TEORIA DA COMPUTAÇÃO

Prova 2 – 09/07/2019 – Prof. Marcus Ramos

1ª Questão (1,5 ponto): Alguém propõe um novo tipo de máquina e argumenta que se trata de uma Máquina Universal. Que estratégias você dispõe para tentar confirmar esta argumentação? Como elas funcionariam?

Por meio de evidências internas: mostrando que novas operações e testes para esta máquina podem ser definidos a partir das operações e testes primitivas da máquina fornecida. Este processo deve continuar até que não restem dúvidas de que todo e qualquer algoritmo pode ser transformado num programa para a referida máquina. Não é uma prova formal, mas essencialmente intuitiva.

Por meio de evidências externas: provando que a referida máquina pode simular a Máquina de Turing e também que a Máquina de Turing pode simular a máquina em questão. Alternativamente, pode-se usar outra máquina universal, como é o caso da Máquina Norma, Máquina de Post, Máquina com Pilhas ou Autômato com Duas Pilhas no lugar da Máquina de Turing. Este método pode ser formalizado, mostrando-se primeiro como construir a máquina que simula a outra, e depois provando-se que a máquina que simula possui um poder computacional que é o mínimo igual ao da máquina que está sendo simulada. Depois, inverte-se o sentido da prova.

2ª Questão (1,5 ponto): Considere a Máquina de Turing determinística com fita ilimitada em ambos os sentidos (fita única, trilha única). Cite três extensões e duas limitações que não modificam o poder computacional da mesma.

Extensões:

- Número arbitrário de trilhas;
- Número arbitrário de fitas;
- Não-determinismo;

Restrições:

- Fita limitada (à esquerda ou à direita);
- A máquina nunca escreve brancos na fita.

3ª Questão (1,5 ponto): Por que, na codificação de problemas de decisão como linguagens, apenas as instâncias do problema com resposta “sim” fazem parte da linguagem?

Porque desta forma é possível garantir que, se a linguagem for recursiva, então o problema é decidível (ou solucionável). Sendo recursiva, existe pelo menos uma Máquina de Turing que pára com toda e qualquer entrada e, além disso, essa máquina aceita apenas as cadeias que pertencem à linguagem (no caso, todas as instâncias de reposta afirmativa), rejeitando todas as demais (as instâncias de resposta negativa). Em outras palavras, aceitação ou rejeição indicam que a resposta para a instância do problema é afirmativa ou negativa e sempre há uma resposta para toda instância do problema em questão.

4ª Questão (1,5 ponto): Como provar que um problema é decidível?

Pode-se codificar o problema na forma de uma linguagem e provar que a mesma é recursiva. De maneira alternativa, porém equivalente, pode-se mostrar que existe um algoritmo (procedimento necanizado que sempre pára e fornece uma resposta para toda entrada) que resolve o problema.

5ª Questão (2,0 pontos): Para provar que um problema é totalmente insolúvel é necessário provar que não existe Máquina de Turing que aceita a linguagem que representa uma codificação do mesmo. Explique, de forma resumida e com suas próprias palavras, como é feita a prova de que não existe nenhuma Máquina de Turing que aceita a linguagem L_d.

A linguagem L_d é composta pelas cadeias w sobre o alfabeto $\{0,1\}$ tais que $w \notin L(M)$, onde M é a Máquina de Turing (com alfabeto de entrada também $\{0,1\}$) representada por w. Em outras palavras, $L_d = \{w_i \in \{0,1\}^* | w_i \notin L(M_i)$ e M_i é a Máquina de Turing codificada pela cadeia $w_i\}$.

Basta considerar cada uma das cadeias do conjunto $\{0,1\}^*$ como sendo uma representação da codificação de alguma Máquina de Turing (as cadeias inválidas representam MTs cuja linguagem é vazia). Em seguida, usa-se o Método Diagonal de Cantor para provar que, se L_d for recursivamente enumerável (e portanto parcialmente solucionável), então isto produz uma contradição em relação à máquina que aceita (ou aceitaria L_d). Logo, L_d não pode ser recursivamente enumerável e o problema correspondente é totalmente insolúvel.

6ª Questão (2,0 pontos): Por que o complemento de uma linguagem recursivamente enumerável não-recursiva não pode ser também uma linguagem recursivamente enumerável não-recursiva? Em outras palavras, por que o complemento de um problema parcialmente solucionável e não-solucionável não pode ser um problema do mesmo tipo?

Porque, se isto fosse verdade, então a linguagem e o seu complemento seriam ambas recursivamente enumeráveis e, de acordo com um importante teorema, ela seria também recursiva. Mas isto contradiz a hipótese de que tanto a linguagem quanto o seu complemento são não-recursivas. Logo, não é possível que o complemento de uma linguagem recursivamente enumerável e não-recursiva seja do mesmo tipo.