Anotações de Linguagens Formais

Hugo Musso Gualandi

2025-03-26

1. Autômatos e Linguagens

Autômatos finitos são um modelo de um processo computacional. Além das aplicações diretas para processamento de texto e especificação de sistemas, eles também são o fundamento para modelos mais complexos, para computação em geral.

Alguns exemplos de perguntas que temos interesse em estudar:

- Para quais entradas um certo algoritmo produz a saída desejada?
- Duas implementações diferentes do mesmo algoritmo são equivalentes?
- Existem problemas que não podem ser solucionados pelo nosso modelo computacional?
- Qual é o impacto de introduzir não-determinismo no nosso modelo?

Vamos partir para um exemplo!

Esse autômato descreve uma disputa de pênaltis entre Brasil e Argentina. Já estamos nas cobranças alternadas, e a Argentina bate primeiro. Começamos no estado inicial E. A primeira linha acontece se a Argentina fizer o gol (a). Avançamos para o estado A, com vantagem para a Argentina, e esperamos a cobrança do Brasil. Se o Brasil também marcar (b), nós voltamos para a estaca zero; se o Brasil errar (x), a Argentina vence (F1). A segunda linha acontece se a argentina errar a cobrança (x), caso em que avançamos do estado E para o B. Desta vez, se o Brasil acertar a sua cobrança (b) ele ganha (F2), mas se errar (x) então voltamos ao início.

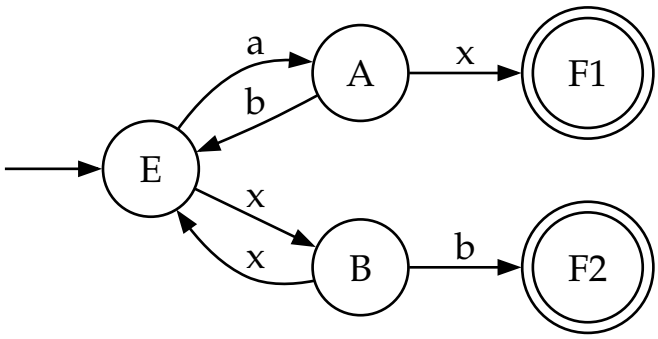


Figura 1: Cobrança de pênaltis entre Brasil e Argentina

Uma cobrança de pênaltis corresponde a um caminho no grafo, que sai do estado inicial e termina em algum dos estados finais. Representamos este caminho pela palavra que ele percorre, juntando as letras de cada aresta percorrida. Em ordem alfabética, as seguintes cobranças de pênaltis são válidas:

- ax, xb
- abax, abxb, xxax, xxbx
- ababax, ababxb, abxxax, abxxbx, xxabax, xxabxb, xxxxax, xxxxbx
- ...

Este conjunto de palavras caracteriza o comportamento do autômato. Às vezes é possível construir um outro autômato que reconhece a mesma linguagem e portanto se comporta de forma equivalente. Por exemplo, as palavras do nosso conjunto não

distinguem quem ganhou a disputa, então não faz diferença se juntarmos os estados F1 e F2 em um estado só.

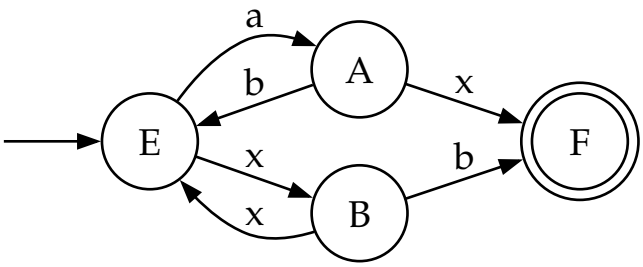
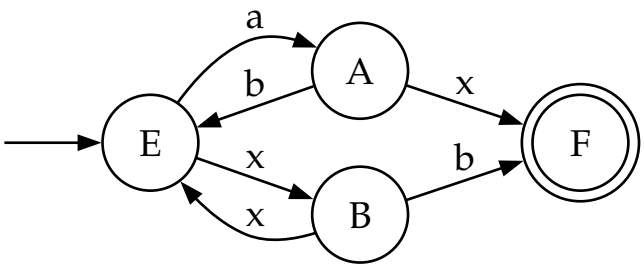


Figura 2: Como o vencedor não importa, podemos juntar F1 e F2



Exercício: As palavras abaixo não são disputas de pênaltis válidas. Justifique isso tanto pelas regras do futebol, quanto pelo comportamento do autômato.

a) a x b) aa c) axa d) ba

1.1. Strings

Nossos programas de computador recebem uma sequência de símbolos e retornam como saída uma resposta de sim ou não. Vamos agora discutir quais são as propriedades esperadas de uma sequência de símbolos.

Alfabeto Σ, um conjunto finito de símbolos.
String/Palavra Uma sequência finita de símbolos.
String vazia A letra ε denota a string com zero símbolos.

O alfabeto pode conter qualquer símbolo. Nos exemplos é comum usarmos letras de a até z, mas a princípio pode ser qualquer coisa inclusive símbolos de pontuação e espaços.

A notação Σ^* se refere ao **conjunto de todas as strings** que podem ser construídas com os símbolos do alfabeto Σ. É comum escrevermos $w \in \Sigma^*$ para dizer que w é uma string com símbolos do alfabeto Σ.

Algumas operações comuns sobre strings:

Concatenação $\mathrm{ab} \cdot \mathrm{cd} = \mathrm{abcd}$
Exponenciação $(\mathrm{ab})^3 = \mathrm{ababab}$
Comprimento $|\mathrm{abcd}| = 4$
Reversão $(\mathrm{abc})^R = \mathrm{cba}$

A operação de concatenação é associativa e tem como elemento neutro a string vazia.

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$
$$\varepsilon \cdot x = x = x \cdot \varepsilon$$

A operação de exponenciação descreve uma concatenação repetida. Podemos implementar esta repetição com uma definição recursiva.

$$w^0 = \varepsilon$$
$$w^{n+1} = w \cdot w^n$$

Uma vantagem da definição recursiva, comparada a uma definição com "produtório" ou "três pontinhos" é que ela facilita provas por indução.

Teorema 1.1.1: A concatenação de duas exponenciações obedece

$$w^n \cdot w^m = w^{n+m}$$

Prova: A prova é por indução em n. Precisamos mostrar que a equação vale para $n = 0$ e também que, se ela vale para n, então vale para $n + 1$.

Caso base: queremos provar $w^0 \cdot w^m = w^{0+m}$.

$$\begin{aligned}
& w^0 \cdot w^m \\
= & \quad \{ \text{ definição da exponenciação } \} \\
& \varepsilon \cdot w^m \\
= & \quad \{ \ \varepsilon \text{ é elemento neutro da concatenação } \} \\
& w^m \\
= & \quad \{ \text{ aritmética } \} \\
& w^{0+m}
\end{aligned}$$

Caso indutivo: Assumimos $w^n \cdot w^m = w^{n+m}$ e queremos $w^{n+1} \cdot w^m = w^{n+m+1}$.

$$\begin{aligned}
& w^{n+1} \cdot w^m \\
= & \quad \{ \text{ definição da exponenciação } \} \\
& (w \cdot w^n) \cdot w^m \\
= & \quad \{ \text{ concatenação é associativa } \} \\
& w \cdot (w^n \cdot w^m) \\
= & \quad \{ \text{ hipótese de indução } \} \\
& w \cdot (w^{n+m}) \\
= & \quad \{ \text{ definição da exponenciação } \} \\
& w^{n+m+1}
\end{aligned}$$

□

1.2. Linguagens

Uma **linguagem** é conjunto de strings. Pudemos descrever o comportamento de um programa através do conjunto de entradas em que o programa responde "sim".

Linguagens podem ser finitas ou infinitas. As strings são sempre finitas, mas a linguagem pode conter infinitas strings. Por exemplo, $\{\varepsilon, \text{a}, \text{aa}, \text{aaa}, ...\}$.

Para representar linguagens, vamos usar operações que criam linguagens mais complexas a partir de linguagens mais simples. Algumas destas operações são operações comuns de conjuntos e outras são extensões das operações sobre strings.

União	$A \cup B = \{w \mid w \in A \vee w \in B\}$
Inserseção	$A \cap B = \{w \mid w \in A \wedge w \in B\}$
Concatenação	$A \cdot B = \{x \cdot y \mid x \in A \wedge y \in B\}$
Exponenciação	$A^0 = \{\varepsilon\}$ $A^{n+1} = A \cdot A^n$
Estrela de Kleene	$A^* = \bigcup_{i=0}^{\infty} A^i = \{\varepsilon\} \cup A \cup A^2 \cup ...$
Complemento	$\overline{A} = \{w \mid w \in \Sigma^* \wedge w \notin A\}$

1.3. Expressões regulares

Escrever $\{\varepsilon\}$ e $\{a\}$ fica repetitivo bem rápido. É comum abreviarmos para ε e a quando estiver claro pelo contexto que estamos falando de conjuntos.

Como veremos mais à frente, as linguagens produzidas pelos autômatos podem ser descritas usando apena concatenação, união, e estrela. Estas são as **linguagens regulares**.

1. A linguagem vazia $\emptyset$ é regular.
2. Para qualquer palavra w, a linguagem $\{w\}$ é regular.
3. Se A e B são regulares, $A \cdot B$ é regular.
4. Se A e B são regulares, $A \cup B$ é regular.
5. Se A é regular, A^* é regular.
6. Nenhuma outra linguagem é regular.

Exercício: toda linguagem finita é regular.

Propriedades importantes das linguagens regulares:

A concatenação é associativa e tem a linguagem ε como elemento neutro. A concatenação com o conjunto vazio resulta no conjunto vazio.

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$
$$\varepsilon \cdot A = A = A \cdot \varepsilon$$
$$\emptyset \cdot A = \emptyset = A \cdot \emptyset$$

A união é associativa e comutativa e tem o conjunto vazio como elemento neutro.

$$A \cup (B \cup C) = (A \cup B) \cup C$$
$$A \cup B = B \cup A$$
$$\emptyset \cup A = A = A \cup \emptyset$$

A união é idempotente

$$A \cup A = A$$

Distributividade:

$$A \cdot (B \cup C) = A \cdot B \cup A \cdot C$$
$$(A \cup B) \cdot C = A \cdot C \cup B \cdot C$$

As operações de concatenação e união são monótonas

$$X \subseteq Y \Longrightarrow A \cdot X \subseteq A \cdot Y$$
$$X \subseteq Y \Longrightarrow A \cup X \subseteq A \cup Y$$

A concatenação se comporta como uma multiplicação e a união se comporta como a soma. O conjunto vazio é similar ao 0 e o conjunto da string de comprimento zero é similar a 1. No entanto, ao contrário dos números convencionais, a multiplicação não é comutativa e não existem operações inversas (subtração e divisão).

A operação A^* descreve a menor solução da inequação $X \supseteq AX \cup \varepsilon$. Isto é:

- $A^* \supseteq AA^* \cup \varepsilon$
- $X \geq AX \cup \varepsilon \Longrightarrow X \supseteq A^*$

Inclusive podemos ir além e dizer que $A^* = AA^* \cup \varepsilon$.

$$\begin{aligned}
& A^* \supseteq AA^* \cup \varepsilon \\
\Longrightarrow \quad & \{ \text{ monotonicidade } \} \\
& AA^* \supseteq A \cdot (AA^* \cup \varepsilon) \\
\Longrightarrow \quad & \{ \text{ monotonicidade } \} \\
& AA^* \cup \varepsilon \supseteq A \cdot (AA^* \cup \varepsilon) \cup \varepsilon \\
\Longrightarrow \quad & \{ \ X \geq AX \cup \varepsilon \Longrightarrow X \supseteq A^* \ \} \\
& AA^* \cup \varepsilon \supseteq A^*
\end{aligned}$$

1.4. Autômatos

Um autômato finito pode ser descrito por uma tupla $\mathcal{A} = (\Sigma, Q, S, F, \Delta)$:

Σ o alfabeto.
Q o conjunto de estados.
S o conjunto de estados iniciais.
F o conjunto de estados finais.
Δ a relação de transição (arestas)

O alfabeto Σ descreve quais caracteres podem aparecer nas arestas do autômato.

Como é de se esperar, em um autômato finito o conjunto de estados Q deve ser finito. Os subconjuntos S e F descrevem os pontos de início e de fim dos caminho.

Os caminhos que percorremos no autômato começam em um estado de S e terminam em algum estado de F.

Uma aresta é uma tripla $(Q \times \Sigma^* \times Q)$, com o estado de origem, o rótulo, e o estado de destino. O conjunto Δ descreve uma relação de transição entre estados. Pense em uma tabela de um banco de dados relacional, em que as colunas são o estado de origem, o rótulo da aresta, e o estado de destino.

Um **Autômato Finito Determinístico (AFD)** é um autômato que podemos testar se uma palavra é reconhecida sem nunca ter que fazer alguma escolha. Ele tem um único estado inicial, e arestas que saem do mesmo estado não compartilham nenhum prefixo. Em particular, não podem começar com a mesma letra, e também não existem arestas com rótulo vazio. Podemos enxergar que a relação de transição tem uma dependência funcional: dado o estado atual e a próxima letra da entrada, há no máximo um estado de destino possível.

No caso geral, temos um **Autômato Finito Não-Determinístico (AFND)**, que não tem estas restrições. Eles podem ter mais de um estado inicial e o conjunto de arestas pode ser uma relação qualquer. São permitidas arestas com rótulos vazios. No entanto, para testar se uma palavra é reconhecida, é preciso testar vários caminhos em paralelo.

VARIAÇÕES: Outros livros podem apresentar autômatos finitos de uma forma diferente da que eu apresentei. Todos exigem que o conjunto de estados seja finito, mas os outros detalhes podem mudar. Porém em geral é possível adaptar um autômato de um formato para o outro, de forma que os formalismos são igualmente poderosos. A seguir eu listo algumas variações comuns. Você consegue pensar em como adaptar os nossos autômatos para esses formatos?

- O autômato deve ter apenas um estado inicial
- O autômato deve ter apenas um estado final
- Os rótulos não podem ser palavras inteiras. Só um símbolo do alfabeto, ou ε.
- Em vez da relação de transição (conjunto de arestas), há uma função que recebe estado atual e símbolo, e retorna um conjunto de estados de destino possíveis.

Exercício: Sabemos que todo autômato determinístico deve ter um único estado inicial. Podemos também exigir que tenha um único estado final? Ou será que existe alguma linguagem que precisa de pelo menos dois estados finais?

2. Caminhos

Um autômato reconhece uma palavra w se existe um caminho rotulado por w, que leva de um estado inicial para um final. Para formalizar estes conceitos, podemos definir uma estrutura de dados para os caminhos, assim como uma função que calcula o rótulo do caminho.

$$\frac{X \in Q}{X! : X \rightsquigarrow X} \qquad\qquad \frac{XaY \in \Delta \quad p : Y \rightsquigarrow Z}{X \xrightarrow{a} p : X \rightsquigarrow Z}$$

Existem duas formas de construir um caminho sobre um dado autômato:

1. (vazio) Se X é um estado, então $X!$ é um caminho que vai de X até X.
2. (passo) Se XaY é uma aresta de X para Y, e p é um caminho que vai de Y até Z, então $X \xrightarrow{a} p$ é um caminho que vai de X até Z.
3. e nada mais é um caminho.

Alguns exemplos de caminho:

- $E \xrightarrow{a} A \xrightarrow{a} X!$
- $E \xrightarrow{b} B \xrightarrow{b} X!$
- $E \xrightarrow{a} A \xrightarrow{b} E \xrightarrow{a} A \xrightarrow{a} X!$

Podemos escrever uma função que recupera a palavra percorrida pelo caminho:

$$\begin{aligned} \mathrm{str}(X!) &= \varepsilon \\ \mathrm{str}\left(X \xrightarrow{a} p\right) &= a \cdot \mathrm{str}(p) \end{aligned}$$

Agora estamos prontos para especificar a linguagem reconhecida por um autômato $\mathcal{A} = (\Sigma, Q, S, F, \delta)$. Dizemos que ele reconhece a palavra w se existe um caminho p que leva de um estado inicial para um final, lendo a palavra w. Isto é:

- $p : X \rightsquigarrow Z$
- $X \in S$
- $Z \in F$
- $\mathrm{str}(p) = w$

3. Relação de Transição

Caminhos representam diretamente a sequência de estados visitados. No entanto, a função $\mathrm{str}()$ é inconveniente na hora de escrever provas.

Uma outra maneira de especificar o comportamento do autômato é modelar uma máquina que testa se a palavra é reconhecida pelo autômato. Esta máquina mantém duas variáveis: o estado atual, e a string do que falta ler. A relação de transição $\vdash$ descreve os passos que a máquina executa. Quando estamos no estado X e o próximo trecho da entrada é a, então se existir uma aresta XaY nós podemos mudar para o estado Y e consumir o trecho a.

$$\frac{XaY \in \Delta}{(X, aw) \vdash (Y, w)}$$

Uma derivação completa consiste de uma sequência destes passos. Usamos a relação $(X, x) \vdash^* (Z, z)$ para dizer que existe uma sequência de zero ou mais transições que levam de (X, x) até (Z, z). No jargão técnico, $\vdash^*$ é o fecho reflexivo e transitivo de $\vdash$.

$$\frac{}{(X, w) \vdash^* (X, w)} \qquad \frac{(X, x) \vdash (Y, y) \quad (Y, y) \vdash^* (Z, z)}{(X, x) \vdash^* (Z, z)}$$

Exemplo de uma derivação completa:

$$(X, \mathrm{ababa}) \vdash (Y, \mathrm{baba}) \vdash (X, \mathrm{aba}) \vdash (Y, \mathrm{ba}) \vdash (X, \mathrm{a}) \vdash (Y, \varepsilon)$$

A linguagem das palavras aceitas a partir de um estado X são as palavras que levam de X até um estado final Z:

$$L(X) = \{w \mid (X, w) \vdash^* (Z, \varepsilon) \wedge Z \in F\}$$

O autômato aceita todas as palavras aceitas por algum dos estados iniciais.

$$L(\mathcal{A}) = \bigcup_{X \in S} L(X) = \{w \mid (X, w) \vdash^* (Z, \varepsilon) \wedge X \in S \wedge Z \in F\}$$

Derivações são tão poderosas quanto caminhos. É possível escrever um algoritmo que converte de uma para a outra.

$$\mathrm{p2d}\left(\frac{X \in Q}{X! : X \rightsquigarrow X}, w\right) = (X, w) \vdash^* (X, w)$$

$$\mathrm{p2d}\left(\frac{XaY \in \Delta \quad p{:}\, Y \rightsquigarrow Z}{X \xrightarrow{a} p : X \rightsquigarrow Z}, w\right) = \text{let } (Y, y) \vdash^* (Z, z) \coloneqq \mathrm{p2d}(p, w) \text{ in}$$
$$(X, ay) \vdash (Y, y) \vdash^* (Z, z)$$

Exercício: Prove que podemos adicionar e remover caracteres do final. Para todo w,

$$(X, x) \vdash (X, y) \iff (X, xw) \vdash (X, yw)$$

Exercício: Extenda a prova para $\vdash^*$. Para todo w,

$$(X, x) \vdash^* (X, y) \iff (X, xw) \vdash^* (X, yw)$$

4. Derivações de gramática

Esta semântica é comumente usada para linguagens livres de contexto, mas é raramente usada para linguagens regulares.

$$\frac{X \in F}{X \Rightarrow \varepsilon} \qquad \frac{XaY \in \Delta}{X \Rightarrow aY} \qquad \frac{w \Rightarrow w'}{uwv \Rightarrow uw'v}$$

Exemplo:

$$\mathrm{E} \Rightarrow \mathrm{aA} \Rightarrow \mathrm{abE} \Rightarrow \mathrm{abbB} \Rightarrow \mathrm{abbbX} \Rightarrow \mathrm{abbb}$$

É claro, também precisamos definir o fecho reflexivo e transitivo de $\Rightarrow$.

$$\frac{}{X \Rightarrow^* X} \qquad \frac{X \Rightarrow Y \quad Y \Rightarrow^* Z}{X \Rightarrow^* Z}$$

5. Semântica operacional big-step

Em breve vamos escrever várias provas que discorrem sobre $L(X)$ e é repetitivo ter que escrever toda hora aquele $(Z, \varepsilon) \wedge Z \in F$. Por isso inventei uma notação nova que abrevia isso. A relação $X \Downarrow w$ codifica que existe um caminho que leva de X para algum estado final, lendo w.

$$\frac{X \in F}{X \Downarrow \varepsilon} \qquad\qquad \frac{XaY \in \Delta \quad Y \Downarrow v}{X \Downarrow av}$$

Por extenso:

1. Se X é um estado final, então ele reconhece a palavra vazia.
2. Se existe uma aresta XaY e Y reconhece v, então X reconhece av.
3. Estados só reconhecem palavras que se encaixam nas regras acima.

Exercício: prove que a definição com $\Downarrow$ equivale à sua especificação com $\vdash^*$:

$$X \Downarrow w \iff \exists Z\colon (X, w) \vdash^* (Z, \varepsilon) \wedge Z \in F$$

6. Semântica por conjuntos

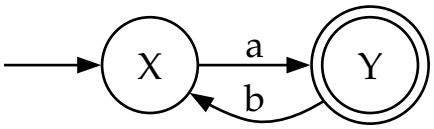
Semânticas operacionais exigem que nós rodemos um programa de computador para saber se uma palavra é aceita. Os detalhes dependem da implementação deste programa de computador. Até agora já vimos mais de uma versão: a semântica com $\vdash$, a com $\Rightarrow$, e a com $\Downarrow$.

Uma outra maneira de pensar sobre a semântica do autômato é especificar quais propriedades nós gostaríamos que fossem verdade sobre as linguagens reconhecidas por cada estado. A especificação da relação $\Downarrow$ parece ser um bom começo:

Definição 6.1 (*Especificação da linguagem do autômato*):
a) Se X é um estado final, então ele deve reconhecer a palavra vazia
b) Se existe uma aresta XaY e Y reconhece v, então X deve reconhecer av.
c) Estados só reconhecem palavras que se encaixam nas regras acima.

Buscamos atribuir uma linguagem à cada estado, obedecendo estas regras. As restrições (a) e (b) dizem quando a solução proposta é viável. Já a regra (c) diz que buscamos a solução mais simples, na qual as linguagens contém apenas as palavras essenciais para que a solução seja viável.

6.1. Um exemplo concreto



Vamos escrever as restrições para o autômato acima. Buscamos uma função $L\colon Q \to \Sigma^*$ que atribui uma linguagem para cada estado. As regras (a) e (b) nos dizem que valores de $L(X)$ e $L(Y)$ são viáveis:

1. $\varepsilon \in L(Y)$
2. $v \in L(Y) \Longrightarrow \mathrm{a} \cdot v \in L(X)$
3. $v \in L(X) \Longrightarrow \mathrm{b} \cdot v \in L(Y)$

Devemos tomar cuidado para colocar o X e Y no lugar certo. Vamos ler com calma o que cada uma dessas fórmulas diz.

1. A linguagem de Y deve conter a palavra vazia.
2. Para mostrar que uma palavra começando em "a" pertence à linguagem de X, é suficiente mostrar que o resto desta palavra pertence à linguagem de Y.
3. Para mostrar que uma palavra começando em "b" pertence à linguagem de Y, é suficiente mostrar que o resto desta palavra pertence à linguagem de X.

Uma solução trivialmente viável é qualquer estado aceitar qualquer palavra.

$$L(X) = \Sigma^*$$
$$L(Y) = \Sigma^*$$

Uma solução mais esperta é

$$L(X) = \mathrm{a}(\mathrm{ba}^*)$$
$$L(Y) = (\mathrm{ba}^*)$$

Vamos conferir que esta solução é viável

1. $\varepsilon \in (\mathrm{ba})^*$
2. $v \in (\mathrm{ba})^* \Longrightarrow \mathrm{a} \cdot v \in \mathrm{a}(\mathrm{ba})^*$
3. $v \in \mathrm{a}(\mathrm{ba})^* \Longrightarrow \mathrm{b} \cdot v \in (\mathrm{ba})^*$

Esta solução esperta também é mínima! Podemos mostrar que qualquer solução viável contém esta solução como sub-solução. Isto é, se L é viável então

a) $\mathrm{a}(\mathrm{ba})^* \subseteq L(X)$
b) $(\mathrm{ba})^* \subseteq L(Y)$

Nossa prova será por indução no comprimento das palavras de L. Para tal, é melhor escrever o enunciado em termos de "$w \in$". Isto é, para toda palavra w queremos provar

a) $w \in \mathrm{a}(\mathrm{ba})^* \Longrightarrow w \in L(X)$
b) $w \in (\mathrm{ba})^* \Longrightarrow w \in L(Y)$

A indução tem que tratar três casos:

- $w = \varepsilon$. A implicação (a) vale vacuosamente pois $\varepsilon \notin \mathrm{a}(\mathrm{ba})^*$. A implicação (b) vale pois a hipótese (1) garante que $\varepsilon \in L(Y)$.
- $w = \mathrm{a}v$. Para a implicação (a) queremos mostrar que $\mathrm{a}v \in \mathrm{a}(\mathrm{ba})^* \Longrightarrow \mathrm{a}v \in L(X)$. Quando $\mathrm{a}v \in \mathrm{a}(\mathrm{ba})^*$, necessariamente $v \in (\mathrm{ba})^*$. Como v é mais curta que w, podemos aplicar a hipótese de indução para obter $v \in L(Y)$. Pela regra (2) da especificação de factível, concluímos $\mathrm{a}v \in L(X)$.

  A implicação (b), $\mathrm{a}v \in (\mathrm{ba}^*) \Longrightarrow \mathrm{a}v \in L(Y)$, vale vacuosamente pois palavras que começam com "a" nunca casam com $(\mathrm{ba})^*$.
- $w = \mathrm{b}v$. A prova é análoga à do caso anterior. A implicação (a) vale vacuosamente e a implicação (b) segue da hipótese de indução junto com a regra (3).

7. Sistemas de inequações

Uma vantagem de especificar a semântica do autômato através de conjuntos é podemos manipulá-los usando equivalências entre expressões regulares.

Veremos que essa estratégia será útil para demonstrar que dois autômatos são equivalentes. A ideia é converter o autômato A_1 para um sistema de restrições C_1 e mostrar que estas restrições equivalem ao sistema C_2, que por sua vez é o sistema do autômato A_2.

```
A1    A2
|     |
C1 -- C2
```

7.1. Notação de subconjuntos

Já que vamos trabalhar bastante com estes sistemas de restrições sobre as linguagens, vale a pena parar para deixar a notação mais leve. Para fins ilustrativos usarei novamente o exemplo do Autômato [?]. Começamos com

1. $\varepsilon \in L(Y)$
2. $v \in L(Y) \implies \mathrm{a} \cdot v \in L(X)$
3. $v \in L(X) \implies \mathrm{b} \cdot v \in L(Y)$

Primeiramente, vou omitir as chamadas $L()$. Nem o Germán Cano aguenta tanto L. Daqui pra frente, assuma que se eu escrever um nome de estado em um contexto que espera um conjunto/linguagem, na verdade se trata de um $L(X)$.

1. $\varepsilon \in Y$
2. $v \in Y \implies \mathrm{a} \cdot v \in X$
3. $v \in X \implies \mathrm{b} \cdot v \in Y$

Em seguida, podemos trocar as implicações por subconjuntos. Preste atenção que o a e o b trocaram de lado, mas o significado é mesmo de antes.

1. $\{\varepsilon\} \subseteq Y$
2. $\mathrm{a} \cdot Y \subseteq X$
3. $\mathrm{b} \cdot X \subseteq Y$

Finalmente, podemos usar união para juntar todas as inequações do Y em uma regra só. Também passei a escrever a expressão regular ε no lugar do conjunto $\{\varepsilon\}$.

$$X \supseteq \mathrm{a}Y$$
$$Y \supseteq \mathrm{b}X \cup \varepsilon$$

Resumindo, temos uma inequação para cada variável, com um termo para cada aresta do autômato, e mais um termo para cada estado final. No caso geral fica

$$L(X) \supseteq \{\varepsilon \mid X \in F\} \cup \bigcup_{XaY \in \Delta} a \cdot L(Y)$$

7.2. Pontos fixos

Quando temos uma equação por variável, nosso sistema se comporta como uma equação $\vec{X} \supseteq f\left(\vec{X}\right)$, onde $\vec{X}$ é um vetor de linguagens. Podemos nos aproveitar da rica teoria de pontos fixos.

Definição 7.2.1 (*Função monótona*): Dizemos que uma função f é monótona quando

$$A \subseteq B \implies f(A) \subseteq f(B)$$

Dá para extender esta definição para vetores de linguagens. Se A e B forem uma lista de linguages (uma para cada estado) nós comparamos componente a componente.

Temos especial interesse nas soluções dos sistemas de equações / inequações.

Definição 7.2.2 (*Ponto prefixo*): $f(x) \subseteq x$

Definição 7.2.3 (*Ponto pósfixo*): $x \subseteq f(x)$

Definição 7.2.4 (*Ponto fixo*): $f(x) = x$

Estamos particularmente interessados no menor dos pontos pré-fixos

Definição 7.2.5 (*Menor ponto prefixo*): Dizemos que μf é o menor ponto prefixo de f se ele é um ponto prefixo que também é menor ou igual a todos os outros.

- $f(\mu f) \subseteq \mu f$
- $\forall x\colon f(x) \subseteq x \implies \mu f \subseteq x$

Lema 7.2.1: O menor ponto prefixo também é um ponto fixo. Portanto, μf é tanto o menor ponto prefixo quanto o menor ponto fixo.

Prova: Já sabemos $f(\mu f) \subseteq \mu f$. Resta mostrar $\mu f \subseteq f(\mu f)$.

$$\begin{aligned}
& f(\mu f) \subseteq \mu f \\
\implies & \quad \{\ f \text{ é monótona } \} \\
& f(f(\mu f)) \subseteq f(\mu f) \\
\implies & \quad \{\ f(\mu f) \text{ é um ponto prefixo, logo } \mu f \text{ é menor } \} \\
& \mu f \subseteq f(\mu f)
\end{aligned}$$

□

Na prática, quando queremos mostrar que algo é um ponto fixo, basta provar que é prefixo, o que dá menos trabalho. Por outro lado, se já soubermos mos que algo é um ponto prefixo, podemos assumir de vez a hipótese mais forte de que é um ponto fixo.

Teorema 7.2.1 (*Bekić*): Um sistema de equações com mais de variável pode ser resolvido uma variável de cada vez.

$$\mu\begin{pmatrix} x \mapsto f(x,y) \\ y \mapsto g(x,y) \end{pmatrix} = \begin{pmatrix} \mu(x \mapsto f(x, \mu(y \mapsto g(x,y)))) \\ \mu(y \mapsto g(\mu(x \mapsto f(x,y)), y)) \end{pmatrix}$$

$$\mu\begin{pmatrix} x \supseteq f(x,y) \\ y \supseteq g(x,y) \end{pmatrix} = \begin{pmatrix} \mu\langle x \supseteq f(x, \mu\langle y \supseteq g(x,y)\rangle)\rangle \\ \mu\langle y \supseteq g(\mu\langle x \supseteq f(x,y)\rangle, y)\rangle \end{pmatrix}$$

$$\mu\begin{pmatrix} x \mapsto f(x,y) \\ y \mapsto g(x,y) \end{pmatrix} = \begin{pmatrix} \mu(x \mapsto f(x, \mu(y \mapsto g(x,y)))) \\ \mu(y \mapsto g(\mu(x \mapsto f(x,y)), y)) \end{pmatrix}$$

Prova: Winskel (1993) capítulo 10. □

Corolário 7.2.1 (*Introdução de equações*): Introduzir uma nova equação não altera o resultado das outras equações.

Corolário 7.2.2 (*Remoção de inequações*): Remover uma equação não altera o resultado das outras equações, caso elas não usem a variável removida.

Lema 7.2.2 (*Substituição de equação*): Substituir uma equação na outra não altera a menor solução do sistema.

$$\mu\begin{pmatrix} x = f(x,y) \\ y = g(x,y) \end{pmatrix} = \mu\begin{pmatrix} x = f(x, g(x,y)) \\ y = g(x,y) \end{pmatrix}$$

Prova: Toda solução do sistema de equações da esquerda é solução do sistema de equações da direita e vice versa. Portanto, o conjunto das soluções é o mesmo e a menor solução também deve ser a mesma.

□

Lema 7.2.3 (*Substituição de inequação*): Substituir uma inequação na outra não altera a menor solução do sistema.

$$\mu\begin{pmatrix} x \supseteq f(x,y) \\ y \supseteq g(x,y) \end{pmatrix} = \mu\begin{pmatrix} x \supseteq f(x, g(x,y)) \\ y \supseteq g(x,y) \end{pmatrix}$$

Prova: Pelo Lema 7.2.1, a menor solução de um sistema de equações também é a menor solução do sistema de inequações. Consequentemente, a substituição também vale o sistema de inequações. □

8. Usando os sistemas

minimize X

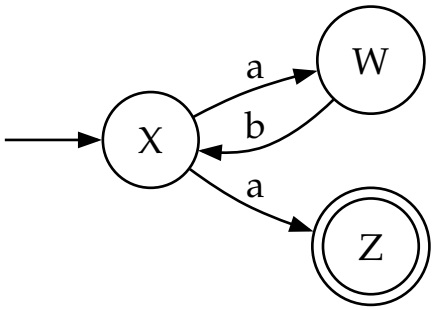
$X \supseteq aW \cup aZ$
$W \supseteq bX$
$Z \supseteq \varepsilon$

$X \supseteq aW \cup aZ$
$W \supseteq bX$
$Z \supseteq \varepsilon$

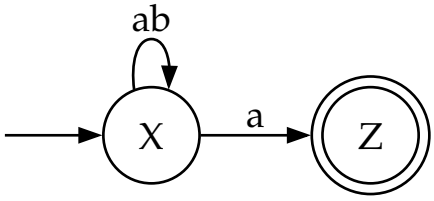
)

$X \supseteq aW \cup aZ$
$W \supseteq bX$
$Z \supseteq \varepsilon$

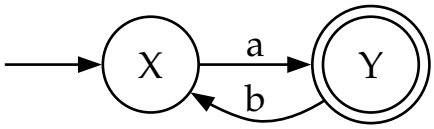
$X \supseteq aW \cup aZ$
$W \supseteq bX$
$Z \supseteq \varepsilon$



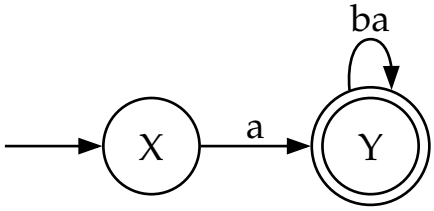
$X \supseteq abX \cup a$



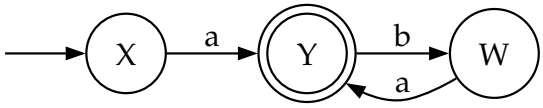
$X \supseteq aY$
$Y \supseteq bX \cup \varepsilon$



$X \supseteq aY$
$Y \supseteq baY \cup \varepsilon$

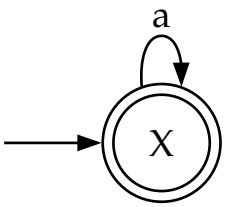


$X \supseteq aY$
$Y \supseteq bZ \cup \varepsilon$
$W \supseteq aY$



9. Semântica Denotacional

Vou fazer as provas para este autômato específico.



Queremos encontrar a menor solução para o sistema $[\![X]\!] = \{\varepsilon\} \cup a \cdot [\![X]\!]$.

Mais formalmente, o menor ponto fixo do operador $f(X) = \{\varepsilon\} \cup a \cdot X$

9.1. Prova de que é ponto fixo

Para provar que a semântica operacional casa com a denotacional, o primeiro passo é mostrar que a linguagem descrita pela semântica operacional é uma solução do sistema de equações pedido pela semântica denotacional.

Teorema 9.1.1: $f(\{w \mid X \Downarrow w\}) = \{w \mid X \Downarrow w\}$

Prova: Um caminho do estado X até um estado final tem duas formas possíveis. Ou X é o estado final do caminho $(X \Downarrow \varepsilon)$, ou visitamos uma aresta e em seguida vamos para o estado final: $X \Downarrow av$, com $X \xrightarrow{a} X$ e $X \Downarrow v$. □

$$
\begin{aligned}
&\{w \mid X \Downarrow w\} \\
&= (\text{dois casos possíveis}) \\
&\{\varepsilon\} \cup \{av \mid X \Downarrow v\} \\
&= (\text{definição de concatenação}) \\
&\{\varepsilon\} \cup a \cdot \{v \mid X \Downarrow v\} \\
&= (\text{renomear}) \\
&\{\varepsilon\} \cup a \cdot \{w \mid X \Downarrow w\}
\end{aligned}
$$

9.2. Prova de que é o menor ponto fixo

Só falta mostrar que a semântica operacional descreve a menor solução possível.

Teorema 9.2.1: Se R é um ponto fixo de F, então $\{w \mid X \Downarrow w\} \subseteq R$

Prova: A intuição por trás da prova é que podemos expendir R usando a propriedade de ser ponto fixo e observar que as palavras de $\{w \mid X \Downarrow w\}$ surgem uma a uma, e portanto pertencem ao conjunto R.

$$\begin{aligned} R &= \varepsilon \cup aR \\ &= \varepsilon \cup a(\varepsilon \cup aR) \\ &= \varepsilon \cup a \cup a^2(\varepsilon \cup aR) \\ &= \varepsilon \cup a \cup a^2 \cup a^3(\varepsilon \cup aR) \\ &= ... \end{aligned}$$

A prova completa é por indução estrutural sobre a derivação $X \Downarrow w$. No caso base temos $X \Downarrow \varepsilon$ e queremos provar $\varepsilon \in R$.

$$\begin{aligned} &\varepsilon \in R \\ &\Leftrightarrow (R \text{ é ponto fixo}) \\ &\varepsilon \in (\{\varepsilon\} \cup a \cdot R) \\ &\Leftarrow \\ &e \in \{\varepsilon\} \end{aligned}$$

No caso indutivo temos $X \Downarrow av$, oriundo de $X \xrightarrow{a} X$ e $X \Downarrow v$, e queremos provar provar que $av \in R$. Podemos assumir, pela hipótese de indução, que $v \in R$.

$$\begin{aligned} &av \in R \\ &\Leftrightarrow (\text{R é ponto fixo}) \\ &av \in (\{\varepsilon\} \cup a \cdot R) \\ &\Leftarrow \\ &av \in a \cdot R \\ &\Leftarrow \\ &v \in R \end{aligned}$$

□

10. Semântica Denotacional (geral)

Agora voou fazer a prova para o caso geral.

Queremos mostrar que nossa semântica denotacional é compatível com a operacional. No caso geral, a semântica denotacional descreve um sistema de equações, com uma linguagem por estado. Buscamos a menor solução do sistema.

$$[\![X]\!] = \varphi(X) \cup \bigcup_{XaY \in \Delta} a \cdot [\![Y]\!]$$

Nosso objetivo é mostar que esta menor solução é a linguagem das palavras aceitas pelo autômato operacionalmente: $[\![X]\!] = L(X)$.

10.1. Prova de que é o menor ponto fixo (big step)

Lema 10.1.1: As linguagens da semântica operacional formam uma solução do sistema de equações da semântica denotacional. Isto é, para todo estado X,

$$L(X) = \varphi(X) \cup \bigcup_{XaY \in \Delta} a \cdot L(Y).$$

Prova: O principal truque é que $X \Downarrow w$ tem dois casos possíveis: $X \Downarrow \varepsilon$ e $X \Downarrow av$.

No primeiro, a unificação nos diz que X deve ser estado final e $w = \varepsilon$. No segundo, temos que w precisa começar com alguma letra a e deve existir uma aresta XaY e a linguagem de Y deve aceitar o resto da palavra..

$$\begin{aligned}
& L(X) \\
= & \text{ (por definição)} \\
& \{w \mid X \Downarrow w\} \\
= & \text{ (quebrar em casos)} \\
& \{\varepsilon \mid X \in F\} \cup \{av \mid XaY \in \Delta \wedge Y \Downarrow v\} \\
= & \text{ (mover o } \forall XaY \text{ para fora da expressão)} \\
& \{\varepsilon \mid X \in F\} \cup \bigcup_{XaY \in \Delta} a \cdot \{v \mid Y \Downarrow v\} \\
= & \text{ (definições de } \varphi(X) \text{ e de } L(Y)) \\
& \varphi(X) \cup \bigcup_{XaY \in \Delta} a \cdot L(Y)
\end{aligned}$$

□

Lema 10.1.2: As linguagens da semântica operacional são um lower bound do sistema de equações da semântica denotacional.

$$\mathcal{F}(R) \leq R \Longrightarrow L \leq R$$

Prova: Primeiro, devemos massagear o enunciado. Nossa hípótese é:

$$H\colon \left(\varphi(X) \cup \bigcup_{XaY \in \Delta} a \cdot R(Y)\right) \subseteq R(X)$$

em outras palavras:

$$H_1\colon \forall X\colon \varphi(X) \subseteq R(X)$$
$$H_2\colon \forall XaY\colon (XaY \in \Delta) \Longrightarrow aR(Y) \subseteq R(X)$$

Assumindo H_1 e H_2, queremos mostrar que, para todo X e w,

$$X \Downarrow w \Longrightarrow w \in R(Y)$$

A prova é por indução em $X \Downarrow w$.

No caso base, temos $w = \varepsilon$ e $X \in F$. Portanto, por H_1, $w \in \varphi(X) \subseteq R(X)$

No caso indutivo, temos $w = av$ e existe uma aresta XaY tal que $Y \Downarrow v$. Pela hipótese de indução, $v \in R(Y)$. Portanto, por H_2, $w = av \in aR(Y) \subseteq R(X)$. □

Prova: Primeiro, devemos massagear o enunciado

Se para todo X, $\left(\varphi(X) \cup \bigcup_{XaY \in \Delta} a \cdot R(Y)\right) \subseteq R(X)$,
então para todo X, $L(X) \subseteq L(X)$.

Se para todos X, w,

$$(w \in \varphi(X) \vee \exists XaY \in \Delta\colon w \in a \cdot R(Y)) \Longrightarrow w \in R(X)$$

então para todos X, Z, w,

$$((X, w) \vdash^* (Z, \varepsilon) \wedge Z \in F) \Longrightarrow w \in R(X)$$

A prova é por indução na evidência $(X, w) \vdash^* (Z, \varepsilon)$

No caso base, temos $X = Z$ e $w = \varepsilon$. Como Z é final, X também é. Portanto, $w \in \varphi(X)$ e a hipótese nos permite concluir $w \in R(X)$.

No caso indutivo, temos $(X, w) \vdash^* (Z, \varepsilon) = (X, av) \vdash (Y, v) \vdash^* (Z, \varepsilon)$. Portanto, existe uma aresta XaY que leva de X a Y, e sabemos que w começa com a ($w = av$). Aplicando a hipótese de indução em $(Y, v) \vdash^* (Z, \varepsilon)$, deduzimos que $v \in R(Y)$. Logo, $w = av \in a \cdot R(Y)$. Com isso, podemos aplicar a hipótese H e obter $w \in R(X)$. □

10.2. Prova de que é o menor ponto fixo (derivações)

Queremos mostrar que nossa semântica denotacional é compatível com a operacional. Isto é, o menor ponto fixo do sistema de equações é a linguagem das palavras reconhecidas pelo autômato.

O sistema:

$$[\![X]\!] = \varphi(X) \cup \bigcup_{XaY \in \Delta} a \cdot [\![Y]\!]$$

Lema 10.2.1: As linguagens da semântica operacional formam uma solução do sistema de equações da semântica denotacional. Isto é, para todo estado X,

$$L(X) = \varphi(X) \cup \bigcup_{XaY \in \Delta} a \cdot L(Y).$$

Prova: O principal truque é quebrar a derivação $(X, w) \vdash^* (Z, \varepsilon)$ em dois casos:
1. $(X, w) \vdash^* (Y, \varepsilon) = (Z, \varepsilon) \vdash (Z, \varepsilon)$
2. $(X, w) \vdash^* (Y, \varepsilon) = (Z, av) \vdash (Y, v) \vdash^* (Z, \varepsilon)$

No primeiro, a unificação nos diz que $X = Z$ e $w = \varepsilon$. No segundo, temos que w precisa começar com alguma letra a e deve existir uma aresta XaY.

$$\begin{aligned}
& L(X) \\
= \ & \text{(definição da semântica operacional)} \\
& \{w \mid (X, w) \vdash^* (Z, \varepsilon) \wedge Z \in F\} \\
= \ & \text{(quebrar em casos)} \\
& \{\varepsilon \mid (X, \varepsilon) \vdash^* (X, \varepsilon) \wedge x \in F\} \cup \\
& \{av \mid (X, av) \vdash (Y, v) \vdash^* (Z, \varepsilon) \wedge XaY \in \Delta \wedge Z \in F\} \\
= \ & \text{(simplificar)} \\
& \{\varepsilon \mid X \in F\} \cup \{av \mid (Y, v) \vdash^* (Z, \varepsilon) \wedge XaY \in \Delta \wedge Z \in F\} \\
= \ & \text{(mover o } \forall XaY \text{ para fora da expressão)} \\
& \{\varepsilon \mid X \in F\} \cup \bigcup_{XaY \in \Delta} a \cdot \{v \mid (Y, v) \vdash^* (Z, \varepsilon) \wedge Z \in F\} \\
= \ & \text{(definições de } \varphi(X) \text{ e de } L(Y)) \\
& \varphi(X) \cup \bigcup_{XaY \in \Delta} a \cdot L(Y)
\end{aligned}$$

□

Lema 10.2.2: As linguagens da semântica operacional são um lower bound do sistema de equações da semântica denotacional.

$$\mathcal{F}(R) \leq R \Longrightarrow L \leq R$$

Prova: Primeiro, devemos massagear o enunciado

Se para todo X, $\left(\varphi(X) \cup \bigcup_{XaY \in \Delta} a \cdot R(Y)\right) \subseteq R(X)$,
então para todo X, $L(X) \subseteq L(X)$.

Se para todos X, w,

$$(w \in \varphi(X) \vee \exists XaY \in \Delta \colon w \in a \cdot R(Y)) \Longrightarrow w \in R(X)$$

então para todos X, Z, w,

$$((X, w) \vdash^* (Z, \varepsilon) \wedge Z \in F) \Longrightarrow w \in R(X)$$

A prova é por indução na evidência $(X, w) \vdash^* (Z, \varepsilon)$

No caso base, temos $X = Z$ e $w = \varepsilon$. Como Z é final, X também é. Portanto, $w \in \varphi(X)$ e a hipótese nos permite concluir $w \in R(X)$.

No caso indutivo, temos $(X, w) \vdash^* (Z, \varepsilon) = (X, av) \vdash (Y, v) \vdash^* (Z, \varepsilon)$. Portanto, existe uma aresta XaY que leva de X a Y, e sabemos que w começa com a ($w = av$). Aplicando a hipótese de indução em $(Y, v) \vdash^* (Z, \varepsilon)$, deduzimos que $v \in R(Y)$. Logo, $w = av \in a \cdot R(Y)$. Com isso, podemos aplicar a hipótese H e obter $w \in R(X)$. □

11. Lema de Arden

Lema 11.1: $X = A^*B$ é solução da equação $X = AX \cup B$.

Prova:

$$\begin{aligned} X &= A^*B \\ &= \left(\bigcup_{i=0}^{\infty} A^i B \right) \\ &= \left(\bigcup_{i=1}^{\infty} A^i B \right) \cup A^0 B \\ &= A \cdot \left(\bigcup_{i=0}^{\infty} A^i B \right) \cup B \\ &= A \cdot (A^*B) \cup B \\ &= AX \cup B \end{aligned}$$

□

Falta mostrar que A^*B é a menor solução. Isto é, qualquer solução de $X = AX \cup B$ contém A^*B. A intuição para isso aparece quando substituímos X por $AX \cup B$ sucessivamente. Repare que na expansão de X os termos de A^*B vão aparecendo um por um: $A^0B, A^1B, A^2B, ...$

$$\begin{aligned} X &= \\ A^1X \cup \underline{B} &= A^1(AX \cup B) \cup B = \\ A^2X \cup \underline{AB \cup B} &= A^2(AX \cup B) \cup AB \cup B = \\ A^3X \cup \underline{A^2B \cup AB \cup B} &= A^3(AX \cup B) \cup A^2B \cup AB \cup B = \\ A^4X \cup \underline{A^3B \cup A^2B \cup AB \cup B} &= ... \end{aligned}$$

Para a prova formal, expandimos a definição de A^*B cxomo uma união infinita e mostramos que todos seus componentes A^nB estão contidos em X.

Lema 11.2: Se $AX \cup B \subseteq X$ então $A^*B \subseteq X$.

Prova: Nosso objetivo é mostrar que $\bigcup_{i=0}^{\infty} A^iB \subseteq X$. Para tal, podemos mostrar que $A^nB \subseteq X$ vale para todo n, por indução em n. Afinal, se o conjunto X contém todos os A^nB, ele deve ser maior ou igual a $\bigcup_{i=0}^{\infty} A^iB$, que por definição é o menor conjunto com esta propriedade.

Caso base: $n = 0$

$A^0B = B \subseteq AX \cup B \subseteq X$

Caso indutivo: $n \geq 1$

Pela hipótese de indução, podemos assumir $A^{n-1}B \subseteq X$. Concatenando A dos dois lados, obtemos $A^nB \subseteq AX$. Portanto, $A^nB \subseteq AX \subseteq AX \cup B \subseteq X$. □

Finalmente, provamos o lema de Arden propriamente dito.

Teorema 11.1 (*Lema de Arden*): A^*B é a menor solução de $X = AX \cup B$.

Prova: O Lema 11.1 nos diz que A^*B é solução e o Lema 11.2 nos diz que A^*B é a menor solução. □

11.1. Por que a menor solução?

Tivemos um bom trabalho para especificar que buscamos a menor solução, e não meramente uma solução qualquer. O que justifica este esforço? Em que casos existem outras soluções para $X = AX \cup B$, além da menor solução A^*B? Um exemplo é a equação

$$X = X \cup \{a\}$$

A menor solução é o conjunto $\{a\}$. No entanto, também são soluções todos os superconjuntos de $\{a\}$: por exemplo, $\{a, b\}$ e Σ^* também são soluções. A culpa disso está no $X = X$ da equação, que corresponde a um loop vazio no autômato.

Como vimos anteriormente, a menor solução surge dos termos A^iB da expansão de X. Para obtermos uma solução diferente de A^*B, é preciso que existam palavras oriundas da parte A^nX da expansão. Isto só é possível quando o conjunto A contém a palavra vazia ε. Nossa prova começa com um lema auxiliar que desenrrola a equação n vezes.

Lema 11.1.1: Se $X \subseteq AX \cup B$, então $\forall n \geq 0.\left(X \subseteq A^{n+1}X \cup \left(\bigcup_{i=0}^{n} A^iB\right)\right)$

Prova: Como sempre, indução em n.

Caso base: $n = 0$

$$\begin{aligned} X &\subseteq AX \cup B \\ &= A^{0+1}X \cup \left(\bigcup_{i=0}^{0} A^iB\right) \end{aligned}$$

Caso indutivo: $n \geq 1$

$$\begin{aligned} X &\subseteq A^{(n-1)+1}X \cup \left(\bigcup_{i=0}^{n-1} A^iB\right) = A^nX \cup \left(\bigcup_{i=0}^{n-1} A^iB\right) \\ &\subseteq A^n(AX \cup B) \cup \left(\bigcup_{i=0}^{n-1} A^iB\right) \\ &= A^{n+1}X \cup A^nB \cup \left(\bigcup_{i=0}^{n-1} A^iB\right) \\ &= A^{n+1}X \cup \left(\bigcup_{i=0}^{n} A^iB\right) \end{aligned}$$

□

Agora podemos mostrar que se A não contém a palavra vazia, então todas as soluções do sistema estão dentro de A^*B.

Lema 11.1.2: Se $\varepsilon \notin A$, e $X \subseteq AX \cup B$, então $X \subseteq A^*B$.

Prova: Seja w uma palavra de X, e seja $|w|$ o seu comprimento. Pelo Lema 11.1.2, deduzimos que $w \in A^{|w|+1}X \cup \left(\bigcup_{i=0}^{|w|} A^i B\right)$. Mas lembre que, como $\varepsilon \notin A$, então A só contém palavras com comprimento maior ou igual a 1. Portanto, $A^{|w|+1}X$ só contém palavras com comprimento pelo menos $|w|+1$, o que impede que w pertença a $A^{|w|+1}X$. Concluimos que $w \in \bigcup_{i=0}^{|w|} A^i B$, que por sua vez está contido em A^*B. □

Finalmente, podemos enunciar a versão tradicional do lema de Arden.

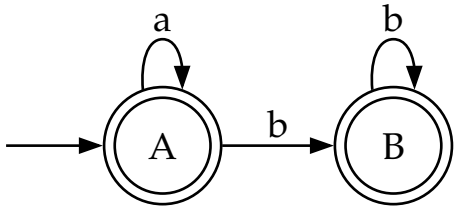
Teorema 11.1.1: Se $\varepsilon \notin A$, e $X = AX \cup B$, então $X = A^*B$.

Prova: Segue do Lema 11.1, Lema 11.2, e Lema 11.1.2. □

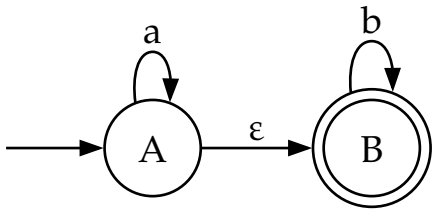
Curiosidade: O Lema 11.1.2 depende crucialmente das palavras terem comprimento finito. Se permitíssemos strings contendo sequências infinitas de caracteres, equações como $X = aX$ permitiriam soluções infinitas como "$aaa...$". Nestas notas de aula, nós não permitimos e nem permitiremos estas strings infinitas. No entanto, eu queria deixar registrado que existem sim situações em que falar de strings infinitas faz sentido. Essa escolha levaria a um teoria alternativa, que trabalharia com maiores pontos fixos ao invés de menores pontos fixos, e provas por co-indução ao invés de por indução.

12. Transições vazias

O autômato a seguir reconhece a linguagem a^*b^*. Porém, isso pode não ser tão óbvio...



Seria legal se pudesemmos construir um autômato que tem uma parte responsável pelo a^* e uma pelo b^*. Uma ferramenta que possibilita isso são arestas ε:



Repare que agora, o autômato é a sequência de dois sub-autômatos. Temos um único estado final, e cada letra da expressão regular a^*b^* corresponde a uma única aresta.

12.1. Eliminando transições ε

Transições ε podem permitir uma representação do autômato com menos arestas, porém não introduzem nenhum poder adicional. Todo autômato com transição ε pode ser reescrito em um autômato equivalente sem ε.

$$A = aA \cup B$$
$$B = bB \cup \varepsilon$$

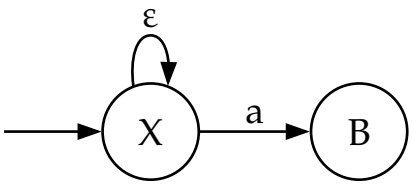
Agora, substituímos a definição de B na equação de A:

$$A = aA \cup (bB \cup \varepsilon)$$
$$B = bB \cup \varepsilon$$

Voilá! Chegamos na versão do autômato sem transição ε. Resumidamente, o estado A herdou todas as transições do estado B, que pode ser alcançado "de graça" a partir de A.

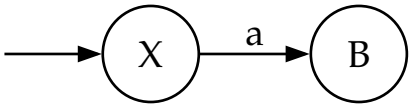
12.2. Loops ε

Tome cuidado com autômatos que tem ciclos de arestas ε, pois nesses casos a regra da substituição não é suficiente.



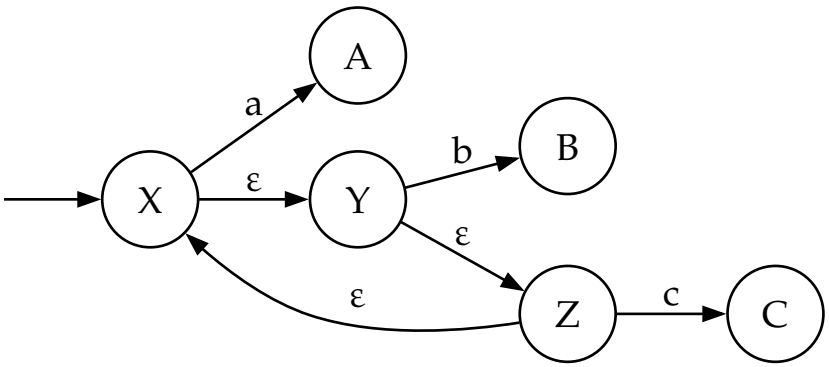
$$X = X + aB$$

Quando chegamos nesse ponto, a solução é usar o Lema de Arden.



$$X = \varepsilon X \cup aB = (\varepsilon^*)aB = aB$$

12.3. Remoção de loops ε maiores



Uma outra situação interessante ocorre quando temos um loop ε com mais de um estado:

$$X = Y + aA$$
$$Y = Z + bB$$
$$Z = X + cC$$

Podemos aplicar a substituição várias vezes até fechar o ciclo:

$$X = aA + bB + Z$$
$$Y = bB + cC + X$$
$$Z = aA + cC + Y$$

$$X = aA + bB + cC + X$$
$$Y = aA + bB + cC + Y$$
$$Z = aA + bB + cC + Z$$

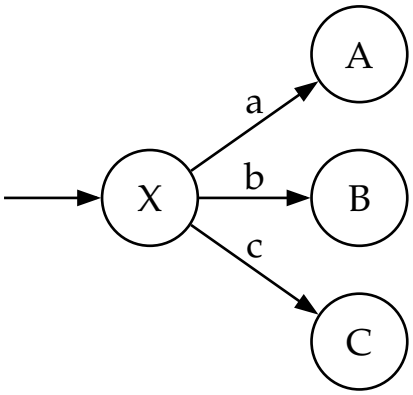
E em seguida, o lema de Arden remove os auto-ciclos

$$X = aA + bB + cC$$
$$Y = aA + bB + cC$$
$$Z = aA + bB + cC$$

Como os estados são equivalentes, podemos combiná-los em um estado só:

$$X = aA + bB + cC$$
$$Y = X$$
$$Z = X$$

O resultado final é um único estado, que contém todas as transições dos estados que participavam daquele loop ε



12.4. Uma prova de $(a^*)^* = (a^*)$

$$X = (a^*)^*$$

$$X = (a^*)X + \varepsilon$$

$$X = Y + \varepsilon$$
$$Y = (a^*)X$$

$$X = Y + \varepsilon$$
$$Y = aY + X$$

$$X = aY + \varepsilon + X$$
$$Y = aY + \varepsilon + Y$$

$$X = aY + \varepsilon$$
$$Y = aY + \varepsilon$$

$$X = Y$$
$$Y = a^*$$

$$X = a^*$$