UFPE - CIn - Comp-Lógica

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
COMPUTABILIDADE E LÓGICA

Tópicos Avançados em Teoria da Computabilidade: O Teorema da Recursão

Gleifer Vaz Alves

UFPE-CIn

gva@cin.ufpe.br

18 de agosto de 2005

Roteiro

Introdução ao teorema da recursão

Construção da máquina de auto-referência

Definição do teorema da recursão

Aplicações

Introdução

- Objetivo: apresentação do teorema da recursão dentro da teoria da computabilidade,
  - implementação do teorema no modelo da máquina de Turing.
- Importantes resultados;
- Correspondências com:
  - lógica matemática,
  - teoria de sistemas auto-reprodutivos,
  - vírus de computador,
  - programação,

Entendimento do teorema da recursão

Paradoxo no estudo da vida

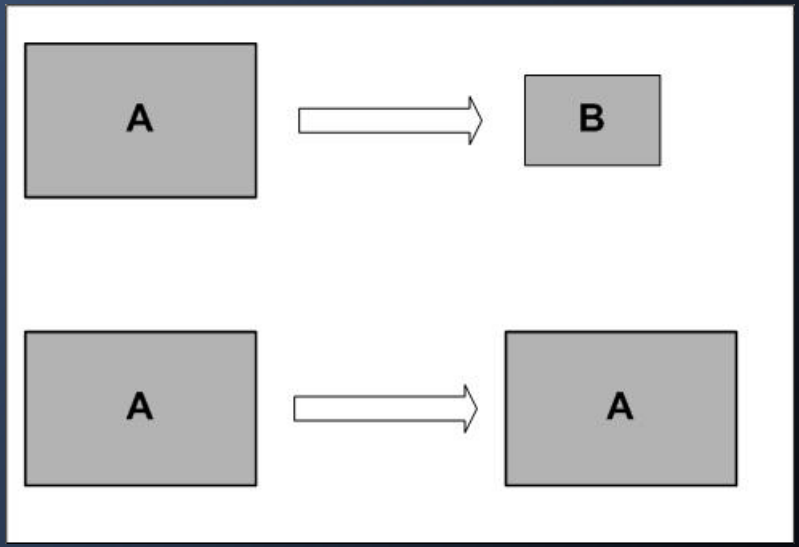
1. Coisas vivas são máquinas.
2. Coisas vivas podem se auto-reproduzir.
3. Máquinas *não* podem se reproduzir.

- As duas primeiras afirmações são verdadeiras.
- Enquanto a terceira afirmação gera um paradoxo...

Máquinas se reproduzem ou não se reproduzem?

- Para uma máquina A construir uma máquina B,
  - A deve ser mais complexa do que B
  - exemplo: fábrica e veículos
- Assim, para a máquina A se reproduzir, ela deve ser mais *complexa* do que si própria,
  - o que é impossível!
- Todavia, surge o *paradoxo*, já que o teorema da *recursão* diz que é possível que as máquinas reproduzam a si próprias.

Paradoxo!



Auto-referência

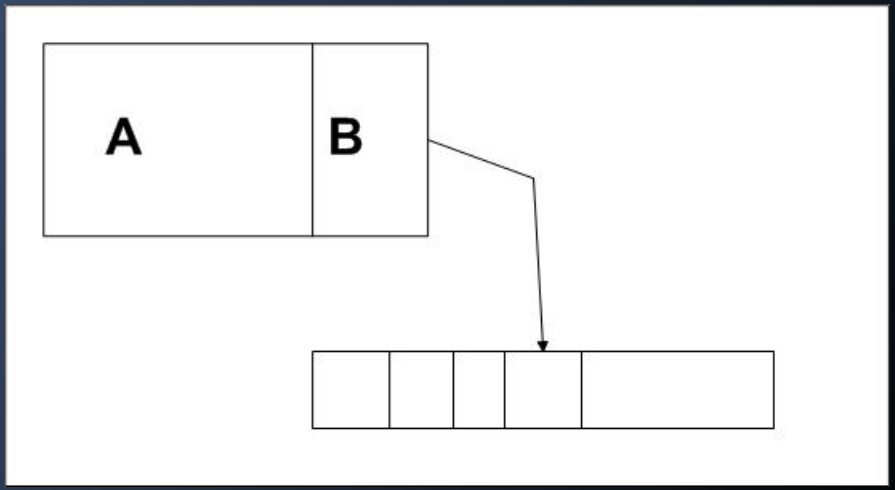
- Máquina de Turing AUTO,
  - é uma máquina de Turing sem entrada, que tem a finalidade de imprimir uma cópia de sua própria descrição.

Lema 0.1 *Existe uma função computável $q : \Sigma^* \to \Sigma^*$, onde se w é uma cadeia qualquer, $q(w)$ é a descrição de uma máquina de Turing P_w que imprime w e pára.*

- MT Q = sobre a cadeia de entrada w:
  1. Construa a máquina de Turing P_w.
     P_w = sobre qualquer entrada:
     (a) Apague a entrada.
     (b) Escreva w na fita.
     (c) Pare.
  2. Dê como saída $\langle P_w \rangle$.

Máquina AUTO

- a máquina é dividida em duas partes: A e B,
- procedimento A + procedimento B $\longrightarrow$ montar AUTO,
- objetivo: $\langle AUTO \rangle = \langle AB \rangle$



Máquina AUTO

- Inicialmente, A é executado e após o término passa o controle para B,
- a tarefa de A é imprimir uma descrição de B, assim como,
  a tarefa de B é imprimir uma descrição de A,
- o resultado é a descrição de AUTO.
- **parte A**: consiste em uma máquina de Turing que imprime $\langle B \rangle$
  existe dependência de A com B, então é necessário construir B.

Máquina AUTO

- Caso B fosse definido em termos de A, teria-se uma definição *circular* de um objeto em termos de si próprio.
- Estratégia: B computa A a partir da saída que A produz.
- para B obter a sua própria descrição $\langle B \rangle$, basta B olhar para a fita!
  isso porque quando A terminou foi deixada na fita a descrição de B,
- Com isso, B combina o A e B em uma única máquina e escreve sua descrição
  $\langle AB \rangle = \langle AUTO \rangle$

Execução de AUTO

1. Inicialmente A é executada e imprime $\langle B \rangle$ na fita.
2. B começa, olha para a fita e encontra sua entrada, $\langle B \rangle$.
3. B calcula $q(\langle B \rangle) = \langle A \rangle$ e combina isso com $\langle B \rangle$ na descrição de uma MT, $\langle AUTO \rangle$.
4. B imprime essa descrição e pára.

Auto-referência

- Exemplos em linguagem natural:
  - Imprima esta sentença
  - Imprima duas cópias do seguinte, a segunda entre aspas:
    “Imprima duas cópias do seguinte, a segunda entre aspas:”

Exemplo em linguagem de programação funcional

```
funcaoB :: Cadeia1 -\rangle  Cadeia2

funcaoB 0                  = 0

funcaoB (head:tail)        = funcaoB tail ++ funcaoA (head tail)

---------

funcaoA :: Letra -\rangle  Cadeia1 -\rangle  Cadeia2

funcaoA head tail              = tail ++ head
```

O teorema da recursão

- Características: tem a capacidade de implementar o auto-referencial *esse* em qualquer linguagem de programação,

  assim, qualquer programa tem como se referir a sua própria descrição.

- teorema: estende a idéia de AUTO, ao invés de apenas imprimir a descrição, o programa pode obter sua própria descrição.

  Teorema 0.1 (Teorema da recursão) *Seja T uma máquina de Turing que computa uma função $t : \Sigma^* \times \Sigma^* \to \Sigma^*$. Existe uma máquina de Turing R que computa uma função $r : \Sigma^* \to \Sigma^*$, onde para toda w,*

  $$r(w) = t(\langle R \rangle, w)$$

- objetivo: montar uma máquina de Turing que obtenha sua própria descrição e então compute com tal descrição.

Idéia do teorema da recursão

- **tarefa 1**: montar uma máquina T com duas entradas: a descrição de R e a palavra.

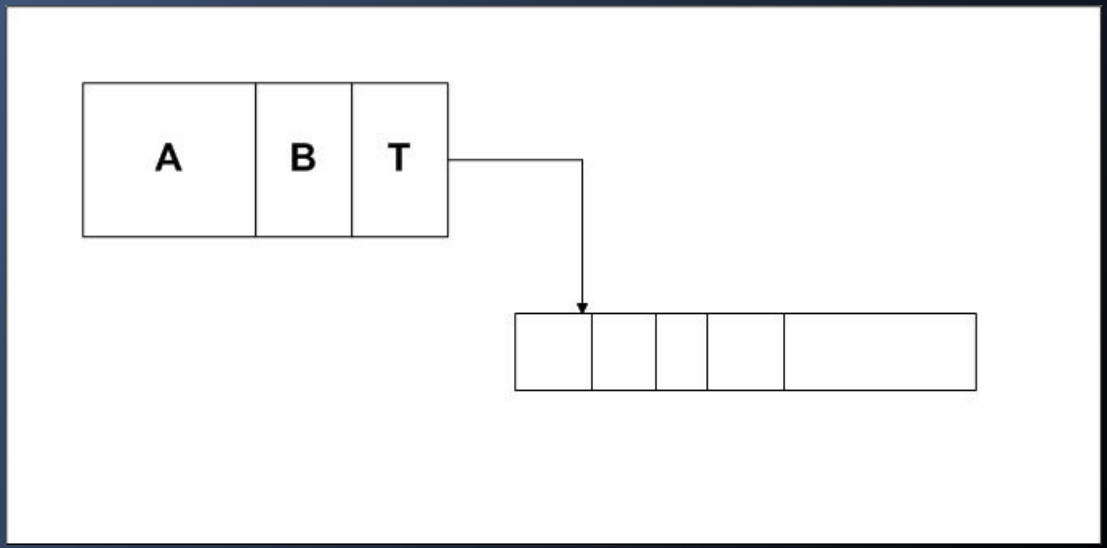
$$t(\langle R \rangle, w)$$

- **tarefa 2**: produzir uma nova máquina R, (*que opera exatamente como* T), sendo que a descrição de R é preenchida automaticamente.
- **passos**:
  - A máquina T tem a descrição de R,
  - a máquina R é clone de T.
  - Logo, a descrição de R é igual a descrição de T, *i.e.*, R obtém sua própria descrição.
- Assim, consegue-se uma máquina que obtenha a própria descrição para ser computada.

Prova teorema da recursão

- Construir uma MT R em três partes: A, B e T.
  - A é uma MT $P_{\langle BT \rangle}$, descrita por $q(\langle BT \rangle)$,
  - B funciona como um procedimento que olha sua fita e aplica q ao conteúdo da fita,
    resultando em $\langle A \rangle$,
  - Com isso, B combina A, B e T em um única máquina,
    obtendo a seguinte descrição $\langle ABT \rangle = \langle R \rangle$
- Por fim, a descrição é codificada em conjunto com w, $\langle R, w \rangle$ e o controle é passado para T.

Controle para a máquina R



Teorema da recursão

- Utilizações:
- é possível utilizar o teorema da recursão para imprimir $\langle M \rangle$,
  para contar o número de estados em $\langle M \rangle$, ou até mesmo simular $\langle M \rangle$.

Teorema da recursão - Aplicações

- Vírus de computador: auto-replicação é a principal tarefa de um vírus, executar cópias de si próprios.
- Apresentação de três teoremas com aplicações do teorema da recursão.

Aplicação - A_{MT} é indecidível

- **Teorema**: A_{MT} é indecidível
- **Prova**: para obter uma contradição, assumir que a máquina de Turing H decide A_{MT},
- Construir a máquina B
  - $B =$ sobre a entrada w:
    1. Obtenha, através do teorema da recursão, a própria descrição de $\langle B \rangle$.
    2. Execute H sobre a entrada $\langle B, w \rangle$.
    3. Faça o oposto do que H faz,
       ou seja, **aceitar** caso H rejeite
       e **rejeitar** caso H aceite.
  - Máquina H funciona como um *inversor* da entrada w, logo não pode estar decidindo A_{MT}.

Aplicação - máquinas de Turing mínimas

- **Definição - máquina de Turing mínima**:

  Se M é uma máquina de Turing, então dizemos que o *comprimento* da descrição $\langle M \rangle$ de M é o número de símbolos na cadeia descrevendo M. Diz-se que M é **mínima** se não existe máquina de Turing equivalente a M que tenha uma descrição mais curta.

  Seja

$$MIN_{MT} = \{\langle M \rangle \mid M \text{ é uma } MT \text{ mínima } \}$$

- **Teorema**: MIN_{MT} não é Turing-reconhecível.

Aplicação - máquinas de Turing mínimas

- **Prova**: Assumir que $MT\ E$ enumera MIN_{MT}, para assim obter uma contradição.

  Construir uma $MT\ C$.

- $C\ =$ sobre a entrada w:

  1. Obtenha, através do teorema da recursão, a própria descrição de $\langle C \rangle$.
  2. Execute o enumerador E até que uma máquina D apareça contendo uma descrição *maior* do que a descrição da máquina C.
  3. Simule D sobre a entrada w.

- **Resultado**: E enumera uma máquina D que *não* é mínima.

Aplicação - máquinas de Turing mínimas

- **Passos**:
  - MIN_{MT} é infinita, logo a lista de E deve ter uma MT que satisfaça a a condição apresentada no passo 2,
  - portanto, em algum momento existirá uma máquina D com uma descrição *maior* do que a descrição da máquina C,
  - Como a máquina C simula a máquina D, elas são consideradas equivalentes,
  - Contudo, C tem uma descrição *mais curta* do que a descrição de D, então, a máquina D não pode ser dita *mínima*,
  - Assim, chega-se em uma contradição.

Aplicação - ponto-fixo e transformação computável

- **Ponto-fixo**: de uma função é um valor que não é modificado pela aplicação da função.
- Aqui, consideram-se as funções que são transformações computáveis de descrições de máquinas de Turing,
  - objetivo: mostrar que para qualquer transformação computável de descrições existe uma MT, cujo comportamento não é modificado pela transformação.
- **Teorema**: Seja $t : \Sigma^* \rightarrow \Sigma^*$ uma função computável. Então, existe uma máquina de Turing F para a qual $t(\langle F \rangle)$ descreve uma máquina de Turing equivalente a F.
- t desempenha o papel da transformação e F é o ponto-fixo.

Recursão e ponto-fixo

- **Prova**: seja F a seguinte máquina de Turing.
  $F =$ sobre a entrada w:
  1. Obtenha, através do teorema da recursão, a própria descrição de $\langle F \rangle$.
  2. Compute $t(\langle F \rangle)$ para obter a descrição de uma máquina de Turing G.
  3. Simule G sobre w.
- Visto que F simula G, sabe-se que ambas descrevem máquinas equivalentes.
- Assim, $\langle F \rangle$ e $t(\langle F \rangle) = \langle G \rangle$.
- Conclui-se que existe uma máquina que não tem seu comportamento afetado por uma transformação.

Primeiro exemplo - linguagem de programação

```
procImprimeDescricao (\langle parâmetros\rangle ):

      print {
              leia x;
              leia z;
              r \langle - multiplica(x,z);
              print r;
      }
```

Segundo exemplo - linguagem de programação

- Alternativa através da programação funcional
  utilizar **programação com mônadas**,
- Com a utilização de mônadas é possível escrever uma função que tenha como retorno (na verdade como *efeito colateral*), a seguinte lista:

```
[ operacao (descricao, resultado) ]
```

  onde cada elemento da lista é uma tupla com a *descrição* e o *resultado* da operação realizada.

Exemplo de programa funcional

```
Saida = [ operacao (descricao, resultado) ]

funcaoVirus :: Saida

funcaoVirus          = do {
                            op1 \langle - operacao1;
                            op2 \langle - operacao2;
                            op3 \langle - operacao3;
                            replicacao ( [ op1,op2,op3 ] );
                          }

op1 ( descricao1, resultado1 )
...
```

Exercício - Prova do teorema de Rice através do teorema da recursão

- **Teorema de Rice**:
  - Sendo P qualquer propriedade não trivial de uma linguagem de uma máquina de Turing.
  - Provar que o problema de determinar se uma dada máquina de Turing tem a propriedade P é indecidível.
  - Especificamente, P é uma linguagem que tem descrições de uma máquina de Turing, onde duas condições devem ser preenchidas:
    1. P é não trivial, *i.e.*, contem algumas, mas não todas as descrições da máquina de Turing.
    2. P é uma propriedade da linguagem da MT, qualquer que seja $L(M_1) = L(M_2)$,
       tem-se $\langle M_1 \rangle \in P$ sse $\langle M_2 \rangle \in P$.

Teorema de Rice através do teorema da recursão

- Assumir que alguma MT X decide uma propriedade P, e P satisfaz as condições do teorema de Rice, umas das condições diz que MTs A e B existem onde:
  $\langle A \rangle \in P$ e $\langle B \rangle \notin P$.
  Utiliza-se A e B para construir MT R:
- $R =$ sobre a entrada w:
  1. Obtenha, através do teorema da recursão, a própria descrição de $\langle R \rangle$.
  2. Rode X sobre $\langle R \rangle$.
  3. Se X aceita $\langle R \rangle$, simule B sobre w.
     Se X rejeita $\langle R \rangle$, simule A sobre w.

Teorema de Rice através do teorema da recursão

- Resultado:
  - Se $\langle R \rangle \in P$,
    X aceita $\langle R \rangle$ e $L(R) = L(B) \rightarrow (\langle R \rangle \in P$ sse $\langle B \rangle \in P)$
    mas, $\langle B \rangle \notin P$.
  - Se $\langle R \rangle \notin P$,
    X rejeita $\langle R \rangle$ e $L(R) \neq L(A) \rightarrow (\langle R \rangle \in P$ sse $\langle A \rangle \in P)$
    mas, $\langle A \rangle \in P$.