Computabilidade e Decidibilidade

Prof^a Jerusa Marchi

INE - UFSC

jerusa.marchi@ufsc.br

Máquinas de Turing

- Até o momento, temos a noção intuitiva de que máquinas de Turing podem ser utilizadas para reconhecer e/ou decidir qualquer linguagem
- Sabemos que qualquer problema computacional pode ser visto como um problema de linguagem
- Também temos a noção de que máquinas de Turing são o modelo mais poderoso de computação
    - Não servindo apenas para aceitar ou rejeitar, mas para produzir saídas
- Estamos prontos, portanto, para definir a noção de algoritmo e definir a tese de Church-Turing

Algoritmos

- De forma intuitiva, um algoritmo é uma coleção de instruções simples para realizar alguma tarefa
- Esta noção intuitiva foi utilizada na matemática durante muitos anos, mas somente no século XX a noção de algoritmo foi definida com precisão

Algoritmos

- Dentre as propriedades de um algoritmo encontram-se:
  - Discretude - um algoritmo é um procedimento que é executado em tempo discreto
  - Exatidão - a configuração atual é obtida pela aplicação de um passo (comando) à configuração anterior
  - Elementaridade dos passos - o passo que determina a configuração deve ser simples e local
  - Massividade - a entrada do algoritmo pode ser escolhida a partir de um conjunto potencialmente infinito
  - Duplicabilidade - a aplicação de uma mesma entrada deve gerar uma mesma saída.
- Contudo, a **decidibilidade** é a propriedade que diferencia algoritmos de procedimentos

Algoritmos

- Exemplo: Verificar se um polinômio possui raizes inteiras

$$6x^3yz^2 + 3xy^2 - x^3 - 10$$

- Uma raiz de um polinômio é uma atribuição de valores às suas variáveis de modo que o valor resultante seja 0.
- O polinômio acima, tem uma raiz inteira em $x = 5$, $y = 3$ e $z = 0$

Algoritmos

- 10^o problema de Hilbert: Seria possível conceber um algoritmo para encontrar as raizes inteiras de um dado polinômio?
  - Descrito como um problema de linguagem:

$$D = \{p \mid p \text{ é um polinômio com raiz inteira}\}$$

  - D é Turing-decidível? D é Turing-Reconhecível?
    - Demonstrado indecidível por Yuri Matijasevic em 1970.
    - porém facilmente demonstrado Turing-Reconhecível - Tendo como entrada o polinômio p, para cada conjunto possível de valores para as variáveis de p ($\{0, 1, -1, 2, -2, \cdots\}$), calcule o valor de p utilizando o conjunto. Se em algum ponto o valor de p resultar em 0, aceite.

Tese de Church-Turing

- Tudo o que é **computável** é computável por uma máquina de Turing
  - Máquinas de Turing que param aceitando ou rejeitando a entrada
  - Máquinas de Turing que podem rodar para sempre com entradas que não são aceitas
- A noção intuitiva de algoritmo corresponde ao conceito do que é possível computar com Máquinas de Turing

Tese de Church-Turing

- Escrito de outra forma:

  Todo o processo efetivo (i.e, para o qual existe algoritmo ou um processo mecânico de computação) pode ser efetuado por meio de uma Máquina de Turing

- Tomemos a Máquina de Turing que pára em respostas a todas as entradas como sendo a noção formal precisa correspondente à intuitiva idéia de algoritmo
  - Máquina de Turing vista como um algoritmo
  - O que é **efetivamente** computável (ou seja o que é decidível)?

UFSC

Decidibilidade

- Antes de definir máquinas de Turing como programas (algoritmos) precisamos definir um "hardware" que execute tais programas
  - Máquinas de Turing Universais

Máquina de Turing Universal

- Uma Máquina de Turing que pode ser programada, podendo assim solucionar “qualquer” problema passível de resolução através de uma Máquina de Turing
- A Máquina de Turing Universal recebe como entrada uma máquina de Turing e a entrada a computar
- A entrada e a descrição da máquina são “programadas” em uma linguagem reconhecida pela Máquina de Turing Universal

- Exemplo:
  - $M = (K, \Sigma, \delta, s, \{h\})$ onde:
    - $K = \{s, q, h\}$
    - $\Sigma = \{\sqcup, \triangleright, a\}$
    - δ :

estado	símbolo	δ
s	a	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\triangleright$	$(s, \rightarrow)$
q	a	(s, a)
q	$\sqcup$	$(s, \rightarrow)$
q	$\triangleright$	$(q, \rightarrow)$

Convensão:

- Um estado da máquina de Turing deverá ser da forma $\{q\}\{0,1\}^*$
- Um símbolo de fita será representado como uma cadeia $\{a\}\{0,1\}^*$
- Seja $M = (K, \Sigma, \delta, s, \{H\})$ uma Máquina de Turing
  - Sejam i e j os menores inteiros, tais que $2^i \geq |K|$ e $2^j \geq |\Sigma| + 2$ ($\leftarrow$ e $\rightarrow$)
    - Cada estado em K será representado como um símbolo q seguido de uma cadeia binária de comprimento i
    - Cada símbolo em Σ será representado como o símbolo a seguido de uma cadeia binária de comprimento j

Convensão:

- Para representar $\sqcup$, $\rhd$, $\rightarrow$ e $\leftarrow$ serão utilizados os quatro menores símbolos em ordem lexicográfica
  - $\sqcup$ - $a0^j$
  - $\rhd$ - $a0^{j-1}1$
  - $\leftarrow$ - $a0^{j-2}10$
  - $\rightarrow$ - $a0^{j-2}11$

Convensão:

- O estado inicial da Máquina será sempre representado como o primeiro estado lexicográfico $q0^i$
- A tabela de transição δ da representação “M” da Máquina de Turing consiste em uma sequência de cadeias da forma

$$(q, a, p, b)$$

onde q e p são representações de estados e a e b representações de símbolos

Convensão:

- As quádruplas são relacionadas em ordem lexicográfica crescente, começando com $\delta(s, \sqcup)$
- O conjunto H é determinado indiretamente pela não ocorrência de seus estados como primeiros componentes em qualquer quádrupla de "M"
  - Se M decide uma linguagem ($H = \{q_{accept}, q_{reject}\}$), q_{accept} será, lexicograficamente, o menor dos dois estados de parada

Máquina de Turing Universal

- Qualquer Máquina de Turing pode ser representada segundo essa convenção
- O mesmo método será utilizado para representar quaisquer cadeias sobre o alfabeto da máquina de Turing
  - para simplificar $\Gamma \subset \Sigma$
- a representação de w, será notacionada “w”

- Exemplo:
  - $M = (K, \Sigma, \delta, s, \{h\})$ onde:
    - $K = \{s, q, h\}$
    - $\Sigma = \{\sqcup, \rhd, a\}$
    - δ :

estado	símbolo	δ
s	a	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\rhd$	$(s, \rightarrow)$
q	a	(s, a)
q	$\sqcup$	$(s, \rightarrow)$
q	$\rhd$	$(q, \rightarrow)$

Máquina de Turing Universal

- $|K| = 3$ então $i = 2$, pois $2^2 \geq 3$
- $|\Sigma| = 3$ então $j = 3$ pois $2^3 \geq 3 + 2$
- Estados e símbolos ficam assim representados

estado/símbolo	representação
s	$q00$
q	$q01$
h	$q11$
$\sqcup$	$a000$
$\triangleright$	$a001$
$\leftarrow$	$a010$
$\rightarrow$	$a011$
a	$a100$

- Exemplo:
  - A representação da cadeia $\triangleright aaa\sqcup$ será

$$"\triangleright aaa\sqcup" = a001a100a100a100a000$$

  - A representação “M” da máquina de Turing M será

$$\begin{aligned}"M" = \; & (q00, a100, q01, a000), (q00, a000, q11, a000),\\ & (q00, a001, q00, a011), (q01, a100, q00, a011),\\ & (q01, a000, q00, a011), (q01, a001, q01, a011)\end{aligned}$$

Máquina de Turing Universal

- Apresentada a representação de qualquer máquina de Turing como "algoritmo", podemos agora introduzir uma Máquina de Turing Universal (*U*)
  - *U* recebe como entrada
    - uma representação "M" de uma certa máquina M
    - uma representação "w" de uma dada sentença de entrada w

- *U* pára em resposta a entrada “M”“w” se e somente se M pára em resposta à entrada w

$$U(''M'',''w'') = ''M(w)''$$

- A máquina *U* é uma MT de 3 fitas que opera da seguinte forma:
  - A cadeia “M”“w” é gravada em sua primeira fita
  - U move “M” para a segunda fita
  - U desloca “w” para a esquerda na primeira fita
  - U grava na terceira fita a codificação do estado inicial *s* de “M”, que é sempre $q0^i$ (os valores de *i* e *j* são obtidos por U via inspeção de “M”)

Continuação:

- U simula os passos de computação de M como segue:
    - percorre a segunda fita até encontrar uma quádrupla cujo primeiro componente corresponda ao estado codificado gravado na terceira fita e cujo segundo componente corresponda ao símbolo codificado apontado na primeira fita
    - Uma vez encontrada esta quádrupla, *U* altera o estado corrente, subtituindo-o pelo terceiro componente da quádrupla e realiza, em sua primeira fita, a ação especificada pelo quarto componente

Continuação:

- Se o quarto componente
  - codifica um símbolo do alfabeto da fita de M, esse símbolo é gravado na primeira fita, substituindo o símbolo corrente
  - for $a0^j10$ ($\leftarrow$), U move o primeiro cabeçote para o primeiro símbolo a à esquerda
  - for $a0^j11$ ($\rightarrow$), U move o primeiro cabeçote para o primeiro símbolo a à direita

Continuação:

- se um $\sqcup$ for encontrado, U o converte para a representação $a0^j$
- se em algum passo a combinação de estado/símbolo não for encontrada na segunda fita, isso significa que o estado corrente é um estado de parada
- U também pára em um estado de parada conveniente

- Antes de demonstrar que algums problemas ou linguagem são **indecidíveis**, vamos ver algumas provas de decidibilidade
  - sobre Linguagens Regulares
    - Testar se um AFD aceita uma cadeia de entrada.
    - Testar se um AFND aceita uma cadeia de entrada.
    - Testar a vacuidade para a linguagem de um AFD
    - Determinar se dois AFD são equivalentes, ou seja reconhecem a mesma linguagem

Linguagens Decidíveis

- sobre Linguagens Livres de Contexto
  - Testar se uma GLC gera uma cadeia de entrada.
  - Testar a vacuidade para a linguagem de uma GLC
  - Determinar se duas GLC são equivalentes, ou seja geram a mesma linguagem

Testar se um AFD aceita uma cadeia de entrada.

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

- A_{AFD} é uma linguagem decidível
- **Idéia da Prova**: Apresentar uma MT M que decide A_{AFD}.
  - M = "Sobre a entrada $\langle B, w\rangle$, onde B é um AFD e w 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*.

Linguagens Decidíveis

- A_{AFD} é uma linguagem decidível
- **Prova**: A entrada $\langle B, w \rangle$ é uma representação do AFD B e uma cadeia w. Uma representação de B consiste em uma lista de seus 5 componentes K, Σ, δ, q_0 e F. Ao receber a entrada $\langle B, w \rangle$, M verifica se esta representa apropriadamente um AFD B e uma cadeia w. Se não, M rejeita.
  Então M realiza a simulação diretamente. M mantém um registro do estado atual de B e da posição atual de B na entrada w escrevendo esta informação em sua fita. Inicialmente o estado atual de B é q_0 e a sua posição na entrada é o símbolo mais à esquerda de w. Os estados e a posição são atualizados segundo a função de transição especificada em δ. Quando M termina de processar o último símbolo de w, M aceita se B estiver em um estado pertencente a F. Caso contrário, rejeita.

Testar se um AFND aceita uma cadeia de entrada.

$$A_{AFND} = \{\langle B, w\rangle \mid B \text{ é um AFND que aceita a cadeia de entrada } w\}$$

- A_{AFND} é uma linguagem decidível
- **Idéia da Prova**: Apresentar uma MT N que decide A_{AFND}.
  - N = "Sobre a entrada $\langle B, w\rangle$, onde B é um AFND e w uma cadeia:
    1. Converta AFND B para um AFD equivalente C
    2. Execute a MT M definida na prova anterior para a entrada $\langle C, w\rangle$
    3. Se M aceita, *aceite*; caso contrário, *rejeite*.

Testar a vacuidade para a linguagem de um AFD

$$V_{AFD} = \{\langle A \rangle \mid A \text{ é um AFD e } L(A) = \emptyset\}$$

- V_{AFD} é uma linguagem decidível
- **Idéia da Prova**: Um AFD aceita alguma cadeia sse é possível atingir um estado de aceitação a partir do estado inicial, passando pelas setas do AFD. Para testar essa condição, podemos projetar uma MT T que usa um algortimo de marcação.
  - T = "Sobre a entrada $\langle A \rangle$, onde A é um AFD:
    1. Marque o estado inicial de A
    2. Repita até que nenhum estado novo seja marcado
    3. Marque qualquer estado que tenha transição chegando nele a partir de qualquer estado que já está marcado
    4. Se nenhum estado de aceitação estiver marcado, *aceite*; caso contrário, *rejeite*.

Determinar se dois AFD são equivalentes, ou seja reconhecem a mesma linguagem

$$EQ_{AFD} = \{\langle A, B\rangle \mid A \text{ e } B \text{ são AFD e } L(A) = L(B)\}$$

- EQ_{AFD} é uma linguagem decidível
- **Idéia da Prova**: Construir um novo AFD C a partir de A e B, tal que C aceita somente aquelas cadeias que são aceitas ou por A ou por B, mas não por ambos.

  $$L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$$

  Essa expressão é chamada de diferença simétrica. Se $L(A) = L(B)$ então C não aceitará nada, ou seja $L(C) = \emptyset$.
  - F = "Sobre a entrada $\langle A, B\rangle$, onde A e B são AFDs:
    1. Construa o AFD C
    2. Rode a MT T sobre a entrada C
    3. Se T aceita, *aceite*. Se T rejeita, *rejeite*.

Linguagens Decidíveis

Testar se uma GLC gera uma cadeia de entrada.

$$A_{GLC} = \{\langle G, w \rangle \mid G \text{ é uma GLC que gera a cadeia } w\}$$

- A_{GLC} é uma linguagem decidível
- **Idéia da Prova**: Se tentarmos enumerar as derivações de G, o procedimento pode nunca parar, caso w não possa ser gerada por G. Para garantir que o processo pare, precisamos assegurar que apenas um número finito de derivações sejam tentadas. Uma GLC pode ser reescrita na Forma Normal de Chomsky, ou seja, $A \rightarrow BC \mid A \rightarrow a$ (adicionalmente permite a regra $S \rightarrow \varepsilon$ desde que S não apareça no corpo de nenhuma produção). Em uma forma normal de Chomsky, uma derivação de w tem no máximo $2 \mid w \mid -1$ passos. A MT S para A_{GLC} é como segue:
  - S = "Sobre a entrada $\langle G, w \rangle$, onde G é uma GLC e w uma cadeia:
    1. Converta G para uma gramática equivalente na forma normal de Chomsky
    2. Liste todas as derivações com até $2 \mid w \mid -1$ passos. Se $\mid w \mid = 0$ liste derivações com 1 passo.
    3. Se alguma dessas derivações gera w aceite, se não, rejeite.

Testar a vacuidade para a linguagem de uma GLC

$$V_{GLC} = \{\langle G \rangle \mid G \text{ é uma GLC e } L(G) = \emptyset\}$$

- V_{GLC} é uma linguagem decidível
- **Idéia da Prova**: Testar se a variável inicial da gramática pode gerar um sequência de terminais. Seja R uma MT, como segue:
  - R = "Sobre a entrada $\langle G \rangle$, onde G é uma GLC:
    1. Marque todos os símbolos terminais em G
    2. Repita até que nenhuma variável venha a ser marcada:
       Marque qualquer variável A em G, onde $A \rightarrow U_1 U_2 \cdots U_k$ e cada símbolo U_i já tenha sido marcado
    3. Se a variável S não está marcada, *aceite*. Caso contrário, *rejeite*.

O problema da Parada

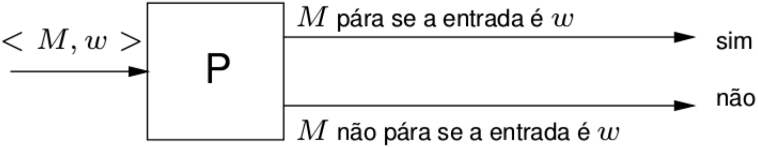
- Dadas uma Máquina de Turing arbitrária M e uma palavra arbitrária w, M pára ao computar w?

O problema da Parada

- A existência de uma MT Universal (U), ou seja, uma máquina de Turing que simula uma MT M para uma entrada w, prova que $L(U)$, ou seja a linguagem formada por todas as Máquinas de Turing e todas as entradas, é recursivamente enumerável
- Ao mostrar a indecidibilidade do problema da parada, mostra-se que não existe uma MT que sempre páre e que seja equivalente a U, ou seja, que $L(U)$ não é recursiva
  - A classe das linguagens recursivas é uma subconjunto estrito da classe das linguagens recursivamente enumeráveis

O problema da Parada

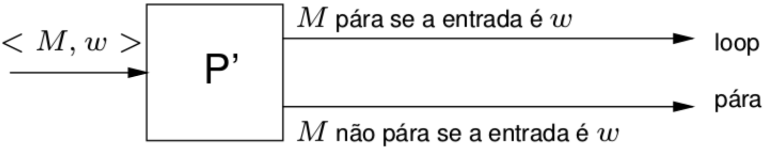
- Teorema: O problema da Parada para MT é indecidível.
- Prova: Suponha que o problema da parada seja decidível. Então existe uma MT P que sempre pára em resposta a uma entrada $< M, w >$



O problema da Parada

Continuação

- A partir da máquina *P* seria possível construir uma máquina P' que entra em loop se e somente se *P* pára em um estado final, ou ainda P' entra em loop se e somente se *M* pára com a entrada *w*



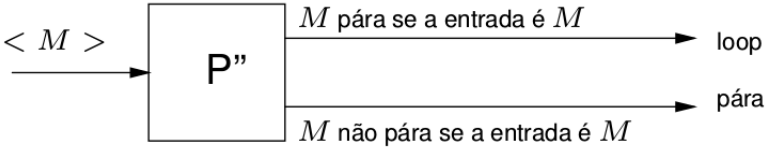
O problema da Parada

Continuação

- Para construir P', faça como segue:
  - para cada par (p, a), com $p \in K$ e $a \in \Gamma$ onde $\delta(p, a)$ é indefinida, $\delta(p, a) \rightarrow (l, a)$ onde l é um novo estado
  - crie novas transições $\delta(l, a) \rightarrow (l, a)$ para todo símbolo $a \in \Sigma$

Continuação

- A partir da máquina P' pode-se obter uma máquina P''



O problema da Parada

Continuação

- Para construir P'' a partir de P' construa transições para:
  - Duplicar a entrada $< M >$ para obter $< M, M >$ (basta copiar a entrada M na fita $M111M$)
  - Agir como P' sobre a entrada

O problema da Parada

Continuação

- agora considere o que acontece se $< P'' >$ for submetida como entrada para a MT P''
  - se P'' entra em loop, para a entrada $< P'' >$ é porque P'' pára se a entrada é $< P'' >$
  - se P'' pára quando a entrada é $< P'' >$ é porque P'' não pára se a entrada é $< P'' >$, ou seja

    P'' pára com entrada $< P'' >$ sse P'' não pára com entrada $< P'' >$

  - Mas P'' pode ser construída a partir de P. Assim, P não pode existir e portanto o problema da parada é indecidível