Prova de propriedades de linguagens regulares usando Coq[1]

Filipe Ramos[2], Cristiano Damiani Vasconcellos[3].

[1] Vinculado ao projeto "Sistemas de tipos para linguagens de programação"
[2] Acadêmico do curso de Ciência da Computação – CCT – Bolsista PROBIC
[3] Orientador, Departamento de Ciência da Computação – CCT – cristiano.vasconcellos@udesc.br

Autômatos finitos determinísticos (AFDs) são modelos abstratos de máquinas que aceitam ou rejeitam entradas. Uma maneira de visualizar esse conceito é na forma de uma função que retorna uma entre duas possibilidades. A entrada de um autômato é uma concatenação de símbolos de um conjunto denominado alfabeto (Σ). Um AFD lê a entrada de modo que cada símbolo lido em sequência acarrete mudança de estado que respeite as regras de transição. Ao fim da leitura, se o último estado de transição é de aceitação, diz-se que o autômato aceita a entrada; senão, ele a rejeita. A linguagem de qualquer AFD é dita regular e pode ser descrita por uma expressão regular. As aplicações de AFDs e linguagens regulares vão do processamento de texto e compiladores à especificação de sistemas a eventos discretos. Tais sistemas são orientados a eventos e não ao tempo; assim, podem ser modelados por autômatos cujos símbolos são eventos. O AFD da Figura 1 exemplifica um protocolo de rede simples, em que cada símbolo ou evento é um comando programado.

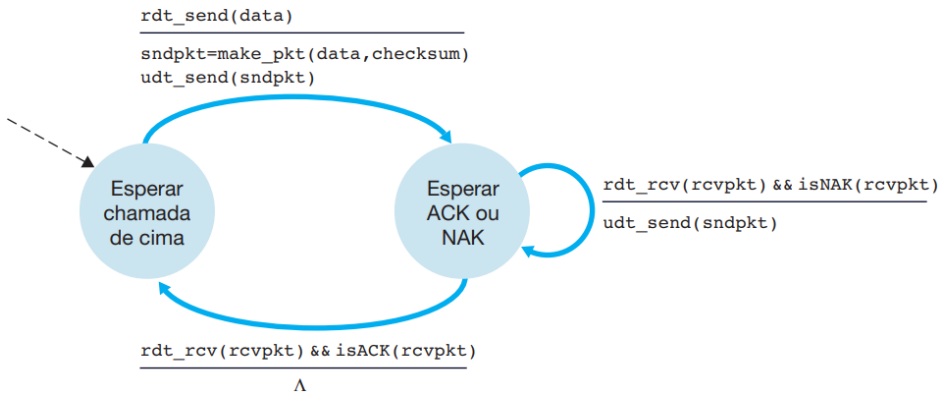

Figura 1. *Exemplo de AFD com saída (KUROSE, 2013)*

Diversas provas acerca de AFDs já foram produzidas, para teoremas como o lema do bombeamento, o teorema de Kleene e o de Myhill-Nerode. Sob outra perspectiva, pode ser requisitada a garantia de propriedades de um sistema modelado por AFD ou ainda ser necessário assegurar a especificação do modelo. Para tanto um assistente de provas auxilia nesses processos ao permitir a composição de provas interativas verificadas por computador. O Coq é um assistente largamente utilizado para provas matemáticas de propósito geral que, conforme sua página oficial na web descreve, "combina uma lógica de ordem superior e uma linguagem de programação funcional ricamente tipada". Esse provador interativo já auxiliou no processo de algumas provas sobre AFDs e afins da teoria da computação (DOCZKAL; KAISER; SMOLKA, 2013).

Antes de começar a provar teoremas sobre AFDs no Coq, deve-se adaptar suas definições matemáticas para a linguagem formal do assistente. Usualmente tem-se que um AFD é uma quíntupla $\langle Q, \Sigma, \delta, q_0, F \rangle$ sendo Q seu conjunto de estados, Σ seu alfabeto, $\delta: Q \times \Sigma \nrightarrow Q$ sua função

parcial de transição, $q_0 \in Q$ seu estado inicial e $F \subseteq Q$ seu conjunto de estados finais ou de aceitação (HOPCROFT, 2006). No Coq todas as funções devem ser totais, motivo pelo qual alguns AFDs precisam ser ajustados com um estado de ralo, que equivale ao estado indefinido na função de transição parcial. Ademais é preciso definir o tipo dos elementos que compõem o autômato. Uma possibilidade para isso é atribuir o tipo string da biblioteca Strings. Os estados e símbolos podem ser agrupados, entre outras formas, em listas. A Figura 2 ilustra uma possível definição de AFD para o Coq. Os respectivos elementos `transition_correct`, `start_correct` e `accept_correct` são restrições para garantir que a função de transição aplicada sobre um elemento da lista de estados e outro da lista de símbolos resulte em um elemento da lista de estados, que o estado inicial esteja na lista de estados e todo elemento da lista de estados de aceitação esteja também na lista de estados.

```
Record dfa := {
      states             : list string;
      alphabet           : list string;
      transition         : string->string->string;
      start              : string;
      accept             : list string;
      transition_correct : forall q a, In q states -> In a alphabet ->
                                 In (transition q a) states;
      start_correct      : In start states;
      accept_correct     : forall q, In q accept -> In q states
}.
```

Figura 2. *Formalização de AFD no Coq*

Utilizando a implementação supracitada foram provados teoremas como o lema do bombeamento, a decidibilidade dos estados acessíveis e teoremas sobre sistemas de filas modelados por esses autômatos. O procedimento consistiu na definição de funções úteis, divisão dos problemas em lemas e aplicação de táticas do Coq, como `apply some_lemma`. A ferramenta mostrou-se eficaz para verificar propriedades gerais referentes aos AFDs.

Referências

DOCZKAL, Christian; KAISER, Jan-Oliver; SMOLKA, Gert. A constructive theory of regular languages in Coq. In: **International Conference on Certified Programs and Proofs**. Springer, Cham, 2013. p. 82-97.

HOPCROFT, J. E. **Introduction to Automata Theory, Languages, and Computation**. 3. ed. Boston: Pearson, 2006.

KUROSE, J. F. **Redes de computadores e a Internet**: uma abordagem top-down. 6. ed. São Paulo: Pearson Education do Brasil, 2013.

Palavras-chave: Autômato finito determinístico. Assistente de provas. Coq.