Decidibilidade

Juliana Kaizer Vizzotto

Universidade Federal de Santa Maria

Disciplina de Teoria da Computação/Slides baseados no livro:
Introdução a Teoria da Computação. Michael Sipser.

Roteiro

- Descrevendo MTs de alto-nível
- Decidibilidade

Descrevendo MTs de alto-nível

- Chegamos a um momento decisivo no estudo da teoria da computação.
- Nosso verdadeiro foco a partir de agora é algoritmos.
- MT server como um modelo preciso para a definição de algoritmo.
- Vamos padronizar uma forma de alto-nível para descrever algoritmos na MT, a partir de agora.
- Vamos usar a língua natural para descrever os algoritmos, ignorando os detalhes de implementação. Nesse nível não precisamos mencionar como a máquina administra sua fita ou sua cabeça de leitura-escrita.

Descrevendo MTs de alto-nível

- A entrada para uma MT é sempre uma cadeia.
- Se desejamos fornecer como entrada um objeto que não uma cadeia, primeiro temos que representar esse objeto como uma cadeia.
- Cadeias podem facilmente representar polinômios, grafos, gramáticas, autômatos, etc.
- Vamos usar a notação $\langle O \rangle$ para a codificação de um objeto O.
- Se tivermos vários objetos $O_1, O_2, ..., O_n$ denotamos sua codificação de em uma única cadeia $\langle O_1, O_2, ..., O_n \rangle$.
- A codificação propriamente dita pode ser feita de muitas formas razoáveis.

Descrevendo MTs de alto-nível

- Quebramos o algoritmo em estágios, cada um possivelmente envolvendo muitos passos **individuais** da computação na MT.
- Indicamos a estrutura em bloco do algoritmo com mais identação.
- A primeira linha do algoritmo descreve a entrada para a máquina.

Descrevendo MTs de alto-nível: Exemplo

- Seja A a linguagem consistindo em todas as cadeias representando grafos não-direcionados que são conexos.
- Lembre-se que um grafo é **conexo** se todo nó pode ser atingido a partir de cada um dos outros nós passando pelas arestas do grafo. Escrevemos:

$$A = \{\langle G \rangle \| G \text{ eh um grafo nao-direcionado conexo}\}$$

- Segue uma descrição de alto-nível de uma MT, que **decide** A.

Descrevendo MTs de alto-nível: Exemplo

- M = "Sobre a entrada $\langle G \rangle$, a codificação de um grafo G:
  1. Selecione o primeiro nó de G e marque-o.
  2. Repita o seguinte estágio até que nenhum novo nó seja marcado:
  3. Para cada nó em G, marque-o, se ele estiver ligado por uma aresta a um nó que já está marcado.
  4. Faça uma varredura em todos os nós de G para determinar se eles estão todos marcados. Se estiverem, *aceite*, caso contrário, *rejeite*."

Descrevendo MTs de alto-nível: Exemplo

- Primeiramente, temos que entender como $\langle G \rangle$ codifica o grafo G como uma cadeia.
- Considere uma codificação que é uma lista dos nós de G seguida de uma lista das arestas de G.
- Cada nó é um número decimal, e cada aresta é o par de números decimais que representam os nós nas duas extremidades da aresta.
- Por exemplo:

$$\langle G \rangle = (1, 2, 3, 4), ((1, 2), (2, 3), (3, 1), (1, 4))$$

Descrevendo MTs de alto-nível: Exemplo

- Quando M recebe a entrada $\langle G \rangle$, ela primeiro faz um teste para verificar se a entrada é a codificação apropriada de um grafo.
- Para fazer isso, M faz uma varredura na fita para ter certeza de que existem duas listas e que elas estão na forma apropriada.
- M então verifica diversas coisas:
  1. a lista de nós não deve conter repetições; (pense em uma MT que verifica a distinção entre elementos)
  2. todo nó da lista de arestas deve também aparecer na lista de nós.
- Se a entrada passa nesses testes , ela é a codificação de algum grafo G.

Descrevendo MTs de alto-nível: Exemplo

- Para o estágio 1, M marca o primeiro nó com um ponto no dígito mais à esquerda.
- Para o estágio 2, M faz uma varredura na lista de nós, para encontrar um nó não marcado n_1 e o marca-o colocando um sinal diferente (sublinhado).
- Então M faz uma varredura na lista novamente para encontrar um nó, n_2, marcado com um ponto e o subinha-o também.
- Agora M varre a lista de arestas e testa se os dois nós n_1, n_2 são aqueles aparecendo nessa aresta.
- Se eles o são, M marca n_1 com um ponto, remove a marca de sublinhar e continua a partir do estágio 2.

Descrevendo MTs de alto-nível: Exemplo

- Se eles não o são, M faz a verificação para a próxima aresta na lista.
- Se não existem arestas, $\{n_1, n_2\}$ não é uma aresta de G.
- Então, M testa um próximo nó marcado com um ponto.
- Se não existem mais nós marcados com um ponto, n_1 não está ligado a nenhum nó marcado com um ponto.
- Então, M põe a marca de sublinhado no próximo nó da lista e repete esse procedimento.
- Se não existem mais nós sem a marca de um ponto, M não foi capaz de encontrar um novo nó para marcar com ponto, portante ela segue para o passo 4.
- Para o estágio 4, M varre a lista de nós para determinar se todos estão marcados com um ponto.

Descrevendo MTs de alto-nível: Exemplo

- Se estiverem, ela entra no estado de aceitação;
- Caso contrário, ela entra no estado de rejeição.
- Com isso está completa a descrição da MT.

Decidibilidade

- Trabalhamos até agora com a MT como um modelo de um computador de propósito geral;
- Definimos a noção de algoritmo em termos de MTs por meio da **Tese de Church-Turing**.
- Lembrando: Linguagem Turing-reconhecível: a coleção de cadeias que M aceita é a linguagem de M, ou a linguagem reconhecida por M.
- Chame de linguagem Turing-reconhecívelm se alguma máquina de Turing a reconhece. ou Linguagem recursivamente-enumerável.

Decidibilidade

- Ao rodar uma MT, três resultados são possíveis: *aceitar*, *rejeitar* ou entrar em *loop*.
- Por entrar em loop queremos dizer que a máquina simplesmente não pára.
- MTs que param sobre todas as entradas, nunca entram em loop.
- Essas máquinas são chamadas de Decisores, pois elas sempre tomam a decisão de aceitar ou rejeitar.
- Chame uma linguagem Turing-decidível ou simplesmente decidível se alguma MT a decide. Ou também chamada de linguagem recursiva em alguns livros-texto.
- Lembre que toda a linguaem decidível é Turing-reconhecível.
- Entretanto, existem linguagens que são Turing-reconhecíveis, porém não decidíveis.

Decidibilidade

- Vamos começar a partir de agora a investigar o poder de algoritmos para resolver problemas.

- Vamos trabalhar com certos problemas que podem ser resolvidos algoritmicamente e outros que não podem.

Decidibilidade: motivação

- **Por que estudar a insolubilidade de problemas?**
  1. Saber quando um problema é algoritmicamente insolúvel é útil, pois você se dá conta que o problema deve ser simplificado ou modificado antes que possa encontrar uma solução algoritmica.
  2. Como qualquer ferramenta, o computador tem capacidades e limitações que devem ser reconhecidas, caso se queira que eles sejam bem usados.
  3. Razão cultural: mesmo se você lidas com problemas que sejam claramente solúveis, um vislumbre do insolúvel pode estimular sua imaginação e ajudá-lo a ganhar uma perspectiva importante sobre a computação.

Decidibilidade: Linguagens Decidíveis

- Veremos exemplos de linguagens que são decidíveis por algoritmos.
- Focamos em linguagens concernentes a autômatos e gramáticas.
- **Linguagem Turing Decidível: linguagem aceita por alguma MT que sempre pára (para qualquer entrada)**

Decidibilidade: problemas decidíveis concernentes a Linguagens Regulares

- **Problema da aceitação** para um AFD de testar se um autômato finito determinístico específico aceita uma dada cadeia pode ser expresso por uma linguagem A_{AFD}.

- Essa linguagem contém as codificações de todos os AFDs juntamente com cadeias que os *AFDs* aceitam:

  $A_{AFD} = \{\langle B, w\rangle | B \text{ eh um AFD que aceita a cadeia de entrada } w\}$.

Decidibilidade: problemas decidíveis concernentes a Linguagens Regulares

- O problema de se testar se um AFD, B, aceita uma entrada w é o mesmo que o problema de se testar se $\langle B, w \rangle$ é um membro da linguagem A_{AFD}.

- **Mostrar que a linguagem é decidível é o mesmo que mostrar que o problema computacional é decidível**.

Decidibilidade: problemas decidíveis concernentes a Linguagens Regulares

- **Teorema**: A_{AFD} é uma linguagem decidível.
- **Prova**: Simplesmente precisamos apresentar uma MT, M, que decide A_{AFD}.
- "Sobre a entrada $\langle B, w \rangle$, onde B é um AFD, e w é uma cadeia:
  1. Simule B sobre a entrada w.
  2. Se a simulação termina em um estado de aceitação, *aceite*. Se ela termina em um estado de não-aceitação, *rejeite*."
- **Exercício:** Escreva um programa em alguma linguagem de programação, para realizar essa simulação.

Decidibilidade: problemas decidíveis concernentes a Linguagens Regulares

- A entrada $\langle B, w \rangle$ é uma representação de um AFD junto com sua entrada w. Uma representação razoável de B é simplesmente uma lista de seus 5 componentes.
- Quando M recebe a sua entrada, M primeiro determina se ela representa apropriadamente um AFD, B, e uma cadeia w. Se não, M já rejeita.
- Então, M realiza a simulação diretamente. Ela mantém o registro do estado atual de B e da posição atual de B na entrada w escrevendo essa informação na sua fita (como??).

Decidibilidade: problemas decidíveis concernentes a Linguagens Regulares

- Inicialmente, o estado atual de B é q_1 e a posição atual de B sobre a entrada, w, é o símbolo mais à esquerda de w.
- Os estados e a posição são atualizados conforme a função de transição do autômato.
- Quando M termina de processar o último símbolo de w, M aceita a entrada se B estiver em um estado de aceitação.
- M rejeita a entrada, se B estiver em um estado de não-aceitação.

Decidibilidade: problemas decidíveis concernentes a Linguagens Regulares

- Podemos provar um teorema semelhante para autômatos finitos não-determinísticos. Seja:

  $A_{AFN} = \{\langle B, w\rangle | B \text{ eh um AFN que aceita a cadeia de entrada } w\}$

- **Teorema**: A_{AFN} é uma linguagem decidível.

- **Exercício**: Prove esse teorema.