Teoria da Computação

Notas de Aula

Prof ^{a} Patrícia Machado, UASC/UFCG

Introdução

O que é Teoria da Computação?

Teoria da Computação é uma área fundamental da Ciência da Computação que estuda as capacidades e limitações dos computadores. Utilizando princípios matemáticos e modelagem, investiga classes de problemas que podem ser computados em abstrações matemáticas de computadores - modelos computacionais - através de um algoritmo, o quão eficiente podem ser computados e o quanto podem ser computados ou aproximados.

Trata-se de uma área central para a compreensão do que os computadores podem ou não podem fazer, bem como a eficiência com que podem realizar tarefas computacionais.

História

A área teve seu início nas décadas de 1930 e 1940, com os trabalhos pioneiros de *Alan Turing*, *Alonzo Church*, *Kurt Gödel* e outros matemáticos e lógicos. Alguns marcos históricos importantes incluem:

- Tese de *Church-Turing*: Proposta por *Alonzo Church* e *Alan Turing*, esta tese sugere que qualquer função que pode ser computada de forma algorítmica pode ser calculada por uma Máquina de *Turing*.
- Máquina de *Turing*: *Alan Turing* introduziu o conceito de uma máquina teórica que podia simular qualquer algoritmo computacional. Este conceito tornou-se uma base fundamental da teoria da computação.
- Problema da Parada: *Alan Turing* demonstrou que não existe um algoritmo geral para decidir se uma Máquina de *Turing* vai parar ou continuar a computar indefinidamente.

Ramos da Teoria da Computação

- **Autômatos e Linguagens Formais** - Estuda modelos matemáticos de máquinas computacionais (autômatos) e as linguagens formais que elas reconhecem. Esta área fornece a base teórica para a análise e a construção de linguagens de programação, compiladores, e sistemas de processamento de linguagem natural, entre outros.

- **Computabilidade** - *Quais problemas podem ser resolvidos por computadores?* A computabilidade estuda a capacidade dos modelos de computação para resolver problemas e estabelece os limites do que é possível computar. Com conceitos centrais como problemas decidíveis, funções computáveis, máquinas de Turing, e reduções, a computabilidade é fundamental para entender a natureza dos algoritmos e a essência da resolução de problemas na ciência da computação. Descobertas nesta área, baseadas na natureza matemática dos problemas, se aplicam a todos os computadores, independente do porte ou tecnologia.

- **Complexidade Computacional** - *O que faz com que alguns problemas sejam computacionalmente difíceis e outros fáceis?* Estuda os recursos necessários para resolver problemas computacionais. Esses recursos podem incluir tempo (quantidade de passos de computação), espaço (quantidade de memória necessária), e outros recursos computacionais. O objetivo principal da complexidade computacional é classificar problemas de acordo com a dificuldade de resolvê-los e entender as limitações práticas da computação.

Conceitos em um Contexto Prático

Depurar um programa é uma tarefa desafiadora devido a diversas questões fundamentais em Teoria da Computação que dificultam ou tornam indecidível a tarefa de localizar um defeito em um código ou determinar sua correção.

Um dos problemas centrais é o "Halting problem", formulado por Alan Turing, que questiona se é possível determinar se um programa irá terminar sua execução para uma entrada específica. Isso significa que não há um algoritmo geral que possa prever com certeza se um programa terminará ou entrará em um loop infinito para todas as entradas possíveis.

Além do problema de parada, outras questões importantes incluem o "Totality problem", que investiga se uma função termina para qualquer entrada possível, e o "No-input halting problem", que indaga se uma função sem entrada sempre termina.

A equivalência de programas ("Program equivalence") também representa um desafio, pois determinar se duas funções sempre retornam o mesmo valor é uma tarefa indecidível, envolvendo aspectos de lógica e estrutura do código. Em outras palavras, não existe um algoritmo geral que possa determinar de forma automática e correta se dois programas são equivalentes para todas as entradas possíveis.

Problemas práticos como variáveis não inicializadas ("Uninitialized variables") e eliminação de código morto ("Dead-code elimination") também contribuem para a dificuldade na depuração. Garantir que todas as variáveis sejam corretamente inicializadas antes de seu uso é crucial para evitar comportamentos inesperados, enquanto a eliminação de código que nunca será executado requer análise sofisticada do fluxo de controle do programa.

Portanto, depurar um programa não se resume apenas a corrigir erros óbvios, mas também a lidar com questões profundas e teóricas que limitam a capacidade de verificar completamente a correção de um software de forma automática e geral.

Máquinas Abstratas e Linguagens Formais

Máquinas abstratas são modelos teóricos que representam computadores idealizados, desprovidos de características específicas de implementação física como arquitetura de hardware real. Esses modelos são utilizados na teoria da computação para estudar a capacidade computacional, a complexidade de algoritmos e a solução de problemas computacionais de forma geral. Como exemplo temos os Autômatos e a Máquina de Turing.

Linguagens Formais são conjuntos de cadeias (*strings*) construídas a partir de um alfabeto e definidas por regras sintáticas específicas. Elas são classificadas em diferentes tipos com base em sua complexidade e no tipo de gramática que as gera. Em Teoria da Computação, linguagens são utilizadas para representar problemas computacionais.

Por que estudar Teoria da Computação?

Teoria da Computação oferece a base teórica para compreendermos como computadores funcionam, os limites da computação e as soluções para problemas algorítmicos. Seu estudo contribui para:

- A concepção de algoritmos eficientes, desenvolvimento de software robusto e compreensão de avanços tecnológicos, propiciando inovações em áreas como computação quântica, bioinformática, realidade virtual e aumentada, entre outras tecnologias emergentes;
- A compreensão dos limites da computação que viabiliza a investigação de algoritmos e soluções em áreas como inteligência artificial, segurança de dados e ciência de dados (*pattern matching*, *cryptography*, *data compression*, ...). Além da exploração de novas fronteiras nestas áreas;
- A construção de conceitos relacionados a concepção de linguagens de programação como gramáticas formais e autômatos, fundamentais para entender a sintaxe e a semântica das linguagens de programação. Isso é essencial para o projeto de compiladores e interpretares, que traduzem linguagens de alto nível para código executável;
- O desenvolvimento da Criptografia que visa garantir a confidencialidade e a integridade dos dados. Conceitos como criptografia de chave pública, *hashing* e assinaturas digitais são fundamentais para a segurança da informação em ambientes digitais.

$x + y$

Referências

- Michael Sipser. Introduction to the Theory of Computation, Third Edition, CENGAGE Learning, 2013. (Introdução à teoria da computação. Tradução da 1a/2a Edição, disponível na biblioteca) - Livro-texto (Seção 0.1)
- Marino H. Catarino. Teoria da Computação. 1a Edição. Editora Freitas Bastos, 2023. (Cap 1)
- Why Theory of Computation?
- Theory of Compuation - Wikipedia
- Theory of Computation: a brief introduction