Capítulo 5: Decidibilidade

Newton José Vieira

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

1 de outubro de 2010

Sumário

Sumário

Sumário

Sumário

Sumário

Sumário

Computação efetiva e formalização

As MTs podem ser usadas para:

- reconhecimento de linguagens (resolver PDs);
- computação de funções em geral.

Computação efetiva tem como características:

- a possibilidade de execução mecânica;
- produção da mesma saída para as mesmas entradas;
- execução em tempo finito; etc.

Formalizando computação efetiva, é possível mostrar que:

- um problema é **computável** (ou decidível, se for PD);
- um problema é **não computável** (ou indecidível, se for PD).

Alguns formalismos

Formalismos igualmente expressivos:

- máquinas de Turing;
- sistemas de Post;
- funções μ-recursivas;
- λ-cálculo;
- máquinas abstratas associadas a linguagens de programação, como Java, C, Pascal.

Observações:

- MT é um dos mais aderentes aos computadores digitais.
- "Computação": o que há de comum entre os formalismos.
- Cada formalismo, uma abordagem para "computação".

A tese de Church-Turing

Tese de Church-Turing

Se uma função é efetivamente computável, então ela é computável por meio de uma máquina de Turing.

(Ou: todo algoritmo pode ser expresso mediante uma MT.)

Como os formalismos são equivalentes, da tese de Church-Turing segue-se que:

- se uma função é efetivamente computável, então ela é computável por meio de um programa escrito na linguagem C etc...

A tese de Church-Turing para PDs

Tese de Church-Turing para PD

Se um problema de decisão tem solução, então existe uma MT que o soluciona.

Auto-referência

Auto-referência: algoritmo recebe outro como entrada.

- Uma MT pode receber uma MT como entrada.
- Um programa pode receber um programa como entrada.
- **Máquina universal**: uma MT (ou programa) que simula uma MT (programa) qualquer suprida como argumento.

Auto-referência levou à descoberta de funções não computáveis:

- Não existe MT que determine se uma MT arbitrária para ou não para certa entrada.
- Não existe programa em C que determine se um programa em C para ou não para certa entrada.
- Vários outros problemas que envolvem o processamento de MTs (programas) por MTs (programas) são insolúveis.

Necessidade de uma representação

- Solução de um PD P: algoritmo (MT) que dá a resposta correta para cada instância $p \in P$.
- MT M_P que soluciona P: reconhece a linguagem constituída de todas as instâncias $p \in P$ para as quais a resposta é "sim".
- Logo: P é decidível sse a linguagem constituída de todas as instâncias $p \in P$ para as quais a resposta é "sim" é **recursiva**.
- Primeiro passo para construir M_P: projetar uma **representação** para as instâncias de P.

Exemplo de representação

PD: "determinar se um número natural n é primo".

Duas representações para a instância "determinar se j é primo":

a) Alfabeto $\{1\}$: 1^j.

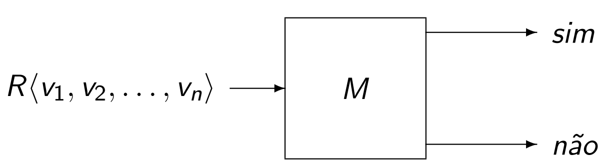
b) Alfabeto $\{0, 1\}$: número j na base 2.
(λ não representa número algum.)

Notação

Seja uma instância $p \in P$ cujos valores para os n parâmetros sejam v_1, v_2, ..., v_n.

$R\langle v_1, v_2, \ldots, v_n\rangle$: palavra que represente a instância p.

Representação esquemática de uma MT M que soluciona P:



´ Para simplificar: a entrada está no formato $R\langle v_1, v_2, \ldots, v_n\rangle$.

Exemplo

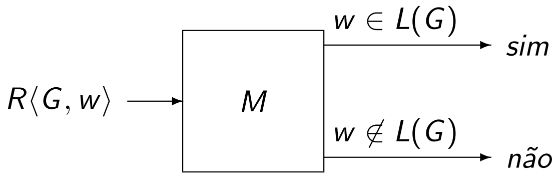
Problema: determinar se uma GLC G gera uma palavra w.

Representação das instâncias usando $\Sigma = \{0, 1\}$, sendo $G = (V, \Gamma, R, P)$, $V = \{X_1, X_2, \ldots, X_n\}$ e $\Gamma = \{a_1, a_2, \ldots, a_k\}$.

- Variável: $R\langle X_i\rangle = 1^i$; $\underline{P = X_1}$.
- Terminal: $R\langle a_j\rangle = 1^{n+j}$.
- Regra: $R\langle X \to A_1A_2 \ldots A_p\rangle = R\langle X\rangle 0 R\langle A_1\rangle 0 R\langle A_2\rangle 0 \ldots R\langle A_p\rangle$.
- Regras: $R\langle\{r_1, r_2, \ldots, r_q\}\rangle = R\langle r_1\rangle 00\langle r_2\rangle 00 \ldots R\langle r_q\rangle$.
- Gramática: $R\langle G\rangle = 1^n 0 1^k 0 R\langle\{r_1, \ldots r_q\}\rangle$.
- Instância: $R\langle G, w\rangle = R\langle G\rangle 000 R\langle w\rangle$.

Exemplo/Esquema da MT solução

Representação esquemática da MT que soluciona o problema:



O PD tem solução sse a linguagem $\{R\langle G, w\rangle \mid w \in L(G)\}$ é recursiva.

Uma representação para MTs

Seja $M = (E, \Sigma, \Gamma, \langle, \sqcup, \delta, i, F)$, $E = \{e_1, \ldots, e_n\}$ e $\Gamma = \{a_1, \ldots, a_k\}$.
Suponha que $e_1 = i$, $a_1 = \langle$, $a_2 = \sqcup$.

Representações dos estados e símbolos do alfabeto:

Estado	**Representação**
$e_1 = i$	1
e_2	11
$\vdots$	$\vdots$
e_n	1^n

Símbolo de Γ	**Representação**
$a_1 = \langle$	1
$a_2 = \sqcup$	11
$\vdots$	$\vdots$
a_k	1^k

Uma representação para MTs (cont.)

Direção: $R\langle D\rangle = 1$, $R\langle E\rangle = 11$.
Supondo $F = \{f_1, f_2, \ldots, f_p\}$:

- $R\langle F\rangle = R\langle f_1\rangle 0 R\langle f_2\rangle 0 \cdots R\langle f_p\rangle$;
- $R\langle \delta(e_i, a_j) = [e'_i, a'_j, d]\rangle = R\langle e_i\rangle 0 R\langle a_j\rangle 0 R\langle e'_i\rangle 0 R\langle a'_j\rangle 0 R\langle d\rangle$.

Sendo $t_1, t_2, \ldots, t_s$ as transições de M, uma representação de M é:

$$R\langle M\rangle = R\langle F\rangle 00 R\langle t_1\rangle 00 R\langle t_2\rangle 00 \cdots R\langle t_s\rangle.$$

Representação para MTs/exemplo

$M = (\{0,1\}, \{\mathtt{a}, \mathtt{b}\}, \{\langle, \sqcup, \mathtt{a}, \mathtt{b}\}, \langle, \sqcup, \delta, 0, \{0,1\})$ com δ contendo:
t_1: $\delta(0, \mathtt{a}) = [1, \mathtt{a}, \mathtt{D}]$
t_2: $\delta(1, \mathtt{b}) = [0, \mathtt{b}, \mathtt{E}]$.

Representação para M:

- Estados: $R\langle 0\rangle = 1$, $R\langle 1\rangle = 11$.
- Símbolos: $R\langle\langle\rangle = 1$, $R\langle\sqcup\rangle = 11$, $R\langle\mathtt{a}\rangle = 111$, $R\langle\mathtt{b}\rangle = 1111$.
- Direção: $R\langle\mathtt{D}\rangle = 1$, $R\langle\mathtt{E}\rangle = 11$.

Representação para MTs/exemplo (cont.)

- Transição 1: $R\langle t_1\rangle = R\langle 0\rangle 0R\langle \mathtt{a}\rangle 0R\langle 1\rangle 0R\langle \mathtt{a}\rangle 0R\langle \mathtt{D}\rangle$ $= 10111011011101.$
- Transição 2: $R\langle t_2\rangle = R\langle 1\rangle 0R\langle \mathtt{b}\rangle 0R\langle 0\rangle 0R\langle \mathtt{b}\rangle 0R\langle \mathtt{E}\rangle$ $= 11011110101111011.$
- Estados finais: $R\langle F\rangle = 1011.$
- $R\langle M\rangle = R\langle F\rangle 00R\langle t_1\rangle 00R\langle t_2\rangle$ $= 1011001011101101110100110111101011111011.$

Máquina de Turing universal

copie $R\langle w\rangle$ na fita 2, $R\langle i\rangle$ na fita 3 e posicione cabeçotes no início;
ciclo
  seja $R\langle a\rangle$ a representação sob o cabeçote da fita 2;
  seja $R\langle e\rangle$ a representação sob o cabeçote da fita 3;
  procure $R\langle e\rangle 0R\langle a\rangle 0R\langle e'\rangle 0R\langle a'\rangle 0R\langle d\rangle$ na fita 1;
  se encontrou **então**
    substitua $R\langle e\rangle$ por $R\langle e'\rangle$ na fita 3
      e volte cabeçote da fita 3 ao seu início;
    substitua $R\langle a\rangle$ por $R\langle a'\rangle$ na fita 2;
    mova cabeçote da fita 2 na direção d
  senão
    se e é estado final **então**
      pare em estado final
    senão
      pare em estado não final
    fimse
  fimse

Máquina de Turing universal

A MT U aceita a linguagem

$$L(U) = \{R\langle M, w\rangle \mid w \in L(M)\}.$$

Se o reconhecimento for *por parada*, tem-se duas "simplificações":

- estados finais estarão ausentes de $R\langle M, w\rangle$;
- após o primeiro **senão**: pare em estado final.

Chamando-se essa nova MT de U_P:

U_P aceita w se, e somente se, M pára se a entrada é w,

ou seja,

$$L(U_P) = \{R\langle M, w\rangle \mid M \text{ pára se a entrada é } w\}.$$

O problema da parada para máquinas de Turing

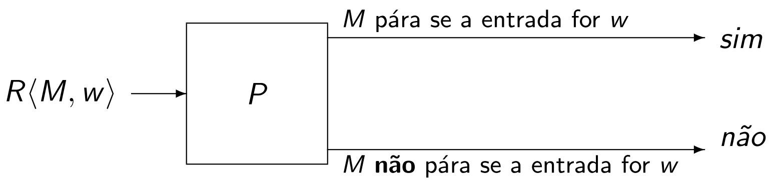
Problema da parada para MTs

Dadas uma MT arbitrária M e uma palavra arbitrária w, determinar se a computação de M com a entrada w para.

- $L(U_P)$ é LRE, pois existe a MT universal U_P.
- Será mostrado que o problema da parada é indecidível.
- Logo, não existe uma MT que sempre pare e que seja equivalente a U_P.
- Ou seja, $L(U_P)$ é LRE, mas não é recursiva.
- Ainda: $\overline{L(U_P)}$ não é LRE. (Por que?)

O problema da parada é indecidível

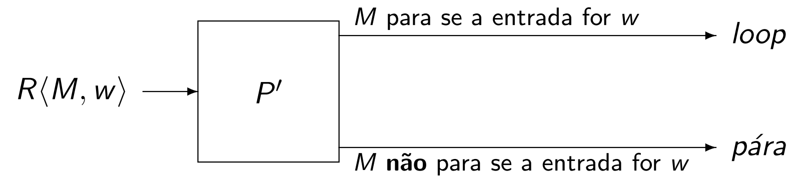
Suponha que o problema seja decidível, e seja uma MT P que solucionasse o problema:



A Tese de Church-Turing
Máquinas de Turing e Problemas de Decisão
Uma Máquina de Turing Universal
O Problema da Parada
Redução de um Problema a Outro
Alguns Problemas Indecidíveis Sobre GLCs

O problema da parada é indecidível (cont.)

A partir da MT P seria possível construir a MT P':



P' é tal que:

P' entra em *loop* se, e somente se, M para se a entrada é w.

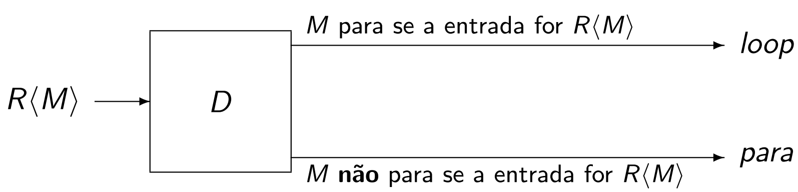
Newton José Vieira Capítulo 5: Decidibilidade

A Tese de Church-Turing
Máquinas de Turing e Problemas de Decisão
Uma Máquina de Turing Universal
O Problema da Parada
Redução de um Problema a Outro
Alguns Problemas Indecidíveis Sobre GLCs

O problema da parada é indecidível (cont.)

Pode-se obter a MT D de 1 parâmetro, $R\langle M\rangle$, tal que:

1. D obtém $R\langle M, R\langle M\rangle\rangle$.
2. D age como P' sobre essa palavra.



Se $R\langle D\rangle$ for submetida como entrada para a MT D:

D para se a entrada for $R\langle D\rangle$
se, e somente se,
D não para se a entrada for $R\langle D\rangle$.

Newton José Vieira Capítulo 5: Decidibilidade

A Tese de Church-Turing
Máquinas de Turing e Problemas de Decisão
Uma Máquina de Turing Universal
O Problema da Parada
Redução de um Problema a Outro
Alguns Problemas Indecidíveis Sobre GLCs

O problema da parada é indecidível/diagonalização

O conjunto das MTs pode ser enumerado:

$$R\langle M_0\rangle, R\langle M_1\rangle, R\langle M_2\rangle, \ldots$$

Considerando-se a MT D, tem-se para todo $i \geq 0$:

D para se a entrada é $R\langle M_i\rangle$
se, e somente se,
M_i não para se a entrada for $R\langle M_i\rangle$

Como D é uma MT, existe k tal que $D = M_k$. Segue-se que:

M_k para se a entrada é $R\langle M_k\rangle$
se, e somente se,
M_k não para se a entrada for $R\langle M_k\rangle$!!!

Newton José Vieira Capítulo 5: Decidibilidade

Problema da parada para linguagens de alto nível

O problema da parada para as linguagens de programação procedurais comuns é indecidível.

1. Suponha que exista a função P tal que: $P(x, w)$ retorna *verdadeiro* se, e somente se, o procedimento de texto x para se sua entrada (também texto) é w.
2. Então pode-se construir o procedimento:

   procedimento $D(x)$:
       enquanto $P(x, x)$ **faça**
       fimenquanto
   fim D.

   Seja T esse texto do procedimento D.
3. Então: $D(T)$ para se, e somente se, $D(T)$ não para.

Algumas consequências da indecidibilidade do problema da parada

Seja $L_P = \{R\langle M, w\rangle \mid M \text{ para se a entrada for } w\}$.
Tem-se:

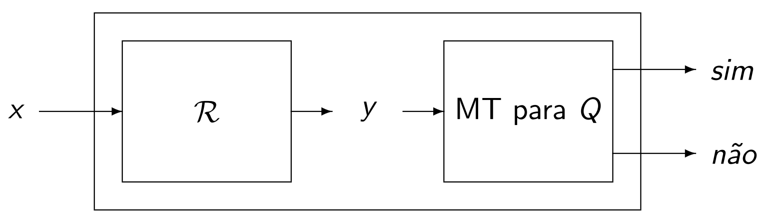
- A linguagem L_P não é recursiva.
- Como L_P é LRE (pois $L_P = L(U_P)$):
  - O problema da parada é semi-decidível.
  - O conjunto das linguagens recursivas é subconjunto próprio do conjunto das LREs.
- A linguagem $\overline{L_P}$ não é LRE. (Por que?)

Redução de um problema a outro

Um PD P é redutível a um PD Q, se existe um algoritmo $\mathcal{R}$ que, recebendo x como entrada, produz um resultado y tal que a resposta de P para a entrada x seja idêntica ou complementar à resposta de Q para a entrada y, qualquer que seja a entrada x.

Solução de P usando-se $\mathcal{R}$ e uma MT para Q:

MT para P



Redução e decidibilildade

Como provar que um PD é decidível ou indecidível usando redução?

- Se P for redutível a um problema decidível, então P será **decidível**.
- Se um problema indecidível for redutível a P, então P será **indecidível**.

Por outro lado:

- Se P pode ser reduzido a um problema indecidível, P pode ser decidível ou não.
- Se um problema decidível pode ser reduzido a P, P pode ser decidível ou não.

Redução para mostrar que um PD é decidível/Exemplo

Problema: determinar se $w \in L(r)$, em que r é uma expressão regular arbitrária.

Decidível: pode ser reduzido ao de determinar se $w \in L(M)$, em que M é um AFD arbitrário.

Redução $\mathcal{R}$: MT que constrói um AFD M_r a partir de ER r:

$$R\langle r, w\rangle \longrightarrow \boxed{\mathcal{R}} \longrightarrow R\langle M_r, w\rangle$$

tal que $w \in L(M_r)$ se, e somente se, $w \in L(r)$.

A Tese de Church-Turing
Máquinas de Turing e Problemas de Decisão
Uma Máquina de Turing Universal
O Problema da Parada
Redução de um Problema a Outro
Alguns Problemas Indecidíveis Sobre GLCs

O problema da fita em branco

Problema: Determinar se $\lambda \in L(M)$ para uma MT M.

Indecidível:
o problema da parada pode ser reduzido a este.

Redução: MT $\mathcal{R}$ que constrói uma MT M' a partir da MT M e entrada w:

$$R\langle M, w\rangle \longrightarrow \boxed{\mathcal{R}} \longrightarrow R\langle M'\rangle$$

tal que M para se entrada é w sse M' para se entrada é λ.

A Tese de Church-Turing
Máquinas de Turing e Problemas de Decisão
Uma Máquina de Turing Universal
O Problema da Parada
Redução de um Problema a Outro
Alguns Problemas Indecidíveis Sobre GLCs

O problema da fita em branco (redução)

A MT $\mathcal{R}$ produz $R\langle M'\rangle$, a partir de $R\langle M, w\rangle$, de forma que:

1. M' escreve w;
2. M' volta o cabeçote para o início da fita;
3. M' se comporta como M.

Com isso, tem-se que:

M para se a entrada é w sse M' para se a entrada é λ.

A Tese de Church-Turing
Máquinas de Turing e Problemas de Decisão
Uma Máquina de Turing Universal
O Problema da Parada
Redução de um Problema a Outro
Alguns Problemas Indecidíveis Sobre GLCs

Um outro problema indecidível

Problema: Determinar se $w \in L(G)$, para uma GI G e $w \in \Sigma^*$.

Indecidível:
o problema da parada pode ser reduzido a este.

Redução: MT $\mathcal{R}$ que constrói uma MT G a partir da MT M:

$$R\langle M, w\rangle \longrightarrow \boxed{\mathcal{R}} \longrightarrow R\langle G, w\rangle$$

tal que M para se a entrada for w se, e somente se, G gera w.

Uma suposição daqui para frente

Daqui para frente, o critério de reconhecimento é o de **parada**:

A MT M para se sua entrada é w sse $w \in L(M)$.

Logo, $L(M)$ será o mesmo que $L_P(M)$.

Uma classe de problemas indecidíveis

Propriedade trivial

Uma propriedade P de LREs é trivial se for satisfeita por toda LRE ou por nenhuma.

Todo problema do tipo:

determinar, para uma MT M, se $L(M)$ satisfaz a propriedade P

em que P é **não trivial** é **indecidível**.

Tal PD é decidível sse $\{R\langle M\rangle \mid L(M) \text{ satisfaz } P\}$ é recursiva.

Uma classe de problemas indecidíveis/exemplo

O problema da fita em branco:

determinar se $\lambda \in L(M)$

envolve uma propriedade não trivial: $\lambda \in L(M)$. Logo, ele é indecidível.

Outros PDs com propriedades não triviais:

- Determinar se $L(M)$ contém alguma palavra.
- Determinar se $L(M)$ contém todas as palavras em Σ^*.
- Determinar se $L(M)$ é finita.
- Determinar se $L(M)$ é regular.
- Determinar se $L(M)$ contém palavra começada com 0.

Outro exemplo usando redução

Determinar se $L(M) \neq \emptyset$ para uma MT arbitrária.

O problema da parada será reduzido a este. A MT redutora produz $R\langle M'\rangle$, a partir de $R\langle M, w\rangle$, de forma que:

1. M' apaga a entrada;
2. M' escreve w;
3. M' volta o cabeçote para o início da fita;
4. M' se comporta como M.

Com isto, tem-se que:

M para se a entrada é w sse M' para com alguma entrada.

O Teorema de Rice

Teorema de Rice

Se P é uma propriedade não trivial de LREs, então $\{R\langle M\rangle \mid L(M) \text{ satisfaz } P\}$ não é recursiva.

Seja P uma propriedade não trivial.

Caso 1: $\emptyset$ não satisfaz P.

Seja uma MT M_X tal que $L(M_X)$ satisfaz P.
(M_X existe pois P é não trivial, e $L(M_X) \neq \emptyset$).
O problema da parada pode ser reduzido ao de se determinar se $L(M')$ satisfaz P, como a seguir

O Teorema de Rice (cont.)

O problema da parada pode ser reduzido ao de determinar se $L(M')$ satisfaz P produzindo-se $R\langle M'\rangle$ a partir de $R\langle M, w\rangle$, onde:

1. M' escreve w na fita após sua entrada para; suponha que a fita fique assim: $\langle x[w \sqcup \ldots$;
2. M' se comporta como M sobre $[w \sqcup \ldots$;
3. na situação em que M para, M' se comporta como M_X sobre $\langle x \sqcup \ldots$.

Então

$$L(M') \text{ satisfaz } P \text{ se, e somente se, } M \text{ pára com entrada } w,$$

como explicado a seguir.

O Teorema de Rice (cont.)

$$L(M') \text{ satisfaz } P \text{ se, e somente se, } M \text{ pára com entrada } w,$$

pois:

- Se M para com entrada w, $L(M') = L(M_X)$; portanto, $L(M')$ satisfaz P.
- Se M não para com entrada w, $L(M') = \emptyset$; dada a suposição inicial desse caso, $L(M')$ não satisfaz P.

Dessa forma, conclui-se que o problema de se determinar se $L(M)$ satisfaz P, para MTs arbitrárias M, é indecidível, ou seja, $\{R\langle M\rangle \mid L(M) \text{ satisfaz } P\}$ não é recursiva.

O teorema de Rice (cont.)

Caso 2: $\emptyset$ satisfaz P.

Nesse caso, $\emptyset$ não satisfaz $\neg P$. E como P não é trivial, $\neg P$ também não é trivial. Pela argumentação do caso 1, $\{R\langle M\rangle \mid L(M) \text{ satisfaz } \neg P\}$ não é recursiva. Como essa linguagem é o *complemento* de $\{R\langle M\rangle \mid L(M) \text{ satisfaz } P\}$ e as linguagens recursivas são fechadas sob complementação, segue-se que a linguagem $\{R\langle M\rangle \mid L(M) \text{ satisfaz } P\}$ não é recursiva.

O problema da correspondência de Post

Sistema de correspondência de Post

Um sistema de correpondência de Post (SCP) é um par (Σ, P), em que P é uma sequência finita de pares (x, y), para $x, y \in \Sigma^+$.

Seja um SCP $S = (\Sigma, [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)])$.

Uma **solução para** S é uma sequência $i_1, i_2, \ldots, i_k$ tal que

$$x_{i_1} x_{i_2} \ldots x_{i_k} = y_{i_1} y_{i_2} \ldots y_{i_k},$$

em que $1 \leq i_j \leq n$ para $1 \leq j \leq k$.

A Tese de Church-Turing
Máquinas de Turing e Problemas de Decisão
Uma Máquina de Turing Universal
O Problema da Parada
Redução de um Problema a Outro
Alguns Problemas Indecidíveis Sobre GLCs

Exemplo de sistema de correspondência de Post

Um SCP: $(\{0, 1\}, [(10, 0), (0, 010), (01, 11)])$.

Solução: 2131, pois 0 10 01 10 = 010 0 11 0.
Para maior clareza, pode ser conveniente apresentar cada par (x_i, y_i) na forma $\frac{x_i}{y_i}$. Nesse caso, a solução seria apresentada assim:

$$\frac{0}{010}\,\frac{10}{0}\,\frac{01}{11}\,\frac{10}{0}.$$

Além destas, quaisquer quantidades de justaposições da seqüência anterior formam soluções: 21312131, 213121312131 etc.

A Tese de Church-Turing
Máquinas de Turing e Problemas de Decisão
Uma Máquina de Turing Universal
O Problema da Parada
Redução de um Problema a Outro
Alguns Problemas Indecidíveis Sobre GLCs

O problema da correpondência de Post

O problema da correspondência de Post (PCP), é:

determinar se um SCP arbitrário tem solução.

Será demonstrado que esse problema é indecidível em dois passos:

1. o PCP modificado (PCPM), será reduzido ao PCP;
2. o problema da parada será reduzido ao PCPM.

As reduções a serem feitas:

$\boxed{\text{parada}}$ $\boxed{\text{PCPM}}$ $\boxed{\text{PCP}}$

$$R\langle M, w\rangle \longrightarrow \boxed{\mathcal{R}} \longrightarrow R\langle S\rangle \longrightarrow \boxed{\mathcal{R}'} \longrightarrow R\langle S'\rangle$$

A Tese de Church-Turing
Máquinas de Turing e Problemas de Decisão
Uma Máquina de Turing Universal
O Problema da Parada
Redução de um Problema a Outro
Alguns Problemas Indecidíveis Sobre GLCs

O problema da correpondência de Post modificado

O **problema da correspondência de Post modificado** (PCPM):

determinar se um SCP arbitrário tem solução iniciada com 1.

Exemplos:

- O SCP $(\{0, 1\}, [(0, 010), (10, 0), (01, 11)])$ tem solução: 1232.
- O SCP $(\{0, 1\}, [(10, 0), (0, 010), (01, 11)])$ não tem solução.

Redução do PCPM ao PCP

Seja um SCP $S = (\Sigma, P)$ com $P = [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$.
Seja "$*$" um símbolo não pertencente a Σ, e sejam:

- x'_i o resultado de colocar "$*$" *após* cada símbolo de x_i. Por exemplo, se $x_i = 11010$, então $x_i = 1*1*0*1*0*$.
- y'_i o resultado de colocar "$*$" *antes* de cada símbolo de y_i. Por exemplo, se $y_i = 0100$, então $y_i = *0*1*0*0$.

Seja o SCP $S' = (\Sigma \cup \{*, \#\}, P')$, sendo P' constituído por:

- $(*x'_1, y'_1)$ (a ser o primeiro par de uma solução);
- (x'_i, y'_i) para $1 \leq i \leq n$; e
- $(\#, *\#)$.

Então: S apresenta solução começada com (x_1, y_1) se,
e somente se, S' tem solução.

Redução do problema da parada ao PCPM: a idéia

<u>A idéia</u>: construir S de forma que, havendo uma solução para S,

$$\frac{x_1}{y_1}\frac{x_{i_2}}{y_{i_2}} \cdots \frac{x_{i_k}}{y_{i_k}},$$

$x_1 x_{i_2} \cdots x_{i_k}$ "represente" a computação de M para a entrada w. A solução deve existir somente se M parar quando a entrada for w.

Redução do problema da parada ao PCPM

Seja uma MT $M = (E, \Sigma, \Gamma, \langle, \sqcup, \delta, i)$ e uma palavra $w \in \Sigma^*$. Seja o SCP $S = (\Delta, P)$ em que:

- $\Delta = E \cup \Gamma \cup \{*, \#\}$;
- o primeiro elemento de P é $(*, *\langle iw*)$;
- os pares restantes de P são:
  - **a)** (c, c), para cada $c \in \Gamma$;
    $(*, *)$.
  - **b)** Para cada $a, b \in \Gamma$ e $e, e' \in E$:
    (ea, be'), se $\delta(e, a) = [e', b, \mathrm{D}]$;
    $(e*, be'*)$, se $\delta(e, \sqcup) = [e', b, \mathrm{D}]$;
    $(cea, e'cb)$, se $\delta(e, a) = [e', b, \mathrm{E}]$, para cada $c \in \Gamma$;
    $(ce*, e'cb*)$, se $\delta(e, \sqcup) = [e', b, \mathrm{E}]$, para cada $c \in \Gamma$.

Redução do problema da parada ao PCPM

- **c)** $(ea, \#)$, se $\delta(e, a)$ é indefinido, para cada $e \in E$ e $a \in \Gamma$;
  $(e*, \#*)$, se $\delta(e, \sqcup)$ é indefinido, para cada $e \in E$.
- **d)** $(c\#, \#)$ para cada $c \in \Gamma$;
  $(\#c, \#)$ para cada $c \in \Gamma$.
- **e)** $(*\#*, *)$.

Então: M para se a entrada é w se,
e somente se, S tem solução iniciada com $(*, *\langle iw*)$.

Duas gramáticas obtidas a partir de um SCP

Seja um SCP $S = (\Sigma, \{(x_1, y_1), \ldots, (x_n, y_n)\})$.
Sejam n símbolos **distintos** s_1, ..., s_n, nenhum deles em Σ.

Duas GLCs:

- $G_x = (\{P_x\}, \Sigma \cup \{s_1, s_2, \ldots, s_n\}, R_x, P_x)$, onde R_x consta das $2n$ regras:
  $P_x \rightarrow x_i P_x s_i$, para cada $1 \leq i \leq n$, e
  $P_x \rightarrow x_i s_i$, para cada $1 \leq i \leq n$.
- $G_y = (\{P_y\}, \Sigma \cup \{s_1, s_2, \ldots, s_n\}, R_y, P_y)$, onde R_y consta das $2n$ regras:
  $P_y \rightarrow y_i P_y s_i$, para cada $1 \leq i \leq n$, e
  $P_y \rightarrow y_i s_i$, para cada $1 \leq i \leq n$.

Relação entre o SCP S e as GLCs G_x e G_y

Se $i_1 i_2 \ldots i_k$ é uma solução de S, então

$$P_x \stackrel{*}{\Rightarrow} x_{i_1} x_{i_2} \ldots x_{i_k} s_{i_k} \ldots s_{i_2} s_{i_1} \text{ e } P_y \stackrel{*}{\Rightarrow} y_{i_1} y_{i_2} \ldots y_{i_k} s_{i_k} \ldots s_{i_2} s_{i_1}$$
$$\text{e } x_{i_1} x_{i_2} \ldots x_{i_k} = y_{i_1} y_{i_2} \ldots y_{i_k}.$$

e vice-versa. Conclui-se, então, que:

$$S \text{ tem solução se, e somente se, } L(G_x) \cap L(G_y) \neq \phi.$$

Problema indecidível sobre GLCs

Não existe algoritmo para determinar se as linguagens de duas GLCs são disjuntas.

O PCP pode ser reduzido a este construindo-se G_x e G_y:

$$R\langle S\rangle \longrightarrow \boxed{\mathcal{R}} \longrightarrow R\langle G_x, G_y\rangle$$

pois S tem solução sse $L(G_x) \cap L(G_y) \neq \phi$.

Outro problema indecidível sobre GLCs

Obs: existem GLCs para $\overline{L(G_x)}$ e $\overline{L(G_y)}$

Não existe algoritmo para determinar se $L(G) = \Sigma^*$.

O PCP pode ser reduzido a este, construindo-se uma GLC para $\overline{L(G_x)} \cup \overline{L(G_y)}$, pois:

$$\overline{L(G_x)} \cup \overline{L(G_y)} = \Sigma^* \text{ se, e somente se, } S \text{ não tem solução,}$$

pois:

$$\begin{aligned}\overline{L(G_x)} \cup \overline{L(G_y)} = \Sigma^* &\leftrightarrow \overline{\overline{L(G_x)} \cup \overline{L(G_y)}} = \phi \\ &\leftrightarrow L(G_x) \cap L(G_y) = \phi \\ &\leftrightarrow S \text{ não tem solução.}\end{aligned}$$

Mais um problema indecidível sobre GLCs

Não existe algoritmo para determinar se uma GLC é ambígua.

O PCP pode ser reduzido a este obtendo-se a partir de um SCP S:

$$G = (\{P, P_x, P_y\}, \Sigma \cup \{s_1, s_2, \ldots, s_n\}, R_x \cup R_y \cup \{P \to P_x, P \to P_y\}, P)$$

pois:

G é ambígua se, e somente se, $L(G_x) \cap L(G_y) \neq \phi$.