ax + b$.*

Prova.

$$\begin{aligned} & a(a^*b) + b \\ = \quad & \{ \text{ distributividade e associatividade } \} \\ & (aa^* + 1)b \\ \leq \quad & \{ \text{ axioma 2.3.11 } \} \\ & a^*b \end{aligned}$$

□

Lema 40. *(b^*a como ponto prefixo) ba^* é ponto prefixo de $f(x) = xa + b$.*

Prova.

$$\begin{aligned} & (ba^*)a + b \\ = \quad & \{ \text{ distributividade e associatividade. } \} \\ & b(a^*a + 1) \\ \leq \quad & \{ \text{ axioma 2.3.12. } \} \\ & ba^* \end{aligned}$$

□

Provas neste formato são chamadas de *calculacionais*, porque derivamos os passos de maneira puramente sintática, de modo que os comentários entre chaves indicam que passo tomamos para ir de uma linha à outra. Baseado nos axiomas de a^*b e ba^* como os menores pontos prefixos, pelo Teorema 26, da Página 25, estes são (os menores) *pontos fixos* também. Além disso, fixando $b = 1$ conseguimos obter as igualdades apresentadas nas Equações 2.1.8, da Página 25, como indicado no Corolário 43.

Corolário 41 (a^*b é ponto fixo)**.** *a^*b é ponto fixo de $f(x) = ax + b$; isto é,*

$$a(a^*b) + b = a^*b. \tag{2.3.15}$$

Corolário 42 (ba^* é ponto fixo)**.** *ba^* é ponto fixo de $f(x) = xa + b$; isto é,*

$$(ba^*)a + b = ba^*. \tag{2.3.16}$$

Corolário 43 (Operação estrela como igualdade)**.** *A operação estrela satisfaz as seguintes igualdades:*

$$aa^* + 1 = a^* = a^*a + 1. \tag{2.3.17}$$

A intuição, portanto, é que a^* é um gerador de novos a justapostos em ambas as direções. O corolário acima nos permite demonstrar a Propriedade 2.1.30, da Página 2.1.30, de comutatividade da estrela.

$$\begin{aligned}
& aa^* \\
= \quad & \{ \text{ definição de } a^* \text{ (Corolário 43). } \} \\
& a(a^*a + 1) \\
= \quad & \{ \text{ distributividade e associatividade. } \} \\
& aa^*a + a \\
= \quad & \{ \text{ distributividade. } \} \\
& (aa^* + 1)a \\
= \quad & \{ \text{ definição de } a^* \text{ (Corolário 43). } \} \\
& a^*a
\end{aligned}$$

Podemos fazer um paralelo direto desta representação com a que estávamos utilizando

anteriormente. Como propriedades do operador estrela de uma álgebra de Kleene temos

$$0^* = 1; \qquad \text{(fecho do elemento neutro da soma)} \qquad (2.3.18)$$

$$1^* = 1; \qquad \text{(fecho do elemento neutro do produto)} \qquad (2.3.19)$$

$$a(a^*b) + b = a^*b; \qquad (a^*b \text{ ponto fixo de } ax + b) \qquad (2.3.20)$$

$$(ba^*)a + b = ba^*; \qquad (b^*a \text{ ponto fixo de } xa + b) \qquad (2.3.21)$$

$$a^* = aa^* + 1; \qquad \text{(definição recursiva da estrela)} \qquad (2.3.22)$$

$$aa^* = a^*a; \qquad \text{(comutatividade da estrela)} \qquad (2.3.23)$$

$$a^* = (a^*)^*; \qquad \text{(idempotência da estrela)} \qquad (2.3.24)$$

$$a^* = a^*a^*; \qquad \text{(transitividade da estrela)} \qquad (2.3.25)$$

$$1 \leq a^*; \qquad \text{(reflexividade da estrela)} \qquad (2.3.26)$$

$$a \leq b^* \iff a^* \leq b^*; \qquad \text{(propriedade de fecho da estrela)} \qquad (2.3.27)$$

$$a \leq b \implies a^* \leq b^*; \qquad \text{(monotonicidade da estrela)} \qquad (2.3.28)$$

$$a \leq a^*; \qquad \text{(elemento menor ou igual a seu fecho)} \qquad (2.3.29)$$

$$(ab)^*a = a(ba)^*; \qquad \text{(regra do } \textit{shift}) \qquad (2.3.30)$$

$$(a + b)^* = a^*(ba^*)^*. \qquad \text{(distributividade da estrela sob a soma)} \qquad (2.3.31)$$

Muitas destas aparecem nas propriedades da Página 26. Isto ocorre devido ao fato que já mencionamos: Linguagens formais podem ser representadas como uma álgebra de Kleene. Com isto em mente, não há a necessidade de reprovarmos os resultados, pois adotamos esta notação apenas para uma maior familiaridade nas manipulações algébricas. Ainda assim, caso o leitor queira buscar provas para algumas das propriedades que não demonstramos aqui, elas podem ser encontradas nos trabalhos de Sakarovitch (SAKAROVITCH, 2009) e Kozen (KOZEN, 1994).

2.4 MATRIZES

Qualquer semianel sobre um conjunto S pode ser generalizado para um novo semianel: o conjunto de matrizes quadradas de dimensão n, cujos elementos pertencem à S, denotado $S^{n\times n}$. Ressalto que este resultado, apesar de também ser verdade para álgebras de Kleene, não é direto devido ao operador de fecho. Precisamos construir esta nova noção de fecho para matrizes de modo que todos os axiomas de uma álgebra de Kleene para a operação de estrela sejam respeitados. Logo abaixo trazemos uma prova construtiva para o fecho de uma matriz, já que não encontramos na literatura.

Teorema 44 (Fecho de uma matriz)**.** *Seja $\mathcal{K} = \langle S, +, \cdot, ^*, 0, 1\rangle$ uma álgebra de Kleene sobre S e sejam A, B, X matrizes em $S^{n\times n}$. Existe matriz M em $S^{n\times n}$ tal que*

$$X \geq A \cdot X + B \implies X \geq M \cdot B. \qquad (2.4.1)$$

Prova. Vamos quebrar nossas matrizes em blocos.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \quad \text{e} \quad M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}. \tag{2.4.2}$$

Para o qual, fixado um k que respeite $0 < k < n$, as dimensões dos blocos seguem

- A_{11} e A_{22} são matrizes quadradas de dimensões $k \times k$ e $(n-k) \times (n-k)$, respectivamente;
- A_{12} e A_{21} são matrizes retangulares de dimensões $k \times (n-k)$ e $(n-k) \times k$, respectivamente;
- X_1 e B_1 são matrizes retangulares $k \times n$;
- X_2 e B_2 são matrizes retangulares $(n-k) \times n$;
- Todos os sub-blocos de M têm as mesmas dimensões de seu correspondente em A.

Note, portanto, que as dimensões dos blocos não possuem tantas exigências sobre seu formato, exigindo somente que os blocos diagonais A_{11} e A_{22} de A sejam quadrados. A prova que faremos será por indução no tamanho da matriz. De caso base, temos a situação onde $n = 1$ e reduzimos o problema para uma matriz de um único elemento de S, que pela hipótese do teorema compõe uma álgebra de Kleene e, portanto, a operação de estrela satisfaz a condição desejada. Para o caso recursivo, assumamos que o teorema vale para valores menores que n, ou seja, todo sub-bloco de $AX + B$ satisfaz a Propriedade 2.4.1.

$$\begin{aligned}
& X \geq MB \\
\iff \quad & \{ \text{ expansão em matrizes por blocos (Equação 2.4.2). } \} \\
& \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \geq \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \\
\iff \quad & \{ \text{ produto. } \} \\
& \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \geq \begin{bmatrix} M_{11}B_1 + M_{12}B_2 \\ M_{21}B_1 + M_{22}B_2 \end{bmatrix} \\
\impliedby \quad & \{ \text{ hipótese do teorema na Equação 2.4.1. } \} \\
& \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \geq \begin{bmatrix} A_{11}X_1 + A_{12}X_2 + B_1 \\ A_{21}X_1 + A_{22}X_2 + B_2 \end{bmatrix} \\
\iff \quad & \{ \text{ fatoração. } \} \\
& \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \geq \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}
\end{aligned}$$

Chegamos em um sistema de duas inequações. As únicas parcelas que desconhecemos são X_1 e X_2, a ideia da construção de M é descrever ambos apenas em termos de blocos de A e B.

$$
\begin{aligned}
& X_1 \geq A_{11}X_1 + A_{12}X_2 + B_1 \\
& X_2 \geq A_{21}X_1 + A_{22}X_2 + B_2 \\
\Longrightarrow \quad & \{ \text{ hipótese de indução na primeira inequação. } \} \\
& X_1 \geq A_{11}^* \left(A_{12}X_2 + B_1\right) \\
& X_2 \geq A_{21}X_1 + A_{22}X_2 + B_2 \\
\Longrightarrow \quad & \{ \text{ monotonicidade. } \} \\
& X_1 \geq A_{11}^* \left(A_{12}X_2 + B_1\right) \\
& X_2 \geq A_{21}\left(A_{11}^* \left(A_{12}X_2 + B_1\right)\right) + A_{22}X_2 + B_2 \\
\Longleftrightarrow \quad & \{ \text{ distributividade na segunda inequação. } \} \\
& X_1 \geq A_{11}^* \left(A_{12}X_2 + B_1\right) \\
& X_2 \geq A_{21}A_{11}^*A_{12}X_2 + A_{21}A_{11}^*B_1 + A_{22}X_2 + B_2 \\
\Longleftrightarrow \quad & \{ \text{ distributividade em } X_2 \text{ na segunda inequação. } \} \\
& X_1 \geq A_{11}^* \left(A_{12}X_2 + B_1\right) \\
& X_2 \geq \left(A_{21}A_{11}^*A_{12} + A_{22}\right)X_2 + A_{21}A_{11}^*B_1 + B_2 \\
\Longrightarrow \quad & \{ \text{ hipótese de indução na segunda inequação. } \} \\
& X_1 \geq A_{11}^* \left(A_{12}X_2 + B_1\right) \\
& X_2 \geq \left(A_{21}A_{11}^*A_{12} + A_{22}\right)^* \left(A_{21}A_{11}^*B_1 + B_2\right) \\
\Longrightarrow \quad & \{ \text{ monotonicidade e } \Delta = A_{21}A_{11}^*A_{12} + A_{22}. \} \\
& X_1 \geq A_{11}^* \left(A_{12}\left(\Delta^* \left(A_{21}A_{11}^*B_1 + B_2\right)\right) + B_1\right) \\
& X_2 \geq \Delta^* \left(A_{21}A_{11}^*B_1 + B_2\right) \\
\Longleftrightarrow \quad & \{ \text{ distributividade na primeira e segunda inequação. } \} \\
& X_1 \geq A_{11}^*A_{12}\Delta^*A_{21}A_{11}^*B_1 + A_{11}^*A_{12}\Delta^*B_2 + A_{11}^*B_1 \\
& X_2 \geq \Delta^*A_{21}A_{11}^*B_1 + \Delta^*B_2 \\
\Longleftrightarrow \quad & \{ \text{ distributividade em } B_1 \text{ e } B_2 \text{ na primeira e segunda inequação. } \} \\
& X_1 \geq \left(A_{11}^*A_{12}\Delta^*A_{21}A_{11}^* + A_{11}^*\right)B_1 + A_{11}^*A_{12}\Delta^*B_2 \\
& X_2 \geq \Delta^*A_{21}A_{11}^*B_1 + \Delta^*B_2
\end{aligned}
$$

Substituindo estes valores de X no lado esquerdo das condições temos a matriz M que

satisfaz ambas.

$$
\begin{aligned}
& X_1 \geq M_{11}B_1 + M_{12}B_2 \\
& X_2 \geq M_{21}B_1 + M_{22}B_2 \\
\implies \quad & \{ \text{ monotonicidade. } \} \\
& \left(A_{11}^* A_{12}\Delta^* A_{21}A_{11}^* + A_{11}^*\right) B_1 + A_{11}^* A_{12}\Delta^* B_2 \geq M_{11}B_1 + M_{12}B_2 \\
& \Delta^* A_{21}A_{11}^* B_1 + \Delta^* B_2 \geq M_{21}B_1 + M_{22}B_2 \\
\iff \quad & \{ \text{ forma matricial. } \} \\
& \begin{bmatrix} A_{11}^* A_{12}\Delta^* A_{21}A_{11}^* + A_{11}^* & A_{11}^* A_{12}\Delta^* \\ \Delta^* A_{21}A_{11}^* & \Delta^* \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \geq \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}
\end{aligned}
$$

Portanto, temos que um candidato a M é justamente

$$
M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = \begin{bmatrix} A_{11}^* A_{12}\Delta^* A_{21}A_{11}^* + A_{11}^* & A_{11}^* A_{12}\Delta^* \\ \Delta^* A_{21}A_{11}^* & \Delta^* \end{bmatrix}. \tag{2.4.3}
$$

Por hipótese, a matriz M é quadrada e pertence à $S^{n\times n}$, portanto A^* está bem definido e $S^{n\times n}$ é fechado sobre a operação estrela. □

A prova do Teorema 44 é a primeira prova construtiva para o fecho de matrizes que temos conhecimento. Provida a forma fechada de A^*, concluímos que o conjunto de matrizes quadradas com elementos em S, de fato, é uma álgebra de Kleene.

Teorema 45 (Álgebras de Kleene estendem para matrizes)**.** *Seja $\mathcal{K}$ álgebra de Kleene sobre um conjunto S e seja $S^{n\times n}$ conjunto de matrizes $n \times n$ cujas entradas pertencem a S. Então*

$$
\mathcal{K}_n = \langle S^{n\times n}, +, \cdot, ^*, 0_{n\times n}, I_{n\times n} \rangle \tag{2.4.4}
$$

também é uma álgebra de Kleene. A matriz $0_{n\times n}$ é a matriz nula de dimensões $n \times n$ com todas as suas entradas valendo 0 e $I_{n\times n}$ é a matriz identidade de dimensões $n \times n$.

Prova. A demonstração para soma e produto é direta. Para o fecho, utilizamos a forma encontrada pelo Teorema 44. □

Pelas matrizes de elementos em S comporem novamente uma álgebra de Kleene, temos todas as propriedades da operação de estrela listadas na Página 34 como válidas para matrizes também. Apesar de a forma de A^* parecer intimidadora, esta traz consigo uma semântica bem interessante e que exploraremos um pouco.

2.4.1 A^* como caminhos em um grafo

A operação de fecho é tão central em nosso trabalho que é válido conseguirmos um pouco mais de intuição acerca de o quê, de fato, o fecho significa. Realmente, se tratarmos A como a matriz de transição de um grafo de dois vértices, a forma de A^* representa todos

os caminhos de comprimento arbitrário entre estes vértices. Ilustremos isto em detalhes. Suponha o grafo da Figura 6, cuja matriz de transição é

$$E = \begin{bmatrix} 0 & h \\ a & 0 \end{bmatrix}. \tag{2.4.5}$$

A semântica de E_{ij} é o rótulo da transição do vértice i ao vértice j. As potências de E aumentam o comprimento deste caminho:

$$E^2 = \begin{bmatrix} 0 & h \\ a & 0 \end{bmatrix} \begin{bmatrix} 0 & h \\ a & 0 \end{bmatrix} = \begin{bmatrix} ha & 0 \\ 0 & ah \end{bmatrix}. \tag{2.4.6}$$

Isto é, E^2_{ij} agora indica todos os caminhos de comprimento dois entre os vértices i e j. Se queremos considerar tanto os de comprimento um quanto os de comprimento dois, basta somarmos ambas as matrizes:

$$E + E^2 = \begin{bmatrix} 0 & h \\ a & 0 \end{bmatrix} + \begin{bmatrix} ha & 0 \\ 0 & ah \end{bmatrix} = \begin{bmatrix} ha & h \\ a & ah \end{bmatrix}. \tag{2.4.7}$$

Podemos generalizar isso para qualquer potência $n \in \mathbb{N}$ de E.

$$E^n := \text{ Matriz de transição de caminhos de comprimento } n \text{ do grafo.}$$

Por definição de estrela temos

$$E^* = \sum_{i \in \mathbb{N}} E^i, \tag{2.4.8}$$

que expõe todos os caminhos do grafo, de qualquer comprimento. Para a matriz E da Equação 2.4.5 em específico obtemos

$$E^* = \begin{bmatrix} h(ah)^*a + 1 & h(ah)^* \\ (ah)^*a & (ah)^* \end{bmatrix} = \begin{bmatrix} (ha)^* & h(ah)^* \\ (ah)^*a & (ah)^* \end{bmatrix}. \tag{2.4.9}$$

A entrada $(ah)^*$ indica que todos os caminhos que se iniciam no estado p e terminam em p possuem uma forma bem definida.

- Primeiro começamos no estado p e transitamos para q através da aresta de rótulo a;
- Em seguida, transitamos do estado q para o estado p via uma aresta de rótulo h.

O operador estrela indica que podemos fazer isto sucessivas vezes, incluindo nenhuma. Convido o leitor a analisar as outras entradas da matriz em detalhes, entretanto deixo na Figura 7 uma descrição da entrada $h(ah)^*$. Esta compreensão de caminhos em grafos é riquíssima e surge em diferentes contextos.

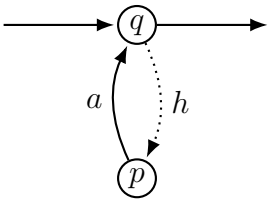


Figura 7 – Representação passo-a-passo de todos os caminhos possíveis no grafo. Para os caminhos $h(ah)^*$ entre q e p podemos percorrer a aresta h e, em seguida, percorrer o caminho ah quantos vezes quisermos, incluindo nenhuma. Percorrer ah siginifica percorrer a aresta a e depois a aresta pontilhada h.

2.5 LINGUAGENS E AUTÔMATOS, AGORA MATRICIALMENTE

Recomecemos mais uma vez, mas agora com todo o arcabouço teórico que desenvolvemos. Representaremos linguagens como somas de elementos de A^*, em que podemos pensar acerca de conjuntos maiores como a união de conjuntos unitários. Estes elementos serão as palavras pertencentes à linguagem, operadas com a operação de soma.

Exemplo 46. *Relembre a linguagem $R = \{ha, hi\}$. No formalismo algébrico a representaríamos como*

$$R = ha + hi. \tag{2.5.1}$$

E seu fecho transitivo e reflexivo como um somatório infinito:

$$R^* = \sum_{i \in \mathbb{N}} R^i. \tag{2.5.2}$$

Para os conhecedores de linguagens formais, estas não são nada mais do que as *expressões regulares*, um modo compacto de representar linguagens, em especial associadas a um autômato conhecido. Autômatos, por sua vez, são um pouco menos diretos. Por esta razão, vamos nos basear na representação gráfica já discutida para defini-los de maneira algébrica. Podemos pensar em um autômato como composto por três componentes principais: um conjunto de estados iniciais, um conjunto de estados finais e, por fim, um grafo.

Definição 47 (AFNDs como matrizes de grafos direcionados (Sakarovitch, Kozen))**.** *Um Autômato Finito Não-Determinístico (AFND) sobre um alfabeto A é um grafo direcionado com dois conjuntos especiais de vértices chamados de iniciais e finais. Denotamos o autômato por $\mathcal{M}$ e o representamos como*

$$\mathcal{M} = \langle u, E, v \rangle, \tag{2.5.3}$$

em que

- *$E \in \mathcal{P}(A)^{n \times n}$ é a matriz de transição do grafo;*
- *$u \in \{0, 1\}^n$ é seu vetor indicador de estados iniciais;*

- $v \in \{0,1\}^n$ *é seu vetor indicador de estados finais.*

Assim como temos pela definição usual, cada elemento possui um significado.

- $E \in \mathcal{P}(A)^{n\times n}$ representa a *matriz de transição* do autômato e lemos E_{ij} como os possíveis valores de transição do estado i para o estado j. Cada dimensão da matriz representa um dos n estados do autômato;

- u é o vetor de *estados iniciais*, ou seja,

$$u_i = \begin{cases} 1, \text{ se, e somente se, } i \text{ é estado inicial do autômato;} \\ 0, \text{ caso contrário.} \end{cases} \tag{2.5.4}$$

- v é o vetor de *estados finais*, ou seja,

$$v_i = \begin{cases} 1, \text{ se, e somente se, } i \text{ é estado final do autômato;} \\ 0, \text{ caso contrário.} \end{cases} \tag{2.5.5}$$

Nesta representação, os estados de Q estão implícitos nas coordenadas de E e dos vetores u e v e, a princípio, podemos fixar qualquer ordenação dos estados na hora de construir u, E e v.

Exemplo 48. *O autômato da Figura 5 teria como forma matricial* $\mathcal{M} = \langle u, E, v\rangle$. *No qual*

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad E = \begin{bmatrix} a & a & a+b \\ 0 & 0 & b \\ 0 & 0 & b \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{2.5.6}$$

A primeira coordenada é referente a q, *a segunda a* p *e a terceira a* r.

Exemplo 49. *O autômato da Figura 6 teria como forma matricial* $\mathcal{N} = \langle u, E, v\rangle$. *No qual*

$$u = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad E = \begin{bmatrix} 0 & h \\ a & 0 \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.5.7}$$

A primeira coordenada é referente a q *e a segunda a* p.

Por definição, a matriz E contém todos os caminhos de comprimento um contidos no autômato, as chamadas *transições*. Se aplicarmos u^T à esquerda de E, conseguimos todas as arestas do grafo cuja origem é algum estado inicial. Este comportamento não é exclusivo de u, qualquer vetor de somente 0 e 1 em suas entradas quando aplicado em E à esquerda possui o efeito de filtrar linhas em particular da matriz de transição.

Exemplo 50. *Computando $u^T E$ para o autômato do Exemplo 49:*

$$u^T E = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & h \\ a & 0 \end{bmatrix} = \begin{bmatrix} 0 & h \end{bmatrix}. \tag{2.5.8}$$

Isto é, estamos filtrando as linhas referentes às coordenadas em que u vale 1. A primeira entrada indica todos os caminhos de comprimento um que se iniciam em q e terminam em q, a segunda entrada indica todos os caminhos de comprimento um entre q e p.

Podemos interpretar a operação de soma como um *ou*, indicando as diferentes alternativas. Como já visto, maiores potências de E dizem respeito a caminhos de comprimento maior.

Exemplo 51. *Computando $u^T E^2$ para o autômato do Exemplo 49:*

$$u^T E^2 = \begin{bmatrix} 0 & h \end{bmatrix} \begin{bmatrix} 0 & h \\ a & 0 \end{bmatrix} = \begin{bmatrix} ha & 0 \end{bmatrix}. \tag{2.5.9}$$

Sendo assim, este resultado representa todos os caminhos de comprimento dois para cada um dos estados do autômato que se iniciem em um estado inicial.

Portanto, estamos apenas restringindo a origem de nossos caminhos ao invés de considerarmos caminhos quaisquer. Vimos que o operador de fecho de uma matriz de transição resulta em todos os caminhos de qualquer comprimento e entre todos os pares de vértices do grafo, logo aplicar u^T à esquerda de E^* resulta em todos os caminhos no grafo que se iniciam em algum estado inicial.

$$u^T E^* := \text{Caminhos que se iniciam em um estado inicial.}$$

E qual o papel de v? Bom, este representa os estados *finais*. Vamos observar o que ocorre se multiplicarmos v pela direita nos exemplos que fizemos.

Exemplo 52. *Computando $u^T Ev$ e $u^T E^2 v$ para o autômato do Exemplo 49:*

$$u^T Ev = \begin{bmatrix} 0 & h \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0, \tag{2.5.10}$$

$$u^T E^2 v = \begin{bmatrix} ha & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = ha. \tag{2.5.11}$$

Observamos que $u^T Ev$ e $u^T E^2 v$ retornam, respectivamente, todos os rótulos de caminhos de comprimento um e 2 que começam em um estado inicial e terminam em um estado final. Entretanto, note que isto é bem próximo da Definição 33, da Página 29, de linguagem de um autômato, porém com comprimentos fixados. Como, ao analisar linguagens, o comprimento do caminho é arbitrário, é de se esperar que E^* seja mencionado.

Definição 53. *$\mathtt{Aut}$ é o conjunto de todos os AFNDs.*

Definição 54 (Linguagem reconhecida pelo autômato)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A. Definimos o operador de linguagem $\mathcal{L} : \mathtt{Aut} \to \mathcal{P}(A^*)$ e chamamos de linguagem de $\mathcal{M}$, denotada por $\mathcal{L}(\mathcal{M})$, o conjunto contendo os rótulos dos caminhos que começam em um estado inicial e terminam em um estado final. Utilizando vetores e matrizes,*

$$\mathcal{L}(\mathcal{M}) = u^T E^* v. \tag{2.5.12}$$

Exemplo 55. *Computando $u^T E^* v$ para o autômato do Exemplo 49,*

$$u^T E^* v = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} (ha)^* & h(ah)^* \\ (ah)^* a & (ah)^* \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = (ha)^*. \tag{2.5.13}$$

Além da noção de linguagem de um autômato, precisamos definir dois conceitos fortemente correlacionados com este. Podemos construir uma noção de *linguagens à esquerda* e *linguagens à direita* de um autômato.

Definição 56 (Linguagens à esquerda)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A. Definimos o operador de linguagem à esquerda $\mathcal{L}_{\mathtt{e}} : \mathtt{Aut} \to \mathcal{P}(A^*)$ e chamamos de linguagens à esquerda de $\mathcal{M}$, denotada por $\mathcal{L}_{\mathtt{e}}(\mathcal{M})$, o seguinte vetor*

$$\mathcal{L}_{\mathtt{e}}(\mathcal{M}) = u^T E^*. \tag{2.5.14}$$

Definição 57 (Linguagens à direita)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A. Definimos o operador de linguagem à direita $\mathcal{L}_{\mathtt{d}} : \mathtt{Aut} \to \mathcal{P}(A^*)$ e chamamos de linguagens à direita de $\mathcal{M}$, denotada por $\mathcal{L}_{\mathtt{d}}(\mathcal{M})$, o seguinte vetor*

$$\mathcal{L}_{\mathtt{d}}(\mathcal{M}) = E^* v. \tag{2.5.15}$$

Exemplo 58. *As linguagens à esquerda do autômato do Exemplo 49 são*

$$\mathcal{L}_{\mathtt{e}}(\mathcal{N}) = u^T E^* = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} (ha)^* & h(ah)^* \\ (ah)^* a & (ah)^* \end{bmatrix} = \begin{bmatrix} (ha)^* & h(ah)^* \end{bmatrix}. \tag{2.5.16}$$

As linguagens à direita são

$$\mathcal{L}_{\mathtt{d}}(\mathcal{N}) = E^* v = \begin{bmatrix} (ha)^* & h(ah)^* \\ (ah)^* a & (ah)^* \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} h(ah)^* \\ (ah)^* \end{bmatrix}. \tag{2.5.17}$$

Se a linguagem de um autômato diz respeito à caminhos com origem e destino fixados, estas duas definições duais fixam somente um deles. Podemos entender as linguagens à esquerda como "e se um dado estado p fosse final? Que linguagem obteríamos?" Note que se observamos a coordenada p deste vetor temos

$$\left(\mathcal{L}_{\mathtt{e}}(\mathcal{N})\right)_p = h(ah)^*, \tag{2.5.18}$$

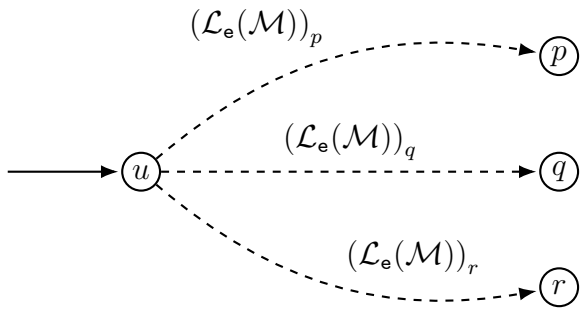


Figura 8 – Representação das entradas do vetor de linguagens à esquerda de um AFND. A linha tracejada entre u, estado inicial, e os demais estados representa os caminhos entre u e eles. Desta forma, $(\mathcal{L}_{\mathtt{e}}(\mathcal{M}))_p$ representa todos os caminhos que comecem em u e terminem em p. Podemos pensar que esta seria exatamente a linguagem do AFND caso p fosse o único estado final.

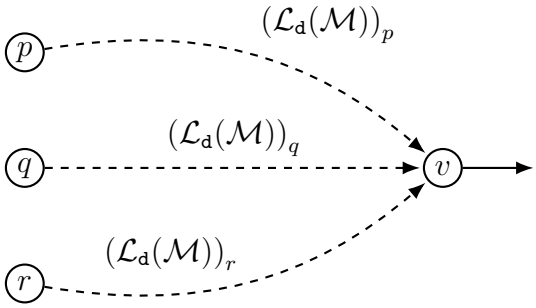


Figura 9 – Representação das entradas do vetor de linguagens à direita de um AFND. A linha tracejada entre v, estado final, e os demais estados representa os caminhos entre eles e v. Desta forma, $(\mathcal{L}_{\mathtt{d}}(\mathcal{M}))_p$ representa todos os caminhos que comecem em p e terminem em v. Podemos pensar que esta seria exatamente a linguagem do AFND caso p fosse o único estado inicial.

que referencia justamente todos os caminhos que partem de um inicial e terminam em p. No caso de p ser o único final do autômato, isto seria exatamente a linguagem de $\mathcal{N}$. O raciocínio para as linguagens à direita é análogo, $(\mathcal{L}_{\mathtt{d}}(\mathcal{N}))_p$ indica todos os caminhos que se iniciem em p e terminem em um estado final. No caso de p ser o único inicial do autômato, $(\mathcal{L}_{\mathtt{d}}(\mathcal{N}))_p$ seria exatamente a linguagem de $\mathcal{N}$. A Figuras 8 e 9 ilustram graficamente estes vetores.

Utilizaremos destas noções futuramente para caracterizar autômatos mínimos, assim como em transformações de redução de tamanho dos autômatos.

2.6 O TEOREMA DA FUSÃO DA ESTRELA

Antes de prosseguirmos com as discussões algébricas e a construção da intuição do algoritmo de Brzozowski, durante todo o tempo estamos lidando com objetos recursivos, como os próprios autômatos, e por esta razão muitas provas em linguagens formais são feitas através de indução na estrutura destes objetos. Suponha que temos uma função

aplicada nas arestas de um grafo direcionado qualquer, representaremos este processo como uma matriz F aplicada a uma matriz de transição A, resultando em FA. Como poderíamos estender este resultado para caminhos de comprimentos diferentes de um? O teorema da fusão é um modo de obter este resultado.

O teorema da fusão da estrela funciona para uma classe de funções específicas. Se F *normaliza* A e B, isto é,

$$FA = BF,$$

então F também normaliza os fechos de A e B. De outra maneira, se F admite tal propriedade para caminhos de comprimento um, então admite a mesma propriedade para caminhos de comprimento arbitrário. O teorema da fusão é uma maneira elegante de lidarmos com indução e o leitor se surpreenderá com o número de vezes em que utillizaremos este teorema.

Teorema 59 (Teorema da fusão da estrela)**.** *Para quaisquer elementos F, A e B de uma álgebra de Kleene, segue que*

$$FA = BF \implies FA^* = B^*F. \tag{2.6.1}$$

Antes da prova, vamos demonstrar de maneira informal o que ela quer dizer. Suponha A e B matrizes de transição e uma função F que as normaliza.

$$\begin{aligned}
& FA^* \\
= \quad & \{ \text{ Definição 2.3.22 de } A^* \text{ (Página 34). } \} \\
& F\,(I + A + AA + AAA + ...) \\
= \quad & \{ \text{ distributividade em } F. \ \} \\
& F + FA + FAA + FAAA + ... \\
= \quad & \{ \text{ comutatividade de } F \text{ sobre } A. \ \} \\
& F + BF + BFA + BFAA + ... \\
= \quad & \{ \text{ comutatividade de } F \text{ sobre } A \text{ sucessivas vezes. } \} \\
& F + BF + BBF + BBBF + ... \\
= \quad & \{ \text{ distributividade em } F. \ \} \\
& (I + B + BB + BBB + ...)F \\
= \quad & \{ \text{ Definição 2.3.22 de } B^* \text{ (Página 34). } \} \\
& B^*F
\end{aligned}$$

Esta demonstração é informal, entretanto serve como uma maneira intuitiva de verificar que tal generalização faz sentido. Estamos propagando o efeito de F para caminhos de cada comprimento possível. Para a prova mais formal, precisamos utilizar dos axiomas propostos para o fecho, lidando melhor com este somatório de infinitos termos.

Prova. Primeiro provemos que $FA \leq BF \implies FA^* \leq B^*F$.

$$
\begin{aligned}
& FA^* \leq B^*F \\
\Longleftarrow \quad & \{ \text{ axioma 2.3.14, da Página 31, com } x := B^*F. \ \} \\
& F + B^*FA \leq B^*F \\
\Longleftarrow \quad & \{ \text{ axioma 2.3.12, da Página 31, e distributividade do produto. } \} \\
& F + B^*FA \leq B^*BF + F \\
\Longleftarrow \quad & \{ \text{ comutatividade da soma. } \} \\
& F + B^*FA \leq F + B^*BF \\
\Longleftarrow \quad & \{ \text{ monotonicidade da soma e do produto. } \} \\
& FA \leq BF
\end{aligned}
$$

De modo semelhante, provamos que $FA \geq BF \implies FA^* \geq B^*F$.

$$
\begin{aligned}
& FA^* \geq B^*F \\
\Longleftarrow \quad & \{ \text{ axioma 2.3.13, da Página 31, com } x := FA^*. \ \} \\
& FA^* \geq F + BFA^* \\
\Longleftarrow \quad & \{ \text{ axioma 2.3.11, da Página 31, e distributividade do produto. } \} \\
& FAA^* + F \geq F + BFA^* \\
\Longleftarrow \quad & \{ \text{ comutatividade da soma. } \} \\
& F + FAA^* \geq F + BFA^* \\
\Longleftarrow \quad & \{ \text{ monotonicidade da soma e do produto. } \} \\
& FA \geq BF
\end{aligned}
$$

□

3 DETERMINISMO E ACESSIBILIDADE

E se nossos autômatos jamais se arrependessem de uma escolha? E se em qualquer ponto da construção de um determinado caminho eles sempre soubessem para onde ir, se houvesse somente uma transição possível sempre? Com os blocos básicos de construção definidos, conseguimos começar a falar mais sobre estes novos objetos matemáticos. Falaremos ainda de autômatos *acessíveis*, uma caracterização relevante e necessária para o problema de minimização.

3.1 ARESTAS VAZIAS

Antes de adentrarmos mais na teoria de autômatos determinísticos, julgo válido fazer um breve comentário sobre uma classe de autômatos em especial. Em alguns contextos, é de interesse considerar *transições vazias* como válidas em autômatos. Transições vazias são arestas da forma $q \xrightarrow{\varepsilon} p$, nomeamos estes casos como ε-AFNDs.

Definição 60 (ε-AFNDs)**.** *Um Autômato Finito Não-Determinístico com ε (ε-AFND) sobre um alfabeto A é um grafo direcionado com dois conjuntos especiais de vértices chamados iniciais e finais. O denotamos por $\mathcal{M}$ e o representamos como*

$$\mathcal{M} = \langle u, E, v \rangle, \tag{3.1.1}$$

no qual

- $E \in \mathcal{P}(A \cup \{1\})^{n \times n}$ *é a matriz de transição do grafo cujas entradas são elementos de um alfabeto A ou a palavra vazia;*
- $u \in \{0,1\}^n$ *é seu vetor indicador de estados iniciais;*
- $v \in \{0,1\}^n$ *é seu vetor indicador de estados finais.*

Note, portanto, que todo AFND é ε-AFND, mas o contrário não é verdade. Podemos fatorar a matriz de transição de um autômato em parte puramente ε e parte ε-livre. Recorde que denotamos ε por 1.

Definição 61 (Fatoração da matriz de transição)**.** *Seja $\mathcal{M} = \langle u, E, v \rangle$ um ε-AFND sobre um alfabeto A, sua matriz de transição E admite a forma*

$$E = J + \sum_{a \in A} aE^{(a)}, \tag{3.1.2}$$

no qual

$$J_{ij} = \begin{cases} 1, \text{ se } i \xrightarrow{1} j \text{ é transição no autômato;} \\ 0, \text{ caso contrário.} \end{cases} \tag{3.1.3}$$

$$E_{ij}^{(a)} = \begin{cases} 1, \text{ se } i \xrightarrow{a} j \text{ é transição no autômato;} \\ 0, \text{ caso contrário.} \end{cases} \tag{3.1.4}$$

Corolário 62 (AFNDs possuem J como matriz nula)**.** *Um AFND é um ε-AFND cuja matriz J é nula.*

Exemplo 63. *Observemos o autômato $\mathcal{M} = \langle u, E, v \rangle$ da Figura 10, cujos vetores de estados iniciais e finais, assim como sua matriz de transição são*

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad E = \begin{bmatrix} a+b & a & 0 \\ 0 & c & 1 \\ 0 & 0 & a+b \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{3.1.5}$$

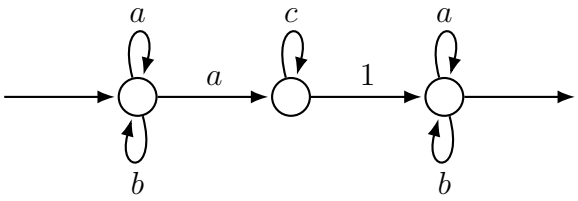


Figura 10 – Autômato com $A = \{a, b, c\}$ e linguagem $(a+b)^* ac^* (a+b)^*$.

Sua fatoração resultará em 4 matrizes, cuja expansão será

$$\begin{aligned} E &= \begin{bmatrix} a+b & a & 0 \\ 0 & c & 1 \\ 0 & 0 & a+b \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} a & a & 0 \\ 0 & 0 & 0 \\ 0 & 0 & a \end{bmatrix} + \begin{bmatrix} b & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & b \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} + a \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} + b \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} + c \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &= J + aE^{(a)} + bE^{(b)} + cE^{(c)} \\ &= J + \sum_{x \in A} xE^{(x)}. \end{aligned} \tag{3.1.6}$$

Uma pergunta natural é se AFNDs são tão expressivos quanto ε-AFNDs. A resposta para essa pergunta é sim e, de fato, felizmente todo ε-AFND pode ser transformado em um AFND sem que sua linguagem seja afetada. Resultado importante que provamos a seguir.

Teorema 64 (Todo ε-AFND pode ser AFND)**.** *Para todo ε-AFND $\mathcal{E} = \langle u, E, v\rangle$, existe um AFND $\mathcal{M} = \langle s, B, t\rangle$ tal que*

$$u^T E^* v = s^T B^* t. \tag{3.1.7}$$

Prova. Considere $E' = \sum_{a\in A} aE^{(a)}$ a parcela sem arestas vazias de $\mathcal{E}$. Por definição, sabemos que

$$\mathcal{L}(\mathcal{E}) = u^T E^* v. \tag{3.1.8}$$

Segue que

$$\begin{aligned}
& u^T E^* v \\
= \quad & \{ \text{ fatoração de } E \text{ em parte 1-livre e matriz binária. } \} \\
& u^T (J + E')^* v \\
= \quad & \{ \text{ distributividade da operação estrela (Equação 2.3.31, Página 34). } \} \\
& u^T J^* (E' J^*)^* v
\end{aligned}$$

Portanto, temos que

$$s = u^T J^*, \qquad B = E' J^* \quad \text{e} \quad s = v. \tag{3.1.9}$$

Agora analisemos $B = E'J^*$ em detalhes, pois precisamos comprovar que este é AFND. Ser AFND implica em não possuir transições 1 e isto equivale a nenhuma entrada de B ser igual a 1. Sem perda de generalidade, fixemos dois estados q e p de $\mathcal{M}$, a entrada equivalente à transição de q para p é

$$B_{qp} = \sum_{r=0}^{n} E'_{qr} J^*_{rp}. \tag{3.1.10}$$

Sabemos que J^*_{rp} sempre vale 1 ou 0. Os casos em que J^*_{rp} vale 0 podem ser ignorados, visto o produto será zero. Foquemos nas parcelas iguais a 1, substituindo no somatório, temos

$$B_{qp} = \sum_{r=0}^{n} E'_{qr} J^*_{rp} = \sum_{J_{rp}=1} E'_{qr} \cdot 1 = \sum_{J_{rp}=1} E'_{qr}. \tag{3.1.11}$$

Entretanto, por definição, E' é a parcela sem arestas vazias de $\mathcal{E}$. Desta forma, toda componente de E' é diferente de 1. Como consequência deste fato e da Equação 3.1.11, temos que $\mathcal{M}$ é AFND. □

Podemos interpretar cada componente. O vetor s contém todos os estados iniciais (u) mais os estados alcançáveis a partir deles através de transições 1 (J^*), a ideia é que suprimimos as transições 1 entre um estado inicial e os demais, tornando-os iniciais também. A matriz B contém todas as transições diferentes de 1 (E') seguidas de zero ou mais transições 1 (J^*). Novamente suprimimos as transições, porém desta vez ao percorrer o caminho. Com a segurança de que quaisquer ε-AFNDs podem ser convertidos em versões AFND, trataremos sempre de AFNDs no restante do texto.

3.2 AUTÔMATOS DETERMINÍSTICOS

Acerca de ser determinístico, nos referimos à classe de autômatos no qual dado um estado q e um valor de transição a, teremos nenhum ou exatamente um estado p tal que $q \xrightarrow{a} p$ seja uma transição do autômato. Ainda, queremos que, em qualquer estado que estejamos, nunca tenhamos dúvidas para onde transitar dado um rótulo a qualquer. As transições vazias são dificultadoras por esta exata razão, pois podemos transicionar de um estado a outro sem que nenhum rótulo seja identificado. Naturalmente isto permite que múltiplos caminhos de valor a existam no autômato e, portanto, viola seu determinismo. De fato, poderíamos definir um autômato determinístico como um AFND que tenha exatamente um estado inicial e que, para todo par $q \in Q$ e $a \in A$, possua até um estado p tal que $q \xrightarrow{a} p$.

A última exigência torna a relação de transição E do autômato em uma função. Mais especificamente, uma *função parcial*, pois podem haver pares q e a em que não exista transição $q \xrightarrow{a} p$ para nenhum p e, portanto, E não é definida para este par. O problema desta definição é puramente teórico, em algumas provas exigiremos que E seja uma *função total*, de modo que quaisquer pares que esta receba, sempre haverá exatamente um estado p no qual $q \xrightarrow{a} p$ exista. Consideremos esta definição que assume um função total.

Definição 65 (Autômato Finito Determinístico)**.** *Seja $\mathcal{M} = \langle Q, A, E, I, T\rangle$ um AFND sobre um alfabeto A, ele é um Autômato Finito Determinístico (AFD) se, e somente se,*

- *Possui exatamente um estado inicial;*
- *A relação de transição E é uma função total de assinatura $Q \times A \to Q$.*

A definição matricial é possível e será a usada neste trabalho. Sabemos bem o significado de possuirmos somente um estado inicial, assim como o de não possuirmos transições vazias. O menos direto, talvez, seja a exigência de função total.

Definição 66 (AFD matricialmente)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A, ele é um Autômato Finito Determinístico (AFD) se, e somente se,*

- *O vetor de estados iniciais é um vetor canônico, ou seja, há exatamente uma entrada não-nula de valor* 1*;*
- *Para cada elemento $a \in A$, cada linha da matriz $E^{(a)}$ tem exatamente uma única entrada não-nula de valor* 1*.*

Exemplo 67. *O autômato da Figura 6 não é determinístico. Apesar de possuir um único estado inicial e nenhuma aresta partindo de um mesmo estado com rótulo repetido, sua função de transição não é definida para o par (q, a). O autômato da Figura 5 não é determinístico, note como o estado q possui três arestas com rótulo a.*

Exemplo 68. *Os autômatos da Figura 11 reconhecem a mesma linguagem e são exemplos de AFND e AFD, respectivamente. Neste caso, ambos possuem o mesmo tamanho. Entretanto, no caso geral, isto não necessariamente é verdade.*

(a) Autômato finito não-determinístico para a linguagem aa^*.

(b) Autômato finito determinístico para a linguagem aa^*.

Figura 11 – Exemplo de um AFND e AFD que reconhecem a mesma linguagem. O autômato da Figura 11a é não-determinístico devido ao estado inicial q possuir duas transições de rótulo a, problema que não aparece no AFD da Figura 11b. Note que o AFND reconhece a^*a e o AFD aa^*, que equivalem ao mesmo valor (Equação 2.3.23, Página 34).

3.2.1 Autômatos completos

Antes de prosseguir para o algoritmo de determinização, acredito ser de valia descrever brevemente uma categoria de autômatos que surgem ao decorrer do processo. Todo autômato resultante do algoritmo de determinização será um *autômato completo.*

Definição 69 (Autômato Completo)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A, este é completo se, e somente se, para todo par de origem e rótulo há pelo menos uma transição correspondente no autômato. Algebricamente, para toda linha i de sua matriz de transição temos*

$$\textit{Para todo } a \textit{ em } A \textit{ existe } j \textit{ tal que } E^{(a)}_{ij} = 1. \tag{3.2.1}$$

Exemplo 70. *O autômato da Figura 12 é completo, mas não é determinístico, pois q possui duas arestas de rótulo a.*

Exemplo 71. *Todo autômato no qual E seja relação total ou função total é completo.*

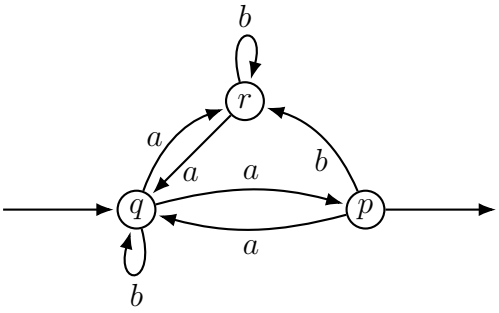


Figura 12 – Exemplo de autômato finito completo sobre o alfabeto $A = \{a, b\}$. Note que há pelo menos uma aresta para cada par de origem e rótulo.

Teorema 72 (Todo autômato pode ser completado)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A, existe um AFND completo $\mathcal{B} = \langle s, B, t\rangle$, também sobre A, tal que*

$$u^T E^* v = s^T B^* t. \tag{3.2.2}$$

Prova. Tome $\mathcal{M}$ e crie um estado novo chamado z. Para todo estado q e símbolo $a \in A$, adicione uma aresta $q \xrightarrow{a} z$ se

$$\sum_j^n E_{qj}^{(a)} = 0. \tag{3.2.3}$$

Isto é, se q é origem de nenhuma aresta de rótulo a. Por fim, adicione arestas de z para si próprio para cada valor a de A e adicione tudo isto a uma nova matriz B que represente este novo grafo. Mantendo os mesmos estados iniciais e finais, ficamos com

$$s = \begin{bmatrix} u \\ 0 \end{bmatrix} \quad \text{e} \quad t = \begin{bmatrix} v \\ 0 \end{bmatrix}. \tag{3.2.4}$$

□

Fato interessante de estados como z é que a partir do momento em que transicionamos para ele não saímos mais de lá. Na verdade, nenhuma palavra reconhecida de $\mathcal{B}$ passa por ele. Estados desta natureza surgem naturalmente em autômatos quaisquer e, especialmente, do processo de determinização que veremos.

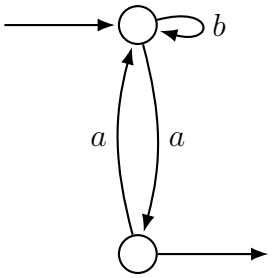
Exemplo 73. *A Figura 13 mostra um exemplo de autômato não-completo sendo transformado em um. Antes da transformação, seus vetores de estados iniciais e finais, assim como sua matriz de transição eram*

$$u = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad E = \begin{bmatrix} b & a \\ 0 & a \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{3.2.5}$$

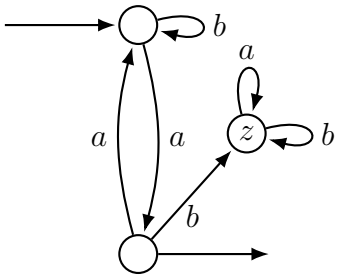
Após a transformação, adicionamos mais uma coordenada, indicando o estado z.

$$s = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad B = \begin{bmatrix} b & a & 0 \\ 0 & a & b \\ 0 & 0 & a+b \end{bmatrix}, \quad t = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}. \tag{3.2.6}$$

Agora todo o alfabeto A está presente em cada uma das linhas, indicando que todo estado possui pelo menos uma aresta para cada valor de rótulo possível.



(a) AFND de exemplo.



(b) Autômato da Figura 13a após o processo de transformação em um autômato completo.

Figura 13 – Exemplo de autômato antes (13a) e após ser transformado em um autômato completo (13b) sobre o alfabeto $A = \{a, b\}$. Note o estado z na Figura 13b.

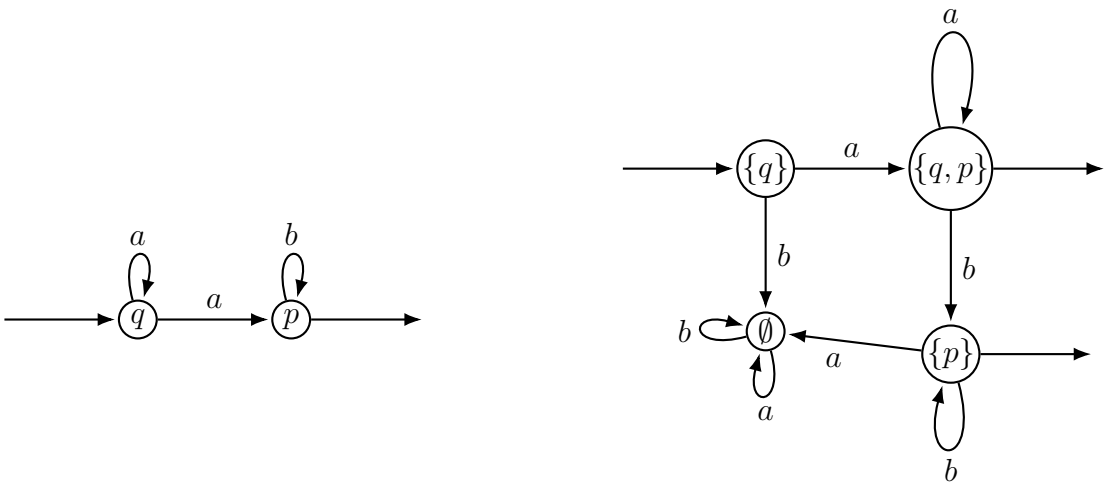
3.2.2 Construção de subconjuntos

Precisamos agora focar nas duas outras exigências de determinismo: Possuirmos somente um estado inicial e, para todo par de origem e rótulo, possuirmos exatamente uma aresta. Felizmente, todo AFND pode ser transformado em um AFD, teorema bem conhecido em teoria de linguagens formais e que se torna fator crucial para a minimização de autômatos. O modo como faremos tal transformação é através de uma troca de *referencial*. Por definição, se um AFND é não-determinístico, então pode haver algum estado q que, ao receber um símbolo a, possua mais de um destino p possível. A ideia consiste em observarmos este conjunto P como um novo estado do autômato, fazendo com que q transicione pare este.

Exemplo 74. *Considere que um estado q transicione para p e r com rótulo a, construiremos novos estados $\{q\}$ e $\{p, r\}$ e faremos $\{q\}$ transicionar para $\{p, r\}$.*

Isto remove por completo a ambiguidade, pois, para quaisquer conjuntos de estados P em que q possua transições de mesmo rótulo, somos capazes sempre de transformá-lo em um único estado novo. Note, portanto, que estamos construindo *subconjuntos* de estados do autômato e consequentemente aumentaremos o número total de estados, relativo à quantidade original, exponencialmente. Se cada subconjunto de Q se torna um novo estado, então teremos $2^{|Q|}$ estados ao fim.

Exemplo 75. *O AFND da Figura 14 traz um caso concreto de construção de subconjuntos. Repare que $\{q\}$, conjunto que representa o estado q no autômato original, agora possui somente uma transição com rótulo a para o conjunto $\{q, p\}$, cujos elementos são justamente os estados em que q possuía transições de rótulo a originalmente.*



(a) AFND para a linguagem a^*ab^*.

(b) AFND para a linguagem a^*ab^* com a representação de subconjuntos. Note que cada subconjunto possível de estados é um novo estado.

Figura 14 – Exemplo de um AFND antes e após a construção de subconjuntos. Construímos um estado para cada subconjunto de Q, fazendo com que este herde as arestas de todos os seus elementos. Repare que, desta forma, todo estado deve ter exatamente uma aresta para cada valor possível de rótulo, resultando em um autômato completo.

Agora possuímos dois autômatos diferentes que reconhecem a mesma linguagem, porém sobre *representações* distintas. A prova de determinismo se baseará em como tais representações se relacionam e, portanto, seria interessante que soubéssemos transitar entre ambas. Para tal, precisaremos de um modo de descrever subconjuntos de estados tanto de maneira usual, no espaço $|Q|$-dimensional, quanto da maneira nova descrita, no espaço $2^{|Q|}$-dimensional. Faremos a transição entre as representações via uma transformação linear.

Definição 76 (Representação de subconjuntos)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND com conjunto de estados Q de tamanho n. Representaremos $P \subseteq Q$ subconjunto de estados de $\mathcal{M}$ de duas formas:*

- *Um vetor n-dimensional s_P em que cada coordenada diz respeito a um estado q do autômato e que definiremos como*

$$(s_P)_q = \begin{cases} 1, & \text{se } q \in P; \\ 0, & \text{caso contrário.} \end{cases} \tag{3.2.7}$$

- *Um vetor canônico 2^n-dimensional e_P em que cada coordenada diz respeito a um subconjunto de Q, no qual sua definição é*

$$(e_P)_R = \begin{cases} 1, & \text{se } R = P; \\ 0, & \text{caso contrário.} \end{cases} \tag{3.2.8}$$

Exemplo 77. *O vetor nulo n-dimensional representa o conjunto vazio e o vetor n-dimensional com todas as entradas valendo* 1 *representa o conjunto de todos os estados, Q.*

Definição 78 (Projeção de estados)**.** *Dadas as representações de subconjuntos apresentadas na Definição 76. Definimos uma transformação linear F que leve uma representação s_P para seu equivalente e_P. Esta transformação é representada por uma matriz de dimensões $2^n \times n$ e satisfaz*

$$e_P^T F = s_P^T. \tag{3.2.9}$$

De maneira implícita, isto define a forma de F, pois sua P-ésima linha será exatamente o vetor s_P:

$$(F)_{Pq} = (s_P)_q. \tag{3.2.10}$$

Nomeamos F a matriz de representação dos subconjuntos de estados de $\mathcal{M}$.

Exemplo 79. *Para autômatos de dois estados, temos que*

$$F = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{3.2.11}$$

Repare que F não depende do autômato, mas sim do tamanho do conjunto de estados. Cada linha representa uma configuração possível de uma sequência de $|Q| = n$ bits, exatamente por isso possuímos 2^n linhas, cada uma representando um subconjunto.

Sabemos descrever subconjuntos de autômatos de duas formas, assim como transitar entre elas. Todavia, não sabemos a forma da matriz de transição do autômato após a construção de subconjuntos, tendo apenas a noção de que este possui 2^n estados. Dedicaremos nossos esforços agora em descrevê-la, assim como provar algumas de suas propriedades.

Definição 80 (Matriz de transição da construção de subconjuntos)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND com conjunto de estados Q de tamanho n e seja $\widehat{\mathcal{M}}$ o autômato após a construção de subconjuntos. A matriz $\widehat{E}$ de transição do autômato $\widehat{\mathcal{M}}$ é a matriz $2^n \times 2^n$*

$$\widehat{E} = \sum_{a \in A} a\widehat{E}^{(a)}, \tag{3.2.12}$$

na qual definimos $\widehat{E}^{(a)}$ de modo que

$$e_P^T \widehat{E}^{(a)} = e^T_{(s_P^T E^{(a)})}. \tag{3.2.13}$$

Aqui, utilizamos dos vetores representantes de seus subconjuntos como índices também. A semântica se preserva, $s_P^T E^{(a)}$ dará resultado a um novo vetor n-dimensional representando um dado subconjunto. Desta forma, $e^T_{(s_P^T E^{(a)})}$ indica sua representação em 2^n coordenadas.

Observando a definição de $\widehat{E}^{(a)}$ de outro modo, $s_P^T E^{(a)}$ filtrará as linhas referentes a todos os estados que pertençam ao subconjunto P. A soma destas linhas dará todas as transições, de cada estado pertencente ao subconjunto P, de rótulo a para qualquer outro estado do grafo. Tal definição é exatamente o significado de $e_P^T \widehat{E}^{(a)}$, P-ésima linha de $\widehat{E}^{(a)}$, em que estamos tratando P como um estado único. De maneira mais precisa,

$$\left(s_P^T E^{(a)}\right)_q = \begin{cases} 1, \text{ se existe transição } p \xrightarrow{a} q \text{ com } p \in P \text{ em } \mathcal{M}; \\ 0, \text{ caso contrário.} \end{cases} \tag{3.2.14}$$

Vamos resumir toda a informação que possuímos até aqui. Temos vetores indicadores de subconjuntos de estados em duas representações, assim como uma matriz de transição relacionada a cada uma delas, podemos ver isto na tabela abaixo.

espaço	dimensão	rep. dos subconj.	matriz
estados	n	$s_P \in \{0,1\}^n$	$E : (A^*)^n \to (A^*)^n$
subconj. de estados	2^n	$e_P \in \{0,1\}^{2^n}$	$\widehat{E} : (A^*)^{2^n} \to (A^*)^{2^n}$

(3.2.15)

A primeira propriedade que provaremos sobre $\widehat{E}$ é que a matriz F normaliza E e $\widehat{E}$. Esta é a chave de ouro para relacionar ambos os autômatos e que utilizaremos na prova do determinismo para mostrar que tal construção preserva a linguagem original do autômato.

Teorema 81 (Projeção de subconjuntos comuta sobre determinismo)**.** *Seja $\mathcal{M} = \langle u, E, v \rangle$ um AFND. Pela Definição 78 temos s_P e e_P representando os subconjuntos de estados de $\mathcal{M}$ em n e 2^n dimensões, respectivamente, e F uma transformação linear que relaciona ambas as representações. Seja $\widehat{E}$ definida pela equação 80, F normaliza E e $\widehat{E}$; isto é,*

$$EF = F\widehat{E}. \tag{3.2.16}$$

Prova. Provaremos, de maneira indireta, que $e_P^T EF = e_P^T F\widehat{E}$ para todo e_P^T.

$$\begin{aligned}
& e_P^T FE \\
= & \quad \{ \text{ definição de } F \text{ (Equação 3.2.9). } \} \\
& s_P^T E \\
= & \quad \{ \text{ expansão de } E \text{ sobre elementos de } A. \} \\
& s_P^T \sum_{a \in A} aE^{(a)} \\
= & \quad \{ \text{ distributividade com } s_P^T. \} \\
& \sum_{a \in A} a s_P^T E^{(a)} \\
= & \quad \{ \ s_P^T E^{(a)} = s^T_{\left(s_P^T E^{(a)}\right)}. \ \} \\
& \sum_{a \in A} a s^T_{\left(s_P^T E^{(a)}\right)} \\
= & \quad \{ \text{ definição de } F \text{ (Equação 3.2.9). } \}
\end{aligned}$$

$$
\begin{aligned}
& \sum_{a\in A} a e^T_{(s^T_P E^{(a)})} F \\
= & \quad \{ \text{ definição de } \widehat{E}^{(a)} \text{ (Equação 3.2.13). } \} \\
& \sum_{a\in A} a e^T_P \widehat{E}^{(a)} F \\
= & \quad \{ \text{ distributividade. } \} \\
& e^T_P \left(\sum_{a\in A} a \widehat{E}^{(a)} \right) F \\
= & \quad \{ \text{ definição de } \widehat{E} \text{ (Definição 80). } \} \\
& e^T_P \widehat{E} F
\end{aligned}
$$

□

Esta comutatividade é exatamente a hipótese do teorema de fusão da estrela, portanto como consequência temos a mesma propriedade para o fecho de ambas as matrizes.

Corolário 82 (F normaliza a estrela)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND, temos F sua matriz de projeção de subconjuntos conforme a Definição 78. Seja $\widehat{E}$ definida pela Equação 80. Segue que*

$$E^* F = F \widehat{E}^*. \tag{3.2.17}$$

3.2.3 A determinização

Teorema 83 (Determinização)**.** *Existe uma transformação $\texttt{det} : \texttt{Aut} \to \texttt{Aut}$ tal que, para todo AFND $\mathcal{M} = \langle u, E, v\rangle$ sobre qualquer alfabeto A, existe um AFD $\widehat{\mathcal{M}} = \langle \hat{u}, \widehat{E}, \hat{v}\rangle$ sobre o mesmo alfabeto que satisfaz*

$$\texttt{det}(\langle u, E, v\rangle) = \langle \hat{u}, \widehat{E}, \hat{v}\rangle; \tag{3.2.18}$$

$$u^T E^* v = \hat{u}^T \widehat{E}^* \hat{v}, \tag{3.2.19}$$

no qual

- $\hat{u} = e_I$;
- *F é definido conforme a Definição 78;*
- *$\widehat{E}$ definido de acordo com a Definição 80;*
- $\hat{v} = Fv$.

Isto é, todo AFND pode ser determinizado.

Prova. Tornando cada subconjunto de estados em um novo estado de $\widehat{\mathcal{M}}$ via construção de subconjuntos, obtemos todas as condições de determinismo satisfeitas.

- Se o autômato possui $I \subseteq Q$ como conjunto de estados iniciais, este se tornará um único estado em $\widehat{\mathcal{M}}$, representado pelo vetor canônico e_I;
- E é função total. Por definição de produto matricial, temos, pela Equação 3.2.13, que a P-ésima linha da matriz $\widehat{E}^{(a)}$ é igual a um vetor canônico, isto significa que sempre há exatamente uma transição para quaisquer pares de origem e rótulo. Esta é exatamente a definição de função total;
- A linguagem do novo autômato $\widehat{\mathcal{M}}$ é a mesma do autômato original $\mathcal{M}$.

O último tópico exige uma prova, que faremos via teorema da fusão da estrela. O interessante é que, fazendo desta forma, obtemos construtivamente as formas de $\hat{u}$ e $\hat{v}$ descritas no enunciado do teorema.

$$
\begin{aligned}
& u^T E^* v \\
= \quad & \{\ u^T = s_I^T .\ \} \\
& s_I^T E^* v \\
= \quad & \{\ e_I^T F = s_I^T \text{ (3.2.9).}\ \} \\
& e_I^T F E^* v \\
= \quad & \{\ F \text{ normaliza } E \text{ e } \widehat{E} \text{ (Corolário 82).}\ \} \\
& e_I^T \widehat{E}^* F v \\
= \quad & \{\ \hat{u} = e_I \text{ e } \hat{v} = Fv .\ \} \\
& \hat{u} \widehat{E}^* \hat{v}
\end{aligned}
$$

□

Exemplo 84. *A Figura 15 exemplifica como o processo de determinização funciona. O autômato de dois estados se torna um de quatro, dos quais cada um representa subconjunto de estados diferente. Note que o resultado difere do indicado na Figura 11, isto se deve ao modo como o algoritmo funciona. Antes do processo de determinização, os vetores de estados iniciais e finais do autômato, assim como sua matriz de transição, eram*

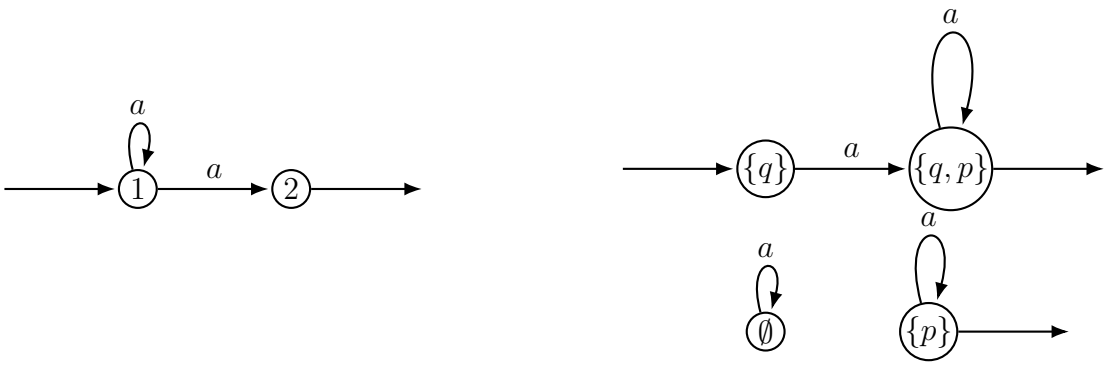
$$u = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad E = \begin{bmatrix} a & a \\ 0 & 0 \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{3.2.20}$$

Após o processo de determinização, estes se tornam

$$\hat{u} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \widehat{E} = \begin{bmatrix} 0 & a & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & a \end{bmatrix}, \quad \hat{v} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}. \tag{3.2.21}$$

Como Fv tem entradas de valor 1 *somente se um estado final pertence a dado subconjunto, o autômato determinizado neste caso possui dois estados finais, justamente os*

estados que representam os dois subconjuntos em que p está presente, estado final do autômato original.



(a) AFND para a linguagem aa^*.

(b) AFD para a linguagem aa^* após o processo de determinização.

Figura 15 – Exemplo de um AFND antes e após o processo de determinização descrito pelo Teorema 83. O autômato da Figura 15a é o mesmo da Figura 11a, porém note que sua versão determinística obtida pelo algoritmo é diferente da presenta na Figura 11b. Isto ocorre pois o Teorema 83 constrói um autômato a partir de subconjuntos de estados. Entretanto, podemos notar que os estados que representam os conjuntos $\emptyset$ e $\{p\}$, a princípio, parecem descartáveis.

Como o leitor pode ter notado, o autômato resultante desta transformação cresce exponencialmente. De fato, se temos um autômato com meros dez estados, após a conversão teremos um total de 1024 estados. Se este número aumenta para 80 estados, que é uma quantidade bem plausível de se alcançar em muitas aplicações, teremos 2^{80} estados após a transformação, que é um número maior do que a quantidade de átomos no universo! Isto se torna um problema grande se notamos que existem aplicações em que tais autômatos possuem milhares, se não milhões, de estados. A questão é se *realmente* precisamos de todos esses novos estados criados. Felizmente, muitos destes novos estados não são úteis e verificaremos agora que processo podemos tomar para removê-los.

3.3 REMOÇÃO DE ESTADOS INACESSÍVEIS

Como bem observamos, o Teorema 83 nos retorna um autômato completo e nosso objetivo agora é identificarmos estados descartáveis. Separaremos os estados em duas categorias: os alcançáveis por algum inicial e os não alcançáveis. Caso a primeira proposição seja verdade, dizemos que o estado é *acessível* no autômato.

Definição 85 (Estado acessível)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A com conjunto de estados Q. Se $q \in Q$ é tal que*

$$\left(\mathcal{L}_e(\mathcal{M})\right)_q \neq 0, \tag{3.3.1}$$

dizemos que q é acessível em $\mathcal{M}$. Caso contrário, será inacessível.

Lema 86 (Todo estado inicial é acessível)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A, todo estado inicial de $\mathcal{M}$ é acessível.*

Prova. Isto decorre diretamente da Definição 85 de acessibilidade. Recorde que o vetor de linguagens à esquerda de um autômato é computado por $u^T E^*$. Fixemos $i \in I$ estado inicial, decorre que

$$
\begin{aligned}
& (\mathcal{L}_{\mathsf{e}}(\mathcal{M}))_i \\
= \quad & \{\ i\text{-ésima coordenada equivale a produto por um canônico.}\ \} \\
& u^T E^* e_i \\
= \quad & \{\ \text{definição de estrela (Corolário 43, Página 33).}\ \} \\
& u^T \left(EE^* + I\right) e_i \\
= \quad & \{\ \text{distributividade.}\ \} \\
& u^T EE^* e_i + u^T I e_i \\
= \quad & \{\ u^T I e_i = u^T e_i = 1, \text{ pois } u_i = 1.\ \} \\
& u^T EE^* e_i + 1
\end{aligned}
$$

Pelo conjunto necessariamente incluir a palavra vazia, ele não é vazio. □

Definição 87 (Autômato acessível)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A com conjunto de estados Q. Se todo estado $q \in Q$ for acessível em $\mathcal{M}$, dizemos que o autômato é acessível. Caso contrário, será inacessível.*

Exemplo 88. *Na Figura 16 somente o estado t é inacessível.*

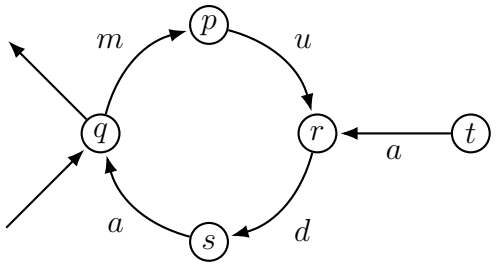


Figura 16 – AFND que reconhece sequências de *muda*. Todos os estados são acessíveis, exceto t.

Exemplo 89. *Os autômatos das Figuras 5, 6, 10, 11, 12, 13, 14 e 15a são todos acessíveis. Os autômatos das Figuras 15b e 16 são inacessíveis.*

Ora, intuitivamente podemos pensar que, se um estado q não pode ser alcançado por nenhum estado inicial do autômato, não haverá caminho algum que comece em um dos iniciais e termine em um dos finais passando por q. Ou seja, não há palavra f aceita pelo autômato que tenha caminho passando por q. Esta ideia nos dá um forte indício de que,

talvez, possamos remover q. Primeiro, vamos entender como um estado isolado contribui para a linguagem de um autômato.

Definição 90 (Linguagem de um estado)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND com conjunto de estados Q. A linguagem de um estado q representa todas as palavras reconhecidas pelo autômato cujo caminho passe por q; isto é,*

$$\begin{aligned}\left(u^T E^* e_q\right)\left(e_q^T E^* v\right) &= \left(\mathcal{L}_e(\mathcal{M}) \cdot e_q\right)\left(e_q^T \cdot \mathcal{L}_d(\mathcal{M})\right) \\ &= \left(\mathcal{L}_e(\mathcal{M})\right)_q \left(\mathcal{L}_d(\mathcal{M})\right)_q .\end{aligned} \tag{3.3.2}$$

Exemplo 91. *A linguagem do estado s do autômato da Figura 17 é $(la)^*ia$ e a linguagem do estado t é $(la)^*ie$. Dos demais, a linguagem é $(la)^*i(a+e)$.*

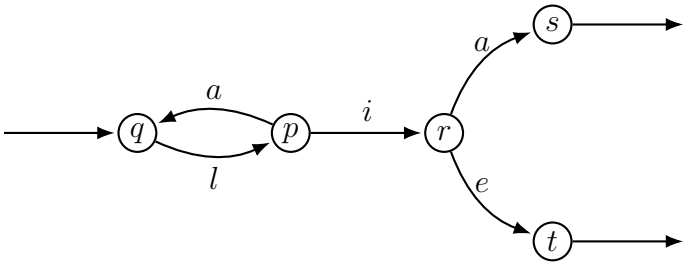


Figura 17 – AFND que reconhece palavras que começam com uma sequência de la e terminam em ia ou ie.

A Definição 90 ilustra exatamente o que gostaríamos. A parcela $u^T E^* e_q$ indica todos os caminhos que comecem em um inicial e terminem em q, ou seja, todas as palavras g tal que haja um caminho

$$s \xrightarrow{g} q,\ s \in I.$$

De maneira semelhante, a parcela $e_q^T E^* v$ mostra todos os caminhos que comecem em q e terminem em algum estado final, isto é, todas as palavras h tal que

$$q \xrightarrow{h} y,\ y \in T.$$

Claramente, como g termina em q e h começa em q, podemos concatenar ambos os caminhos e formar uma palavra $f = gh$. De maneira direta também, podemos concluir que f é reconhecida por $\mathcal{M}$ pela Definição 54 de linguagem de um autômato. Fato relevante para nós é que a linguagem do autômato é exatamente a soma da linguagem individual de seus estados.

Lema 92 (A soma das linguagens dos estados resulta na linguagem do autômato)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND com conjunto de estados Q, temos que*

$$u^T E^* v = \sum_{q\in Q}\left(u^T E^* e_q\right)\left(e_q^T E^* v\right) = \sum_{q\in Q} u^T E^*\left(e_q e_q^T\right) E^* v. \tag{3.3.3}$$

Em outras palavras, $\mathcal{L}(\mathcal{M})$ é a soma das linguagens de cada um de seus estados individualmente.

Prova. Observando uma parcela do somatório,

$$u^T E^* \left(e_q e_q^T\right) E^* v. \tag{3.3.4}$$

Como a única entrada não-nula de e_q é a coordenada q, temos que $e_q e_q^T$ será uma matriz $n \times n$ com todas as suas entradas valendo 0, exceto no elemento da diagonal qq. Cada estado q dá origem a uma matriz desta forma e, se somarmos todas, obtemos a matriz identidade:

$$\sum_{q \in Q} e_q e_q^T = I. \tag{3.3.5}$$

Sendo assim,

$$\begin{aligned}
& u^T E^* v \\
= \quad & \{ \text{ transitividade da estrela (Equação 2.3.25, Página 34). } \} \\
& u^T E^* E^* v \\
= \quad & \{ \text{ matriz identidade. } \} \\
& u^T E^* I E^* v \\
= \quad & \{ \text{ Equação 3.3.5. } \} \\
& u^T E^* \left(\textstyle\sum_{q \in Q} e_q e_q^T\right) E^* v \\
= \quad & \{ \text{ distributividade. } \} \\
& \textstyle\sum_{q \in Q} u^T E^* \left(e_q e_q^T\right) E^* v
\end{aligned}$$

□

Exemplo 93. *A linguagem do autômato da Figura 17 pode ser obtida somando a linguagem de seus estados. Como visto, suas linguagens assumem três formas distintas:* $(la)^* ia$, $(la)^* ie$ *e* $(la)^* i(a + e)$. *Segue que a soma resulta exatamente em* $(la)^* i(a + e)$.

Com uma noção mais clara de como cada estado contribui para o todo, podemos confirmar um fato importante: estados inacessíveis *não* contribuem com nenhuma palavra nova para a linguagem do autômato.

Lema 94 (Estados inacessíveis não agregam à linguagem)**.** *Seja* $\mathcal{M} = \langle u, E, v \rangle$ *um AFND sobre um alfabeto A. Seja q um estado inacessível em* $\mathcal{M}$*, então não existe palavra f reconhecida por* $\mathcal{M}$ *que possua um caminho que passe por q.*

Prova.

$$\begin{aligned}
& u^T E^* v \\
= \quad & \{ \text{ definição de linguagem de um estado (Definição 90). } \} \\
& \left(\mathcal{L}_{\mathtt{e}}(\mathcal{M}) \cdot e_q\right) \left(e_q^T \cdot \mathcal{L}_{\mathtt{d}}(\mathcal{M})\right) \\
= \quad & \{ \ q \text{ é inacessível em } \mathcal{M} \text{ (Definição 85). } \}
\end{aligned}$$

$$
\begin{aligned}
& 0 \cdot \left(e_q^T \cdot \mathcal{L}_{\mathtt{d}}(\mathcal{M})\right) \\
= \quad & \{ \text{ produto pelo elemento neutro da soma. } \} \\
& 0
\end{aligned}
$$

Portanto, não há caminho que comece em um inicial e termine em um final passando por q. Outro modo de ver é que o conjunto de palavras reconhecidas por $\mathcal{M}$ e que passam por q é vazio. □

Com este lema, podemos provar que qualquer AFND pode ser transformado em uma versão acessível apenas descartando todos os estados inacessíveis nele.

Teorema 95 (Poda)**.** *Existe uma transformação* $\mathtt{pod} : \mathtt{Aut} \to \mathtt{Aut}$ *tal que, para todo AFND* $\mathcal{M} = \langle u, E, v\rangle$ *sobre qualquer alfabeto* A*, existe um AFND acessível* $\mathcal{B} = \langle s, B, t\rangle$ *sobre o mesmo alfabeto que satisfaça*

$$\mathtt{pod}(\langle u, E, v\rangle) = \langle s, B, t\rangle; \tag{3.3.6}$$

$$u^T E^* v = s^T B^* t. \tag{3.3.7}$$

Chamamos $\mathtt{pod}$ *de poda e dizemos que* $\mathcal{B}$ *é o autômato* $\mathcal{M}$ *podado.*

Prova. Pelo Lema 94, sabemos que apenas os estados acessíveis influenciam na linguagem do autômato, portanto podemos descartar o restante sem que deixemos de reconhecer palavras ou passemos a reconhecer novas. O conjunto de estados que são acessíveis no autômato necessariamente será um subconjunto de estados dele, nomearemos este conjunto por $\mathcal{A}(\mathcal{M}) \subseteq Q$. Portanto, se o conjunto de estados de $\mathcal{B}$ for exatamente $\mathcal{A}(\mathcal{M})$, o novo autômato será necessariamente do mesmo tamanho ou menor que $\mathcal{M}$. Provar este teorema é construir uma transformação linear que leve os n estados de $\mathcal{M}$ para os $|\mathcal{A}(\mathcal{M})| = m$ estados de $\mathcal{B}$.

Para o processo de *poda* utilizaremos de uma transformação linear intuitiva, que manterá todos os estados pertencentes a $\mathcal{A}(\mathcal{M})$ e descartará os demais. Para tal, sem perda de generalidade, vamos considerar que todos os estados acessíveis de $\mathcal{M}$ ocupam as primeiras coordenadas dos vetores e matrizes no decorrer da prova. Isso pode ser assumido, pois todos os resultados que desenvolvermos aqui serão invariantes por permutações. Deste modo, a matriz N que efetua este comportamento é

$$N = \begin{bmatrix} I_{m\times m} & 0_{m\times(n-m)} \end{bmatrix}. \tag{3.3.8}$$

Isto é, as primeiras m coordenadas, referentes aos acessíveis, são mantidas sem alteração e as $n - m$ restantes são todas zeradas. Com N, podemos extrair a parcela acessível de u:

$$u^T = \begin{bmatrix} s^T_{1\times m} & | & 0_{1\times(n-m)} \end{bmatrix}. \tag{3.3.9}$$

Segue que

$$u^T = s^T \begin{bmatrix} I_{m\times m} & 0_{m\times(n-m)} \end{bmatrix}. \tag{3.3.10}$$

Na verdade, por N ser uma matriz retangular com mais colunas do que linhas, ela pode admitir infinitas inversas à direita. Pelo modo como a definimos, uma possível inversa seria N^T:

$$NN^T = \begin{bmatrix} I_{m\times m} & 0_{m\times(n-m)} \end{bmatrix} \begin{bmatrix} I_{m\times m} \\ 0_{m\times(n-m)} \end{bmatrix} = I_{m\times m}. \tag{3.3.11}$$

Com isto, se multiplicarmos por N^T à direita em ambos os lados da equação 3.3.10, somos capazes de isolar u.

$$\begin{aligned} & u^T = s^T N \\ \implies & \quad \{ \text{ produto por } N^T \text{ à direita em ambos os lados. } \} \\ & u^T N^T = s^T N N^T \\ \iff & \quad \{ \ N^T \text{ é inversa à direita de } N. \ \} \\ & u^T N^T = s^T \end{aligned}$$

Note que, na verdade, poderíamos colocar qualquer matriz $(n-m)\times m$ no lugar do bloco nulo da matriz inversa que utilizamos, porém manter desta forma facilita o raciocínio e torna a inversa exatamente a transposta. Como passo seguinte, faremos a mesma coisa para a matriz de transição E do autômato original. Através da ordenação que fixamos, podemos quebrar as transições de $\mathcal{M}$ em quatro categorias:

- Transições entre dois estados acessíveis em $\mathcal{M}$, que denotaremos por $E_{\mathtt{ac}\to\mathtt{ac}}$;
- Transições entre dois estados inacessíveis em $\mathcal{M}$, denotados por $E_{\mathtt{inac}\to\mathtt{inac}}$;
- Transições de acessíveis para inacessíveis e vice-versa, no qual denotamos por $E_{\mathtt{ac}\to\mathtt{inac}}$ e $E_{\mathtt{inac}\to\mathtt{ac}}$, respectivamente.

Decorre que

$$E = \begin{bmatrix} E_{\mathtt{ac}\to\mathtt{ac}} & E_{\mathtt{ac}\to\mathtt{inac}} \\ E_{\mathtt{inac}\to\mathtt{ac}} & E_{\mathtt{inac}\to\mathtt{inac}} \end{bmatrix}. \tag{3.3.12}$$

Podemos inferir de antemão que $E_{\mathtt{ac}\to\mathtt{inac}} = 0$. Existir uma aresta de um estado acessível para um estado inacessível é gerar um caminho entre um estado inicial e ele, o que não existe por hipótese. Portanto,

$$E = \begin{bmatrix} E_{\mathtt{ac}\to\mathtt{ac}} & E_{\mathtt{ac}\to\mathtt{inac}} \\ E_{\mathtt{inac}\to\mathtt{ac}} & E_{\mathtt{inac}\to\mathtt{inac}} \end{bmatrix} = \begin{bmatrix} E_{\mathtt{ac}\to\mathtt{ac}} & 0 \\ E_{\mathtt{nac}\to\mathtt{ac}} & E_{\mathtt{nac}\to\mathtt{nac}} \end{bmatrix}. \tag{3.3.13}$$

Intuitivamente, o que queremos é preservar o bloco $E_{\mathtt{ac}\to\mathtt{ac}}$, pois é ele quem contém as transições entre estados pertencentes a $\mathcal{A}(\mathcal{M})$. Aplicando N em E,

$$\begin{aligned} & N_{m\times n} E_{n\times n} \\ = & \quad \{ \text{ definição de } N \text{ (Equação 3.3.8) e } E \text{ (Definição 61, Página 47). } \} \end{aligned}$$

$$
\begin{aligned}
& \begin{bmatrix} I_{m\times m} & 0_{m\times(n-m)} \end{bmatrix} \begin{bmatrix} E_{\mathtt{ac}\to\mathtt{ac}} & 0 \\ E_{\mathtt{inac}\to\mathtt{ac}} & E_{\mathtt{inac}\to\mathtt{inac}} \end{bmatrix} \\
= & \quad \{ \text{ produto. } \} \\
& \begin{bmatrix} E_{\mathtt{ac}\to\mathtt{ac}} & 0_{(n-m)\times m} \end{bmatrix} \\
= & \quad \{ \text{ distributividade. } \} \\
& E_{\mathtt{ac}\to\mathtt{ac}} \begin{bmatrix} I_{m\times m} & 0_{(n-m)\times m} \end{bmatrix} \\
= & \quad \{ \ B = E_{\mathtt{ac}\to\mathtt{ac}}. \ \} \\
& BN
\end{aligned}
$$

Aqui temos dois fatos interessantes: de fato, fomos capazes de extrair o bloco de interesse e N normaliza E e B:

$$NE = BN. \tag{3.3.14}$$

Esta é exatamente a hipótese do teorema da fusão da estrela e o podemos utilizar para provar que a linguagem do autômato original é preservada após a transformação.

$$
\begin{aligned}
& u^T E^* v \\
= & \quad \{ \ u^T = s^T N \text{ (Página 63). } \} \\
& s^T N E^* v \\
= & \quad \{ \text{ teorema da fusão da estrela, da (Teorema 59, Página 44). } \} \\
& s^T B^* N v \\
= & \quad \{ \ Nv = t. \ \} \\
& s^T B^* t
\end{aligned}
$$

Como queríamos provar. Repare que pelo fato de N sempre ser inversível à direita temos uma fórmula fechada para computar a poda de $\mathcal{M}$.

$$
\begin{aligned}
& NE = BN \\
\Longrightarrow \quad & \{ \text{ Produto por } N^T \text{ à direita em ambos os lados e } NN^T = I. \ \} \\
& NEN^T = B
\end{aligned}
$$

Por fim,

$$\mathit{pod}(\mathcal{M}) = \langle Nu, NEN^T, Nv \rangle. \tag{3.3.15}$$

□

Ressalto que N depende diretamente de $\mathcal{M}$ e não é uma matriz genérica. De fato, podemos reparar que o efeito de filtro que N exerce aparece em todos os elementos do autômato, como se propagássemos o descarte. Afinal de contas, precisamos ser consistentes, se removemos um estado do grafo, este não pode aparecer no conjunto de iniciais, finais, nem haver transições alguma que o envolva.

4 CODETERMINISMO E COACESSBILIDADE

Assim como conceitos de dualidades surgem em diversas áreas matemáticas como álgebra linear, otimização e álgebra abstrata, teremos esta característica aqui também. Podemos definir codeterminismo, coacessibilidade e cocoisas *co*mais. O interessante é que muito do que já fizemos basta para provar propriedades deste "mundo *co*" de maneira direta ou quase direta.

4.1 REVERSÃO

Começaremos desenvolvendo a noção de dualidade com a qual trabalharemos. Toda a teoria será fundamentada na operação de *reversão* que, apesar de gramaticalmente ser um nome incorreto, representará o ato de inverter a ordem de uma palavra qualquer. Primeiro definiremos o processo para palavras e depois estenderemos para linguagens, vide Definições 96 e 98, respectivamente.

Definição 96 (Reversão de palavras)**.** *A operação* $\texttt{rev}_w : A^* \to A^*$*, que chamaremos de reversão, satisfaz*

$$\texttt{rev}_w(\varepsilon) = \varepsilon; \tag{4.1.1}$$

$$\texttt{rev}_w(a \cdot f) = \texttt{rev}_w(f) \cdot a, \qquad a \in A, f \in A^*. \tag{4.1.2}$$

Exemplo 97. $\texttt{rev}_w(botafogo) = ogfatob$ *e* $\texttt{rev}_w(rio\ de\ janeiro) = orienaj\ ed\ oir$*.*

Definição 98 (Linguagem reversa)**.** *Seja* $L \subseteq \mathcal{P}(A^*)$ *uma linguagem sobre um alfabeto* A*. A operação* $\texttt{rev}_{\mathcal{L}} : \mathcal{P}(A^*) \to \mathcal{P}(A^*)$ *define a linguagem reversa de* L *por*

$$\texttt{rev}_{\mathcal{L}}(L) = \{\texttt{rev}_w(f) \mid f \in L\}. \tag{4.1.3}$$

Podemos estender tal noção para matrizes e vetores. Se temos um elemento $E \in \mathcal{P}(A^*)^{m\times n}$*, então*

$$(\texttt{rev}_{\mathcal{L}}(E))_{ij} = \texttt{rev}_{\mathcal{L}}(E_{ji}). \tag{4.1.4}$$

Isto é, transpomos a matriz e aplicamos $\texttt{rev}_{\mathcal{L}}$ *em cada uma de suas entradas.*

Definição 99 (Notação compacta para reversão)**.** *Para ambas as reversões definidas, podemos simplificar a notação para*

$$\texttt{rev}_w(f) = f^R, \tag{4.1.5}$$

$$\texttt{rev}_{\mathcal{L}}(L) = L^R. \tag{4.1.6}$$

Exemplo 100. $\texttt{rev}_{\mathcal{L}}(\{botafogo, libertadores, 2024\}) = \{ogofatob, serodatrebil, 4202\}$.

Exemplo 101. *Considere* $E = \begin{bmatrix} 0 & haha \\ muahaha & 0 \end{bmatrix}$. *Então*

$$\begin{aligned} \texttt{rev}_{\mathcal{L}}(E) &= \texttt{rev}_{\mathcal{L}}\left(\begin{bmatrix} 0 & haha \\ muahaha & 0 \end{bmatrix}\right) \\ &= \begin{bmatrix} 0 & \texttt{rev}_{\mathcal{L}}(muahaha) \\ \texttt{rev}_{\mathcal{L}}(haha) & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & ahahaum \\ ahah & 0 \end{bmatrix}. \end{aligned} \tag{4.1.7}$$

Em alguns contextos, não estamos interessados em matrizes cujos elementos pertençam à $\mathcal{P}(A^*)$ como, por exemplo, em matrizes de transição. Conseguimos outras duas igualdades caso restrinjamos os elementos da matriz.

Corolário 102 (Reversão em matrizes restritas)**.** *Dadas matrizes* $X \in (A^*)^{m\times n}$ *e* $Y \in \mathcal{P}(A \cup \{1\})^{m\times n}$*, é verdade que*

$$(\texttt{rev}_{\mathcal{L}}(X))_{ij} = \texttt{rev}_w(X_{ji}); \tag{4.1.8}$$

$$\texttt{rev}_{\mathcal{L}}(Y) = Y^T. \tag{4.1.9}$$

Corolário 103 (Involução de $\texttt{rev}_w$ e $\texttt{rev}_{\mathcal{L}}$)**.** *Sejam* $f, g \in A^*$*, segue que*

$$\texttt{rev}_w(fg) = \texttt{rev}_w(g) \cdot \texttt{rev}_w(f). \tag{4.1.10}$$

De maneira semelhante, se temos $X \in \mathcal{P}(A^*)^{m\times n}$ *e* $Y \in \mathcal{P}(A^*)^{n\times p}$ *matrizes, então*

$$\texttt{rev}_{\mathcal{L}}(XY) = \texttt{rev}_{\mathcal{L}}(Y) \cdot \texttt{rev}_{\mathcal{L}}(X), \tag{4.1.11}$$

no qual $\texttt{rev}_{\mathcal{L}}(XY) \in \mathcal{P}(A^*)^{p\times m}$.

Exemplo 104. *A palavra mesopotâmia tem origem das palavras gregas "mésos" (meio) e "pótamus" (rio), vamos quebrá-la nestas duas partes. Revertendo-a teremos*

$$\texttt{rev}_w(meso \cdot potâmia) = \texttt{rev}_w(potâmia) \cdot \texttt{rev}_w(meso) = aimâtoposem.$$

Exemplo 105. *Considere* $X = \begin{bmatrix} 1 & aa \\ 0 & uu \end{bmatrix}$ *e* $Y = \begin{bmatrix} 0 & ha \\ 0 & hi \end{bmatrix}$. *Então*

$$\begin{aligned} \texttt{rev}_{\mathcal{L}}(XY) &= \texttt{rev}_{\mathcal{L}}(Y) \cdot \texttt{rev}_{\mathcal{L}}(X) \\ &= \texttt{rev}_{\mathcal{L}}\left(\begin{bmatrix} 0 & ha \\ 0 & hi \end{bmatrix}\right) \cdot \texttt{rev}_{\mathcal{L}}\left(\begin{bmatrix} 1 & aa \\ 0 & uu \end{bmatrix}\right) \\ &= \begin{bmatrix} 0 & 0 \\ ah & ih \end{bmatrix} \begin{bmatrix} 1 & 0 \\ aa & uu \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 \\ ah + ihaa & ihuu \end{bmatrix} \end{aligned} \tag{4.1.12}$$

Note que podemos pensar em elementos de $\mathcal{P}(A^*)$ como matrizes 1×1 e, desta forma, a definição para matrizes coincide exatamente com a de elementos individuais. Para dar sequência a outros resultados, precisaremos de uma propriedade bem interessante entre a operação de reversão e a estrela.

Lema 106 (Reversão e ponteciação comutam)**.** *Seja $L \in \mathcal{P}(A^*)$ uma linguagem, então, para todo $n \in \mathbb{N}$,*

$$\mathtt{rev}_{\mathcal{L}}(L^n) = (\mathtt{rev}_{\mathcal{L}}(L))^n \,. \tag{4.1.13}$$

Prova. Como caso base teremos $\mathtt{rev}_{\mathcal{L}}(L^0) = (\mathtt{rev}_{\mathcal{L}}(L))^0$, de modo que $L^0 = 1$. Isto é fácil verificar, visto que, por definição de potenciação, $1^0 = 1$. Agora para o caso recursivo assumamos $\mathtt{rev}_{\mathcal{L}}(L^n) = (\mathtt{rev}_{\mathcal{L}}(L))^n$ como hipótese de indução. Segue que

$$\begin{aligned}
& \mathtt{rev}_{\mathcal{L}}\,(L^{n+1}) \\
= \quad & \{ \text{ definição de potência. } \} \\
& \mathtt{rev}_{\mathcal{L}}\,(L \cdot L^n) \\
= \quad & \{ \text{ distributividade da reversão (Corolário 103). } \} \\
& \mathtt{rev}_{\mathcal{L}}\,(L^n) \cdot \mathtt{rev}_{\mathcal{L}}\,(L) \\
= \quad & \{ \text{ hipótese de indução. } \} \\
& (\mathtt{rev}_{\mathcal{L}}(L))^n \cdot \mathtt{rev}_{\mathcal{L}}\,(L) \\
= \quad & \{ \text{ definição de produto. } \} \\
& (\mathtt{rev}_{\mathcal{L}}(L))^{n+1}
\end{aligned}$$

□

Lema 107 (Reversão distribui sobre a união)**.** *Para toda linguagem L e S, segue que*

$$\mathtt{rev}_{\mathcal{L}}(L + S) = \mathtt{rev}_{\mathcal{L}}(L) + \mathtt{rev}_{\mathcal{L}}(S). \tag{4.1.14}$$

Prova. Decorre da definição de reversão:

$$\begin{aligned}
& \mathtt{rev}_{\mathcal{L}}(L + S) \\
= \quad & \{ \text{ definição de reversão de linguagens (Definição 98). } \} \\
& \sum_{f \in L \text{ ou } f \in S} f^R \\
= \quad & \{ \text{ definição de união de conjuntos. } \} \\
& \textstyle\sum_{f \in L} f^R + \sum_{g \in S} g^R \\
= \quad & \{ \text{ definição de reversão de linguagens (Definição 98). } \} \\
& \mathtt{rev}_{\mathcal{L}}(L) + \mathtt{rev}_{\mathcal{L}}(S)
\end{aligned}$$

□

Lema 108 (Reversão e estrela de linguagens comutam)**.** *Seja $L \in \mathcal{P}(A^*)$ uma linguagem, então*

$$\texttt{rev}_{\mathcal{L}}(L^*) = (\texttt{rev}_{\mathcal{L}}(L))^* . \tag{4.1.15}$$

Prova. Recorde que podemos definir a operação de estrela sobre linguagens como

$$L^* = \sum_{i \in \mathbb{N}} L^i.$$

Aplicando ao somatório,

$$\begin{aligned}
& \texttt{rev}_{\mathcal{L}}\left(L^*\right) \\
= \quad & \{ \text{ definição de estrela (Equação 2.5.2, Página 39). } \} \\
& \texttt{rev}_{\mathcal{L}}\left(\textstyle\sum_{i \in \mathbb{N}} L^i\right) \\
= \quad & \{ \text{ reversão distribui sobre união (Lema 107). } \} \\
& \textstyle\sum_{i \in \mathbb{N}} \texttt{rev}_{\mathcal{L}}\left(L^i\right) \\
= \quad & \{ \text{ reversão e potenciação comutam (Lema 108). } \} \\
& \textstyle\sum_{i \in \mathbb{N}} \left(\texttt{rev}_{\mathcal{L}}\left(L\right)\right)^i \\
= \quad & \{ \text{ definição de estrela (Equação 2.5.2, Página 39). } \} \\
& \left(\texttt{rev}_{\mathcal{L}}\left(L\right)\right)^*
\end{aligned}$$

□

Teorema 109 (Reversão e estrela comutam)**.** *Seja $E \in \mathcal{P}(A^*)^{n \times n}$ uma matriz quadrada, então*

$$\left(\texttt{rev}_{\mathcal{L}}\left(E\right)\right)^* = \texttt{rev}_{\mathcal{L}}\left(E^*\right) \tag{4.1.16}$$

Prova. Primeiro, a reversão de matrizes blocadas funciona exatamente igual à de matrizes convencionais. Logo, se

$$E = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix},$$

então sua reversão seguirá a mesma lógica convencional:

$$E^R = \begin{bmatrix} E_{11}^R & E_{21}^R \\ E_{12}^R & E_{22}^R \end{bmatrix}.$$

Pela definição de estrela temos

$$E^* = \begin{bmatrix} E_{11}^* E_{12} \Delta^* E_{21} E_{11}^* + E_{11}^* & E_{11}^* E_{12} \Delta^* \\ \Delta^* E_{21} E_{11}^* & \Delta^* \end{bmatrix}, \tag{4.1.17}$$

$$\left(E^R\right)^* = \begin{bmatrix} (E_{11}^R)^* E_{21}^R \left(\Delta^R\right)^* E_{12}^R (E_{11}^R)^* + (E_{11}^R)^* & (E_{11}^R)^* E_{21}^R \left(\Delta^R\right)^* \\ \left(\Delta^R\right)^* E_{12}^R (E_{11}^R)^* & \left(\Delta^R\right)^* \end{bmatrix}. \tag{4.1.18}$$

Relembrando que $\Delta = E_{21}E_{11}^{*}E_{12} + E_{22}$ e, portanto, $\Delta^R = E_{12}^R \left(E_{11}^R\right)^* E_{21}^R + E_{22}^R$. Faremos a prova por indução. O caso base é meramente uma matriz 1×1, equivalente à um elemento L de $\mathcal{P}(A^*)$ e que já foi provado pelo Lema 108. Como hipótese de indução suponha que, para quaisquer sub-blocos de E, a comutatividade seja verdadeira. Logo,

$$
\begin{aligned}
& \left(\texttt{rev}_{\mathcal{L}}\left(E\right)\right)^* \\
= \quad & \{ \text{ definição de } \left(\texttt{rev}_{\mathcal{L}}\left(E\right)\right)^* \text{ (Equação 4.1.18). } \} \\
& \begin{bmatrix} (E_{11}^R)^* E_{21}^R \left(\Delta^R\right)^* E_{12}^R (E_{11}^R)^* + (E_{11}^R)^* & (E_{11}^R)^* E_{21}^R \left(\Delta^R\right)^* \\ \left(\Delta^R\right)^* E_{12}^R (E_{11}^R)^* & \left(\Delta^R\right)^* \end{bmatrix} \\
= \quad & \{ \text{ hipótese de indução. } \} \\
& \begin{bmatrix} (E_{11}^*)^R E_{21}^R \left(\Delta^*\right)^R E_{12}^R (E_{11}^*)^R + (E_{11}^*)^R & (E_{11}^*)^R E_{21}^R \left(\Delta^*\right)^R \\ \left(\Delta^*\right)^R E_{12}^R (E_{11}^*)^R & \left(\Delta^*\right)^T \end{bmatrix} \\
= \quad & \{ \text{ definição de reversão nos blocos (Equação 4.1.4). } \} \\
& \begin{bmatrix} \left(E_{11}^* E_{12} \Delta^* E_{21} E_{11}^* + E_{11}^*\right)^R & \left(\Delta^* E_{21} E_{11}^*\right)^R \\ \left(E_{11}^* E_{12} \Delta^*\right)^R & \left(\Delta^*\right)^R \end{bmatrix} \\
= \quad & \{ \text{ definição de reversão (Equação 4.1.4). } \} \\
& \begin{bmatrix} E_{11}^* E_{12} \Delta^* E_{21} E_{11}^* + E_{11}^* & E_{11}^* E_{12} \Delta^* \\ \Delta^* E_{21} E_{11}^* & \Delta^* \end{bmatrix}^R \\
= \quad & \{ \text{ definição de } E^* \text{ (Equação 4.1.17). } \} \\
& \texttt{rev}_{\mathcal{L}}\left(E^*\right)
\end{aligned}
$$

□

Corolário 110 (Transposição e estrela comutam)**.** *Seja $E \in \mathcal{P}\left(A \cup \{\varepsilon\}\right)^{n \times n}$ uma matriz quadrada, então*

$$\left(E^T\right)^* = \left(E^*\right)^T. \tag{4.1.19}$$

O resultado do corolário intuitivamente já poderia fazer sentido. Podemos pensar que a matriz de transição E^T representa o exato mesmo grafo que E, mas com o sentido de todas as arestas invertido. Desta forma, $\left(E^T\right)^*$ computa todos os caminhos, mas baseado nas arestas invertidas. Por outro lado, $\left(E^*\right)^T$ computa todos caminhos e, somente após este processo, os inverte.

Por termos definido a linguagem reversa, é natural pensarmos se não há uma transformação que leve um autômato para uma versão que reconheça a linguagem reversa da linguagem do autômato original. De fato, podemos pensar em palavras como rótulos de caminhos e reverter uma palavra equivaleria a invertermos todas arestas de seu caminho, lendo-o a partir do último vértice. Isto é, se temos um caminho

$$q \xrightarrow{a} p \xrightarrow{b} .... \xrightarrow{x} r \xrightarrow{y} s,$$

após a inversão, teremos

$$s \xrightarrow{y} r \xrightarrow{x} .... \xrightarrow{b} p \xrightarrow{a} q.$$

Desta forma, sabemos que basta trocarmos o sentido de todas as arestas do grafo do autômato. Este efeito é obtido transpondo a matriz de transição E, visto que $E^T_{ij} = E_{ji}$. Adicional a isso, por estarmos lendo os caminhos "de trás para frente", o caminho de palavras reconhecidas deve ser visto dos finais aos iniciais, portanto os vetores u e v serão trocados. Formalizamos isso através do Teorema 111.

Teorema 111 (Linguagens regulares são fechadas sobre reversão)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A que reconheça uma linguagem $\mathcal{L}(\mathcal{M})$. Existe um autômato finito $\mathcal{M}^R$, nomeado autômato reverso de $\mathcal{M}$, cuja linguagem é $(\mathcal{L}(\mathcal{M}))^R$. Definiremos a função que efetua tal transformação como $\mathtt{rev}_{\mathtt{Aut}} : \mathtt{Aut} \to \mathtt{Aut}$, de modo que*

$$\mathcal{L} \circ \mathtt{rev}_{\mathtt{Aut}} = \mathtt{rev}_{\mathcal{L}} \circ \mathcal{L}; \tag{4.1.20}$$

$$\mathtt{rev}_{\mathcal{L}} \left(\mathcal{L} \left(\mathcal{M}\right)\right) = v^T \left(E^T\right)^* u. \tag{4.1.21}$$

Prova. Primeiro definamos a forma fechada da linguagem reversa:

$\mathtt{rev}_{\mathcal{L}}(\mathcal{L}(\mathcal{M}))$

$=$ { definição de linguagem (Definição 54, Página 42). }

$\mathtt{rev}_{\mathcal{L}}(u^T E^* v)$

$=$ { involução de $\mathtt{rev}_{\mathcal{L}}$ (Corolário 102). }

$\mathtt{rev}_{\mathcal{L}}(v) \cdot \mathtt{rev}_{\mathcal{L}}(E^*) \cdot \mathtt{rev}_{\mathcal{L}}(u^T)$

$=$ { estrela e reversão comutam (Teorema 109). }

$\mathtt{rev}_{\mathcal{L}}(v) \cdot (\mathtt{rev}_{\mathcal{L}}(E))^* \cdot \mathtt{rev}_{\mathcal{L}}(u^T)$

$=$ { u e v vetores de 0 e 1. }

$v^T (\mathtt{rev}_{\mathcal{L}}(E))^* u$

$=$ { reverter uma matriz de transição equivale a transpor (Equação 4.1.9). }

$v^T (E^T)^* u$

Note que o produto já está em um formato adequado para definirmos o autômato. O vetor v será seu vetor de estados iniciais, E^T sua matriz de transição e u seu vetor de estados finais; isto é,

$$\mathcal{M}^R = \langle v, E^T, u\rangle. \tag{4.1.22}$$

□

Exemplo 112. *A Figura 18 mostra um autômato e sua forma reversa. Antes da reversão, seus vetores de estados iniciais e finais, assim como sua matriz de transição, eram*

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad E = \begin{bmatrix} i & h & 0 \\ 0 & a & a \\ i & 0 & 0 \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.1.23}$$

Após o processo de reversão, trocamos os estados iniciais e finais e transpomos a matriz de transição, obtendo

$$v = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad E^T = \begin{bmatrix} i & 0 & i \\ h & a & 0 \\ 0 & a & 0 \end{bmatrix}, \quad u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{4.1.24}$$

(a) AFND de exemplo antes da reversão. Sua linguagem é $(i^*haa^*)^*$.

(b) AFND da Figura 18a após o processo de reversão. Sua linguagem é $\mathtt{rev}_{\mathcal{L}}\left((i^*haa^*)^*\right) = (aa^*hi^*i)^*$.

Figura 18 – Exemplo de autômato antes (18a) e após ser revertido (18b). Note que todas as arestas do autômato tiveram seu sentido trocado, assim como o estado inicial se tornou final e o final se tornou inicial.

Corolário 113 (Reversão de autômatos é involutiva)**.** *Para qualquer autômato $\mathcal{M} \in \mathtt{Aut}$, segue-se que*

$$\mathtt{rev}_{\mathtt{Aut}}(\mathtt{rev}_{\mathtt{Aut}}(\mathcal{M})) = \mathcal{M}. \tag{4.1.25}$$

Baseado nas noções de autômatos reversos, conseguimos demonstrar uma regra de preservação entre as linguagens do autômato antes e após a reversão. Esta preservação é quem nos garante que processos que veremos mais à frente, como codeterminização e remoção de estados coinacessíveis, são consistentes.

Teorema 114 (Consistência em $\mathcal{M}$ implica consistência em $\mathcal{M}^R$)**.** *Seja $\varphi : \mathtt{Aut} \to \mathtt{Aut}$ uma transformação de autômatos. Se tal transformação não altera a linguagem de $\mathcal{M}$, consequentemente não altera a de $\mathcal{M}^R$; isto é,*

$$\mathcal{L} = \mathcal{L} \circ \varphi \implies \mathcal{L} = \mathcal{L} \circ \mathtt{rev}_{\mathtt{Aut}} \circ \varphi \circ \mathtt{rev}_{\mathtt{Aut}}. \tag{4.1.26}$$

Prova.

$$
\begin{aligned}
& \mathcal{L} \circ \mathit{rev}_{\mathit{Aut}} \circ \varphi \circ \mathit{rev}_{\mathit{Aut}} \\
= & \quad \{ \text{ linguagens de } \mathcal{M} \text{ e } \mathcal{M}^R \text{ (Equação 4.1.20). } \} \\
& \mathit{rev}_{\mathcal{L}} \circ \mathcal{L} \circ \varphi \circ \mathit{rev}_{\mathit{Aut}} \\
= & \quad \{ \text{ hipótese de } \varphi. \ \} \\
& \mathit{rev}_{\mathcal{L}} \circ \mathcal{L} \circ \mathit{rev}_{\mathit{Aut}} \\
= & \quad \{ \text{ linguagens de } \mathcal{M} \text{ e } \mathcal{M}^R \text{ (Equação 4.1.20). } \} \\
& \mathcal{L} \circ \mathit{rev}_{\mathit{Aut}} \circ \mathit{rev}_{\mathit{Aut}} \\
= & \quad \{ \ \mathit{rev}_{\mathit{Aut}} \text{ é involutiva (Equação 4.1.25). } \} \\
& \mathcal{L}
\end{aligned}
$$

□

4.2 AUTÔMATOS CODETERMINÍSTICOS

Agora redefiniremos alguns conceitos de autômatos que vimos no capítulo anterior, mas para seus duais. A ideia principal se manterá a mesma, mas agora daremos atenção ao destino das arestas, não mais suas origens. Se ser determinístico implicava que nenhum estado possuía duas arestas de mesmo rótulo para dois vizinhos distintos, ser *codeterminístico* implicará que nenhum estado possuirá duas arestas de mesmo rótulo apontando para ele. Como os caminhos são lidos do fim ao início, se antes exigíamos um único inicial, agora exigimos um único final. Para a definição matricial, como a matriz de transição do autômato reverso é a transposta da matriz de transição do autômato original, analisaremos agora suas colunas e não linhas.

Definição 115 (Autômato Finito Codeterminístico)**.** *Seja $\mathcal{M} = \langle Q, A, E, I, T \rangle$ um AFND sobre um alfabeto A, este é um Autômato Finito Codeterminístico (AFCD) se, e somente se,*

- *Possui exatamente um estado final;*
- *A relação de transição E^T é uma função total de assinatura $Q \times A \to Q$.*

Definição 116 (AFCD matricialmente)**.** *Seja $\mathcal{M} = \langle u, E, v \rangle$ um AFND sobre um alfabeto A, este é dito um Autômato Finito Codeterminístico (AFCD) se, e somente se,*

- *O vetor de estados finais é um vetor canônico, ou seja, há exatamente uma entrada não-nula;*
- *Para cada elemento $a \in A$, cada coluna da matriz $E^{(a)}$ é canônica, isto é, tem exatamente uma única entrada não-nula.*

Exemplo 117. *A Figura 19 mostra um exemplo de um autômato que não é codeterminístico. O estado r é destino de duas arestas de rótulos iguais.*

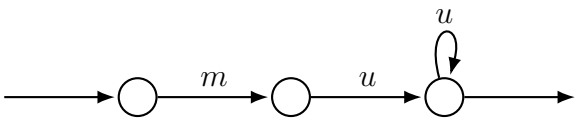


Figura 19 – AFND não-codeterminístico de exemplo.

Exemplo 118. *A Figura 20 mostra um exemplo de um autômato que é codeterminístico. Nenhum estado é destino de duas arestas de mesmo rótulo.*

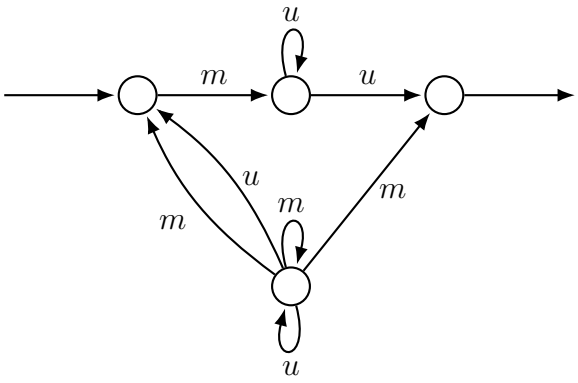


Figura 20 – AFND codeterminístico de exemplo.

O teorema de codeterminização garante que todo AFND pode ser codeterminizado. A prova deste teorema se assemelha bastante à prova de determinização que já fizemos, portanto focaremos em descrevê-la à partir desta óptica. Em verdade, podemos pensar na codeterminização como um processo de reversão e de determinização.

Teorema 119 (Codeterminização)**.** *Dado um alfabeto A, existe uma função* $\texttt{codet} : \texttt{Aut} \to \texttt{Aut}$ *tal que, para todo AFND $\mathcal{M} = \langle u, E, v\rangle$ sobre A, existe um AFCD $\widetilde{\mathcal{M}} = \langle \tilde{u}, \widetilde{E}, \tilde{v}\rangle$ sobre o mesmo alfabeto, de modo que*

$$\texttt{codet}(\langle u, E, v\rangle) = \langle \tilde{u}, \widetilde{E}, \tilde{v}\rangle; \tag{4.2.1}$$

$$u^T E^* v = \tilde{u}^T \widetilde{E}^* \tilde{v}. \tag{4.2.2}$$

Prova. Pensaremos em $\texttt{codet}$ como uma composição de três transformações:

$$\texttt{codet} = \texttt{rev}_{\texttt{Aut}} \circ \texttt{det} \circ \texttt{rev}_{\texttt{Aut}}. \tag{4.2.3}$$

Isto é, primeiro encontramos o autômato reverso de $\mathcal{M}$, o determinizamos e revertemos o autômato novamente.

$$\begin{aligned} & \texttt{codet}(\mathcal{M}) \\ = \quad & \{ \text{ definição de codeterminização como composições (Equação 4.2.3). } \} \end{aligned}$$

$$
\begin{aligned}
& \mathtt{rev}_{\mathtt{Aut}}(\mathtt{det}(\mathtt{rev}_{\mathtt{Aut}}(\mathcal{M}))) \\
= \quad & \{ \text{ definição de } \mathcal{M}. \ \} \\
& \mathtt{rev}_{\mathtt{Aut}}(\mathtt{det}(\mathtt{rev}_{\mathtt{Aut}}(\langle u, E, v\rangle))) \\
= \quad & \{ \text{ definição de reversão de autômatos (Equação 4.1.22). } \} \\
& \mathtt{rev}_{\mathtt{Aut}}(\mathtt{det}(\langle v, E^T, u\rangle)) \\
= \quad & \{ \text{ definição de determinização (Teorema 83, Página 56). } \} \\
& \mathtt{rev}_{\mathtt{Aut}}(\langle \hat{v}, \hat{M}, \hat{u}\rangle) \\
= \quad & \{ \text{ definição de reversão de autômatos (Equação 4.1.22). } \} \\
& \langle \hat{u}, \hat{M}^T, \hat{v}\rangle
\end{aligned}
$$

O autômato final deste processo é codeterminístico, visto que todas as condições são satisfeitas:

- $\hat{v}$ é um vetor canônico, dado que antes da última operação de reversão este era o vetor de estados iniciais de $\mathtt{det}(\langle v, E^T, u\rangle)$ e, pela definição de autômatos determinísticos, estes possuem sempre um único inicial (Definição 65, Página 49);
- As colunas de M^T são todas canônicas, este fato vale pois a matriz M do autômato $\mathtt{det}(\langle v, E^T, u\rangle)$ possuía todas as suas linhas canônicas pela definição de determinísmo (Definição 66, Página 50). Logo, ao transpormos M, suas colunas passam a ser canônicas;
- A linguagem do autômato original é preservada após todo este processo, como consequência do Teorema 114.

□

Corolário 120 (Consistência na codeterminização)**.** *Seja $\mathcal{M}$ um AFND, sua linguagem se mantém a mesma após o processo de codeterminização; isto é,*

$$\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathtt{codet}(\mathcal{M})). \tag{4.2.4}$$

Prova. Utilize o Teorema 114 de consistência de linguagens com $\varphi = \mathtt{det}$. Pela Equação 4.2.3, temos exatamente $\mathtt{codet}$ e, portanto, codeterminizar preserva a linguagem. □

Exemplo 121. *A Figura 21 exemplifica o processo de codeterminização. O autômato original passa de três para oito estados, cada um representando um subconjunto de estados diferente. Os vetores de estados iniciais e finais, assim como sua matriz de transição, eram*

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad E = \begin{bmatrix} 0 & m & 0 \\ 0 & 0 & u \\ 0 & 0 & u \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.2.5}$$

Aplicando a operação de reversão, o autômato resultante é o presente na Figura 21b, que matricialmente se torna

$$v = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad E^T = \begin{bmatrix} 0 & 0 & 0 \\ m & 0 & 0 \\ 0 & u & u \end{bmatrix}, \quad u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{4.2.6}$$

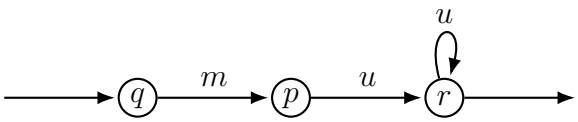
Aplicando a determinização, obtemos o autômato da Figura 21c, que matricialmente equivale a

$$\hat{u} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \widehat{E} = \begin{bmatrix} 0 & 0 & 0 & 0 & m+u & 0 & 0 & 0 \\ m & u & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u & 0 & 0 & m & 0 & 0 & 0 \\ m & 0 & 0 & 0 & u & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & m+u & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & m+u & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & m+u & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & m+u & 0 & 0 & 0 \end{bmatrix}, \quad \hat{v} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \tag{4.2.7}$$

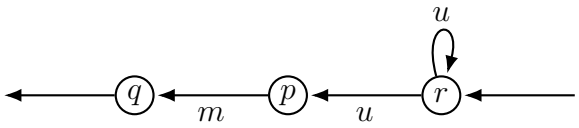
Para obtermos o resultado final da codeterminização, precisamos reverter o autômato novamente, obtendo o autômato da Figura 21d. Matricialmente temos

$$\tilde{u} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \ \widetilde{E} = \begin{bmatrix} 0 & m & 0 & m & 0 & 0 & 0 & 0 \\ 0 & u & u & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ m+u & 0 & m & u & m+u & m+u & m+u & m+u \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \ \tilde{v} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{4.2.8}$$

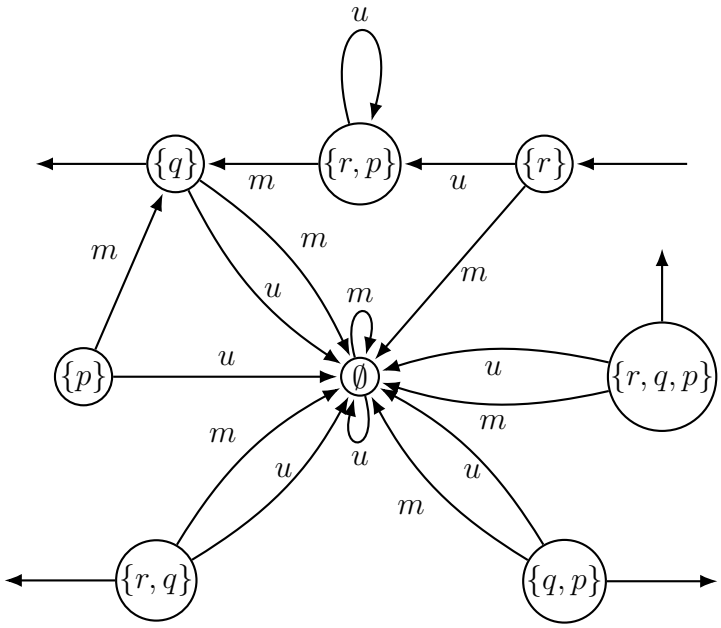
Note que cada símbolo aparece uma vez em cada coluna de $\widetilde{E}$. Além disso, todos os estados que representam subconjuntos aos quais q pertence se tornaram iniciais também, pois q era estado inicial do autômato original.



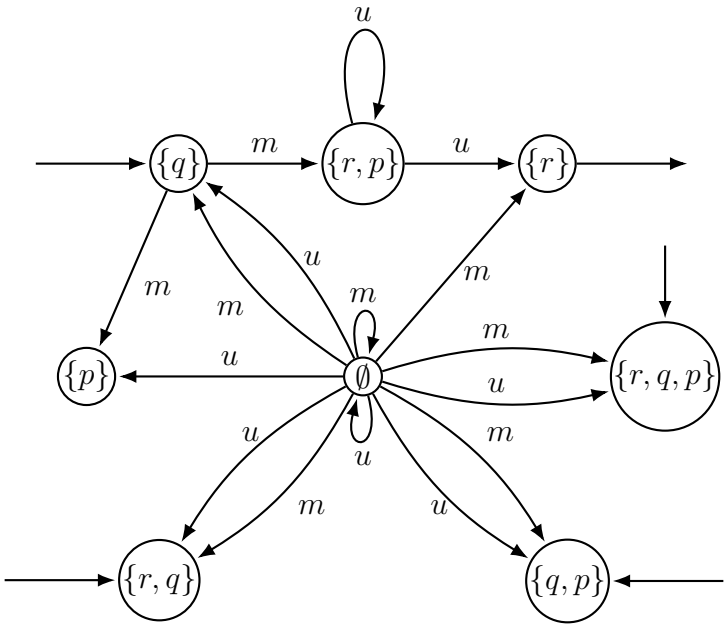
(a) Autômato finito não-codeterminístico para a linguagem de palavras que começam com mu e são seguidas de uma sequência de u.



(b) Autômato finito da Figura 21a após o processo de reversão.



(c) Autômato finito da Figura 21b após o processo de determinização.



(d) Autômato da Figura 21a após o processo de codeterminização.

Figura 21 – Exemplo de um AFND antes (21a) e após o processo de codeterminização (21d) descrito pelo Teorema 119.

4.2.1 Autômatos cocompletos

Se determinização gera um autômato completo, codeterminização gera um autômato *cocompleto*. Sua definição é direta: se antes observávamos pares de origem e rótulo, agora observaremos pares de destino e rótulo.

Definição 122 (Autômato Cocompleto)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A, este é cocompleto se, e somente se, para todo par de destino e rótulo há, pelo menos, uma transição correspondente no autômato. Algebricamente, para toda coluna i de sua matriz de transição temos*

$$\text{Para todo } a \text{ em } A \text{ existe } j \text{ tal que } E^{(a)}_{ji} = 1. \tag{4.2.9}$$

Naturalmente, se podemos tornar um AFND em completo, podemos torná-lo em cocompleto de uma maneira bem semelhante.

Exemplo 123. *O autômato da Figura 22 é cocompleto. Em verdade, é codeterminístico.*

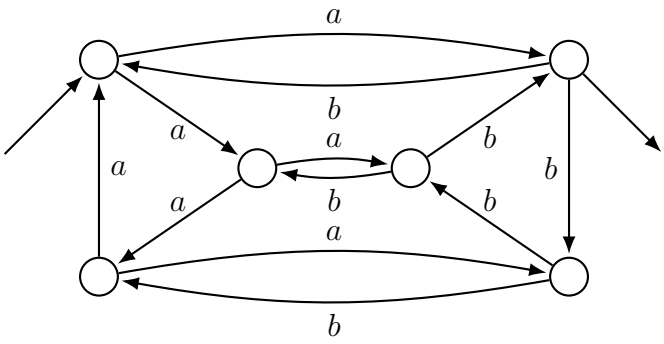


Figura 22 – Autômato finito não-determinístico cocompleto de exemplo. Todo estado possui arestas de rótulo a e b chegando em si.

Teorema 124 (Todo autômato pode ser cocompletado)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A, existe um AFND cocompleto $\mathcal{B} = \langle s, B, t\rangle$, também sobre A, tal que*

$$u^T E^* v = s^T B^* t. \tag{4.2.10}$$

Prova. Tome $\mathcal{M}$ e crie um estado novo chamado m. Para todo estado q e símbolo $a \in A$, adicione uma aresta $m \xrightarrow{a} q$ se

$$\sum_i^n E^{(a)}_{iq} = 0. \tag{4.2.11}$$

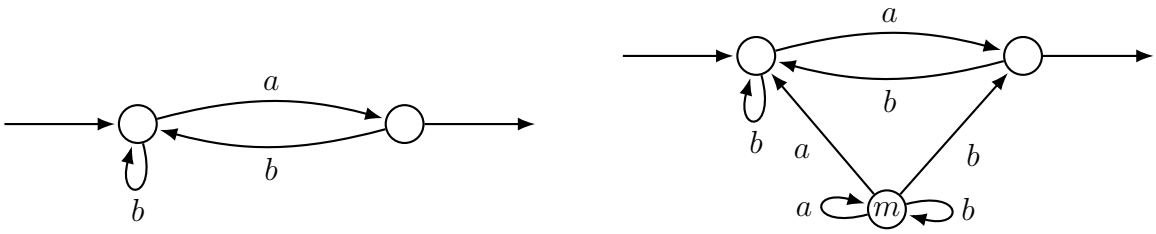
Isto é, se q é destino de nenhuma aresta de rótulo a. Por fim, adicione arestas de m para si próprio para cada valor a de A e adicione tudo isto a uma nova matriz B que represente este novo grafo. Mantendo os mesmos estados iniciais e finais ficamos com

$$s = \begin{bmatrix} u \\ 0 \end{bmatrix}, \quad t = \begin{bmatrix} v \\ 0 \end{bmatrix}. \tag{4.2.12}$$

□

Por vezes, m é chamado de *estado morto* devido ao fato de que não há arestas que cheguem nele, de modo que ele nunca pode ser acessado a partir de um inicial. Estados com essa característica são comuns e podem aparecer após o processo de codeterminização.

Exemplo 125. *Um exemplo do processo de cocompletar é ilustrado na Figura 23.*



(a) AFND de exemplo.

(b) Autômato da Figura 23a após o processo de transformação em um autômato cocompleto.

Figura 23 – Exemplo de autômato antes (23a) e após ser transformado em um autômato cocompleto sobre o alfabeto $A = \{a, b\}$ (23b).

4.3 REMOÇÃO DE ESTADOS COINACESSÍVEIS

Como estamos recriando os conceitos de maneira dual, o leitor pode esperar que haja uma "copoda" que mitigue este problema. De fato há! Vamos descrevê-lo em detalhes. Entretanto, antes de adentrarmos no teorema de remoção de estados *co*inacessíveis, precisamos definir o que é *coacessibilidade*.

Definição 126 (Estado coacessível)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A com conjunto de estados Q. Se $q \in Q$ tal que*

$$\left(\mathcal{L}_{\mathbb{d}}(\mathcal{M})\right)_q \neq 0, \tag{4.3.1}$$

dizemos que q é coacessível em $\mathcal{M}$. Caso contrário, será coinacessível.

Definição 127 (Autômato coacessível)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A com conjunto de estados Q. Se todo estado $q \in Q$ for coacessível em $\mathcal{M}$, dizemos que o autômato é coacessível. Caso contrário, será coinacessível.*

Exemplo 128. *Na Figura 24 somente os estados r e s são coinacessíveis.*

Exemplo 129. *Os autômatos das Figuras 5, 6, 10, 11b, 12, 13a, 14a, 15a, 16, 17, 18, 19, 20, 22 e 23 são todos coacessíveis. Os das Figuras 13b, 14b, 15b e 24 são coinacessíveis.*

Lema 130 (Estados coinacessíveis não agregam à linguagem)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND sobre um alfabeto A. Seja q um estado coinacessível em $\mathcal{M}$, então não existe palavra f reconhecida por $\mathcal{M}$ que possua um caminho que passe por q.*

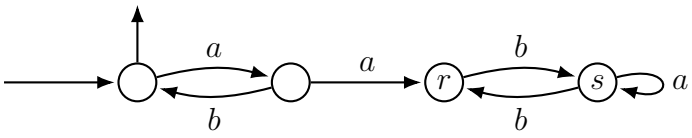


Figura 24 – Autômato finito não-determinístico para a linguagem de zero ou mais repetições de ab, $(ab)^*$.

Prova.

$$
\begin{aligned}
& u^T E^* v \\
= \quad & \{ \text{ definição de linguagem de um estado (Definição 90, Página 60). } \} \\
& (\mathcal{L}_{\mathsf{e}}(\mathcal{M}) \cdot e_q) \left(e_q^T \cdot \mathcal{L}_{\mathsf{d}}(\mathcal{M})\right) \\
= \quad & \{ \ q \text{ é coinacessível em } \mathcal{M} \text{ (Definição 126). } \} \\
& (\mathcal{L}_{\mathsf{e}}(\mathcal{M}) \cdot e_q) \cdot 0 \\
= \quad & \{ \text{ produto pelo elemento neutro da soma. } \} \\
& 0
\end{aligned}
$$

□

Teorema 131 (Copoda)**.** *Existe uma transformação* $\texttt{copod} : \texttt{Aut} \to \texttt{Aut}$ *tal que, para todo AFND* $\mathcal{M} = \langle u, E, v\rangle$ *sobre um alfabeto* A*, existe um AFND coacessível* $\mathcal{B} = \langle s, B, t\rangle$ *sobre o mesmo alfabeto que satisfaz*

$$\texttt{copod}(\langle u, E, v\rangle) = \langle s, B, t\rangle; \tag{4.3.2}$$

$$u^T E^* v = s^T B^* t. \tag{4.3.3}$$

Chamamos $\texttt{copod}$ *de copoda e dizemos que* $\mathcal{B}$ *é o autômato* $\mathcal{M}$ *copodado.*

Prova. Assim como fizemos na prova de codeterminização, separaremos a operação de copoda em uma composição de três funções:

$$\texttt{copod} = \texttt{rev}_{\texttt{Aut}} \circ \texttt{pod} \circ \texttt{rev}_{\texttt{Aut}}. \tag{4.3.4}$$

Desta forma, primeiro revertemos o autômato, aplicamos o processo de poda e, em seguida, aplicamos a função de reversão novamente.

$$
\begin{aligned}
& \texttt{copod}(\mathcal{M}) \\
= \quad & \{ \text{ definição de copoda como composições (Equação 4.3.4). } \} \\
& \texttt{rev}_{\texttt{Aut}}(\texttt{pod}(\texttt{rev}_{\texttt{Aut}}(\mathcal{M}))) \\
= \quad & \{ \text{ definição de } \mathcal{M}. \ \} \\
& \texttt{rev}_{\texttt{Aut}}(\texttt{pod}(\texttt{rev}_{\texttt{Aut}}(\langle u, E, v\rangle))) \\
= \quad & \{ \text{ definição de reversão de autômatos (Equação 4.1.22). } \}
\end{aligned}
$$

$$
\begin{aligned}
& \mathtt{rev}_{\mathtt{Aut}}\left(\mathtt{pod}\left(\langle v, E^T, u\rangle\right)\right) \\
= \quad & \{ \text{ definição de poda (Teorema 95, Página 62). } \} \\
& \mathtt{rev}_{\mathtt{Aut}}\left(\langle Nv, NE^TN^T, Nu\rangle\right) \\
= \quad & \{ \text{ definição de reversão de autômatos (Equação 4.1.22). } \} \\
& \langle Nu, N^TEN, Nv\rangle
\end{aligned}
$$

Reescrevendo,

$$s = Nv, \quad B = NE^TN^T \quad \text{e} \quad t = Nu. \tag{4.3.5}$$

Para ser coacessível, o vetor de linguagens à direita não pode conter zeros. Analisando o autômato $\langle t, B^T, s\rangle$ temos, pela definição de linguagem à direita,

$$\mathcal{L}_{\mathtt{d}}\left(\langle t, B^T, s\rangle\right) = \left(B^T\right)^* s. \tag{4.3.6}$$

Entretanto, note que isto é exatamente o vetor de linguagens à esquerda de seu reverso, $\langle s, B, t\rangle$, que é acessível pela definição de poda (Teorema 95, Página 62).

$$
\begin{aligned}
& \left(B^T\right)^* s \\
= \quad & \{ \text{ transposição comuta com a estrela (Corolário 110, Página 69). } \} \\
& \left(B^*\right)^T s \\
= \quad & \{ \text{ definição de transposição. } \} \\
& \left(s^T B^*\right)^T \\
= \quad & \{ \text{ definição de linguagem à esquerda (Definição 56, Página 56). } \} \\
& \mathcal{L}_{\mathtt{e}}\left(\langle s, B, t\rangle\right)
\end{aligned}
$$

Como $\langle s, B, t\rangle$ é acessível, não há entradas nulas no vetor de linguagens à esquerda e, portanto, não haverá também na linguagem à direita de $\langle t, B^T, s\rangle$. Restaria verificar que a linguagem do autômato é preservada após esse processo, que novamente sai diretamente do Teorema 114. □

Corolário 132 (Consistência na copoda)**.** *Seja $\mathcal{M}$ um AFND, sua linguagem se mantém a mesma após o processo de copoda; isto é,*

$$\mathcal{L}\left(\mathcal{M}\right) = \mathcal{L}\left(\mathtt{copod}(\mathcal{M})\right). \tag{4.3.7}$$

Prova. Utilize o Teorema 114 de consistência de linguagens com $\varphi = \mathtt{pod}$. Pela Equação 4.3.4, temos exatamente $\mathtt{copod}$ e, portanto, remover estados coinacessíveis preserva a linguagem. □

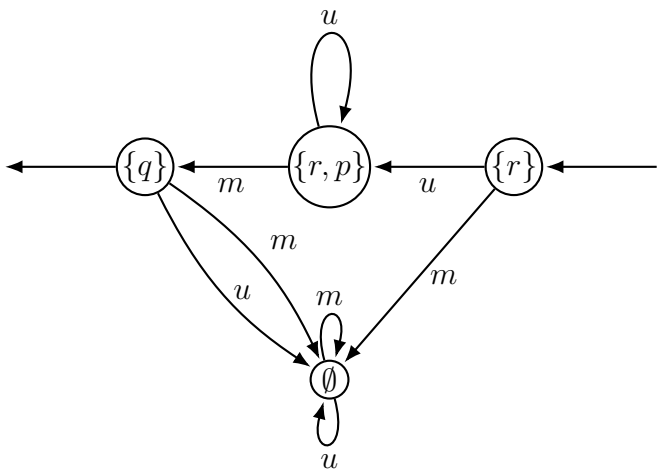
Exemplo 133. *Vamos remover os estados coinacessíveis do autômato da Figura 21d. Primeiro o revertemos e obtemos exatamente o autômato da Figura 21c; após isso, removemos todos os estados inacessíveis dele, obtendo o autômato da Figura 25a. Seu vetor de estados iniciais, vetor de estados finais e matriz de transição se tornam*

$$t = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad B^T = \begin{bmatrix} 0 & 0 & 0 & m+u \\ m & u & 0 & 0 \\ 0 & u & 0 & m \\ 0 & 0 & 0 & m+u \end{bmatrix}, \quad s = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{4.3.8}$$

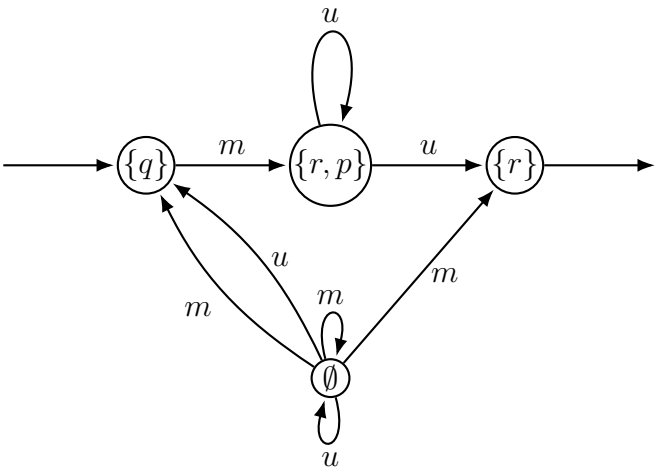
Após o processo de reversão, obtemos um autômato coacessível, cujo vetor de estados iniciais, vetor de estados finais e matriz de transição são

$$s = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & m & 0 & 0 \\ 0 & u & u & 0 \\ 0 & 0 & 0 & 0 \\ m+u & 0 & m & m+u \end{bmatrix}, \quad t = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}. \tag{4.3.9}$$

O resultado é justamente o autômato da Figura 20. O número de estados do autômato final, se comparado com o coinacessível, foi cortado pela metade. Este caso é um exemplo claro de o quão simplificado o autômato copodado pode ser se comparado com o original, para casos maiores isto se torna mais evidente.



(a) Autômato finito da Figura 21c após removermos todos os seus estados coinacessíveis.



(b) Autômato finito da Figura 25a após o processo de reversão.

Figura 25 – Exemplo do processo de remoção de estados coinacessíveis aplicado ao autômato da Figura 21d.

5 MINIMIZAÇÃO

Após tantas discussões, definições e teoremas, finalmente estamos prontos para descrever o processo de minimização de Brzozowski em maiores detalhes. Primeiro começaremos caracterizando o autômato determinístico mínimo e, em seguida, traremos resultados intermediários importantes. Por fim, trazemos a prova.

5.1 BRZOZOWSKI E SUA HIPÓTESE DE AUTÔMATO DETERMINÍSTICO MÍNIMO

Antes de adentrarmos no algoritmo de Brzozowski e como este funciona, precisamos definir o que é um autômato determinístico mínimo.

Definição 134 (Autômato determinístico mínimo de uma linguagem)**.** *Um AFND $\mathcal{M}$ que reconhece uma linguagem L é o autômato determinístico mínimo de L se, e somente se, é o autômato determinístico com o menor número de estados possíveis que reconhece L.*

Apesar de correta e de deixar claro quem é este autômato, esta definição não nos dá pista alguma de como computá-lo. Portanto, antes de qualquer tentativa de minimização, precisamos ter uma forma compatível de caracterização para um autômato determinístico mínimo. Há algumas caracterizações presentes na literatura, a que selecionamos tem a vantagem de se adequar bem com a teoria desenvolvida até aqui.

Teorema 135 (Condições necessárias e suficientes para minimalidade)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFD sobre um alfabeto A, este é mínimo se, e somente se,*

- *$\mathcal{M}$ é determinístico;*
- *$\mathcal{M}$ é acessível;*
- *A linguagens à direita de $\mathcal{M}$ são todas diferentes entre si.*

$$\left(\mathcal{L}_{\mathtt{d}}(\mathcal{M})\right)_p \neq \left(\mathcal{L}_{\mathtt{d}}(\mathcal{M})\right)_q, \text{ para todo } p, q \in Q \text{ tal que } p \neq q. \tag{5.1.1}$$

Neste trabalho provaremos que tais condições são *necessárias*, mas não provaremos que são *suficientes*. Além disso, um fato importante é que o autômato determinístico mínimo de uma linguagem é *único* a menos de permutações entre os estados, este fato também não provaremos.

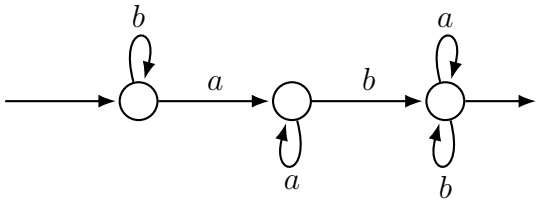


Figura 26 – Autômato determinístico mínimo para a linguagem de palavras que contêm ab.

Exemplo 136. *O autômato da Figura 26 é um exemplo de autômato determinístico mínimo. Seus vetores de iniciais e finais, assim como sua matriz de transição, são*

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad E = \begin{bmatrix} b & a & 0 \\ 0 & a & b \\ 0 & 0 & a+b \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{5.1.2}$$

Ele é determinístico, pois possui somente um inicial e todo par de origem e rótulo possui exatamente uma aresta correspondente. Quanto às linguagens à direita temos

$$E^*v = \begin{bmatrix} b^* & b^*a & b^*aa^*b(a+b)^* \\ 0 & a^* & a^*b(a+b)^* \\ 0 & 0 & (a+b)^* \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} b^*aa^*b(a+b)^* \\ a^*b(a+b)^* \\ (a+b)^* \end{bmatrix}. \tag{5.1.3}$$

No qual todas as entradas são diferentes. Não são disjuntas porque, por exemplo, b está presente na linguagem das duas últimas coordenadas. Por fim, todos os três estados são acessíveis, tornando o autômato acessível. Portanto, pelo Teorema 135, ele é mínimo.

As duas primeiras condições são mais diretas: Se estamos interessados no autômato *determinístico* mínimo, é esperado que ele seja determinístico; a segunda condição é compreensível através do Teorema 95, da Página 62. Se existe estado inacessível no autômato, então podemos removê-lo sem que a linguagem seja afetada e, portanto, o autômato original não era mínimo. Para entendermos a terceira condição, precisamos entender mais a fundo a relação de linguagens à esquerda e à direita com a linguagem de um autômato.

Teorema 137 (A linguagem de um autômato pode ser fatorada em esquerda e direita)**.** *Seja $\mathcal{M} = \langle u, E, v \rangle$ um AFND sobre um alfabeto A, a sua linguagem $\mathcal{L}(\mathcal{M})$ é o produto de seus vetores de linguagens à esquerda e à direita; isto é,*

$$\mathcal{L}(\mathcal{M}) = \mathcal{L}_e(\mathcal{M}) \cdot \mathcal{L}_d(\mathcal{M}). \tag{5.1.4}$$

Prova.

$\mathcal{L}_{\mathtt{e}}(\mathcal{M}) \cdot \mathcal{L}_{\mathtt{d}}(\mathcal{M})$

$=$ { definição de linguagens à esquerda (Definição 56) e linguagens à direita (Definição 57). }

$$
\begin{aligned}
& \left(u^T E^*\right) \cdot (E^* v) \\
= \quad & \{ \text{ associatividade. } \} \\
& u^T E^* E^* v \\
= \quad & \{ \ E^* E^* = E^* \text{ (Equação 2.3.25, Página 34). } \} \\
& u^T E^* v \\
= \quad & \{ \text{ definição de linguagem (Definição 54, Página 42). } \} \\
& \mathcal{L}(\mathcal{M})
\end{aligned}
$$

□

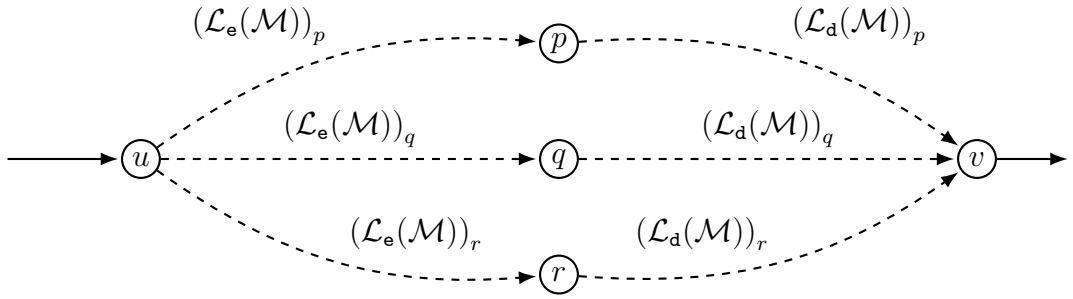


Figura 27 – Ilustração da linguagem de um autômato, $\mathcal{L}(\mathcal{M})$, fatorada em duas partes. A primeira é a linguagem à esquerda, equivalente a todos os caminhos que começam nos estados iniciais e terminam em qualquer estado do autômato; a segunda é a linguagem à direita, equivalente a todos os caminhos que começam em qualquer estado do autômato e terminam nos estados finais.

Desta forma, como ilustra a Figura 27, podemos partir a linguagem do autômato em

- Caminhos que começam nos estados iniciais e terminam em qualquer estado do autômato (linguagens à esquerda);
- Caminhos que começam em qualquer estado do autômato e terminam nos estados finais (linguagens à direita).

Devido a isso, para qualquer fatoração $f = gh$ de $f \in \mathcal{L}(\mathcal{M})$, podemos fixar um estado intermediário q no caminho tal que

$$f = gh, \quad g \in (\mathcal{L}_{\mathtt{e}}(\mathcal{M}))_q \quad \text{e} \quad h \in (\mathcal{L}_{\mathtt{d}}(\mathcal{M}))_q . \tag{5.1.5}$$

Exemplo 138. *Para o autômato da Figura 26, se $f = abab$, então vale que $ab \in (\mathcal{L}_{\mathtt{e}}(\mathcal{M}))_p$ e $ab \in (\mathcal{L}_{\mathtt{d}}(\mathcal{M}))_p$.*

Exemplo 139. *Para o autômato da Figura 26, se $f = ababa$, então vale que $abab \in (\mathcal{L}_{\mathtt{e}}(\mathcal{M}))_r$ e $a \in (\mathcal{L}_{\mathtt{d}}(\mathcal{M}))_r$.*

Se dois estados possuem a mesma linguagem à direita, então estão reconhecendo as mesmas palavras h e, portanto, podem ser unidos e o autômato não é mínimo. Desta forma, o autômato determinístico mínimo necessariamente terá todas as linguagens à direita diferentes.

Estamos restringindo as linguagens à direita, mas nada falamos das linguagens à esquerda. A princípio, seria plenamente plausível exigirmos o mesmo para a primeira parte dos caminhos, não exigimos tal condição de maneira explícita, pois autômatos determinísticos sempre possuem linguagens à esquerda *disjuntas*, como provaremos mais à frente. Intuitivamente, as condições estão garantindo que "enxugamos" o autômato ao máximo e remover um estado a mais faria com que perdêssemos palavras, deixando de reconhecê-las. Brzozowski comprovou que se aplicarmos $\mathtt{Min}$ em um AFND qualquer, de modo que

$$\mathtt{Min} = \mathtt{pod} \circ \mathtt{det} \circ \mathtt{copod} \circ \mathtt{codet}, \tag{5.1.6}$$

obtemos como resultado exatamente o autômato determinístico mínimo para a sua linguagem. Outro modo de descrever o processo de minimização é desmembrando as operações de codeterminização e copoda, de modo que

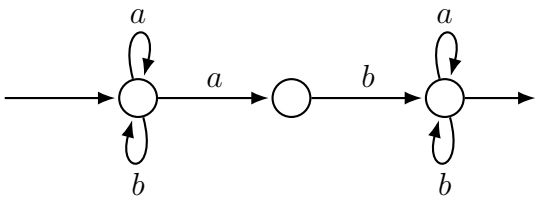
$$\begin{aligned} \mathtt{Min} &= \mathtt{pod} \circ \mathtt{det} \circ \underbrace{\mathtt{rev}_{\mathtt{Aut}} \circ \mathtt{pod} \circ \mathtt{rev}_{\mathtt{Aut}}}_{\mathtt{copod}} \circ \underbrace{\mathtt{rev}_{\mathtt{Aut}} \circ \mathtt{det} \circ \mathtt{rev}_{\mathtt{Aut}}}_{\mathtt{codet}} \\ &= \mathtt{pod} \circ \mathtt{det} \circ \mathtt{rev}_{\mathtt{Aut}} \circ \mathtt{pod} \circ \mathtt{det} \circ \mathtt{rev}_{\mathtt{Aut}}. \end{aligned} \tag{5.1.7}$$

Isto é razoável, pois, como vimos, se um autômato determinístico possui suas linguagens à esquerda disjuntas, o seu dual, codeterminístico, possuirá as linguagens *à direita* disjuntas. Entretanto, exigimos que estas sejam somente diferentes, mas por quê? Isto se deve ao fato de que nem sempre somos capazes de obter um autômato que tenha ambas as linguagens, à esquerda e à direita, disjuntas ao mesmo tempo, embora possamos sempre garantir que ao menos uma delas seja disjunta. A exigência de diferente é o *melhor* que conseguimos garantidamente e isto se deve ao fato de que, determinizar um autômato codeterminístico e coacessível, torna a *disjunção* em *diferença*.

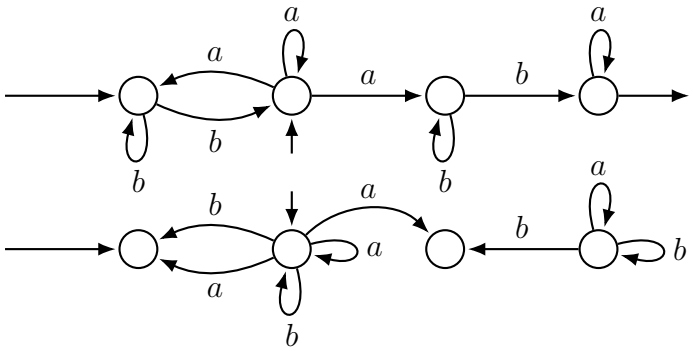
Exemplo 140. *Ilustremos o algoritmo com um exemplo. Podemos pensar no algoritmo de Brzozowski dividido em quatro etapas:*

- *Antes de tudo, temos o autômato original, representado na Figura 28a;*
- *A primeira etapa é aplicar a operação de codeterminização (Teorema 119, Página 73), restringindo as linguagens à direita. O resultado pode ser visto na Figura 28b;*
- *Após a codeterminização, o autômato tende a crescer e gerar estados coinacessíveis, removemos eles através da operação de copoda (Teorema 131, Página 79). O resultado é representado na Figura 28c;*

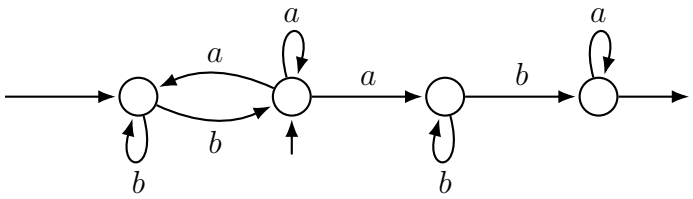
- *Com o autômato codeterminizado e coacessibilizado, resta garantirmos que ele é determinístico. Isto é feito através do processo de determinização (Teorema 83, Página 56), podendo acrescentar estados inacessíveis ao autômato; aplicamos a operação de remoção de estados inacessíveis (Teorema 119, Página 73) para removê-los. O resultado de ambas as operações é representado na Figura 28d.*



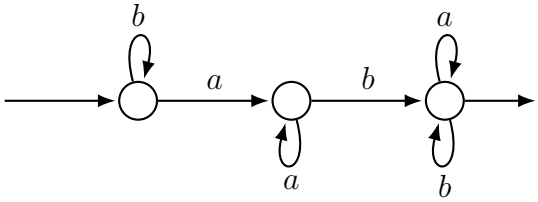
(a) Autômato finito não-determinístico para a linguagem de palavras com que contêm *ab*.



(b) Autômato da Figura 28a após o processo de codeterminização.



(c) Autômato da Figura 28b após seus estados coinacessíveis removidos. Seu tamanho foi reduzido pela metade.



(d) Autômato da Figura 28c após ser determinizado e, em seguida, ter seus estados inacessíveis removidos. Dos 16 estados presentes após a determinização, apenas 3 restaram após o processo de remoção de estados inacessíveis.

Figura 28 – Exemplo de aplicação do algoritmo de Brzozowski. Primeiro codeterminizamos o autômato para tornar suas linguagens a direita disjuntas; em seguida, removemos seus estados coinacessíveis, fazendo com que não haja estado com linguagem à direita de valor zero; após isto, determinizamos o autômato e removemos os estados inacessíveis.

Fizemos as operações de determinização e remoção de estados inacessíveis de uma só vez, pois o após a determinização do autômato da Figura 28c resultaria em um autômato de 16 *estados, que seria difícil de desenhar de maneira clara. O resultado sempre será um autômato determinístico mínimo para a mesma linguagem que o autômato original reconhecia. Note que o autômato da Figura 28d é o mesmo da Figura 26. O Exemplo 136 comprova que é mínimo.*

5.2 ONDE DETERMINISMO E CODETERMINISMO SE ENCONTRAM

Suponda válida a hipótese de Brzozowski de que o processo descrito na Equação 5.1.6 sempre resulta no autômato mínimo, precisamos de alguns resultados preliminares antes de provarmos seu teorema. Primeiro, precisamos comprovar que as linguagens à direita de um autômato codeterminístico são disjuntas, assim como garantir que remover seus estados coinacessíveis mantém tal propriedade. Por fim, precisamos mostrar como determinismo e codeterminismo se relacionam. Em nosso caso, isto requer que provemos que determinizar um autômato codeterminístico e coacessível torna suas linguagens à direita *diferentes*, que é a terceira exigência do Teorema 135. Todos estes resultados, sem excessões, são verdadeiros para seu dual, determinismo. Comecemos provando que as linguagens à direita de um autômato finito codeterminístico são realmente disjuntas.

Definição 141 (Vetor disjunto)**.** *O vetor $x \in \mathcal{P}(A^*)^n$ é dito disjunto se suas entradas são duas a duas disjuntas. Definimos o operador $\texttt{disj} : \mathcal{P}(A^*)^n \to \{0,1\}$ tal que*

$$\texttt{disj}(x) = \forall_{i \neq j} \left(x_i \cap x_j = 0\right). \tag{5.2.1}$$

Lema 142 (Disjunção da soma equivale à disjunções individuais)**.** *Sejam $x, y \in \mathcal{P}(A^*)^n$ vetores. Se $x_i \cap y_j = 0$ para todo i e j, então*

$$\texttt{disj}(x + y) \iff \texttt{disj}(x) \wedge \texttt{disj}(y). \tag{5.2.2}$$

Prova. Se x e y são disjuntos, então à medida que somamos y_i a x_i temos a garantia de que y_i não adicionará elementos já existentes ao vetor x. A única condição necessária e suficiente para que a soma siga disjunta, portanto, é que os vetores x e y sejam disjuntos, pois podem haver casos em que $x_i \cap x_j \neq 0$, tornando a parcela $(x_i + y_i) \cap (x_j + y_j) \neq 0$. □

Corolário 143 (Potências disjuntas)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFND e $x \in \mathcal{P}(A^*)^n$ disjunto. Segue-se que*

$$\texttt{disj}\left(\sum_{i \in \mathbb{N}} x^T E^i\right) \iff \forall i \in \mathbb{N}.\ \ \texttt{disj}(x^T E^i). \tag{5.2.3}$$

Prova. Sucessivas aplicações do Lema 142. □

Teorema 144 (Transitar em um codeterminístico lhe mantêm disjunto)**.** *Seja* $\mathcal{M} = \langle u, E, v\rangle$ *um AFCD sobre um alfabeto* A*. Sejam* $y \in \{0,1\}^n$ *disjunto com, no máximo, uma entrada não nula e* $x \in \mathcal{P}(A^*)^n$ *vetor disjunto, então* $Ex + y$ *é novamente disjunto.*

Prova. Por $\mathcal{M}$ ser codeterminismo, sempre haverá uma transição existente para cada par de destino e rótulo, isto é, nenhuma coluna de E é nula. Portanto, as entradas de Ex terão todas comprimentos maiores ou iguais a um, representando este efeito de transicionar um passo. Por outro lado, o vetor y, por ser um vetor de valores 0 ou 1, terá comprimento zero em todas as suas entradas. Assumindo que Ex seja disjunto, pelo Lema 142, $Ex + y$ é disjunto também. Para provar a hipótese de Ex ser disjunto, podemos expandi-lo sobre o alfabeto de $\mathcal{M}$. Deste modo,

$$Ex = \left(\sum_{a\in A} a \cdot E^{(a)}\right) x = \left(\sum_{a\in A} a \cdot E^{(a)} x\right). \tag{5.2.4}$$

Os diferentes termos do somatório serão necessariamente disjuntos entre si, pois começarão com símbolos distintos sempre. Isto é, para $a \neq b$, com $a, b \in A$, toda entrada de $a \cdot E^{(a)}x$ começará em a, inviabilizando a existência de qualquer elemento em comum com $b \cdot E^{(b)}x$. Portanto, suponha $a \in A$ sem perda de generalidade, para qualquer coordenada q vale que

$$\left(a \cdot E^{(a)}x\right)_q = a \cdot \sum_{j=1}^{n} x_q \cdot E^{(a)}_{qj}. \tag{5.2.5}$$

Como a é símbolo inicial de todos os elementos deste vetor, focaremos apenas na parte de $E^{(a)}x$. Estamos supondo que para quaisquer coordenadas p e q, $p \neq q$,

$$\left(E^{(a)}x\right)_q \cap \left(E^{(a)}x\right)_p = 0. \tag{5.2.6}$$

Decorre que

$$\begin{aligned}
&\left(E^{(a)}x\right)_q \cap \left(E^{(a)}x\right)_p \\
= \quad &\{ \text{ definição de } i\text{-ésima coordenada de } E^{(a)}x \text{ (Equação 5.2.5). } \} \\
&\left(\textstyle\sum_{j=1}^{n} x_j \cdot E^{(a)}_{qj}\right) \cap \left(\sum_{j=1}^{n} x_j \cdot E^{(a)}_{pj}\right) \\
= \quad &\{ \text{ distributividade da intersecção sobre a união. } \} \\
&\textstyle\sum_{i=1}^{n}\sum_{j=1}^{n} \left(\left(x_i \cdot E^{(a)}_{qi}\right) \cap \left(x_j \cdot E^{(a)}_{pj}\right)\right)
\end{aligned}$$

Para o caso $i = j$,

$$\begin{aligned}
&\left(x_i \cdot E^{(a)}_{qi}\right) \cap \left(x_j \cdot E^{(a)}_{pj}\right) \\
= \quad &\{ \ i = j. \ \}
\end{aligned}$$

$$
\begin{aligned}
& \left(x_i \cdot E_{qi}^{(a)}\right) \cap \left(x_i \cdot E_{pi}^{(a)}\right) \\
= & \quad \{ \text{ distributividade. } \} \\
& x_i \cdot \left(E_{qi}^{(a)} \cap E_{pi}^{(a)}\right) \\
= & \quad \{ \text{ codeterminismo não admite duas arestas de mesmo destino e rótulo. } \} \\
& x_i \cdot 0 \\
= & \quad \{ \text{ produto pelo elemento neutro da soma. } \} \\
& 0
\end{aligned}
$$

Para $i \neq j$, teremos dois casos:

- Ao menos uma das entradas $E_{qi}^{(a)}$ e $E_{pj}^{(a)}$ é 0;
- Ambas as entradas $E_{qi}^{(a)}$ e $E_{pj}^{(a)}$ valem 1. Neste caso,

$$
\begin{aligned}
& \left(x_i \cdot E_{qi}^{(a)}\right) \cap \left(x_j \cdot E_{pj}^{(a)}\right) \\
= & \quad \{ \ E_{qi}^{(a)} = E_{pj}^{(a)} = 1. \ \} \\
& x_i \cap x_j \\
= & \quad \{ \text{ hipótese de } \mathit{disj}(x). \ \} \\
& 0
\end{aligned}
$$

Segue que Ex é disjunto e, portanto, $Ex + y$ é disjunto pelo Lema 142. □

Teorema 145 (Linguagens à direita de um AFCD são disjuntas)**.** *Seja $\mathcal{M} = \langle u, E, v \rangle$ um AFCD sobre um alfabeto A. Seu vetor de linguagens à direita $\mathcal{L}_{\mathtt{d}}(\mathcal{M})$ é disjunto.*

Prova. Pelo Corolário 143, precisamos somente mostrar que $E^i v$ é disjunto para qualquer potência. Pelo Teorema 144, para $x = v$ e $y = 0$, temos que Ev é disjunto. Aplicando sucessivas vezes este teorema com $x = E^n v$ e $y = 0$, concluímos que $E^{n+1} v$ será disjunto também. Como toda potência é disjunta, podemos aplicar o Corolário 143 e concluir que $E^* v$ é disjunto. □

Desta forma, se uma dada palavra h está presente em $(\mathcal{L}_{\mathtt{d}}(\mathcal{M}))_q$, *não* há outro estado $p \neq q$ no autômato tal que h esteja em $(\mathcal{L}_{\mathtt{d}}(\mathcal{M}))_p$ e, portanto, unir dois estado é deixar de reconhecer palavras gh antes reconhecidas. Agora que provamos tal condição, precisamos mostrar que ao remover os estados coinacessíveis de um AFCD não fazemos com que ele deixe de ser codeterminístico. Este fato é importante para a garantia de que determinizar um autômato após os processos de codeterminização e remoção de estados coinacessíveis mantém as linguagens diferentes.

Teorema 146 (Remoção de estados coinacessíveis preserva codeterminismo)**.** *Seja $\mathcal{M} = \langle u, E, v \rangle$ um AFCD, o autômato $\mathtt{copod}(\mathcal{M})$ é novamente AFCD.*

Prova. A remoção de estados coinacessíveis remove somente estados e transições. Para violar o codeterminismo teríamos de adicionar novas transições ou possuir mais de um estado final. □

Portanto, note que as exigências do Teorema 135 são plausíveis e têm o papel de restringir as linguagens de cada estado, forçando que cada palavra possua uma, e unicamente uma, forma de ser reconhecida, tornando todo estado presente necessário.

Falta-nos provar como $\texttt{det}$ e $\texttt{codet}$ se relacionam, pois isto será de importância central na prova de corretude do algoritmo que se segue.

Lema 147 (F deixa disjuntos diferentes)**.** *Seja F conforme a Definição 78, da Página 54, sobre um conjunto Q. Se $x \in \mathcal{P}(A^*)^n$ é disjunto e sem entradas nulas, então Fx tem entradas diferentes; isto é,*

$$(Fx)_q \neq (Fx)_p \text{, para todo } q \neq p. \tag{5.2.7}$$

Prova. Sem perda de generalidade, fixemos P e S subconjuntos de Q, de modo que $S = S' \cup \{s\}$ e $s \in Q$ não pertença a P.

$$\begin{aligned}
& (Fx)_P \\
= \quad & \{ \text{ definição de produto matricial. } \} \\
& \textstyle\sum_{q \in Q} F_{(Pq)} x_q \\
= \quad & \{ \ F \text{ matriz de 0 ou 1 indicando se } q \text{ está em } P \text{ (Definição 78, Página 54). } \} \\
& \textstyle\sum_{p \in P} x_p
\end{aligned}$$

Por hipótese, $s \notin P$, logo $\sum_{p \in P} x_p$ não possui a parcela x_s, nem nenhum elemento presente em x_s, pelo fato de x ser disjunto. Finalizamos o argumento de $(Fx)_P$ ser diferente de $(Fx)_S$ pelo fato de x não conter entradas nulas, então existe pelo menos um elemento em $(Fx)_S$ que não está em $(Fx)_P$. Segue-se que os conjuntos são diferentes:

$$x_s \subseteq (Fx)_S \quad \text{e} \quad x_s \nsubseteq (Fx)_P. \tag{5.2.8}$$

□

Teorema 148 (Um autômato codeterminístico e coacessível mantém linguagens à direita diferentes se determinizado)**.** *Seja $\mathcal{M}$ um AFCD coacessível. O autômato $\texttt{det}(\mathcal{M})$ possui suas linguagens à direita diferentes; isto é,*

$$(\mathcal{L}_d(\texttt{det}(\mathcal{M})))_q \neq (\mathcal{L}_d(\texttt{det}(\mathcal{M})))_p \text{, para todo } q \neq p. \tag{5.2.9}$$

Prova. Definamos

$$\texttt{det}\,(\mathcal{M}) = \langle \hat{u}, \hat{E}, \hat{v} \rangle, \tag{5.2.10}$$

de modo que, conforme o teorema de determinização (Teorema 83, Página 56) descreve,

$$\hat{v} = Fv. \tag{5.2.11}$$

Pela Definição 78, da Página 54, de F, cada uma de suas linhas representa um subconjunto de Q, conjunto de estados de $\mathcal{M}$. Portanto, Fv extrai todos os subconjuntos que possuam o único estado final de $\mathcal{M}$. Se t é este estado, então

$$\left(Fv\right)_R = \begin{cases} 1, \text{ se } t \in R; \\ 0, \text{ caso contrário.} \end{cases} \tag{5.2.12}$$

Como agora estamos somando diferentes conjuntos disjuntos, precisamos provar que as linguagens à direita são todas diferentes, isto é,

$$\left(\hat{E}^*\hat{v}\right)_q \neq \left(\hat{E}^*\hat{v}\right)_p, \text{ para todo par de estados } q \neq p. \tag{5.2.13}$$

Segue-se que

$$\begin{aligned} & \hat{E}^*\hat{v} \\ = & \quad \{ \text{ definição de } \hat{v} \text{ (Equação 5.2.11). } \} \\ & \hat{E}^*\left(Fv\right) \\ = & \quad \{ \text{ associatividade do produto. } \} \\ & \hat{E}^*Fv \\ = & \quad \{ \ \hat{E}^*F = E^*F \text{ (Corolário 82, Página 56). } \} \\ & FE^*v \\ = & \quad \{ \text{ associatividade do produto. } \} \\ & F(E^*v) \end{aligned}$$

Por hipótese, $\mathcal{L}_{\mathtt{d}}(\mathcal{M}) = E^*v$ é disjunto e não possui entradas nulas, pois $\mathcal{M}$ é co-determinístico e coacessível. Logo, pelo Lema 147, $\mathcal{L}_{\mathtt{d}}(\mathcal{M})$ tem todas as suas entradas diferentes entre si. □

5.2.1 Resultados estendem para determinismo

Esta breve seção serve somente para enunciarmos todos os teoremas provados para AFCDs para sua versão dual, os AFDs.

Teorema 149 (Transicionar em um determinístico lhe mantém disjunto)**.** *Seja $\mathcal{M} = \langle u, E, v \rangle$ um AFD sobre um alfabeto A. Sejam $y \in \{0,1\}^n$ disjunto com, no máximo, uma entrada não nula e $x \in \mathcal{P}(A^*)^n$ vetor disjunto, então $x^TE + y^T$ é novamente disjunto.*

Prova. Presente no apêndice, entretanto simétrica à prova para o caso de AFCD. □

Teorema 150 (Linguagens à esquerda de um AFD são disjuntas)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFD sobre um alfabeto A. Seu vetor de linguagens à esquerda $\mathcal{L}_e(\mathcal{M})$ é disjunto.*

Prova. Presente no apêndice, entretanto simétrica à prova para o caso de AFCD. □

Teorema 151 (Remoção de estados inacessíveis preserva determinismo)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFD, o autômato $\mathtt{pod}(\mathcal{M})$ é novamente AFD.*

Prova. Presente no apêndice, entretanto simétrica à prova para o caso de AFCD. □

5.3 A CORRETUDE DO ALGORITMO DE BRZOZOWSKI

Após dezenas de páginas, é chegada a hora de provar a corretude do algoritmo de Brzozowski. Este teorema atesta que sempre há um autômato determinístico mínimo que respeite as condições do Teorema 135 para qualquer linguagem regular. Faremos esta prova de forma puramente construtiva e baseada na hipótese de Brzozowski (Equação 5.1.6), obtendo a forma do autômato determinístico mínimo no decorrer da prova, assim como a garantia de preservação de linguagem.

Teorema 152 (Brzozowski - Minimização)**.** *Existe uma transformação $\mathtt{Min} : \mathtt{Aut} \to \mathtt{Aut}$ tal que, para qualquer alfabeto A e qualquer autômato finito $\mathcal{M}$ sobre este alfabeto, $\mathtt{Min}(\mathcal{M})$ é determinístico mínimo. Temos que,*

$$\mathtt{Min} = \mathtt{pod} \circ \mathtt{det} \circ \mathtt{copod} \circ \mathtt{codet}; \tag{5.3.1}$$

$$\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathtt{Min}(\mathcal{M})). \tag{5.3.2}$$

Prova. Vamos nos basear nas três condições para ser mínimo listadas no Teorema 135. Comecemos avaliando a condição das linguagens à direita de $\mathtt{Min}(\mathcal{M})$ serem diferentes. Pelo Teorema 145, o vetor de linguagens à direita de um autômato codeterminístico é *disjunto*, então aplicamos $\mathtt{codet}$ em $\mathcal{M}$. Tendo em vista que desejamos reduzir o tamanho do autômato e visando manter as linguagens à direita diferentes, removemos seus estados coinacessíveis. Podemos tomar este passo porque $\mathtt{copod}$ preserva o codeterminismo, vide Teorema 146. Deste modo, temos

$$\widetilde{\mathcal{M}} = \mathtt{copod}(\mathtt{codet}(\mathcal{M})) = \langle \tilde{u}, \tilde{E}, \tilde{v}\rangle. \tag{5.3.3}$$

A exigência seguinte requer que o autômato resultante seja determinístico. Conseguimos isto facilmente aplicando $\mathtt{det}$ em $\widetilde{\mathcal{M}}$ conforme o Teorema 83, da Página 56. Pelo Teorema 148, o autômato resultante terá as linguagens à direita *diferentes*. Assim como o processo de codeterminização, a determinização incrementa o número de estados do autômato exponencialmente, potencialmente trazendo consigo estados inacessíveis que

não contribuem para a linguagem, vide Lema 94, da Página 61. Removemos todos os inacessíveis através do processo de poda descrito pelo Teorema 95, da Página 62. Pelo Teorema 151, o autômato resultante permanece determinístico, respeitando a primeira exigência. Após todas estas transformações, o autômato resultante satisfaz todas as três condições de minimalidade listadas no Teorema 135.

$$\mathit{Min}(\mathcal{M}) = \mathit{pod}(\mathit{det}(\mathit{copod}(\mathit{codet}(\mathcal{M})))) \,. \tag{5.3.4}$$

Segue que

$$\mathit{Min} = \mathit{pod} \circ \mathit{det} \circ \mathit{copod} \circ \mathit{codet}.$$

Resta-nos provar que a linguagem é preservada.

$$\begin{aligned}
& \mathcal{L}\,(\mathit{Min}(\mathcal{M})) \\
= \quad & \{ \text{ definição de } \mathit{Min} \text{ (Equação 5.1.6). } \} \\
& \mathcal{L}\,(\mathit{pod}(\mathit{det}(\mathit{copod}(\mathit{codet}(\mathcal{M}))))) \\
= \quad & \{ \ \mathit{pod} \text{ preserva a linguagem (Equação 3.3.7, Página 62). } \} \\
& \mathcal{L}\,(\mathit{det}(\mathit{copod}(\mathit{codet}(\mathcal{M})))) \\
= \quad & \{ \ \mathit{det} \text{ preserva a linguagem (Equação 3.2.19, Página 56). } \} \\
& \mathcal{L}\,(\mathit{copod}(\mathit{codet}(\mathcal{M}))) \\
= \quad & \{ \ \mathit{copod} \text{ preserva a linguagem (Equação 4.3.3, Página 79). } \} \\
& \mathcal{L}\,(\mathit{codet}(\mathcal{M})) \\
= \quad & \{ \ \mathit{codet} \text{ preserva a linguagem (Equação 4.2.2, Página 73). } \} \\
& \mathcal{L}\,(\mathcal{M})
\end{aligned}$$

Cada passo de preservação utilizando o teorema da fusão da estrela. □

Com isto, provamos que o algoritmo de Brzozowski sempre resulta em um autômato determinístico mínimo e que este preserva a linguagem do autômato original. Devido ao ferramental teórico que viemos desenvolvendo, a prova ficou pequena e vai direto ao ponto, encaixando-se bem ao o formalismo que utilizamos.

6 CONCLUSÕES

Neste trabalho, apresentamos uma prova calculacional para a corretude do algoritmo de Brzozowski para a minimização de autômatos finitos como uma tentativa de demonstrar como o teorema da fusão pode simplificar (ou reduzir) provas indutivas. O algoritmo de Brzozowski foi escolhido devido ao fato de, até nosso conhecimento, não possuir uma prova calculacional existente e por depender de inúmeros resultados intermediários que são provados através de indução no comprimento de caminhos. Junto à prova da corretude do algoritmo em si, trazemos a intuição do porquê o algoritmo funciona, provido de definições e provas acerca das três transformações utilizadas ao decorrer dele. Em especial, os principais resultados foram

- Uma prova construtiva para a fórmula fechada do fecho transitivo e reflexivo de uma matriz E cujos elementos pertencem à uma álgebra de Kleene;
- Uma prova construtiva da corretude do algoritmo de remoção de estados inacessíveis, provada com teorema da fusão e utilizando do formalismo de vetores e matrizes;
- Uma prova de que se um autômato é codeterminístico e coinacessível, então determinizá-lo torna suas linguagens à direita diferentes. Prova feita utilizando o teorema da fusão via formalismo matricial;
- Provas intermediárias necessárias para a prova da corretude do algoritmo de Brzozowski utilizando a representação matricial;
- Uma prova construtiva para a corretude do algoritmo de Brzozowski, com a prova feita utilizando o teorema da fusão e o formalismo de vetores e matrizes.

E por fim a prova de corretude do algoritmo. Todas as provas citadas acima foram feitas através de um mesmo teorema, nomeado teorema da fusão da estrela e serviu como um modo excelente de encurtar determinados trechos que exigiriam uma prova por indução convencional e potencialmente mais confusa. Acreditamos que este trabalho é mais um indício de que linguagens formais e álgebra linear podem ser vistas de maneira unificada e que tal conexão ajuda a simplificar certas noções e obter novos resultados enriquecedores, abrindo margem para outras contribuições nesta linha.

Como trabalhos futuros, temos uma gama, já que vários conceitos de linguagens formais podem ser representados usando álgebra linear. Para nós, o principal seria explorar o teorema da fusão *generalizado*. A versão do teorema da fusão que utilizamos no decorrer da monografia é restrito, em nosso caso, estamos lidando sempre com composições de funções lineares e não abrangendo o caso geral. Acreditamos que, através do teorema

mais geral, poderíamos provar alguns outros resultados presentes nesta monografia, entre os quais temos os teoremas

- De vetores disjuntos permanecerem disjuntos, quando transitam em autômatos determinísticos e codeterminísticos;
- De comutatividade entre a operação de reversão e transposição de matrizes;
- Das linguagens disjuntas tanto para determinismo quanto para codeterminismo.

Outro ponto onde o teorema da fusão generalizado poderia vir a ser usado é na teoria de linguagens livres-de-contexto. Em verdade, se linguagens regulares podem ser descritas através de pontos fixos de funções lineares, as livres-de-contexto são descritas através de pontos fixos de funções *não-lineares*.

REFERÊNCIAS

BACKHOUSE, R. Galois connections and fixed point calculus. In: _____. **Algebraic and Coalgebraic Methods in the Mathematics of Program Construction: International Summer School and Workshop Oxford, UK, April 10–14, 2000 Revised Lectures**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 89–150. ISBN 978-3-540-47797-6. Disponível em: https://doi.org/10.1007/3-540-47797-7_4.

CONWAY, J. H. **Regular algebra and finite machines, [by] J.H. Conway.** London: Chapman and Hall, 1971. (Chapman and Hall mathematics series). ISBN 0412106205.

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. Introduction to automata theory, languages, and computation, 2nd edition. **SIGACT News**, Association for Computing Machinery, New York, NY, USA, v. 32, n. 1, p. 60–65, mar. 2001. ISSN 0163-5700. Disponível em: https://doi.org/10.1145/568438.568455.

KOZEN, D. A completeness theorem for kleene algebras and the algebra of regular events. **Information and Computation**, v. 110, n. 2, p. 366–390, 1994. ISSN 0890-5401. Disponível em: https://www.sciencedirect.com/science/article/pii/S0890540184710376.

SAKAROVITCH, J. **Elements of Automata Theory**. Cambridge: Cambridge University Press, 2009.

SILVA, A. **Kleene Coalgebra**. Tese (Doutorado) — Radboud University Nijmegen, December 2010. Disponível em: https://www.cs.cornell.edu/courses/cs6861/2024sp/Handouts/Silva-thesis.pdf.

APÊNDICE A – RESULTADOS DE DETERMINISMO

Teorema 153 (Transicionar em um determinístico lhe mantém disjunto)**.** *Seja $\mathcal{M} = \langle u, E, v\rangle$ um AFD sobre um alfabeto A. Sejam $y \in \{0,1\}^n$ disjunto com, no máximo, uma entrada não nula e $x \in \mathcal{P}(A^*)^n$ vetor disjunto, então $x^T E + y^T$ é novamente disjunto.*

Prova. Por $\mathcal{M}$ ser determinístico, sempre haverá uma transição existente para cada par de origem e rótulo, isto é, nenhuma linha de E é nula. Portanto, as entradas de $x^T E$ terão todas comprimentos maiores ou iguais a um, representando este efeito de transicionar um passo. Por outro lado, o vetor y^T, por ser um vetor de valores 0 e 1, terá comprimento zero em todas as suas entradas. Assumindo que $x^T E$ seja disjunto, pelo Lema 142, da Página 88, $x^T E + y^T$, é disjunto também. Para provar a hipótese de $x^T E$ ser disjunto, podemos expandi-lo sobre o alfabeto $\mathcal{M}$. Deste modo,

$$x^T E = x^T \sum_{a \in A} a \cdot E^{(a)} = \left(\sum_{a \in A} x^T E^{(a)} \cdot a \right). \tag{A.0.1}$$

Os diferentes termos do somatório serão necessariamente disjuntos entre si, pois terminarão com símbolos distintos sempre. Isto é, para $a \neq b$, com $a, b \in A$, toda entrada de $x^T E^{(a)} \cdot a$ terminará em a, inviabilizando a existência de qualquer elemento em comum com $x^T E^{(b)} \cdot b$. Portanto, suponha $a \in A$ sem perda de generalidade, para qualquer coordenada q vale que

$$\left(x^T E^{(a)} \cdot a\right)_q = \left(\sum_{i=1}^{n} x_q \cdot E_{iq}^{(a)} \right) \cdot a \tag{A.0.2}$$

Como a é símbolo final de todos os elementos deste vetor, focaremos apenas na parte $x^T E^{(a)}$. Estamos supondo que para qualquer p e q, $p \neq q$,

$$\left(x^T E^{(a)}\right)_q \cap \left(x^T E^{(a)}\right)_p = 0 \tag{A.0.3}$$

Decorre que

$$\begin{aligned}
& \left(x^T E^{(a)}\right)_q \cap \left(x^T E^{(a)}\right)_p \\
= \quad & \{ \text{ definição de } i\text{-ésima coordenada de } x^T E^{(a)} \text{ (Equação A.0.2). } \} \\
& \left(\sum_{i=1}^{n} x_i \cdot E_{iq}^{(a)}\right) \cap \left(\sum_{i=1}^{n} x_i \cdot E_{ip}^{(a)}\right) \\
= \quad & \{ \text{ distributividade. } \} \\
& \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\left(x_i \cdot E_{iq}^{(a)}\right) \cap \left(x_j \cdot E_{jp}^{(a)}\right) \right)
\end{aligned}$$

Para o caso $i = j$,

$$\begin{aligned}
& \left(x_i \cdot E_{iq}^{(a)}\right) \cap \left(x_j \cdot E_{jp}^{(a)}\right) \\
= \quad & \{ \ i = j. \ \}
\end{aligned}$$

$$
\begin{aligned}
& \left(x_i \cdot E_{iq}^{(a)}\right) \cap \left(x_i \cdot E_{ip}^{(a)}\right) \\
= & \quad \{ \text{ distributividade. } \} \\
& x_i \cdot \left(E_{iq}^{(a)} \cap E_{ip}^{(a)}\right) \\
= & \quad \{ \text{ determinismo não admite duas arestas de mesma origem e rótulo. } \} \\
& x_i \cdot 0 \\
= & \quad \{ \text{ produto pelo elemento neutro da soma. } \} \\
& 0
\end{aligned}
$$

Para $i \neq j$, teremos dois casos:

- Ao menos uma das entradas $E_{iq}^{(a)}$ e $E_{jp}^{(a)}$ é 0;
- Ambas as entradas $E_{iq}^{(a)}$ e $E_{jp}^{(a)}$ valem 1. Neste caso,

$$
\begin{aligned}
& \left(x_i \cdot E_{iq}^{(a)}\right) \cap \left(x_j \cdot E_{jp}^{(a)}\right) \\
= & \quad \{ \ E_{iq}^{(a)} = E_{jp}^{(a)} = 1. \ \} \\
& x_i \cap x_j \\
= & \quad \{ \text{ hipótese de } \mathtt{disj}(x^T). \ \} \\
& 0
\end{aligned}
$$

Segue que $x^T E$ é disjunto e, portanto, $x^T E + y^T$ é disjunto pelo Lema 142. □

Teorema 154 (Linguagens à esquerda de um AFD são disjuntas)**.** *Seja $\mathcal{M} = \langle u, E, v \rangle$ um AFD sobre um alfabeto A. Seu vetor de linguagens à esquerda $\mathcal{L}_e(\mathcal{M})$ é disjunto.*

Prova. Pelo Corolário 143, da Página 88, precisamos somente mostrar que $u^T E^i$ é disjunto para qualquer potência. Pelo Teorema 153, para $x = u$ e $y = 0$, temos que $u^T E$ é disjunto. Aplicando sucessivas vezes este teorema com $x = u^T E^n$ e $y = 0$, concluímos que $u^T E^{n+1}$ será disjunto também. Como toda potência é disjunta, podemos aplicar o Corolário 143 e concluir que $u^T E^*$ é disjunto. □

Teorema 155 (Remoção de estados inacessíveis preserva determinismo)**.** *Seja $\mathcal{M} = \langle u, E, v \rangle$ um AFD, o autômato $\mathtt{pod}(\mathcal{M})$ é novamente AFD.*

Prova. A remoção de estados inacessíveis remove somente estados e transições. Para violar o determinismo teríamos de adicionar novas transições ou possuir mais de um estado inicial. □