3 The Church-Turing Thesis

Yajun Yang

yjyang@tju.edu.cn

School of Computer Science and Technology
Tianjin University

2016

Outline

1. Turing Machines
2. Variants of Turing Machines
3. The Definition of Algorithm

Outline

1 Turing Machines
- Formal Definition of a Turing Machine
- Examples of Turing Machines

2 Variants of Turing Machines

3 The Definition of Algorithm

Turing Machines

***Turing machine* 图灵机**

- first proposed by Alan Turing in 1936

Turing Machines

***Turing machine* 图灵机**

- first proposed by Alan Turing in 1936
- a much more powerful model than DFA/NFA and PDA

Turing Machines

***Turing machine* 图灵机**

- first proposed by Alan Turing in 1936
- a much more powerful model than DFA/NFA and PDA
- with an unlimited and unrestricted memory

Turing Machines

***Turing machine* 图灵机**

- first proposed by Alan Turing in 1936
- a much more powerful model than DFA/NFA and PDA
- with an unlimited and unrestricted memory
- a much more accurate model of a general purpose computer

Turing Machines

***Turing machine* 图灵机**

- first proposed by Alan Turing in 1936
- a much more powerful model than DFA/NFA and PDA
- with an unlimited and unrestricted memory
- a much more accurate model of a general purpose computer
- can do everything that a real computer can do

Turing Machines

***Turing machine* 图灵机**

- first proposed by Alan Turing in 1936
- a much more powerful model than DFA/NFA and PDA
- with an unlimited and unrestricted memory
- a much more accurate model of a general purpose computer
- can do everything that a real computer can do

Even a Turing machine cannot solve certain problems

- these problems are beyond the theoretical limits of computation

艾伦·图灵（Alan Turing）

艾伦·图灵（Alan Turing）

June 23, 1912 – June 7, 1954 (aged 41)

英国数学家、逻辑学家

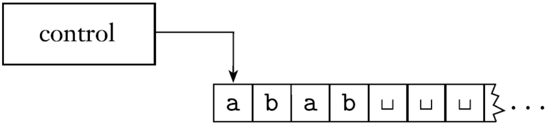
“计算机科学之父”

“人工智能之父”

Turing Machines

The Turing machine model uses an infinite tape as its unlimited memory.

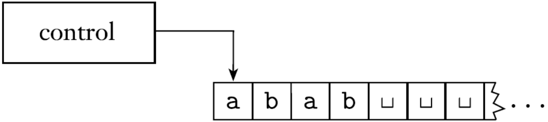
- It has a tape head that can read and write symbols and move around on the tape.
- Initially the tape contains only the input string and is blank everywhere else.
- If the machine needs to store information, it may write this information on the tape.



Turing Machines

The Turing machine model uses an infinite tape as its unlimited memory.

- To read the information that it has written, the machine can move its head back over it.
- The machine continues computing until it decides to produce an output.
- The outputs ***accept*** and ***reject*** are obtained by entering designated accepting and rejecting states.
- If it doesn't enter an accepting or a rejecting state, it will go on forever, never ***halting***.



Turing Machines

The differences between finite automata and Turing machines

1. A Turing machine can both write on the tape and read from it.
2. The read–write head can move both to the left and to the right.
3. The tape is infinite.
4. The special states for rejecting and accepting take effect immediately.

Turing Machines

Example (Turing machine M_1)

Let's introduce a Turing machine M_1 for testing membership in the language

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

We want M_1 to accept if its input is a member of B and to reject otherwise.

Turing Machines

Example (Turing machine M_1)

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

On input string w:

1. Zig-zag across the tape to corresponding positions on either side of the $\#$ symbol to check whether these positions contain the same symbol. If they do not, or if no $\#$ is found, reject. Cross off symbols as they are checked to keep track of which symbols correspond.

Turing Machines

Example (Turing machine M_1)

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

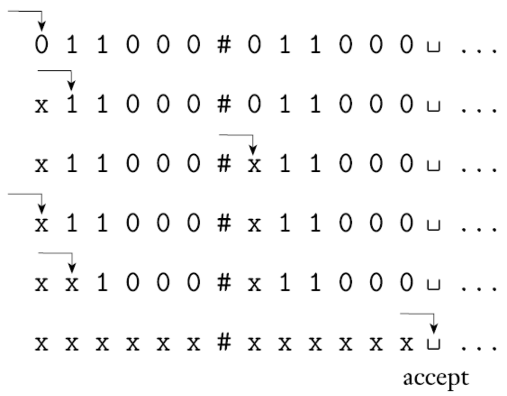
On input string w:

1. Zig-zag across the tape to corresponding positions on either side of the $\#$ symbol to check whether these positions contain the same symbol. If they do not, or if no $\#$ is found, reject. Cross off symbols as they are checked to keep track of which symbols correspond.
2. When all symbols to the left of the $\#$ have been crossed off, check for any remaining symbols to the right of the $\#$. If any symbols remain, reject; otherwise, accept.

Turing Machines

Example (Turing machine M_1)

several nonconsecutive snapshots of M_1's tape after it is started on input $011000\#011000$.



Formal Definition of a Turing Machine

Definition (TM (图灵机))

Formal Definition of a Turing Machine

Definition (TM (图灵机))

A ***Turing machine*** 图灵机 (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

Formal Definition of a Turing Machine

Definition (TM (图灵机))

A ***Turing machine*** 图灵机 (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of ***states***,

Formal Definition of a Turing Machine

Definition (TM (图灵机))

A ***Turing machine*** 图灵机 (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of ***states***,
2. Σ is the ***input alphabet*** not containing the ***blank symbol*** $\sqcup$,

Formal Definition of a Turing Machine

Definition (TM (图灵机))

A ***Turing machine* 图灵机** (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of ***states***,
2. Σ is the ***input alphabet*** not containing the ***blank symbol*** $\sqcup$,
3. Γ is the ***tape alphabet***, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,

Formal Definition of a Turing Machine

Definition (TM (图灵机))

A ***Turing machine*** 图灵机 (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{accept}}, q_{\mathsf{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of ***states***,
2. Σ is the ***input alphabet*** not containing the ***blank symbol*** $\sqcup$,
3. Γ is the ***tape alphabet***, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$ is the ***transition function***,

Formal Definition of a Turing Machine

Definition (TM (图灵机))

A ***Turing machine* 图灵机** (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of ***states***,
2. Σ is the ***input alphabet*** not containing the ***blank symbol*** $\sqcup$,
3. Γ is the ***tape alphabet***, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$ is the ***transition function***,
5. $q_0 \in Q$ is the ***start state***

Formal Definition of a Turing Machine

Definition (TM (图灵机))

A ***Turing machine* 图灵机** (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of ***states***,
2. Σ is the ***input alphabet*** not containing the ***blank symbol*** $\sqcup$,
3. Γ is the ***tape alphabet***, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$ is the ***transition function***,
5. $q_0 \in Q$ is the ***start state***
6. $q_{\text{accept}} \in Q$ is the ***accept state***, and

Formal Definition of a Turing Machine

Definition (TM (图灵机))

A ***Turing machine*** 图灵机 (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of ***states***,
2. Σ is the ***input alphabet*** not containing the ***blank symbol*** $\sqcup$,
3. Γ is the ***tape alphabet***, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$ is the ***transition function***,
5. $q_0 \in Q$ is the ***start state***
6. $q_{\text{accept}} \in Q$ is the ***accept state***, and
7. $q_{\text{reject}} \in Q$ is the ***reject state***, where $q_{\text{accept}} \neq q_{\text{reject}}$.

Configurations of a Turing Machine

A ***configuration*** of the Turing machine

Configurations of a Turing Machine

A ***configuration*** of the Turing machine

- the current state

Configurations of a Turing Machine

A ***configuration*** of the Turing machine

- the current state
- the current tape contents

Configurations of a Turing Machine

A ***configuration*** of the Turing machine

- the current state
- the current tape contents
- the current head location

A configuration of the Turing machine

uqv

- the current state is q
- the current tape contents is uv
- the current head location is the first symbol of v

The tape contains only blanks following the last symbol of v.

Configurations of a Turing Machine

A ***configuration*** of the Turing machine

- the current state
- the current tape contents
- the current head location

A configuration of the Turing machine

uqv

- the current state is q
- the current tape contents is uv
- the current head location is the first symbol of v

The tape contains only blanks following the last symbol of v. $1011q_701111$

Configurations of a Turing Machine

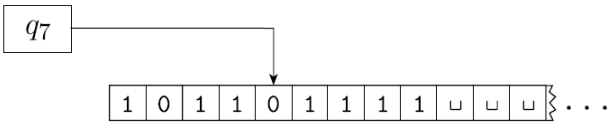
A ***configuration* 格局** of the Turing machine

A configuration of the Turing machine

uqv

- the current state is q
- the current tape contents is uv
- the current head location is the first symbol of v

The tape contains only blanks following the last symbol of v. $1011q_701111$



How does a Turing machine compute

Configuration C_1 ***yields* 产生** configuration C_2

- if the Turing machine can legally go from C_1 to C_2 in a single step.

Formalization

Suppose that $a, b \in \Gamma$, $u, v \in \Gamma^*$, $q_i, q_j \in Q$

- uaq_ibv yields uq_jacv

if $\delta(q_i, b) = (q_j, c, \mathsf{L})$

- uaq_ibv yields $uacq_jv$ if $\delta(q_i, b) = (q_j, c, \mathsf{R})$

How does a Turing machine compute

Configuration C_1 ***yields*** configuration C_2

- if the Turing machine can legally go from C_1 to C_2 in a single step.

Formalization

Special cases occur when the head is at one of the ends.

- For the left-hand end
  - left-moving: $q_i bv$ yields $q_j cv$
  - right-moving: $q_i bv$ yields $cq_j v$
- For the right-hand end
  - uaq_i is equivalent to $uaq_i\sqcup$

How does a Turing machine compute

TM M on input w

- ***start configuration***: $q_0 w$
- ***accepting configuration***: $\cdots q_{\text{accept}} \cdots$
- ***rejecting configuration***: $\cdots q_{\text{reject}} \cdots$

Accepting and rejecting configurations are ***halting configurations*** and do not yield further configurations.

How does a Turing machine compute

TM M on input w

- ***start configuration***: $q_0 w$
- ***accepting configuration***: $\cdots q_{\text{accept}} \cdots$
- ***rejecting configuration***: $\cdots q_{\text{reject}} \cdots$

Accepting and rejecting configurations are ***halting configurations*** and do not yield further configurations.

Because the machine is defined to halt when in q_{accept} and q_{reject}, we equivalently could have defined the transition function to have the more complicated form

- $\delta : Q' \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$, where $Q' = Q - \{q_{\text{accept}}, q_{\text{reject}}\}$

How does a Turing machine compute

A Turing machine M ***accepts*** input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where

- C_1 is the start configuration of M on input w,
- each C_i yields C_{i+1}, and
- C_k is an accepting configuration.

The collection of strings that M accepts is ***the language of*** M, or ***the language recognized by*** M, denoted $L(M)$.

Turing-recognizable

Definition (Turing-recognizable)

Call a language ***Turing-recognizable*** 图灵可识别的 if some Turing machine recognizes it.

Turing-recognizable

Definition (Turing-recognizable)

Call a language ***Turing-recognizable*** **图灵可识别的** if some Turing machine recognizes it.

(It is called a ***recursively enumerable language*** **递归可枚举语言**.)

When we start a Turing machine on an input, three outcomes are possible.

1. accept
2. reject
3. loop

By ***loop*** we mean that the machine simply does not halt.

Turing-decidable

A Turing machine M can fail to accept an input by entering the q_{reject} state and rejecting, or by looping

Turing-decidable

A Turing machine M can fail to accept an input by entering the q_{reject} state and rejecting, or by looping

- Sometimes distinguishing amachine that is looping from one that is merely taking a long time is difficult.

Turing-decidable

A Turing machine M can fail to accept an input by entering the q_{reject} state and rejecting, or by looping

- Sometimes distinguishing amachine that is looping from one that is merely taking a long time is difficult.

For this reason, we prefer Turing machines that halt on all inputs; such machines never loop.

Turing-decidable

A Turing machine M can fail to accept an input by entering the q_{reject} state and rejecting, or by looping

- Sometimes distinguishing amachine that is looping from one that is merely taking a long time is difficult.

For this reason, we prefer Turing machines that halt on all inputs; such machines never loop.

- These machines are called ***deciders*** because they always make a decision to accept or reject.

Turing-decidable

A Turing machine M can fail to accept an input by entering the q_{reject} state and rejecting, or by looping

- Sometimes distinguishing amachine that is looping from one that is merely taking a long time is difficult.

For this reason, we prefer Turing machines that halt on all inputs; such machines never loop.

- These machines are called ***deciders*** because they always make a decision to accept or reject.
- A decider that recognizes some language also is said to ***decide*** that language.

Turing-decidable

Turing-decidable

- These machines are called ***deciders*** because they always make a decision to accept or reject.

Turing-decidable

- These machines are called ***deciders*** because they always make a decision to accept or reject.
- A decider that recognizes some language also is said to ***decide*** that language.

Turing-decidable

- These machines are called ***deciders*** because they always make a decision to accept or reject.
- A decider that recognizes some language also is said to ***decide*** that language.

Definition (Turing-decidable)

Call a language ***Turing-decidable*** 图灵可判定的 or simply ***decidable*** 可判定的 if some Turing machine decides it.

Turing-decidable

- These machines are called ***deciders*** because they always make a decision to accept or reject.
- A decider that recognizes some language also is said to ***decide*** that language.

Definition (Turing-decidable)

Call a language ***Turing-decidable*** 图灵可判定的 or simply ***decidable*** 可判定的 if some Turing machine decides it.

(It is called a ***recursively language*** 递归语言.)

Turing-decidable

- These machines are called ***deciders*** because they always make a decision to accept or reject.
- A decider that recognizes some language also is said to ***decide*** that language.

Definition (Turing-decidable)

Call a language ***Turing-decidable*** 图灵可判定的 or simply ***decidable*** 可判定的 if some Turing machine decides it.

(It is called a ***recursively language*** **递归语言**.)

- Every decidable language is Turing-recognizable.

Relationship among the classes of languages

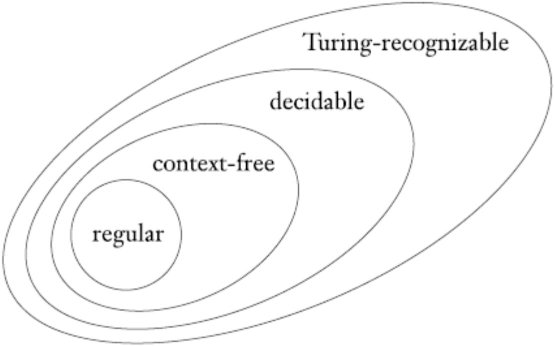
Theorem

Every context-free language is decidable.

Relationship among the classes of languages

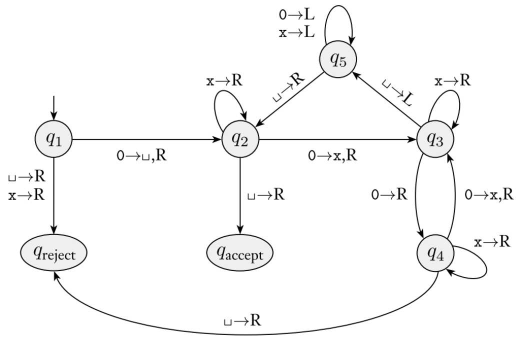
Theorem

Every context-free language is decidable.



Example (TM M_2 that decides $A = \{0^{2^n} | n \geq 0\}$, Input: 0000)

Example (TM M_2 that decides $A = \{0^{2^n} | n \geq 0\}$, Input: 0000)

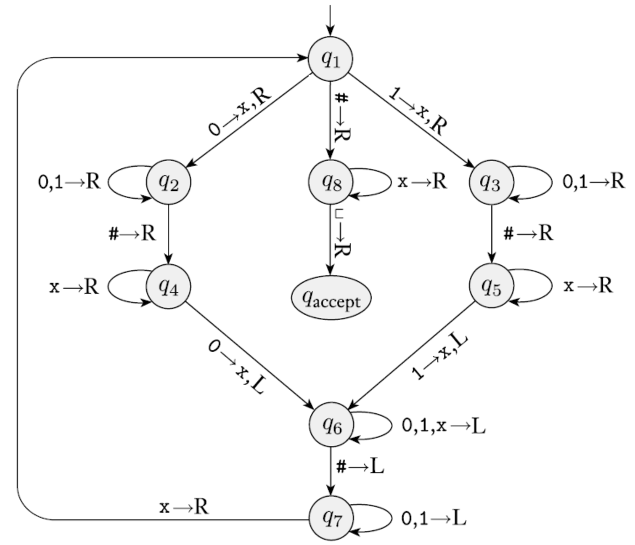


$q_1 0000$
$\sqcup q_2 000$
$\sqcup \mathrm{x} q_3 00$
$\sqcup \mathrm{x} 0 q_4 0$
$\sqcup \mathrm{x} 0 \mathrm{x} q_3 \sqcup$
$\sqcup \mathrm{x} 0 q_5 \mathrm{x} \sqcup$
$\sqcup \mathrm{x} q_5 0 \mathrm{x} \sqcup$

$\sqcup q_5 \mathrm{x} 0 \mathrm{x} \sqcup$
$q_5 \sqcup \mathrm{x} 0 \mathrm{x} \sqcup$
$\sqcup q_2 \mathrm{x} 0 \mathrm{x} \sqcup$
$\sqcup \mathrm{x} q_2 0 \mathrm{x} \sqcup$
$\sqcup \mathrm{xx} q_3 \mathrm{x} \sqcup$
$\sqcup \mathrm{xxx} q_3 \sqcup$
$\sqcup \mathrm{xx} q_5 \mathrm{x} \sqcup$

$\sqcup \mathrm{x} q_5 \mathrm{xx} \sqcup$
$\sqcup q_5 \mathrm{xxx} \sqcup$
$q_5 \sqcup \mathrm{xxx} \sqcup$
$\sqcup q_2 \mathrm{xxx} \sqcup$
$\sqcup \mathrm{x} q_2 \mathrm{xx} \sqcup$
$\sqcup \mathrm{xx} q_2 \mathrm{x} \sqcup$
$\sqcup \mathrm{xxx} q_2 \sqcup$
$\sqcup \mathrm{xxx} \sqcup q_{accept}$

Example (TM M_1)



Outline

1 Turing Machines

2 Variants of Turing Machines

- Multitape Turing Machines
- Nondeterministic Turing Machines 非确定型图灵机
- Enumerators
- Equivalence With Other Models

3 The Definition of Algorithm

Multitape Turing Machines 多带图灵机

多带图灵机

- A *multitape Turing machine* is like an ordinary Turing machine with several tapes. Each tape has its own head for reading and writing. Initially the input appears on tape 1, and the others start out blank.

Multitape Turing Machines 多带图灵机

多带图灵机

- A *multitape Turing machine* is like an ordinary Turing machine with several tapes. Each tape has its own head for reading and writing. Initially the input appears on tape 1, and the others start out blank.
- **转移函数：** $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$, where k is the number of tapes.

Multitape Turing Machines 多带图灵机

多带图灵机

- A *multitape Turing machine* is like an ordinary Turing machine with several tapes. Each tape has its own head for reading and writing. Initially the input appears on tape 1, and the others start out blank.
- **转移函数：** $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$, where k is the number of tapes.
- $\delta(q_i, a_1, \cdots, a_k) = (q_j, b_1, \cdots, b_k, L, R, \cdots, L)$

Multitape Turing Machines 多带图灵机

多带图灵机

- A *multitape Turing machine* is like an ordinary Turing machine with several tapes. Each tape has its own head for reading and writing. Initially the input appears on tape 1, and the others start out blank.
- **转移函数：** $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$, where k is the number of tapes.
- $\delta(q_i, a_1, \cdots, a_k) = (q_j, b_1, \cdots, b_k, L, R, \cdots, L)$

Theorem

Every multitape Turing machine has an equivalent single-tape Turing machine.

Multitape Turing Machines 多带图灵机

Theorem

Every multitape Turing machine has an equivalent single-tape Turing machine.

Multitape Turing Machines 多带图灵机

Theorem

Every multitape Turing machine has an equivalent single-tape Turing machine.

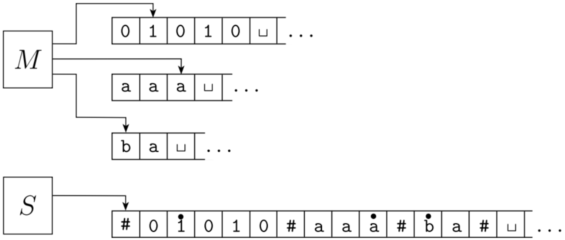
Proof idea: Convert a multi-tape TM M to an equivalent single-tape TM S. The key point is to simulate M with S.

Multitape Turing Machines 多带图灵机

Theorem

Every multitape Turing machine has an equivalent single-tape Turing machine.

Proof idea: Convert a multi-tape TM M to an equivalent single-tape TM S. The key point is to simulate M with S.



Nondeterministic Turing Machines

- $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\})$

Theorem

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Nondeterministic Turing Machines

Theorem

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Nondeterministic Turing Machines

Theorem

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

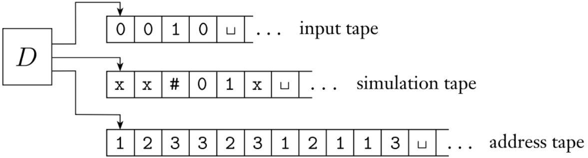
Proof idea: Simulate any nondeterministic TM N with a deterministic TM D. Try all possible branches of N's nondeterministic computation. If D ever finds the accept state on one of these branches, D accepts. Otherwise, D's simulation will not terminate.

Nondeterministic Turing Machines

Theorem

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Proof idea: Simulate any nondeterministic TM N with a deterministic TM D. Try all possible branches of N's nondeterministic computation. If D ever finds the accept state on one of these branches, D accepts. Otherwise, D's simulation will not terminate.



Enumerators 枚举器

枚举器是一个k带图灵机，最后一带作为输出带。

Enumerators 枚举器

枚举器是一个k带图灵机，最后一带作为输出带。枚举器产生的语言：

Enumerators 枚举器

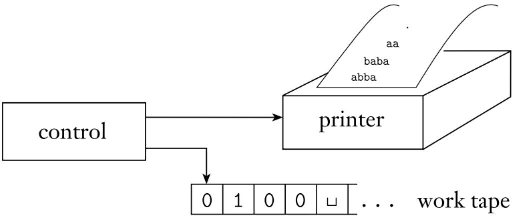
枚举器是一个k带图灵机，最后一带作为输出带。枚举器产生的语言：

$L(M) = \{w | w \in \Sigma^*, w\text{能被}M\text{打印在输出带上}, \# \notin \Sigma\}$

Enumerators 枚举器

枚举器是一个k带图灵机，最后一带作为输出带。枚举器产生的语言：

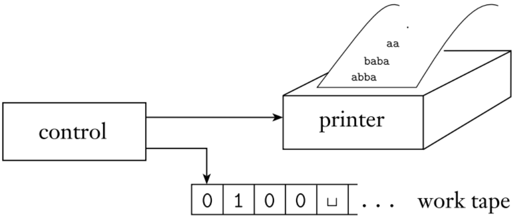
$L(M) = \{w|w \in \Sigma^*, w\text{能被}M\text{打印在输出带上}, \# \notin \Sigma\}$



Enumerators 枚举器

枚举器是一个k带图灵机，最后一带作为输出带。枚举器产生的语言：

$L(M) = \{w|w \in \Sigma^*, w\text{能被}M\text{打印在输出带上}, \# \notin \Sigma\}$



Theorem

A language is Turing-recognizable if and only if some enumerator enumerates it.

Outline

1. Turing Machines

2. Variants of Turing Machines

3. The Definition of Algorithm
   - Hilbert's Problems

Church–Turing thesis

Intuitive notion of algorithms	equals	*Turing machine algorithms*

The Church–Turing Thesis 邱奇-图灵论题