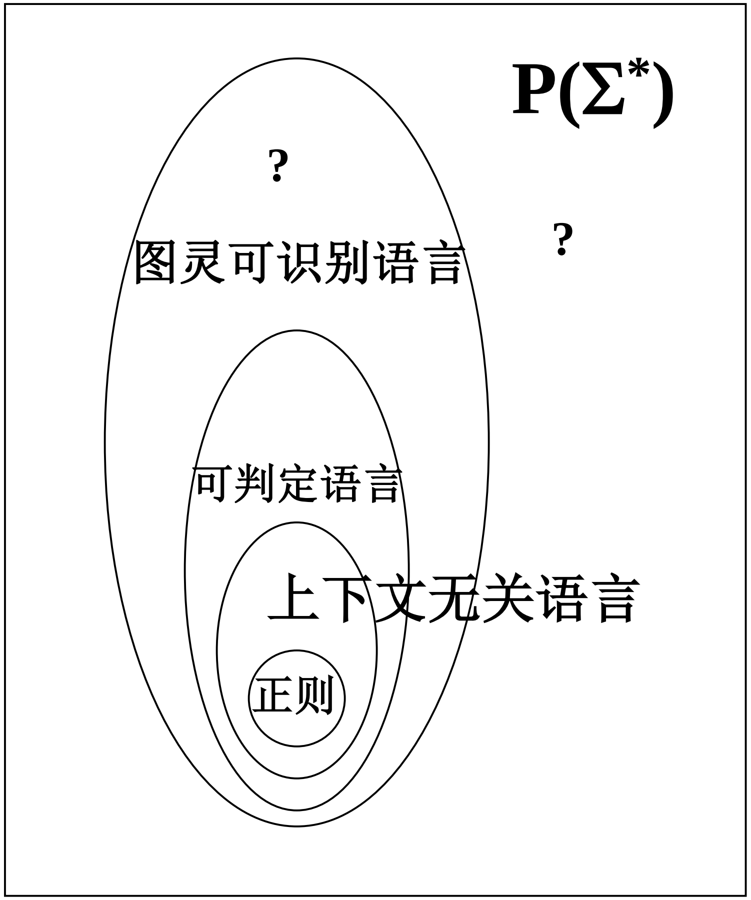
计算理论
第二部分 可计算理论

第4章 可判定性

可判定=有算法

- **Halt** 图灵可识别 非图灵可判定
- **Halt的补** 非图灵可识别
- 可判定问题举例
- 不可判定问题举例



Church-Turing 论题

1930’s人们开始考虑算法的精确定义

- 1933, Kurt Gödel, 递归函数
- 1936, Alonzo Church, λ-calculus
- 1936, Alan Turing, 判定图灵机(判定器)
- Church 和 Turing 证明这三种定义等价
- 计算机能力的极限
- 即使未来几年量子计算机制造成功,
  人们能解决的问题类并不会变大

一些自然构造的问题

停机问题:

$Halt = \{ <M,x> \mid$ 图灵机M在串x上会停机 $\}$

成员测试:

$A_{DFA} = \{<B,w> \mid B$是DFA,w是串,B接受$w\}$

$A_{CFG} = \{<B,w> \mid B$是CFG,w是串,B派生$w\}$

$A_{TM} = \{<M,w> \mid M$是一个TM,且接受$w\}$

空性质测试:

$E_{DFA} = \{\langle A\rangle \mid A$是DFA,$L(A)=\varnothing\}$

$E_{CFG} = \{\langle G\rangle \mid G$是CFG,$L(G)=\varnothing\}$

等价性质测试:

$EQ_{DFA} = \{<A,B> \mid A$和$B$都是DFA,且$L(A)=L(B)\}$

$EQ_{CFG} = \{<A,B> \mid A$和$B$都是CFG,且$L(A)=L(B)\}$

定理:停机问题Halt是图灵可识别的

Halt={ <M,x> | 图灵机M在串x上会停机 }

证明: 构造识别Halt的图灵机T,

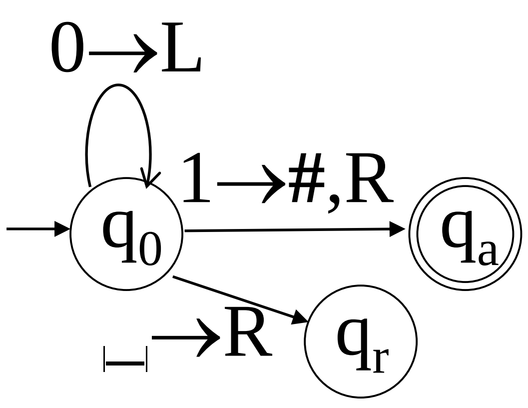
T = “ 对于输入<M,x>, M是图灵机, x是串

1. 在x上模拟M,
2. 若M停机(接受或拒绝), 则接受.”

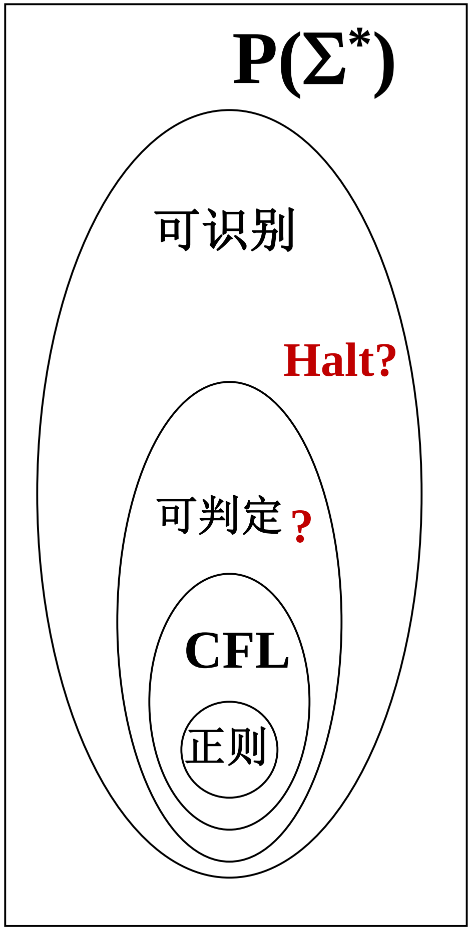
T的语言是Halt, 证毕.

注：T不是判定器 (?)

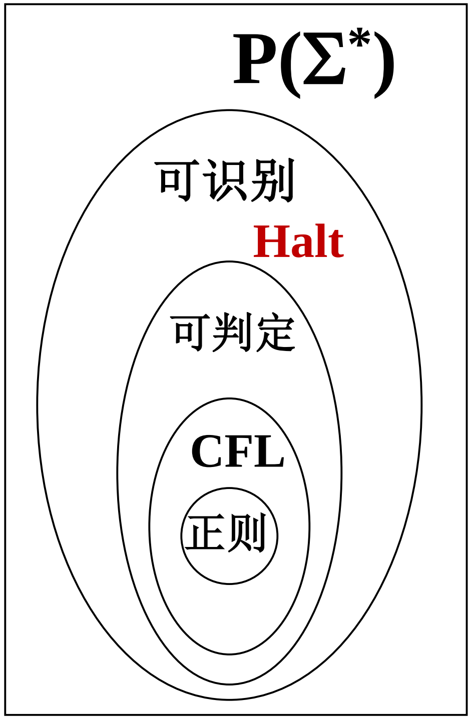
例T上运行<M,01>



M



定理:停机问题Halt不可判定



Halt={ <M,x> | 图灵机M在串x上会停机 }

证明: 假设Halt有判定器H, 构造D使用H:

Diagonal = “ 对于输入<M>, M是图灵机,

1. 在<M,<M>>上运行H,
2. 若H接受, 则返回1;
3. 若H拒绝, 则停机.”

- 在Diagonal上输入串<Diagonal>”是否会停机?
- 若D停机, 即<D,<D>>∈HALT, H接受<D,<D>>, 则由2, D不停机
- 若D不停机, 即<D,<D>>∉HALT, H拒绝<D,<D>>, 则由3, D停机
- 矛盾, 所以H不存在.

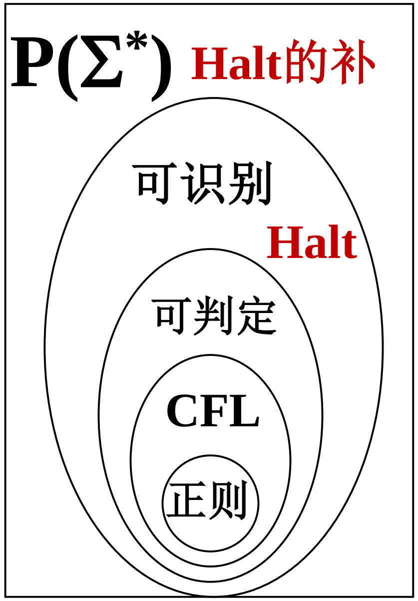
定理: Halt的补不是图灵可识别的

定理: 若A和A的补都是图灵可识别, 则A图灵可判定

证明: 设图灵机T和Q分别识别A和A的补, 构造***H***:

H = “ 对于输入x, x是串,

1. 在x上同步模拟T和Q, 直到有一个停机,
2. 若T接受x或Q拒绝x, 则接受x;
3. 若Q接受x或T拒绝x, 则拒绝x.”



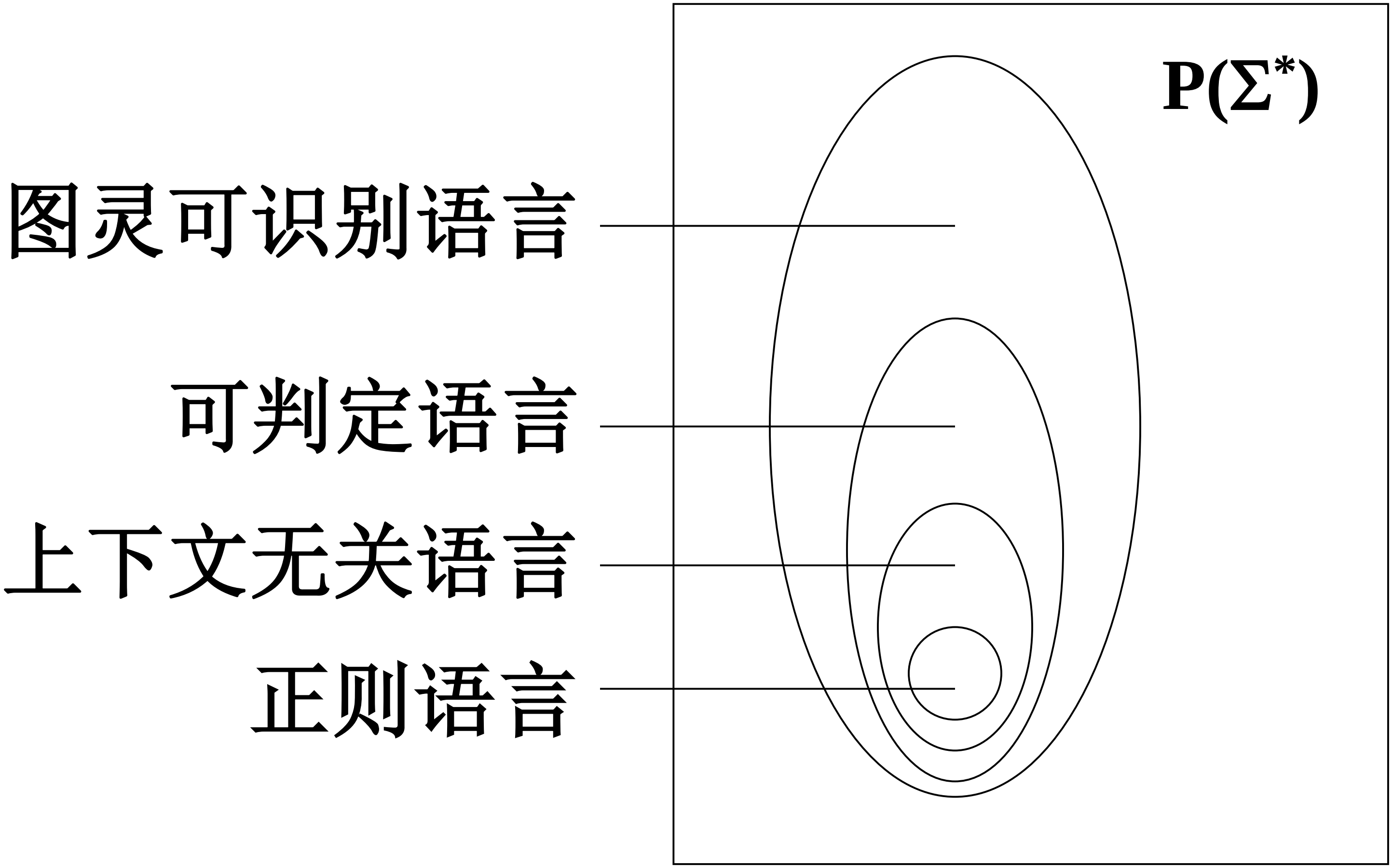
H是判定器, ***H***的语言是A.

$\forall x \in A \Rightarrow$ T一定停机接受, Q可能停机拒绝 $\Rightarrow$ ***H***停机

$\forall x \notin A \Rightarrow$ Q一定停机接受, T可能停机拒绝 $\Rightarrow$ ***H***停机

推论: 停机问题Halt的补不是图灵可识别的.

各语言类之间的关系



可判定性

停机问题:

Halt = { <M,x> | 图灵机M在串x上会停机 } 不可判定

成员测试:

A_{DFA} = {<B,w>|B是DFA,w是串,B接受w}　可判定

A_{CFG} = {<B,w>|B是CFG,w是串,B派生w}　可判定

A_{TM} = {<M,w>|M是一个TM,且接受w}　不可判定

空性质测试: E_{DFA} = {<A>|A是DFA,L(A)=∅} 可判定

E_{CFG} = {<G>|G是CFG,L(G)=∅}可判定

E_{TM} = {<M>|M是TM,L(M)=∅} 不可判定

等价性质测试:

EQ_{DFA} = {<A,B>|A和B都是DFA,且L(A)=L(B)} 可判定

EQ_{CFG} = {<A,B>|A和B都是CFG,且L(A)=L(B)} 不可判定

A_{DFA}={<B,w>|DFA B接受串w}可判定

证明:如下构造A_{DFA}的判定器:

M="对于输入<B,w>,其中B是DFA,w是串:

1)在输入w上模拟B.

2)如果模拟以接受状态结束,则接受;

如果以非接受状态结束,则拒绝."

$L(M) = A_{DFA}$. 将B视为子程序或实现细节:

- 检查输入. ((p,q,…)(a,…)((p,a,q),…)(q_0)(F), w)
- 模拟. 初始,B的状态是 q_0,读写头位于w的最左端,

  状态的更新由B的转移函数决定.

A_{NFA}={<B,w>|NFA B接受串w}可判定

思路1: 直接模拟NFA?

思路2: 先将NFA转换成DFA.

证明:如下构造A_{NFA}的判定器:

N="在输入<B,w>上,其中B是NFA,w是串:

1)将NFA B转换成一个等价的DFA C.

2)在输入<C,w>上运行A_{DFA}的判定器M.

3)如果M接受,则接受,否则拒绝."

运行TM M: M作为子程序加进N的设计中.

$L(N) = A_{NFA}$.

空性质测试

定理: E_{DFA}={<A>|A是DFA,L(A)=∅}可判定.

证明: 若A为一个DFA, 则

L(A)≠∅ ⇔ 存在从起始状态到某接受状态的路径.

T="对于输入<A>, 其中A是一个DFA:

1)标记起始状态.

2)重复下列步骤,直到没有新标记出现.

3) 对任一未标记状态, 若有从已标记状态到它的转移, 则将它标记.

4)如果无接受状态被标记,则接受;否则拒绝."

$L(T) = E_{DFA}$.

TM成员测试A_{TM}

A_{TM}={<M,w>|M是一个TM,且接受w}

定理 A_{TM}是不可判定的.

命题 A_{TM}是图灵可识别的.

U="对于输入<M,w>,其中M是TM,w是串:

1) 在输入w上模拟M;

2) 若M进入接受状态,则接受;

若M进入拒绝状态,则拒绝."

$L(U) = A_{TM}$.

注:若M在w上不停机,则U在<M,w>上不停机.

定理 A_{TM}不可判定

证明:假设A_{TM}可判定, 且设H是其判定器, 构造

D="对于输入<M>,其中M是TM:

1)在串<M, <M>>上运行H.

2)若H接受(<M, <M>>), 则D拒绝(<M>);

若H拒绝(<M, <M>>), 则D接受(<M>)."

$\langle D\rangle \in L(D)$ (?)

$\Leftrightarrow \quad \langle D, \langle D\rangle\rangle \in A_{TM}$

$\Leftrightarrow$ H接受$\langle D, \langle D\rangle\rangle$

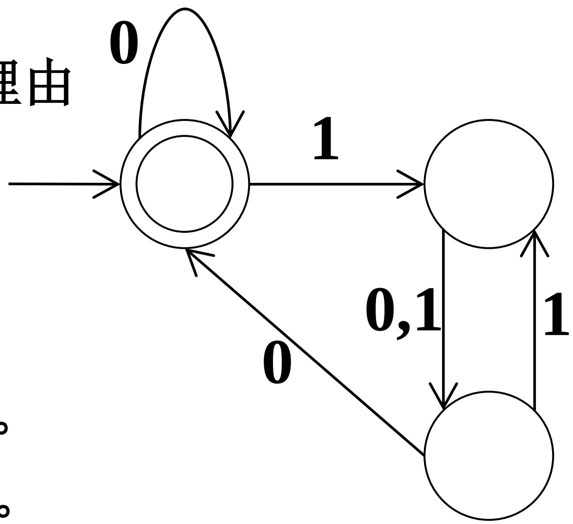
$\Leftrightarrow$ D拒绝 $\langle D\rangle$

$\Leftrightarrow \quad \langle D\rangle \notin L(D)$

矛盾, 所以H不存在.

计算理论第4章作业

4.1 对于右图所示的**DFA M,** 回答下列问题，并说明理由



a. $<M,0100> \in A_{DFA}$? **b.** $<M,011> \in A_{DFA}$?

c. $<M> \in A_{DFA}$?

e. $<M> \in E_{DFA}$? **f.** $<M,M> \in EQ_{DFA}$?

4.2 考虑一个**DFA**和一个正则表达式是否等价的问题。

将这个问题描述为一个语言并证明它是可判定的。

4.3 设 $ALL_{DFA} = \{\langle A\rangle \mid A$是一个识别$\Sigma^*$的DFA$\}$.

证明ALL_{DFA}可判定.

4.15 设$A = \{\langle R\rangle \mid R$是一个正则表达式,

其所描述的语言中至少有一个串w以111为子串 $\}$.

证明A是可判定的。

不可判定问题举例

Hilbert第十问题: “多项式是否有整数根”有没有算法?

1970’s 被证明不可判定 (没有判定器, 即没有算法)

M = “对于输入 “p”, p是k元多项式,

1. 取k个整数的向量x (绝对值和从小到大)

2. 若p(x) = 0, 则停机接受.

3. 否则转1. ”

这个图灵机对输入 $p(x,y) = x^2+y^2-3$不停机

对比: 一个可判定问题

一元多项式是否有整数根?

M = “对于输入“p”, k次1元多项式p(x),

1. 计算解的绝对值上界N
2. 对所有|x|≤N
3.   若p(x) = 0, 则停机接受.
4. 停机拒绝. ”