MATA51: Teoria da Computação

Semestre 2021.1

Prof. Laís Nascimento

Alberto Lucas e Renata Ribeiro

Lista de Exercícios 5 - Problema Halts

---

1. **Na execução do programa diagonal, o que acontece se Halts (diagonal, diagonal) retorna 0.**

   Ao retornar 0, a função Halts entrará em loop infinito, pois o primeiro parâmetro é um programa em C válido que para quando lê uma entrada, ou seja, o segundo parâmetro.

2. **Na execução do programa diagonal, o que acontece se Halts (diagonal, diagonal) retorna 1.**

   Ao retornar 1, a função Halts não entrará em loop infinito, pois o primeiro parâmetro é um programa em C que não é válido, portanto não para quando ler uma entrada, ou seja, o segundo parâmetro.

3. **O que podemos inferir após as análises das respostas às questões 1 e 2?**

   Podemos inferir que dependendo do valor do retorno da função, saberemos se se a entrada é válida ou não, ou seja, o programa para ou não.

4. **Podemos especificar o resultado que diagonal retorna sobre a entrada diagonal.c?**

   Sim, pois sempre obteremos apenas duas respostas, ou o programa para, ou o programa não para. Desse modo, podemos especificar o resultado como sendo sim ou não, ou ainda, como decidível ou não decidível.

5. **Eu consigo escrever a função Halts em C? Consigo escrevê-la em qualquer outra linguagem de programação?**

   Não é possível escrever Halts em C ou qualquer outra linguagem pois não existe algoritmo genérico executando sobre um computador que resolve o problema da parada. A prova da afirmação anterior foi dada por Alan Turing em 1936, especificamente para uma Máquina de Turing.

6. **Este texto descreve o halting problem (Problema da Parada). Este problema é decidível?**

O problema da parada é indecidível. Isto é, não existe nenhuma máquina de Turing que decida a linguagem **L = {⟨M, w⟩ : M é uma MT e M aceita w}** para todas as entradas.

Para vermos isso, suponhamos que existe uma MT **H** que aceita **L.** Isto é, **H** para e aceita a entrada ⟨**M, w**⟩ se **M** aceita **w**. Então podemos usar **H** para construir uma nova MT **D** que se comporta da seguinte maneira:

> Na entrada de **M**, onde **M** é uma MT, **D** executará **H** na entrada ⟨**M,**⟨**M**⟩⟩. Isto é, **D** executa **H** para checar se **M**, quando dado si mesmo como entrada, para e aceita.
>
> Se **H** aceita, o que significa que **M** aceita ⟨**M**⟩, então **D** rejeitará a entrada. Por outro lado, se **M** rejeitar ⟨**M**⟩, **D** aceitará.
>
> E o que acontece quando executamos **D** com ⟨**D**⟩ como entrada? Se **D** aceita ⟨**D**⟩, então isso significa que **H** executa **D** sobre ⟨**D**⟩ e retornou rejeição, pois é como definimos **D**. De modo semelhante, se **D** rejeita ⟨**D**⟩, isso significa que **D** aceitou ⟨**D**⟩.

E portanto, temos uma contradição. Concluímos que **D** não pode existir, e consequentemente **H** também não.

7. **Especifique a linguagem associada ao Problema da Parada (vamos chamá-la de linguagem Halts). Esta linguagem é recursiva?**

Se a linguagem Halts existisse ela seria recursiva, tendo em vista que retornaria 1 ou 0. Na verdade, se o Problema da Parada fosse decidível então toda Linguagem Recursivamente Enumerável deveria ser uma Linguagem Recursiva (Decidível).

Vale lembrar que para toda linguagem recursivamente enumerável existe uma máquina de Turing que irá **parar e aceitar** quando se roda com qualquer cadeia da linguagem na entrada e pode **parar e rejeitar ou entrar em loop** quando se roda com qualquer cadeia que não é da linguagem. **Esta é a diferença entre linguagem recursiva**, que exige que a máquina de Turing sempre pare.

8. **É possível construir uma Máquina de Turing para reconhecer as cadeias da linguagem Halts? Se sim, mostre o esboço (diagrama) da MT. Se não, justifique a negativa.**

Não é possível construir uma Máquina de Turing para reconhecer as cadeias de linguagem Halts. Isso se dá pelo problema ser indecidível. A prova foi feita por Alan Turing em 1936 em seu artigo “On Computable Numbers, with an Application to the Entscheidungsproblem”. Em sua prova de que o Entscheidungsproblem (Problema da Parada) não pode ter solução, Turing partiu de duas provas que deveriam levar à sua prova final. Seu primeiro teorema é mais relevante para o problema da parada, o segundo é mais relevante para o teorema de Rice .

a) **Primeira prova:** que não existe nenhuma "máquina de computação" que possa decidir se uma "máquina de computação" arbitrária (representada por um inteiro 1, 2, 3,...) É "livre de círculos", ou seja, continua imprimindo seu número em binário ad infinitum não temos um processo geral para fazer isso em um número finito de etapas. A prova de Turing, embora pareça usar o "processo diagonal", na verdade mostra que **sua máquina** (chamada H) não pode calcular seu próprio número, muito menos o número diagonal inteiro (argumento diagonal de Cantor): "A falácia do argumento reside em a suposição de que B [o número diagonal] é computável ". A prova não requer muita matemática.

b) **Segunda prova :** esta é talvez mais familiar aos leitores como o teorema de Rice : "Podemos mostrar ainda que *não pode* *haver nenhuma máquina E que, quando fornecida com o SD [" programa "] de uma máquina M arbitrária, determinará se M alguma vez imprime um determinado símbolo (0 digamos)*".

c) **Terceira prova :** "Correspondendo a cada máquina de computação M, construímos uma fórmula Un (M) e mostramos que, se existe um método geral para determinar se Un (M) é demonstrável, então existe um método geral para determinar se M alguma vez imprime 0 ". A terceira prova requer o uso de lógica formal para provar um primeiro lema, seguido por uma breve prova de palavras do segundo:

   i) Lema 1: Se S1 [símbolo" 0 "] aparecer na fita em alguma configuração completa de M, então Un (M) é demonstrável;

   ii) Lema 2: [O inverso] Se Un (M) for demonstrável, então S1 [símbolo" 0 "] aparece na fita em alguma configuração completa de M.

A prova final é dada por redução ao absurdo que o problema de Hilbert Entscheidungs não pode ter solução e, portanto, não existe uma Máquina de Turing que possa reconhecer as cadeias da linguagem.

9. Podemos descrever o Problema da Parada usando a diagonal de Cantor: mostre o argumento da diagonalização para esse problema.

$$A_{MT}\{< M, w > \ | \, M \text{ é } uma\ MT\ e\ M\ aceita\ w\}$$

Queremos provar que A_{MT}é indecidível.

Suponhamos que A_{MT}é decidível e H é um decisor. Daí:

$$H\,(< M, w >) \ = \{\,aceite\ se\ \ M\ aceita\ w\ e\ rejeite\ se\ M\ \text{não}\ aceita\ w\}$$

Suponhamos D outra MT que usa H para determinar o que M faz com <M> e faz o oposto:

D = “Sobre a entrada <M>, onde M é uma MT:

1. Rode H sobre a entrada <M, <M>>. (Por exemplo, um compilador Java, escrito em Java, que é compilado neste compilador)
2. Dê como saída o oposto do que H dá como saída.

Daí:

$$D\,(< M >) \; = \{\, aceite\ se\ \ M\, não\ aceita\ \ < M > \quad e\ rejeite\ se\ M\ aceita\ \ < M > \}$$

Logo, se D tiver <D> como entrada:

$$D\,(< D >) \; = \{\, aceite\ se\ \ D\, não\ aceita\ \ < D > \quad e\ rejeite\ se\ D\ aceita\ \ < D > \}$$

Daí, por contradição, H e D não podem existir, como queríamos provar.
Descrevendo essa contradição por diagonalização temos que:

A)

	<M1>	<M2>	<M3>	<M4>	...
M1	aceite		aceite		
M2	aceite	aceite	aceite	aceite	
M3					...
M4	aceite	aceite			
...					

A entrada i,j é aceite se Mi aceita <Mj>. Entradas em branco: Mi rejeita ou entra em loop.

B)

	<M1>	<M2>	<M3>	<M4>	...
M1	aceite	rejeite	aceite	rejeite	
M2	aceite	aceite	aceite	aceite	
M3	rejeite	rejeite	rejeite	rejeite	...
M4	aceite	aceite	rejeite	rejeite	
...					

A entrada i,j é o valor de H sobre a entrada <Mi, <Mj>>.

C)

	<M1>	<M2>	<M3>	<M4>	...	<D>
M1	**aceite**	rejeite	aceite	rejeite		aceite
M2	aceite	**aceite**	aceite	aceite		aceite
M3	rejeite	rejeite	**rejeite**	rejeite	...	rejeite
M4	aceite	aceite	rejeite	**rejeite**		aceite
...					...	
D	rejeite	rejeite	aceite	aceite		**?**
...						...

Se D estiver na figura uma contradição ocorre em “?”.

Referências

1. Turing, AM (1937), "On Computable Numbers, with an Application to the Entscheidungsproblem", Proceedings of the London Mathematical Society , 2, 42 (1), pp. 230-65, doi : 10.1112 / plms / s2-42.1. 230
2. Martin Davis (1965), The Undecidable, Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions , Raven Press, Nova York. Outros artigos incluem os de Gödel, Church, Rosser e Kleene.