Máquinas Universais

Teoria da Computação

INF05501

Algoritmo

- Termo **intuitivamente** usado para designar a solução de um problema

- Forma de descrever se determinada **propriedade** verifica-se ou não **para uma dada classe de entrada**

- Solucionabilidade de um problema corresponde à **existência de um algoritmo que o resolva**

Noção Intuitiva de Algoritmo

- **Descrição** deve ser
  - Finita
  - Não-ambígua
- Deve consistir de **passos**
  - Discretos
  - Executáveis mecanicamente
  - Realizáveis em um tempo finito

Noção Intuitiva de Algoritmo (cont.)

- **Não há restrições teóricas**
- Porém, limitações de **tempo** ou de **espaço** podem **impedir aplicação prática**
- Nosso estudo será restrito aos **algoritmos naturais**, ou seja, definidos sobre $\mathbb{N}$
- Qualquer **conjunto contável** pode ser **equivalente ao dos naturais**, através de uma **codificação**

De um Algoritmo Para um Programa

- **Conceito de programa**, como definido anteriormente, **satisfaz a noção intuitiva de algoritmo**

- Desta forma, um programa é uma implementação de um algoritmo

- Se podemos implementar um algoritmo através de um programa, então podemos **executá-lo usando este programa**

- No entanto, precisamos definir a **máquina** na qual tal programa deverá ser executado

Máquina para Algoritmos

- Uma máquina a ser considerada para a formalização de algoritmos deve ser:
  - Simples
    - Para possibilitar estudos de propriedades através de **abstrações**
    - Para permitir estabelecer conclusões gerais sobre a **classe de funções computáveis**
  - Poderosa
    - Para ser capaz de **simular qualquer máquina** (real ou teórica)
    - Para garantir que **resultados obtidos** sejam **válidos mesmo para modelos que possam ter mais recursos**

Máquina Universal

- Se houver um máquina tal como descrito, na qual seja **possível representar qualquer algoritmo como um programa para esta máquina**, então ela é dita uma máquina universal

- Como saber/provar se uma máquina é universal?
  - Evidência interna: Demonstração de que **qualquer extensão** das capacidades da máquina universal proposta computa, no máximo, a mesma classe de funções; ou seja, **não aumenta o seu poder computacional**
  - Evidência externa: Exame de **outros modelos** que definem a noção de algoritmo, juntamente com a prova de que **são, no máximo, computacionalmente equivalentes**

Exemplos de Máquinas Universais

- Máquina Norma (*Number TheOretic Register MAchine*)
- Máquina de Turing

Exemplos de Máquinas Universais: Máquina Norma

- Proposta por **Richard Bird** (1976)
- É uma **máquina de registradores**
  - Sua memória é composta por um **conjunto infinito de registradores naturais**
  - Possui 3 operações:
    - **Incremento** (adição de um)
    - **Decremento** (subtração de um)
    - **Teste se valor armazenado é zero**

Exemplos de Máquinas Universais: Máquina de Turing

- Proposta por **Alan Turing** (1936), é o **modelo mais utilizado para a formalização de algoritmos**

- Baseia-se na ideia de uma pessoa realizando cálculos utilizando um **instrumento de escrita** e um **apagador**

- Modelo formal constitui-se de:
  - Uma fita, usada como entrada, saída e memória
  - Uma unidade de controle
  - Um programa

Hipótese de Church

- Apresentada por Alonzo Church em 1936
- Afirma que qualquer função computável pode ser processada por uma Máquina de Turing
- Isto é, para qualquer função computável, **existe um algoritmo expresso na forma de Máquina de Turing** capaz de processá-la
- Desta forma, a Hipótese de Church determina que a Máquina de Turing é o mais genérico dispositivo de computação existente

Hipótese de Church (cont.)

- Como a noção intuitiva de algoritmo não é matematicamente precisa, é **impossível formalizar uma demonstração**

- Entretanto, **todas as evidências internas e externas imaginadas foram sempre verificadas**

- Também verificou-se que os **demais modelos propostos**, bem como **qualquer extensão** de suas capacidades, possuem, **no máximo, a mesma capacidade computacional** da Máquina Turing

Problema da Codificação de Conjuntos Estruturados

- Como visto anteriormente, por simplicidade, **trabalharemos com o conjunto** $\mathbb{N}$

- No entanto, os **tipos de dados existentes não se restringem ao naturais**

- Disto, surge o problema da codificação de conjuntos estruturados, que diz respeito a **como representar tipos estruturados como números naturais**

Formalização do Problema

Para um dado conjunto estruturado X, quer-se definir uma função injetora $c : X \to \mathbb{N}$, onde, $\forall x, y \in X$:

$$\text{se } c(x) = c(y), \text{ então } x = y$$

sendo $c(x)$ e $c(y)$ números naturais que codificam x e y, respectivamente

Exemplo de Codificação

- Suponha que queiramos **codificar, de forma unívoca, elementos de** $\mathbb{N}^n$ **como números naturais**

- Ou seja, queremos obter uma função injetora $c : \mathbb{N}^n \rightarrow \mathbb{N}$

- Como poderíamos obter esta função?

Exemplo de Codificação (cont.)

- Pelo Teorema Fundamental da Aritmética, **todo número natural pode ser univocamente decomposto em seus fatores primos**

- Suponha os n primeiros números primos denotados por $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, e assim por diante

- Assim, podemos definir a seguinte codificação unívoca, supondo que $(x_1, x_2, ..., x_n) \in \mathbb{N}^n$ e que o símbolo $\bullet$ denota a operação de multiplicação:

$$c(x_1, x_2, ..., x_n) = p_1^{x_1} \bullet p_2^{x_2} \bullet ... \bullet p_n^{x_n}$$

Exemplo de Codificação (cont.)

- Note que essa codificação **não constitui uma função bijetora**
- Ou seja, nem todo número natural corresponde a uma n-upla
- Entretanto, todo número natural decomponível nos n primeiros números primos corresponde a uma n-upla

Exemplo de Codificação 2

- Um **programa monolítico** também pode ser **codificado como um número natural**

- Suponha um programa monolítico $P = (I, r_0)$, com m **instruções rotuladas**, onde $\{F_1, F_2, ..., F_f\}$ e $\{T_1, T2, ..., T_t\}$ são os correspondentes **conjuntos de identificadores de operações e testes**, respectivamente

- Seja $P' = (I, 1)$ como P, exceto pelos **rótulos**, os quais são **renomeados como números naturais**, onde 1 **é o rótulo inicial**, e o 0 **único rótulo final (se existir)**

Exemplo de Codificação 2 (cont.)

- Assim, uma instrução rotulada pode ser de **uma das duas seguintes formas**:

  - Operação: r_1: `faça` F_k `vá_para` r_2
  - Teste: r_1: `se` T_k `então vá_para` r_2 `senão vá_para` r_3

Exemplo de Codificação 2 (cont.)

- **Cada instrução rotulada** pode ser denotada por uma quádrupla ordenada, onde a **primeira componente identifica o tipo** da instrução:
  - Operação: $(0, k, r_2, r_2)$
  - Teste: $(1, k, r_2, r_3)$

Exemplo de Codificação 2

- **Usando a codificação do exemplo anterior**, o programa monolítico P', visto como quádruplas ordenadas pode ser codificado como segue:

  – **Cada quádrupla** (instrução rotulada) é codificada como um **número natural**

  – A m**-upla correspondente ao programa monolítico** P' é codificada como um **número natural**

- Note que esta codificação também **não constitui uma função bijetora**

Exemplo de Codificação 2

- A partir da codificação proposta, pode-se **recuperar um programa original a partir de um número natural**

- Por exemplo: $p = (2^{150}) \bullet (3^{105})$

- Quantas são as instruções rotuladas?

Exemplo de Codificação 2

- Por exemplo: $p = (2^{150}) \bullet (3^{105})$

- Disto, identifica-se que o programa possui **duas instruções rotuladas**: uma representada pelo número 150 e outra, pelo número 105

- Decompondo estes números em seus 4 primeiros fatores primos:

$$150 = 2^1 \bullet 3^1 \bullet 5^2 \bullet 7^0$$

$$105 = 2^0 \bullet 3^1 \bullet 5^1 \bullet 7^1$$

Exemplo de Codificação 2

- **Pelos expoentes**, identificam-se as quádruplas que codificam as **instruções rotuladas**:

$$(1, 1, 2, 0) \text{ e } (0, 1, 1, 1)$$

- Estas quádruplas codificam quais instruções rotuladas?

Exemplo de Codificação 2

- **Pelos expoentes**, identificam-se as quádruplas que codificam as **instruções rotuladas**:

$$(1, 1, 2, 0) \text{ e } (0, 1, 1, 1)$$

- Estas quádruplas codificam quais instruções rotuladas?

1 : se T_1 então vá_para 2 senão 0
2 : faça F_1 vá_para 1

Exercícios

1. Codifique o seguinte programa monolítico usando um número natural:

```
1 : faça F vá_para 2
2 : se T então vá_para 3 senão vá_para 5
3 : faça G vá_para 4
4 : se T então vá_para 1 senão vá_para 0
5 : faça F vá_para 6
6 : se T então vá_para 7 senão vá_para 2
7 : faça G vá_para 8
8 : se T então vá_para 6 senão vá_para 0
```

2. Recupere o programa monolítico codificado pelo número $2^{3675} \bullet 3^{42}$