UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FILIPE RAMOS

ESPECIFICAÇÃO E PROVA DE PROPRIEDADE ACERCA DE AUTÔMATOS FINITOS DETERMINÍSTICOS ASSISTIDAS POR COQ

JOINVILLE - SC

2019

FILIPE RAMOS

ESPECIFICAÇÃO E PROVA DE PROPRIEDADE ACERCA DE AUTÔMATOS FINITOS DETERMINÍSTICOS ASSISTIDAS POR COQ

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Dra. Karina Girardi Roggia
Coorientador: Me. Rafael Castro Gonçalves Silva

JOINVILLE - SC

2019

Filipe Ramos
Especificação e prova de propriedade acerca de autômatos finitos determinísticos assistidas por Coq/ Filipe Ramos. – Joinville - SC, 2019-
35 p. : il. (algumas color.) ; 30 cm.

Dra. Karina Girardi Roggia

– Universidade do Estado de Santa Catarina - Udesc, 2019.

1. Tópico 01. 2. Tópico 02. I. Prof. Dr. xxxxx. II. Universidade do Estado de Santa Catarina. III. Centro de Educação do Planalto Norte. IV. identificação xxxx

CDU 02:121:005.7

Filipe Ramos

Especificação e prova de propriedade acerca de autômatos finitos determinísticos assistidas por Coq

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação

Banca examinadora:

Dra. Karina Girardi Roggia
Instituto Superior Técnico de Lisboa

Dr. Cristiano Damiani Vasconcellos
Universidade Federal de Minas Gerais

Dr. Roberto Silvio Ubertino Rosso Junior
Loughborough University

Joinville - SC, xx de xxxx de 2019

Dedico este trabalho a...

AGRADECIMENTOS

Gostaria de agradecer...

Aqui devem ser colocadas os agradecimentos às pessoas que de alguma forma contribuíram para a realização do trabalho.

“frase”

autor

RESUMO

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chaves: latex, abntex e editoração de texto.

ABSTRACT

Resumo em inglês

Key-words: latex, abntex e text editoration.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

AFD Autômato finito determinístico

SED Sistema a eventos discretos

VANT Veículo aéreo não tripulado

SUMÁRIO

1 INTRODUÇÃO

2 ASSISTENTES DE PROVAS

Um assistente de provas interativo é um artifício de software que auxilia no processo de provas matemáticas ...

2.1 TEORIA DOS TIPOS

A teoria dos tipos fornece o encontro da ciência da computação com a matemática e a lógica. Ela é simultaneamente um sistema formal e linguagem de programação que permite raciocinar enquanto se programa. Sua relevância se explica por possuir simplicidade, ser robusta, apresentar a propriedade da decidibilidade, ser uma linguagem funcional, entre outros (LUO, 1994).

[EXPLICAR O PARADOXO DE RUSSEL]

$$X = \{A \mid A \notin A\}$$

Seria X pertencente a si mesmo? Essa pergunta incorre no paradoxo de Russel ...

[OQ É UM TIPO?]

2.2 TEORIA DOS TIPOS INTUICIONISTA

A teoria dos tipos intuicionista foi estabelecida por Per Martin-Löf em ...

2.3 COQ

(CHLIPALA, 2013)

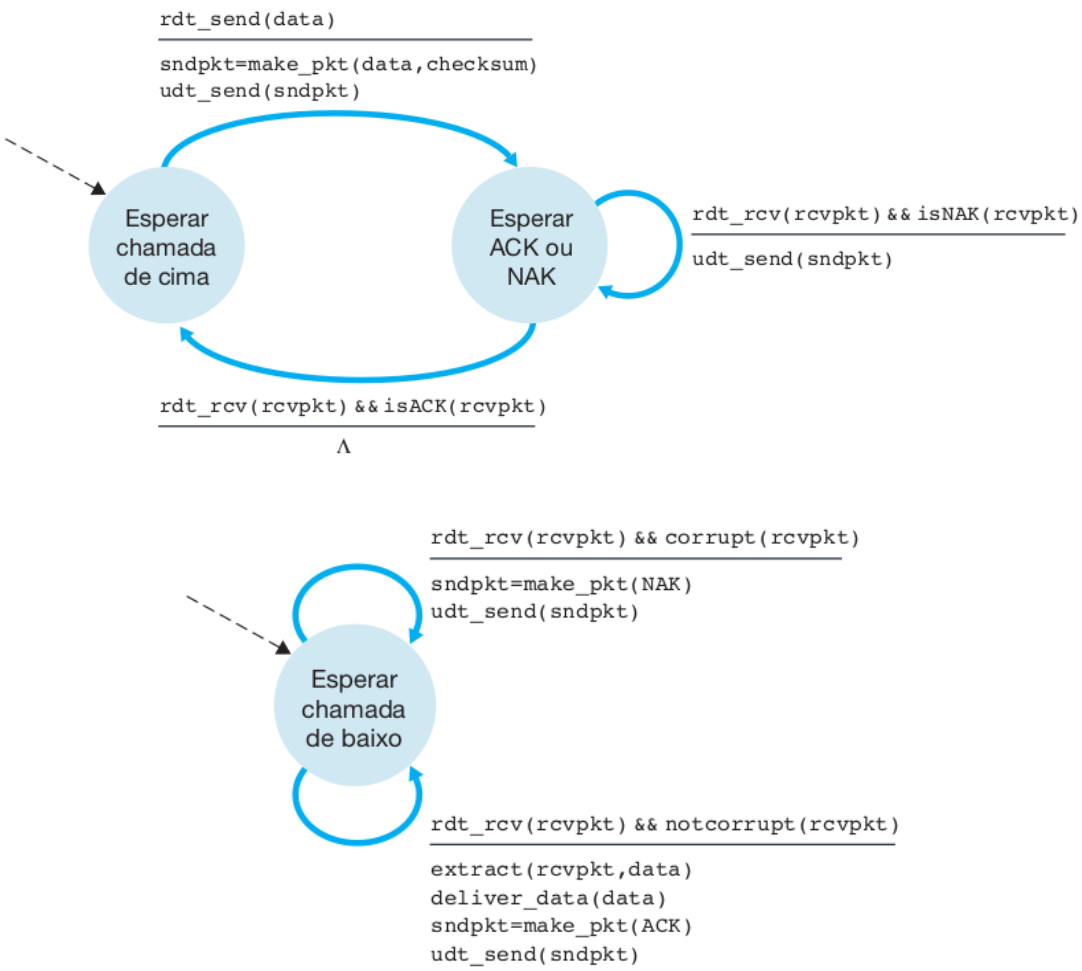
3 SISTEMAS A EVENTOS DISCRETOS

Segundo Lafortune (2019, p. 142, tradução do autor)

> Sistemas a eventos discretos (SEDs) são sistemas dinâmicos com duas características definidoras: seus espaços de estados são discretos e potencialmente infinitos, e sua dinâmica é orientada a eventos, em vez de tempo. (...) Eventos que ocorrem de forma assíncrona (em geral) causam um salto no espaço de estados, de um estado para outro.

A exemplo disso, protocolos de comunicação em redes de computadores são frequentemente modelados por SEDs, como indica a Figura 1.

Figura 1 – Protocolo para um canal de redes de computadores com erros de bits: lado remetente e (abaixo) destinatário



Fonte: KUROSE, J.; ROSS K. **Redes de computadores e a internet**: uma abordagem top-down. 6. ed. São Paulo: Pearson, 2013.

Na Figura 1, tem-se representado um protocolo de redes em que há estados de espera e transições que dependem de eventos discretos – neste caso, procedimentos computacionais. A representação supracitada é uma evidência do caráter de SED do protocolo, e a figura é um diagrama de estados, que será introduzido mais adiante.

...

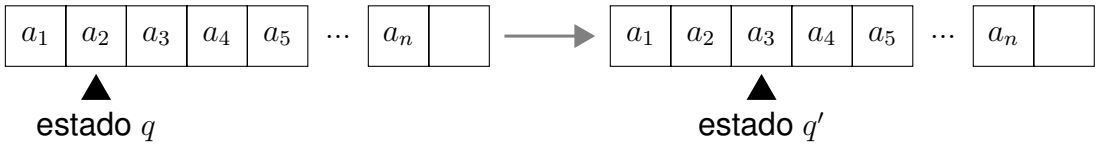
SEDs são comumente representados por redes de Petri e autômatos determinísticos. falar um pouco sobre redes de Petri. Este trabalho de conclusão de concurso, todavia, enfoca a representação por autômatos determinísticos, que podem ser finitos ou infinitos. A diferença entre eles é que naqueles os conjuntos de estados e eventos são finitos e nestes, infinitos. A seção que segue introduz os principais conceitos de autômatos finitos determinísticos usados para este trabalho.

3.1 AUTÔMATOS FINITOS DETERMINÍSTICOS

Autômato – palavra derivada do termo em latim *automatu* – é “maquinismo que se move por meios mecânicos” e “imita os movimentos humanos” (FERREIRA, 2010, p. 81). Esse termo tem sido usado na ciência da computação desde a década de 1930 para descrever importantes modelos da Teoria dos Autômatos, que aborda máquinas de Turing, por exemplo. Nesse contexto, um autômato é uma máquina abstrata descrita matematicamente e idealizada em termos de limitações físicas (HOPCROFT; MOTWANI; ULLMAN, 2007).

Um autômato finito determinístico (AFD), ou máquina de estados finitos determinística, é uma máquina dotada de fita, unidade de controle e função de transição. A fita de um autômato é um espaço ilimitado utilizado para armazenar uma sequência de símbolos que serão lidos e computados. Já a unidade de controle contém as variáveis do estado atual da máquina, que servem de parâmetro para a computação da função de transição. Para determinar o estado de um autômato, há uma cabeça de leitura sobre a fita e um conjunto de elementos abstratos que norteiam a evolução do funcionamento da máquina: os estados (HOPCROFT; MOTWANI; ULLMAN, 2007). A Figura 2 esquematiza a transição de estado de um AFD com a cabeça de leitura inicialmente posicionada sobre a segunda célula da fita.

Figura 2 – Transição de um AFD



Fonte: Elaborada pelo autor, 2019.

Durante a computação de uma cadeia de entrada, o AFD lê o símbolo da célula atual da fita, apontada pela cabeça de leitura, e avança o cabeçote em uma posição para a direita. Inicialmente, ao receber uma entrada, a cabeça de leitura estará posicionada na extremidade esquerda da fita e, por conseguinte, da cadeia de símbolos. Se a computação de um símbolo lido não acarretar um estado definido ou a cabeça de leitura estiver posicionada sobre uma célula vazia, o funcionamento do autômato será interrompido.

Na Ciência da Computação, os AFDs formam a base de alguns componentes de software e partes de compiladores. Um artifício muito empregado no desenvolvimento de software são as expressões regulares, que podem ser convertidas em AFDs e permitem encontrar padrões em textos (HOPCROFT; MOTWANI; ULLMAN, 2007). Já no contexto dos SEDs, os AFDs são modelos matemáticos que descrevem sistemas com base nos eventos que podem ocorrer. Dessa maneira, as cadeias de símbolos que são enviadas à entrada dos AFDs constituem sequências de eventos cuja computação resulta em uma descrição do sistema baseada nas variáveis de controle da máquina de estados (CASSANDRAS; LAFORTUNE, 1999).

3.1.1 Definição formal

A definição de AFD que segue foi inspirada e adaptada de Hopcroft, Motwani e Ullman (2007) e Cassandras e Lafortune (1999).

Um AFD G é uma quíntupla

$$\langle Q, E, \delta, q_0, Q_m \rangle \tag{3.1}$$

em que

Q é o conjunto finito de estados

E é o conjunto finito de símbolos ou eventos

$\delta : Q \times E \nrightarrow Q$ é a função de transição

q_0 é o estado inicial

$Q_m \subseteq Q$ é o conjunto de estados marcados

A função de eventos ativos de G, que será denotada por $\Gamma_G : Q \to 2^E$, relaciona cada estado com os eventos possíveis a partir dele. Formalmente

$$(\forall q \in Q)(\forall e \in E), e \in \Gamma_G(q) \Leftrightarrow \delta(q, e) \in Q$$

isto é, $e \in \Gamma_G(q)$ se e somente se $\delta(q, e)$ é definido.

Ao fecho de Kleene sobre um conjunto de eventos E, denotado por $E^\star$, pertencem todas as possíveis cadeias de eventos pertencentes a E. Uma cadeia de eventos é uma sequência finita $e_1 e_2 ... e_{|w|}$ em que $e_i \in E$ ($\forall i = 1..|w|$). Quando $|w| = 0$, a cadeia é dita vazia e será simbolizada por ε.

A função de transição estendida $\hat{\delta} : Q \times E^\star \nrightarrow Q$ é definida recursivamente destarte:

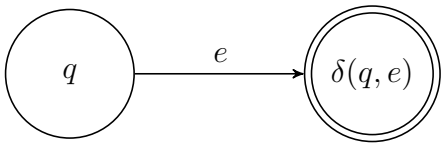
$$\hat{\delta}(q, w) = \begin{cases} q & se\ w = \varepsilon \\ \hat{\delta}(\delta(q, e), w') & se\ w = ew' \end{cases}$$

e terá valor indefinido quando $\delta(q, e)$ assim for.

3.1.2 Diagrama de estados

Para auxiliar na visualização das transições entre os estados dos autômatos, os AFDs são comumente representados por diagramas de estados. Nessa representação, os estados são nós de uma estrutura semelhante a de grafos, e as transições, arestas que interligam dois nós, conforme a Figura 3. Representam-se as transições cuja origem e destino são o mesmo estado por *loops*: arestas que partem de um nó e terminam no mesmo.

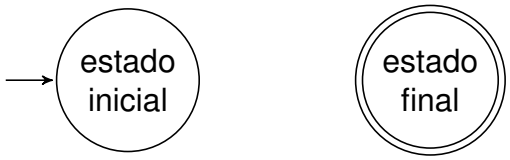
Figura 3 – Representação da transição de estados em um diagrama



Fonte: Elaborada pelo autor, 2019.

Nesta classe de diagramas, os estados inicial e final podem ser destacados de alguma forma. Para este trabalho, uma seta sem origem aponta sempre para o nó do estado inicial, e uma circunferência dupla enfatiza o de um estado final, como demonstra a Figura 4.

Figura 4 – Representação de estados inicial (à esquerda) e final (à direita) em um diagrama



Fonte: Elaborada pelo autor, 2019.

Pode-se citar outros aspectos desta representação de autômatos: a possibilidade de adicionar rótulos aos nós, a opção de omitir os nomes dos estados nos nós quando não forem necessários e a aglutinação de transições que partem e terminam no mesmo estado em uma mesma aresta, com os símbolos separados por vírgula.

3.1.3 Linguagem marcada

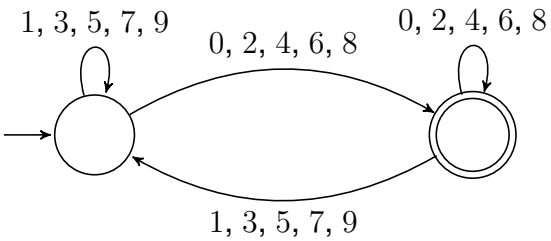
A linguagem marcada por um AFD G é o conjunto

$$L_m(G) = \{w \in E^\star \mid \hat{\delta}(q_0, w) \in Q_m\}$$

Sendo assim, a linguagem marcada pelo autômato são todas as cadeias de eventos que o fazem transicionar do estado inicial a um estado marcado. Isso significa que, quando uma cadeia $w \in L_m(G)$ for posicionada na fita do autômato, a computação de cada evento, da esquerda para a direita da sequência, sempre resultará em um estado definido e terminará em um estado marcado.

Denomina-se linguagem regular a linguagem marcada por qualquer AFD (HOPCROFT; MOTWANI; ULLMAN, 2007). Desse modo, este trabalho de conclusão de curso versará sobre linguagens exclusivamente regulares, a exemplo das quais é possível citar o conjunto dos números naturais pares. Sabendo que um número par é aquele cujo último algarismo – da esquerda para a direita – pertence a $\{0, 2, 4, 6, 8\}$, pode-se construir o AFD da Figura 5, demonstrando a validade da afirmação.

Figura 5 – Diagrama de estados do AFD que reconhece números naturais pares



Fonte: Elaborada pelo autor, 2019.

Quando um autômato se destina a verificar padrões em cadeias de eventos, ou palavras, diz-se que ele é um formalismo reconhecedor (MENEZES, 2005) e, portanto, o autômato da Figura 5 reconhece todos os números naturais pares. No que tange aos SEDs, alcançar um estado marcado implica a conclusão de alguma tarefa (CASSANDRAS; LAFORTUNE, 1999). A impossibilidade de concluir tarefas, os bloqueios ocorrem quando não é possível sair de um estado não marcado e chegar a um marcado, tema da subseção seguinte, 3.1.4.

3.1.4 Linguagem gerada e bloqueios

A linguagem gerada por um AFD G é o conjunto

$$L(G) = \{w \in E^\star \mid \hat{\delta}(q_0, w) \in Q\}$$

Neste caso, a linguagem são todas as cadeias de eventos que fazem o autômato transicionar do estado inicial a um estado definido.

Caso o autômato G chegue a um estado $q \notin Q_m$ tal que $\Gamma_G(q) = \varnothing$, diz-se que há um *deadlock*, já que não existe evento que pode ser executado a partir de q. Um *livelock* se configura quando, mesmo não havendo *deadlock*, não é possível alcançar um estado marcado. Se qualquer bloqueio acontece

$$\overline{L_m(G)} \subset L(G)$$

em que $\overline{L_m(G)}$ é o conjunto de todos os prefixos de todas as cadeias pertencentes a $L_m(G)$, ou

$$\overline{L_m(G)} = \{w \in E^\star \mid \exists w' \in E^\star, ww' \in L_m(G)\}$$

Isso é devido à constatação de que, tendo sido executada uma cadeia de eventos $w \in E^\star$ e estando em um bloqueio, w pertencerá a $L(G)$, mas não será prefixo de alguma cadeia da linguagem marcada: não haverá $w' \in E^\star$ tal que $ww' \in L_m(G)$, por definição (CASSANDRAS; LAFORTUNE, 1999).

3.1.5 Função de transição total

Haja vista que não é possível formular funções parciais no assistente de provas Coq, algumas mudanças na definição de AFD supracitada são necessárias a fim de representar AFDs nessa ferramenta. A começar, é impreterível alterar a função de transição de um AFD G para torná-la total. Seja $\delta' : Q \cup \{\otimes\} \times E \to Q \cup \{\otimes\}$ a seguinte função total:

$$\delta'(q, e) = \begin{cases} \delta(q, e) & \text{se } \delta(q, e) \in Q \\ \otimes & \text{senão} \end{cases}$$

em que $\otimes$ é um estado novo, não pertencente a Q: o estado de ralo.

O autômato G é muito semelhante ao

$$G' = \langle Q \cup \{\otimes\}, E, \delta', q_0, Q_m \rangle$$

uma vez que a única diferença entre eles é que, em G', ao realizar-se uma transição que seria indefinida em G, alcança-se um estado do qual não se pode sair.

Como a função δ' é total, tem-se que

$$L(G') = E^\star$$

em termos do que se estabeleceu como linguagem gerada anteriormente. Pode-se, não obstante, modificar a definição dela de forma que G e G' sejam equivalentes em se tratando de linguagens:

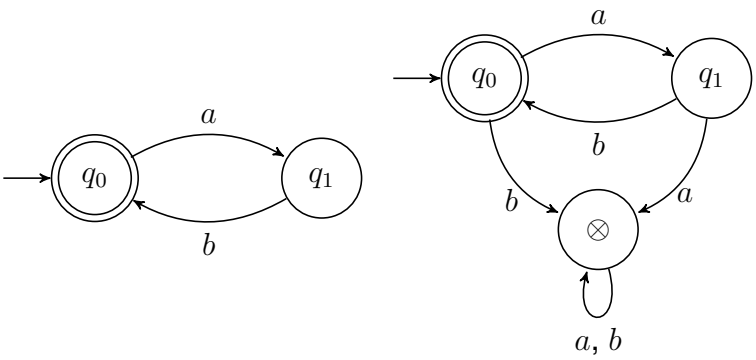
$$L'(G') = \{w \in E^\star \mid \hat{\delta}(q_0, w) \in Q \wedge \hat{\delta}(q_0, w) \neq \otimes\} \tag{3.2}$$

é a linguagem gerada pelo autômato G'. Então

$$L'(G') = L'(G) = L(G)$$

É possível visualizar as formulações de um AFD usando função de transição parcial e total na Figura 6, que a exemplifica para um AFD de alfabeto $\{a, b\}$.

Figura 6 – Um AFD simples (à esquerda) e seu correspondente com função de transição total (à direita)



Fonte: Elaborada pelo autor, 2019.

Isso posto, é evidente que um AFD cuja função de transição não é total pode ser modelado matematicamente utilizando funções totais. Esse fato é importante para que se possa representá-lo na linguagem do Coq.

3.1.6 Formulação em Coq

Todas as definições acerca de AFDs supracitadas utilizam conjuntos, mas, para representar os autômatos no assistente de provas Coq, é necessário que sejam usados tipos. Assim, estados e eventos terão tipos específicos.

Conforme a subseção 3.1.5, a formulação de AFDs em Coq precisa considerar o estado de ralo para que a função de transição seja sempre total. Nesse sentido, pode-se definir os estados como do seguinte tipo paramétrico:

```
Inductive state {Q : Type} := sink_state | proper_state (q : Q).
```

sendo *proper state* – estado próprio – qualquer estado do autômato exceto o de ralo. Ao invés de usar apenas o tipo `Q`, essa definição é importante para facilitar a caracterização das linguagens geradas, já que o estado de ralo tem de estar bem delimitado para defini-las, como demonstra a equação 3.2.

A função de transição será do tipo `Q→E→state`, e não `state→E→state`, o que permitiria quaisquer transições partindo do estado de ralo. Ademais, é importante especificar quais estados são marcados com uma função do tipo `Q→bool` que retorna `true` se e somente se o estado for marcado.

Com isso, um autômato determinístico `g : dfa` terá estes elementos:

- `Q`: o tipo dos estados próprios
- `E`: o tipo dos símbolos ou eventos
- `transition : Q→E→@state Q`: a função de transição
- `initial_state : Q`: o estado inicial
- `is_marked : Q→bool`: a função de demarcação dos estados

Esta definição permite que instâncias do tipo `dfa` sejam, com efeito, autômatos infinitos determinísticos. O presente trabalho, porém, assume que os tipos dos estados e eventos são sempre finitos. A implementação do tipo `dfa` proposta é apresentada pela Figura 7.

Figura 7 – Implementação do tipo dos AFDs em Coq

```
Record dfa (Q E : Type) := {
  transition: Q -> E -> @state Q;
  initial_state: Q;
  is_marked: Q -> bool
}.
```

Fonte: Elaborada pelo autor, 2019.

A definição de AFD anterior é equivalente a esta. Sejam q_1, q_2, ... e q_n, q^* e e_1, e_2, ... e e_m, respectivamente, todos os estados, o estado inicial e todos os possíveis símbolos de um AFD, podemos construir os tipos que seguem:

$$\texttt{Inductive Q :=}\ q_1\ \texttt{|}\ q_2\ \texttt{|}\ \ldots\ \texttt{|}\ q_n\texttt{.}$$

$$\texttt{Inductive E :=}\ e_1\ \texttt{|}\ e_2\ \texttt{|}\ \ldots\ \texttt{|}\ e_m\texttt{.}$$

que são os respectivos tipos dos estados e eventos do autômato na definição para o Coq. Seja $\delta : \{q_1, q_2, ..., q_n, \otimes\} \times \{e_1, e_2, ..., e_m\} \to \{q_1, q_2, ..., q_n, \otimes\}$ a função de transição de estados tal que

$$\delta(q, e) = \begin{cases} q' & \text{se } q \neq \otimes \text{ e } \texttt{transition}\ q\ e\ =\ \texttt{proper\_state}\ q' \\ \otimes & \text{senão} \end{cases}$$

ou, por outro lado, seja `transition : Q→E→@state Q` definida de acordo com o quadro **??**. Além disso, determinemos o conjunto de estados marcados:

$$Q_m = \{q_i \mid i \in \{1, 2, ..., n\} \wedge \texttt{is\_marked}\ q_i\ =\ \texttt{true}\}$$

ou, de outra parte, construamos a função `is_marked` com base no quadro **??**. O autômato $\langle\{q_1, q_2, ..., q_n\}, \{e_1, e_2, ..., e_m\}, \delta, q^*, Q_m\rangle$ pode ser formulado em termos dos elementos `Q`, `E`, `transition` e `is_marked` e vice-versa.

Toda sequência de eventos será representada por uma lista, e a função de transição estendida `extended_transition` será do tipo `dfa→@state Q→list E→@state Q` e definida recursivamente. Se o estado de origem for o de ralo, ele será o próprio retorno da função; senão, `extended_transition` computará a função de transição para a cabeça da lista de eventos e fará recursão destarte:

```
extended_transition g (proper_state q) [] = proper_state q;
extended_transition g (proper_state q) e::w =
  extended_transition g (transition q e) w
```

Por definição, uma lista de eventos `w` será gerada por `g`, ou `g ==> w`, se

```
extended_transition g (proper_state initial_state) w != sink_state
```

ou seja, se a computação da função de transição estendida da lista partindo do estado inicial não resultar no estado de ralo, em conformidade com a equação 3.2. Sendo G o AFD definido conforme a equação 3.1 e equivalente ao `g`, podemos notar

$$L(G) = \{a_1a_2...a_n \in E^\star \mid \texttt{g ==> [}a_1\texttt{;}a_2\texttt{;...;}a_n\texttt{]}\}$$

evidenciando também a correlação entre as diferentes definições supracitadas para a mesma classe de máquinas abstratas.

4 PROBLEMAS DE FILAS

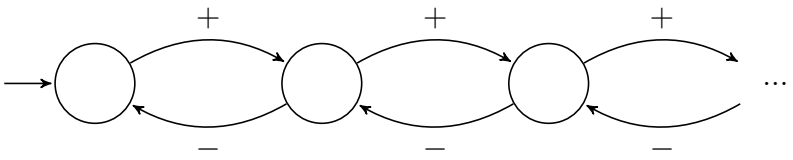
Os sistemas de filas constituem importante classe de sistemas a eventos discretos (SEDs) em que entidades ou objetos devem esperar para obter um determinado recurso. Há três elementos básicos que compõem esses sistemas:

- consumidor: entidade que espera por um recurso;
- servidor: o recurso que é aguardado;
- fila: o espaço em que os consumidores esperam.

Os recursos são genericamente denominados servidores por geralmente proverem serviço (CASSANDRAS; LAFORTUNE, 1999). O presente trabalho aborda uma ótica abstrata dos sistemas de filas, já que há a possibilidade de reduzir problemas a esses sistemas.

Para ser atendido por um servidor de banco, uma pessoa deve esperar em uma fila até que as pessoas a sua frente sejam atendidas. Esse sistema de filas pode ser modelado pelo autômato infinito determinístico ilustrado pela Figura **??**, em que $+$ e $-$ são os respectivos eventos de chegada e partida de pessoa da fila.

Figura 8 – Um sistema de filas simples



Fonte: Elaborada pelo autor, 2019.

Na Figura 8 é assumido que a fila de pessoas pode crescer indefinidamente, o que não é possível na realidade. Assumindo que as filas têm um tamanho máximo, faz-se possível modelar vários sistemas de filas por meio de autômatos finitos determinísticos (AFDs).

Um exemplo de sistema de filas é apresentado por Ferreira (2019) e consiste em uma planta de fila de demandas para veículos aéreos não tripulados (VANTs). Nele as demandas são os consumidores, e os VANTs, os servidores.

4.1 PROPRIEDADES DESEJADAS

Um sistema de filas geralmente requer que duas propriedades inerentes sejam satisfeitas: que o tamanho máximo de cada fila não possa exceder uma capacidade máxima e não seja possível realizar operações de retirada em uma fila vazia. Isso decorre do fato de que normalmente as filas destes sistemas são representações de espaços físicos, com limitações que devem ser respeitadas nos modelos de SEDs.

O problema do produtor-consumidor é um exemplo de aplicação destas propriedades. Na ciência da computação, a concorrência entre threads sobre os mesmos recursos impõe que alguns artifícios de software e hardware sejam empregados a fim de respeitá-las. Sem estes, o desenvolvimento de aplicações computacionais com recursos compartilhados gera defeitos que são manifestados no uso. exemplificar. Para a automação industrial, exemplificar.

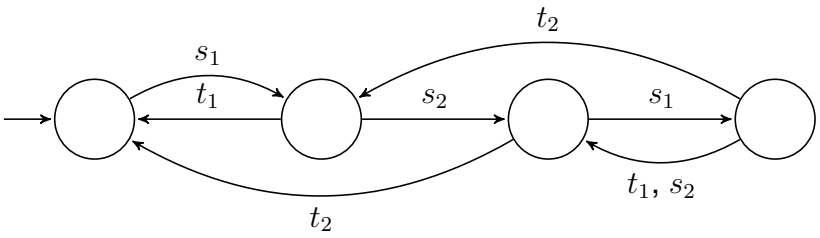
Apesar de as representações de filas físicas requererem um volume mínimo de zero itens, esta pode não ser uma condição para filas abstratas. A subseção a seguir aborda estes e outros casos, em que é possível reduzir problemas abstraindo a noção de filas, servidores e consumidores.

4.1.1 Redução de problemas

É possível reduzir problemas à validação das propriedades supracitadas. Seja um SED com a seguinte restrição: há dois subconjuntos de eventos E_1 e E_2 tais que, em qualquer cadeia de eventos, o número de eventos pertencentes a E_1 menos o número de eventos pertencentes a E_2 é maior ou igual a n e/ou menor ou igual a m. Pode-se modelá-lo como um sistema de filas em que os eventos de E_1 e E_2 sejam substituídos respectivamente por operações de adição e retirada de fila. Garantido que a lista tem capacidade máxima de $m - n$ itens e volume mínimo de zero itens, a restrição é satisfeita.

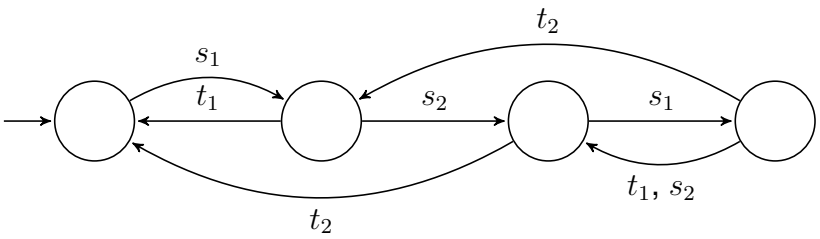
A exemplo disso, seja o autômato da Figura 10. ... O exemplo é simples, mas ilustra facilmente esta redução de problemas.

Figura 9 – Autômato para o funcionamento de uma linha de produção simples



Fonte: Elaborada pelo autor, 2019.

Figura 10 – Autômato da Figura 10 reduzido para um sistema de filas



Fonte: Elaborada pelo autor, 2019.

4.1.2 Aplicações

5 GARANTIA DE PROPRIEDADES EM SISTEMAS DE FILAS

ESCREVER MAIS

Para representar as operações de adição e retirada de fila, utilizaremos os respectivos símbolos $+$ e $-$. Seja $G = \langle Q, E, \delta, q_0, Q_m \rangle$ um AFD definido conforme a equação 3.1, em que $E \supseteq \{+, -\}$, queremos provar que o sistema de filas modelado por G satisfaz as duas propriedades supracitadas no capítulo 4: que a fila tenha uma capacidade máxima e não seja permitido efetuar operação de retirada de item quando ela estiver vazia.

Precisamos de uma função $c : E^\star \to \mathbb{Z}$ que conte o número de itens armazenados em uma fila após uma cadeia de eventos w. Uma forma de defini-la é

$$c(w) = \begin{cases} c(w') + 1 & \text{se } w = +w' \\ c(w') - 1 & \text{se } w = -w' \\ 0 & \text{senão} \end{cases}$$

Baseado nas propriedades aritméticas da soma de inteiros, podemos observar para quaisquer w_1 e $w_2 \in E^\star$

$$c(w_1 w_2) = c(w_1) + c(w_2)$$

que será útil para a prova dos teoremas acerca das propriedades dos sistemas de filas.

A fim de facilitar a leitura dos teoremas, simplifiquemos a notação para a transição estendida de um estado q por uma cadeia de eventos w da seguinte maneira:

$$qw \equiv \hat{\delta}(q, w)$$

Pelo mesmo motivo, definamos a função $t : E \to \mathbb{Z}$ desta forma:

$$t(e) = \begin{cases} 1 & \text{se } e = + \\ -1 & \text{se } e = - \\ 0 & \text{senão} \end{cases}$$

5.1 GARANTIA DE CAPACIDADE MÁXIMA

Teorema 1. *Existe uma função $f : Q \to \mathbb{Z}$ e um inteiro n tal que*

$$f(q_0) = 0 \wedge [(\forall q \in Q)(\forall e \in E), f(qe) \geq f(q) + t(e) \wedge f(q) \leq n]$$

se e somente se a fila do sistema modelado por G tem sempre no máximo n itens.

INTRODUZIR PROVA Provemos antes o lema que segue.

Lema 1. *Para toda função $f : Q \to \mathbb{Z}$, é válido que*

$$f(q_0) = 0 \wedge [(\forall q \in Q)(\forall e \in E), f(qe) \geq f(q) + t(e)] \Rightarrow \forall w \in L(G), c(w) \leq f(q_0 w)$$

Demonstração. A prova deste lema será por indução em w, tendo como base

$$c(\varepsilon) \leq f(q_0 \varepsilon)$$

o que é verdade, pois $c(\varepsilon) = 0$ e é pressuposto $f(q_0) = 0$.

Para o passo indutivo, temos de provar que, para todo evento $u \in E$, se wu é gerada por G, então

$$c(wu) = c(w) + t(u) \leq f(q_0 wu)$$

ou

$$c(w) \leq f(q_0 wu) - t(u) \tag{5.1}$$

é válido a partir da hipótese de indução.

Sabemos que, se wu é gerada por G, então o prefixo w também é. Diante disso

$$c(w) \leq f(q_0 w) \tag{5.2}$$

é obtido da hipótese de indução.

Com base na hipótese do lema, obtemos

$$f(q_0 w) \leq f(q_0 wu) - t(u) \tag{5.3}$$

A partir das equações 5.2 e 5.3, verifica-se

$$c(w) \leq f(q_0 w) \leq f(q_0 wu) - t(u)$$

demonstrando a equação 5.1, como queríamos. □

INTRODUZIR PROVA ⇒ DO TEOREMA

Demonstração. Aplicando o lema 1 na hipótese do teorema, obtemos

$$\forall w \in L(G), c(w) \leq f(q_0 w)$$

Da hipótese também temos

$$f(q_0 w) \leq n$$

Portanto

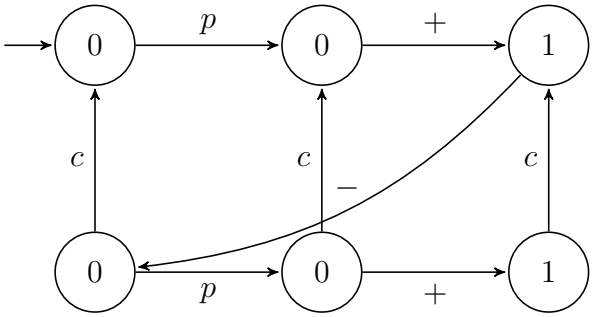
$$c(w) \leq f(q_0 w) \leq n$$

como desejávamos demonstrar. □

A demonstração do lema 1 e teorema 1 foram auxiliadas pelo Coq.

Na figura 11, há um exemplo que ilustra um sistema de produtor e consumidor. Nela os eventos p e c são, respectivamente, a produção e consumo de item. Seja $f_1 : Q \to \mathbb{Z}$ uma função cuja imagem está rotulada nos nós da figura, indicando para cada estado o valor mapeado. Como f_1 respeita o teorema 1, podemos concluir que o sistema modelado permite volume máximo de um item na fila.

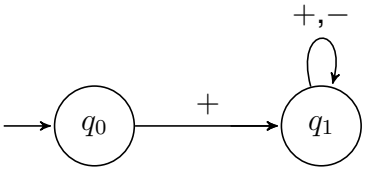
Figura 11 – Exemplo de aplicação do teorema 1 em um sistema que atende à propriedade



Fonte: Elaborada pelo autor, 2019.

Sob outra perspectiva, a figura 12 apresenta um exemplo de AFD que modela um sistema que permite, notavelmente, adição e retirada de um número indefinido de itens da fila. Supondo que tal sistema atenda à propriedade da capacidade máxima, seja $f_2 : Q \to \mathbb{Z}$ qualquer função que mapeie cada estado a um número inteiro de forma que $f_2(q_0) = 0$ e $f_2(qe) \geq f_2(q) + t(e)$ para todo evento $e \in \{+, -\}$. É fácil notar que f_2 não existe, pois, como $q_1 e = q_1$, podemos obter o absurdo: $f(q_1) \geq f(q_1) + t(e)$. Com isso, o teorema 1 conclui que o sistema não admite capacidade máxima.

Figura 12 – Exemplo de sistema que não respeita a propriedade



Fonte: Elaborada pelo autor, 2019.

Além de provar que determinados sistemas de filas cumprem ou não com a

especificação de uma fila com volume máximo, muitas vezes quer-se atestar a impossibilidade de efetuar operação de retirada de fila vazia. É sobre isso que trata a seção 5.2, a seguir.

5.2 GARANTIA DE VOLUME MÍNIMO

Todo sistema de filas especificado com base em Cassandras e Lafortune (1999) deve impedir a remoção de itens de filas quando estas forem vazias. Isso é em razão do caráter físico e não abstrato com que se configuram as filas por aquela ótica. Mesmo para filas abstratas, pode ser interessante que esta propriedade seja respeitada. A fim de garantir isso, o teorema 2 nos possibilita obter resposta à pergunta: a fila tem volume mínimo de zero itens?

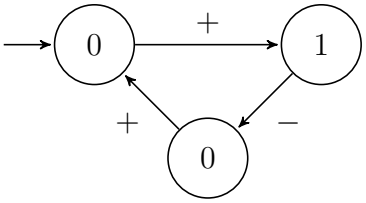
Teorema 2. *Existe uma função* $f : Q \to \mathbb{Z}$ *tal que*

$$f(q_0) = 0 \wedge [(\forall q \in Q)(\forall e \in E), f(qe) \leq f(q) + t(e) \wedge f(q) \geq 0]$$

se e somente se a fila do sistema modelado por G *nunca tem volume negativo.*

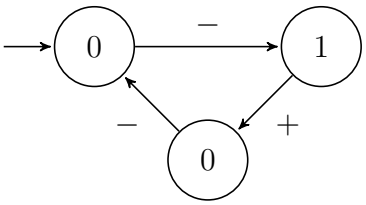
Considerando o autômato da figura 14, verificamos que o sistema nela representado tem uma fila que respeita esta propriedade, embora não garanta capacidade máxima. Tomando a função $f_3 : Q \to \mathbb{Z}$ rotulada nos nós da figura, pode-se notar que ela respeita o teorema 2, evidenciando que tal SED foi modelado de modo a não permitir o evento $-$ nos momentos em que a fila está vazia.

Figura 13 – Exemplo de sistema que atende à propriedade



Fonte: Elaborada pelo autor, 2019.

Figura 14 – Exemplo de sistema que atende à propriedade



Fonte: Elaborada pelo autor, 2019.

6 CONSIDERAÇÕES PARCIAIS

REFERÊNCIAS

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to discrete event systems**. Boston: Kluwer Academic, 1999.

CHLIPALA, A. **Certified programming with dependent types: a pragmatic introduction to the Coq proof assistant**. Cambridge, MA: The MIT Press, 2013.

FERREIRA, A. B. d. H. **Míni Aurélio: o dicionário da língua portuguesa**. Curitiba: Editora Positivo, 2010.

FERREIRA, V. L.

Coordenação de múltiplas aeronaves não tripuladas baseada na alocação dinâmica de tarefas associadas à logística de entregas — Universidade do Estado de Santa Catarina, 2019. Disponível em: <http://sistemabu.udesc.br/pergamumweb/vinculos/000071/0000714c.pdf>. Acesso em: 16 out. 2019.

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. **Introduction to automata theory, languages, and computation**. Boston: Pearson/Addison Wesley, 2007.

LAFORTUNE, S. Discrete event systems: Modeling, observation, and control. **Annual Review of Control, Robotics, and Autonomous Systems**, Annual Reviews, v. 2, p. 141–159, 2019. Disponível em: <https://www.annualreviews.org/doi/abs/10.1146/annurev-control-053018-023659>. Acesso em: 18 out. 2019.

LUO, Z. **Computation and reasoning: a type theory for computer science**. Oxford New York: Clarendon Press Oxford University Press, 1994.

MENEZES, P. F. B. **Linguagens formais e autômatos**. Porto Alegre: Sagra-Luzzatto, 2005.