Exame de Qualificação 08/03/2016

Leila Ribeiro e Luciana S. Buriol

March 28, 2017

Teoria da Computação

Perguntas

1. **(2.5 pontos) Na área de Computação, um conceito fundamental é o da *Máquina Universal*.**

   (a) **Defina a Máquina de Turing Universal, explicando a definição e discutindo os requisitos para que esta função seja possível (por exemplo, o fato do alfabeto de uma Máquina de Turing ser finito é importante para o definição da Máquina Universal?);**

   - Uma Máquina de Turing $M*$ é dita *universal* no que diz respeito a computação de uma função se, toda vez que $M*$ inicia lendo o 1 mais a esquerda de $m+1$ 1s seguido por um caracter branco, e seguido por uma string de n_2+1 1s, seguido por um caracter branco, ..., seguindo por uma string contínua de n_k+1 1s em uma fita cujo restante é vazia, onde m é o número de gödel de uma Máquina de Turing M que computa a função numérico-teórica (parcial) f para $k0$, então $M*$ irar parar realizando a leitura do 1 mais a esquerda de uma sequência de $f(n_1, n_2, ..., n_k)+1$ 1s em uma fita cujo restante é vazia.
   - De maneira mais geral, assim como uma Máquina de Turing convencional, a Máquina de Turing Universal também possui um alfabeto finito de entrada, um alfabeto finito de saída, um estado inicial, uma função de transição. Propriedades que podem ser codificadas de diferentes formas, uma delas é a representação de uma MTD por meio de números de *Euler-Gödel*. Nesse tipo de representação cada um dos possíveis símbolos apresentados pela máquina deve ser codificado por um respectivo número natural.
   - O número de Gödel que codifica a MTD desejada é obtido através da multiplicação da sequência de números primos elevado ao número natural corresponde ao símbolo na função de transição. Entretanto, uma MTD pode possuir mais de um número de Gödel possível, e a maioria dos números naturais não codifica um MTD. Uma MTU pode decodificar e executar qualquer MT, recebendo como entrada o número natural que codifica uma determinada máquina e a sua entrada.
   - As regras válidas para MTD também são válidas para MTU, como é caso da utilização de alfabetos finitos. Fato que permite que a funções de transição também sejam finitas, e por sua vez, computáveis através de uma máquina de Turing.

   (b) **Explique a relevância desta definição para a Ciência da Computação;**

   - A noção de Máquina de Turing Universal foi importante no desenvolvimento da Ciência da Computação. A forma que uma MTU, dadas uma entrada m e n, simula uma operação arbitrária de uma Máquina de Turing M_m em uma entrada n, é sugestiva a noção de plasticidade de hardware, que se refere a capacidade de máquina com arquitetura de propósito geral capaz de imitar outras diversas arquiteturas de máquinas através de um software específico. Além disso, através da utilização de MTU Turing mostrou que é *impossível* construir uma TM para algumas funções, posteriormente provado que tais funções são *não-computáveis*.

(c) **Existem conceitos análogos ao da Máquina de Turing Universal em outros modelos de computação, por exemplo, em Funções Recursivas Parciais? Explique.**

- Uma Máquina de Turing Universal pode calcular qualquer função recursiva parcial, decidir qualquer linguagem recursiva, e aceitar qualquer linguagem recursiva enumerável. De acordo com a tese de CT, os problemas resolvíveis por uma MTU são exatamente os mesmos resolviveis por um algoritmo ou método eficiente de computação. Por estas razões, uma Máquina de Turing Universal serve como um padrão utilizado para comparar sistemas computacionais, e um sistema que pode simular uma Máquina de Turing Universal é chamado de Turing completo.

2. **Disserte sobre *Complexidade de Problemas*, incluindo os seguintes tópicos, entre outros que você julgue importantes (identifique claramente no seu texto onde estes tópicos estão desenvolvidos):**

(a) **Classes de Complexidade;**

- O principal foco de estudo da Teoria da Complexidade é a complexidade intrinseca de problemas computacionais. Dentre os objetivos desta área pode-se mencionar a descoberta da complexidade concreta de um problema, ou seja, quanto tempo e espaço um problema leva para resolvido, e também a identificação de conexões entre as diferentes complexidades apresentadas pelos problemas. Este segundo objetivo, por sua vez, levou a descoberta e definição de diferentes classes de complexidade, tais como as de problemas P, NP, NP-Completos, dentre outros.

(b) **Redução de problemas (defina e dê exemplos);**

-

(c) **Teorema de Cook-Levin (explique intuitivamente o que diz este teorema, bem como sua importância);**

- Teorema de Cook-Levin: estabelece que o problema da satisfatibilidade booleana (SAT) por expressões na forma normal conjuntiva (CNF) é NP-Completo. O problema SAT foi o primeiro da história da computação a ser provado como NP-Completo. A prova foi realizada de forma independente por Cook e Levin, os quais provaram que SAT está em NP e propuseram um problema geral em NP (representando todos os problemas em NP) que foi reduzido a SAT em tempo polinomial. Dessa forma, ambos provaram que todo problema em NP pode ser reduzido a SAT em tempo polinomial.
- O Teorema de Cook-Levin é de extram importância para a Cincência da Computação por vários aspectos. Um dos principais é o fato de ser o percursor das discussões acerca da NP-Completude, conceito que não existia à época. Outro fator importantíssimo é que, por provar o primeiro NP-Completo, o Teorema facilitou muito a inclusão de outros problemas a essa classe. Seguem duas possibilidades de provar que um novo problema é NP-Completo: (1) provar que o problema é tanto NP quanto NP-Hard. Para isso, prove uma possível solução pode ser verificada em tempo polinomial (prova de que está em NP), e também prove que o problema pode ser reduzido a um problema NP-Completo já conhecido (prova indireta que o problema é NP-Hard) ou então mostre que qualquer problema que já pertence ao conjunto de problemas em NP pode ser reduzido à este problema (prova direta que o problema é NP-Hard); (2) utilize a inter-redutibilidade entre os problemas NP-Completos. Para isso escolha um problema NP-Completo X já conhecido, reduza o novo problema a X e, depois, reduza X ao novo problema. Com isso estará provado que o novo problema não é pior do que X, quanto ele não pode ser melhor que X, logo o novo problema também NP-Completo.

(d) **A questão *P = NP?* (o que significa e sua relevância a Ciência da Computação.**

- P vs. NP?: esta é considerada a questão aberta central na Teoria da Complexidade e é também uma questão importante para a ciência como um todo. A questão P vs. NP pode ser vista como uma dúvida relacionada à complexidade de um determinado problema. Sabemos que é possível resolver um problema que pertence a classe P em tempo polinomial, e também sabemos que é possível verificar se uma solução em tempo polinomial para um problema que pertence a classe NP. Caso a firmação seja verdadeira P = NP, isso significa que todos os problemas verificáveis em tempo polinomial, também podem ser resolvidos em tempo polinomial, caso seja falsa P NP, então problemas que estão em NP apenas podem ser resolvidos em tempo super-polinomial. Supondo que P = NP, o impacto no mundo seria enorme, algoritmos de criptografia como SSL, RSA e PGP (que pertencem a NP) possuiriam uma solução em tempo polinomial.