第八讲　递归定理及应用

罗　欢
Email：hluo@fzu.edu.cn

递归定理及其证明
(自我复制)

悖论

- 生物都是机器
  - 现代假设
- 生物都能自再生
  - 显然
- 机器不能自再生
  - 错误!
  - 递归定理

引理7.1

引理7.1: 存在可计算函数

$q : \Sigma^* \to \Sigma^*$，对任意串w,

q(w)是图灵机P_w的描述,

TM P_w打印出w, 然后停机.

说明:

- $\forall w$, $q(w)=\langle P_w \rangle$
- $\forall x$, $P_w(x)=w$
- 对任何w, P_w都存在
- 从w得到$\langle P_w \rangle$的过程,
  是可计算的

引理7.1证明

证明: TM Q="对输入串w:

1) 构造下列TM P_w:

  P_w="对任何输入:

   a) 抹去输入;

   b) 写下w;

   c) 停机."

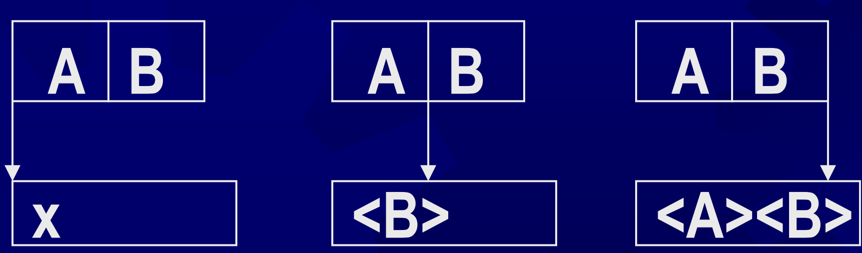
2) 输出<P_w>. "   #

自我复制机

- 自己打印自己的TM
  - TM SELF
    - ∀x, SELF(x)=<SELF>

main(){printf(“main(){printf(“main(){… ”);}

  - <SELF>=<AB>
    - ∀x, A(x)=<B>,B(x)=<A>



TM SELF

TM SELF打印<SELF>

构造: <SELF>=<AB>.

$A=P_{<B>}$，$<A>=<P_{<B>}>=q(<B>)$

B="对输入<M>,

<M>是TM的部分描述:

1) 计算q(<M>);

2) 用q(<M>)和<M>合成完整的 TM描述;

3) 打印这个描述, 然后停机. "

递归定理

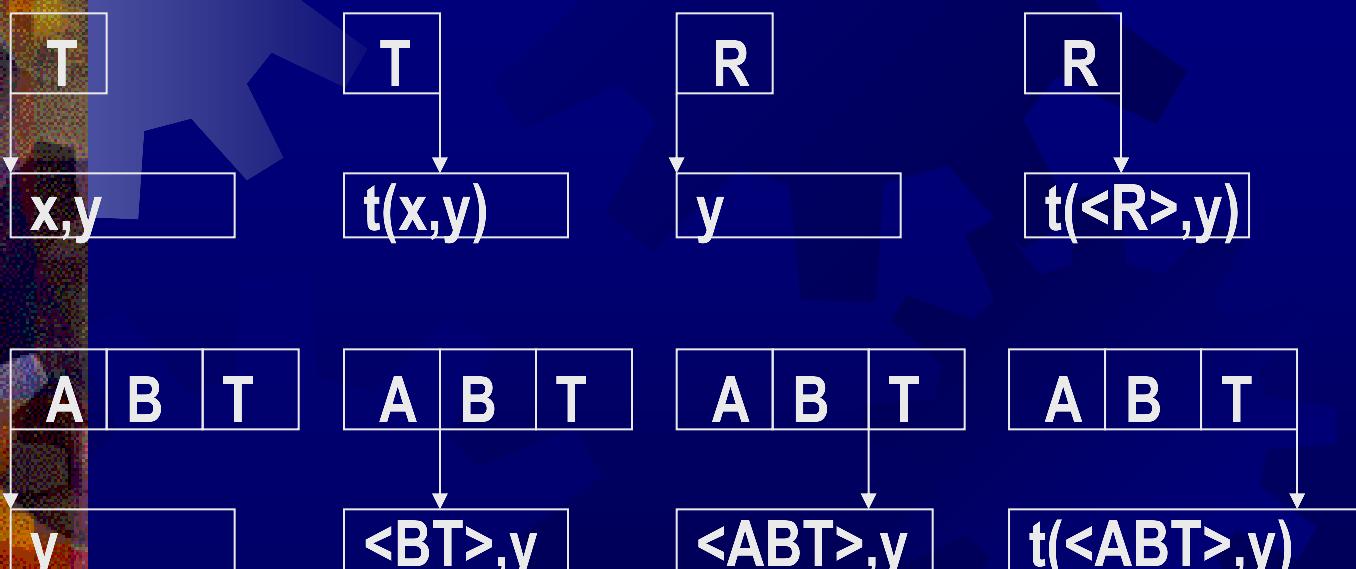
- **定理7.2**(递归定理): 设TM T计算函数 $t: \Sigma^* \times \Sigma^* \to \Sigma^*$, 则存在TM R计算函数 $r: \Sigma^* \to \Sigma^*$, 使得
  $\forall w$, $r(w)=t(\langle R\rangle, w)$.
- **说明**: 若t(x,y)是2元函数，则固定输入x=c后, 变成1元函数t(c,y)
  - 若 t(x,y) 可计算, 则对任意 c，t(c,y) 也可计算
  - 递归定理说: 存在 c, 使得 c 就是计算 t(c,y) 的程序!

递归定理证明图示

- $\forall t:\Sigma^*\times\Sigma^*\to\Sigma^*,\ \exists r:\Sigma^*\to\Sigma^*,$

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

- $\langle R\rangle=\langle ABT\rangle$



递归定理的证明

证明: <R>=<ABT>,

A=$P_{<BT>}$, <A>=q(<BT>).

B="对输入w, 输出q(w)".

TM R="对输入w:

1) 运行A, 在带上写下<BT>;

2) 运行B, 在带上写下<A>,

拼装出<ABT>;

3) 对<ABT,w>运行T. " #

注: q见引理7.1.

递归定理的应用
(通用机)

递归定理的用途

- TM自引用定义
  - 可以在M的定义中使用<M>, 即

    TM M="……<M>……" 是合法定义!
    - 先定义TM T="对输入x, y: ……x……"
    - 根据递归定理, 存在这样的R
    - TM R="对输入y: ……<R>……"
- 代替对角化
  - 证明A_{TM}不可判定, MIN_{TM}不可识别
- 不动点定理
  - ∀可计算函数t, ∃TM F, L(t(<F>))=L(F)

应用递归定理的术语

可以在M的定义中使用<M>

TM M=“…得到自己的一个描述<M>, …<M>…”

或：TM M=“……<M>……”

例：TM SELF=“对任意输入,

1) 利用递归定理得到自己的描述<SELF>;

2) 打印<SELF>. ”

或: TM SELF=“对任意输入,打印<SELF>”

图灵机接受性问题

- **TM接受性问题**
  - 检查一个图灵机是否接受一个给定的串

- 语言

A_{TM} = { <M,w> | TM M接受串w }

引理

- 引理: A_{TM}是图灵可识别的
- 证明思路:
  - 设计TM U模拟M在w上的计算
    - 当M在w上停机接受时,U接受
    - 当M在w上停机拒绝时,U拒绝
    - 当M在w上不停机时,U也不停机
  - 这样的U称为通用机
    - 注意U是固定的, M和w是任意的
    - 程序设计经验表明这样的U存在

通用机

- 存在一个固定的TM U,
对于任意的TM M和输入w,
U在<M,w>上的运行结果与
M在w上的运行结果是一样的,
即 U(<M,w>)=M(w).

- U在<M,w>上模拟M在w上的运行

引理的证明

证明: 设计TM U来识别A_{TM}.

U=“ 对于输入<M,w>,

M是TM, w是串:

1) 在输入w上模拟M.

2) 如果M进入接受状态,

则接受;

如果M进入拒绝状态,

则拒绝. ” #

定理7.3

定理7.3: A_{TM}是不可判定的.

证明: (反证) 假设TM H判定A_{TM}.
如下构造TM B.

B="对于输入w:

1) 由递归定理得到它自己的描述<B>

2) 在输入<B,w>上运行H

3) 若H接受,则拒绝; 若H拒绝,则接受. "

于是，B接受w $\Leftrightarrow$ <B,w>$\in A_{TM}$ $\Leftrightarrow$
H接受<B,w> $\Leftrightarrow$ B拒绝w，矛盾！ #

极小图灵机

- 定义7.4: 设M是图灵机,
 M的描述是字符串<M>,
 M的描述的长度是
 <M>含有的符号个数,
 如果没有与M等价的图灵机
 具有更短的描述,
 则说M是极小图灵机

图灵机的极小性问题

- 图灵机的极小性问题
  - 检测一个给定的图灵机是否极小图灵机
- MIN_{TM} = { <M> | M是极小TM }
  - 注意: 有无穷多种TM,
    每种TM都有等价的极小TM,
    所以MIN_{TM}是无穷语言

定理7.5

- **定理7.5**: $\mathrm{MIN_{TM}}$不是图灵可识别的.
- **证明**: 假设TM E枚举$\mathrm{MIN_{TM}}$.

构造下列TM C.

C="对于输入w:

1) 由递归定理得到它自己的描述<C>
2) 运行E, 直到比C的描述更长的
   TM D出现
3) 在输入w上模拟D. "

因为E枚举D, 所以D是极小TM.

但是D的描述比C长, 且D与C等价.

矛盾! #

不动点

- 如果x=f(x)，则x称为f的不动点.
- 定理7.6(递归定理的不动点形式):
对于任何一个可计算函数
$t : \Sigma^* \to \Sigma^*$, 都存在一个图灵机F,
使得t(<F>)描述一个与F等价的
图灵机.

定理7.6证明

证明: TM F=“ 对输入w:

1) 由递归定理得到自己的一个描述<F>

2) 计算t(<F>)得到一个图灵机G的描述

3) 在输入w上模拟G. ” #

内容的总结

总结

- 概念
  - 图灵机,图灵机变种, 图灵机描述
  - 图灵可识别, 图灵可判定
  - 丘奇-图灵论题, 算法
  - 递归定理, 通用机
- 等价性
  - 单带TM模拟多带TM
  - DTM模拟NTM
  - 枚举器与识别器互化
- 问题
  - 希尔伯特第十问题
  - 图灵机接受性问题
  - 极小图灵机问题

作业

- 6.1 6.2