Redação LFA - Decidibilidade

André Marcelino de Souza Neves
andreneves3@gmail.com

26 de Junho de 2019

Introdução

É inegável a contribuição de *Alan Mathison Turing* na ciência da computação, tanto que sua proposta de uma máquina formal para reconhecimento de linguagens sensíveis ao contexto possui, para resolução de problemas de computação, expressividade semelhante a qualquer linguagem de programação moderna. Essa conclusão foi enunciada como **A Tese de Church-Turing**: *Se uma função é efetivamente computável então ela é computável por meio de uma máquina de Turing* [1].

Foi proposto por *Hilbert* um problema denominado *Entscheidungsproblem (Problema de decisão)*, o qual questiona se existe um conjunto de passos ordenados (denominado posteriormente como **Algoritmo)** que possa determinar se um determinado problema ou enunciado lógico tem solução possível [2]. A conclusão de *Turing* é que é impossível decidir se um problema pode ou não ser provado, o que introduz o conceito de **Decidibilidade**.

Para um problema decidível existe uma máquina de *Turing* que, para cada instância, se a resposta for válida, a MT repousa em estado final, e se for inválida, a MT repousa em um estado não final [1]. Em um problema indecidível, a máquina de *Turing* para em um estado final apenas quando for possível determinar que a resposta é positiva.

Redutibilidade

Na literatura é possível encontrar vários problemas clássicos que foram provados serem indecidíveis. Além disso, corriqueiramente surgem outros problemas que são necessários provar sua decidibilidade. É possível obter essa prova por meio de comparação de qualquer problema com algum outro que já tenha sido provado ser decidível ou indecidível, por meio da redutibilidade de um problema a outro.

Um problema de decisão P é redutível a um problema Q se é possível ter um algoritmo que transforme um valor x em um valor y, e tal que a resposta de P para x é idêntica ou complementar à resposta de Q para y. Dessa forma, afirma-se que o problema P é redutível a Q [1].

Com a análise de redutibilidade, é possível propagar a definição de decidibilidade entre dois problemas, conforme mostrado a seguir [1]:

- Se P é redutível a um problema decidível, então P é decidível;
- Se um problema indecidível é redutível a um problema P, então P é indecidível.

Problema da parada

O problema da parada é um exemplo adequado para ser usado em contextos de execução de programas de computadores. Segue seu enunciado: Dadas uma MT arbitrária M e uma palavra arbitrária w, determinar se a computação de M com a entrada w tem finalização [1].

O problema tem sua fundamentação na ideia de que, para que um algoritmo determine se M tem finalização com a computação de w, será necessário acompanhar toda a execução de M. Caso a computação tenha fim, M chegará em um estado final, de forma que o algoritmo possa percebê-lo. Em suposição de um caso contrário, o algoritmo nunca poderá assumir que a computação não terá fim.

Isso acontece pois podem haver casos em que a computação da palavra de entrada seja demasiadamente demorada, e por isso, o algoritmo não pode assumir que a computação é infinita. Com isso, o algoritmo tem resposta positiva apenas quando a computação de w por M tem um fim, e caso não tenha, o algoritmo irá esperar indefinidamente, de forma que também não repousará em algum estado de aceitação ou não aceitação.

Problema do mundo real semelhante: Determinação de servidores offline

Um sistema de gerenciamento de servidores é um *gateway* que recebe todas as requisições antes de

repassá-las para um servidor. Esse sistema precisa determinar se o servidor está online ou offline. Um servidor que está travado com alguma computação não deve ser considerado offline.

A questão é saber se houve falha de energia ou problema de conexão de rede. A determinação de que o servidor está offline é baseada no acontecimento de algum desse dois problemas. Contudo, se o servidor está travado, não é permitido assumir tal situação.

O sistema sabe qual é a configuração do servidor, programas instalados, e tem a lista de requisições (análogo à uma palavra de entrada de uma Máquina de *Turing*). Caso o servidor esteja online e destravado, obtém-se uma resposta válida imediatamente. Caso esteja offline, não é possível obter resposta, mas não é permitido assumir que está offline, pois o servidor pode estar travado.

É necessário agora, determinar se o servidor está travado ou não. De posse da configuração do servidor e da última requisição recebida, o sistema deve emular a execução, de forma que possa concluir se o servidor está ocupado com essa requisição ou se realmente está offline.

A requisição é a palavra de entrada do problema Q. Ela também será a palavra de entrada para o teste da parada, que será feito no sistema gerenciador, de forma a emular a requisição feita no servidor. O teste de que o servidor será ou não destravado é o clássico problema da parada.

O problema dos servidores possui a mesma palavra de entrada do problema da parada, e em termos de decidibilidade, ambos produzirão respostas semelhantes: Caso possua um fim, a resposta será positiva. Caso não possua, ambos os problemas não tem finalização. Isso prova que o problema da parada é redutível ao problema dos servidores, e por consequência da redutibilidade, é provado que o problema dos servidores também é indecidível.

Décimo problema de Hilbert

O décimo problema de *Hilbert* também é um exemplo clássico de indecidibilidade. O problema consiste no questionamento sobre um algoritmo que, dado uma equação com qualquer número de variáveis e que tenha coeficientes inteiros, determine se é possível encontrar alguma tupla que contenha apenas números inteiros e que solucionem a equação. Um exemplo de equação é: $4x + 2y = 2$, a qual uma possível solução é a tupla $\{x = 1,\ y = -1\}$.

Em uma possível solução, o problema cai novamente na situação semelhante ao problema da parada, onde é necessário fazer uma varredura sobre os números inteiros e testar se uma determinada tupla é solução para a equação. Caso não haja solução do tipo para determinada equação de entrada, o algoritmo procura infinitamente por uma resposta válida.

Problema do mundo real semelhante: Busca de raízes exatas de uma equação

Um problema do mundo real semelhante é uma função que utiliza algum método numérico para procurar por raízes exatas de uma equação qualquer. O algoritmo pode ser o método de bisseção, o qual confina, repetidamente, um intervalo inicial na busca pela raiz da função. Caso a equação possua alguma raiz exata, o algoritmo poderá encontrá-la. Caso não possua, o algoritmo irá, infinitamente, refinar o intervalo considerado até que seu ponto médio seja uma raiz, execução que não será finalizada.

Aproximação da solução computacional

Um problema indecidível terá parada apenas quando sua entrada for satisfatória, ou então não terá fim. A solução para aproximar uma solução em ambos os problemas é usar um tempo limite para execução de um algoritmo. É possível usar um tempo limite de acordo com os requisitos de sistema de tempo de resposta.

Para otimização da execução, uma solução viável é:

- Dada uma entrada, definir um tempo limite, como um dia. Se o algoritmo der retorno positivo, armazenar a resposta em um dicionário, de forma que nas próximas execuções o sistema seja capaz de dar uma resposta em tempo satisfatório.

As consequências práticas dessa técnica é que, se houver alguma entrada tal que, apesar de ser válida para o problema, tem um tempo de computação muito elevado, ela será ignorada e levará a uma situação onde o sistema assume uma informação falsa.

Referências

[1] N. J. Vieira, "Linguagens e Máquinas: Uma Introdução aos Fundamentos da Computação," Tech. Rep., 2004. [Online]. Available: http://homes.dcc.ufba.br/~leandrojsa/livroFTC.pdf

[2] D. G. Bispo, "A teoria da computação de Alan Turing," Tech. Rep., 2018. [Online]. Available: https://tede2.pucsp.br/bitstream/handle/21265/2/Danilo%20Gustavo%20Bispo.pdf