Mathematical Foundation of Computer Sciences III

Turing Machine

Guoqiang Li

School of Software, Shanghai Jiao Tong University

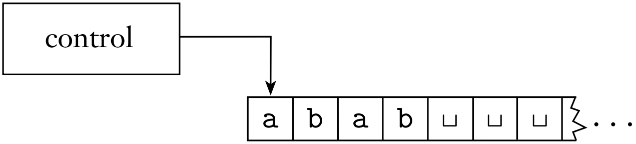
Turing Machine

Turing machines

Alan Turing in 1936 proposed Turing machines M:

- M uses an infinite tape as its unlimited memory, with a tape head reading and writing symbols and moving around on the tape. The tape initially contains only the input string and is blank everywhere else.
- If M needs to store information, it may write this information on the tape. To read the information that it has written, M can move its head back over it.
- M continues computing until it decides to produce an output. The outputs accept and reject are obtained by entering designated accepting and rejecting states.
- If M doesn't enter an accepting or a rejecting state, it will go on forever, never halting.

Schematic of a Turing machine



The difference 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.

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

zig-zag:是判断两段有特殊位置关系的字符串是否相等的常用方法。现在特定符号左侧某个位置找一个字符并标记，之后移动到符号右边对应位置再找一个相同的字符并做相同的标记，之后回到左边标记好的字符的下个字符处并重复上述过程。

M_1 on input string w:

1. Zig-zag across the tape to corresponding positions on either side of the $\sharp$ symbol to check whether these positions contain the same symbol. If they do not, or if no $\sharp$ 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 $\sharp$ have been crossed off, check for any remaining symbols to the right of the $\sharp$. If any symbols remain, reject; otherwise, accept.

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

0 1 1 0 0 0 # 0 1 1 0 0 0 ⊔ . . .

x 1 1 0 0 0 # 0 1 1 0 0 0 ⊔ . . .

x 1 1 0 0 0 # x 1 1 0 0 0 ⊔ . . .

x 1 1 0 0 0 # x 1 1 0 0 0 ⊔ . . .

x x 1 0 0 0 # x 1 1 0 0 0 ⊔ . . .

x x x x x x # x x x x x x ⊔ . . .

accept

Formal definition of a Turing machine

Definition

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

- Q is set of states,
- Σ is the input alphabet not containing the blank symbol $\sqcup$,
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
- $q_0 \in Q$ is the start state,
- $q_{accept} \in Q$ is the accept state, and
- $q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$.

Computation by M

Initially, M receives its input $w = w_1 w_2 \dots w_n \in \Sigma^*$ on the leftmost n squares of the tape, and the rest of the tape is blank (i.e., filled $\sqcup$).

The head starts on the leftmost square of the tape.

As Σ does not contain $\sqcup$, so the first blank appearing on the tape marks the end of the input.

Once M has started, the computation proceeds according to the rules described by the transition function.

If M ever tries to move its head to the left off the left-hand end of the tape, the head stays in the same place for that move, even though the transition function indicates L.

The computation continues until it enters either the accept or reject states, at which point it halts. If neither occurs, M goes on forever.

Configurations

A configuration of a Turing machine consists of

- the current state,
- the current tape contents, and
- the current head location.

当前指针指向的字符，并写在该指针右面

Configurations

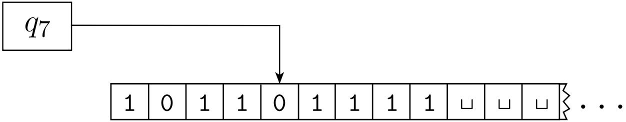
A configuration of a Turing machine consists of

- the current state,
- the current tape contents, and
- the current head location.

By $u\,q\,v$ we mean the configuration where

- the current state is q,
- the current tape contents is uv, and
- the current head location is the first symbol of v.
- The tape contains only blanks following the last symbol of v.

Configurations



A Turing machine with configuration 1011 q_7 01111

Formal definition of computation

Let $a, b, c \in \Gamma$, $u, v \in \Gamma^*$, and $q_i, q_j \in Q$.

1. If $\delta(q_i, b) = (q_j, c, L)$, then
$$ua\, q_i\, bv \text{ yields } u\, q_j\, acv \qquad .$$
2. If $\delta(q_i, b) = (q_j, c, R)$, then
$$ua\, q_i\, bv \text{ yields } uac\, q_j\, v \qquad .$$

Formal definition of computation

Let $a, b, c \in \Gamma$, $u, v \in \Gamma^*$, and $q_i, q_j \in Q$.

1. If $\delta(q_i, b) = (q_j, c, L)$, then
$$ua\, q_i\, bv \text{ yields } u\, q_j\, acv \quad .$$
2. If $\delta(q_i, b) = (q_j, c, R)$, then
$$ua\, q_i\, bv \text{ yields } uac\, q_j\, v \quad .$$

Special cases occur when the head is at one of the ends of the configuration:

1. For the left-hand end, the configuration $q_i\, bv$ yields $q_j\, cv$ if the transition is left moving (because we prevent the machine from going off the left-hand end of the tape), and it yields $c\, q_j\, v$ for the right-moving transition. 意思是说移动到最左端之后只能进行字符的重写
但是在不能向左移动
2. For the right-hand end, the configuration $ua\, q_i$ is equivalent to $ua\, q_i \sqcup$ because we assume that blanks follow the part of the tape represented in the configuration. 在此课件中的TM中，默认TM接受的字符
都是以空格符号结尾的。

Special configurations

The start configuration of M on input w is the configuration $q_0\, w$.

In an accepting configuration, the state of the configuration is q_{accept}.

In a rejecting configuration, the state of the configuration is q_{reject}.

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

Formal definition of computation

M accepts w if there are sequence of configurations $C_1, C_2, \ldots, C_k$ such that

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

Formal definition of computation

M accepts w if there are sequence of configurations $C_1, C_2, \ldots, C_k$ such that

- C_1 the start configuration of M on 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 languages

Definition

A language is Turing-recognizable, if some Turing machine recognizes it.

Turing-decidable languages

On an input, the machine M may accept, reject, or loop. By loop we mean that the machine simply does not halt.

Turing-decidable languages

On an input, the machine M may accept, reject, or loop. By loop we mean that the machine simply does not halt.

If M always halts, then it is a decider. A decider that recognizes some language is said to decide that language.

Turing-decidable languages

On an input, the machine M may accept, reject, or loop. By loop we mean that the machine simply does not halt.

If M always halts, then it is a decider. A decider that recognizes some language is said to decide that language.

Definition

A language is Turing-decidable or simply decidable if some Turing machine decides it.

Example: $A = \{0^{2^n} \mid n \geq 0\}$

On input string w:

1. Sweep left to right across the tape, crossing off every other 0.
2. If in stage 1 the tape contained a single 0, accept.
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, reject.
4. Return the head to the left-hand end of the tape.
5. Go to stage 1.

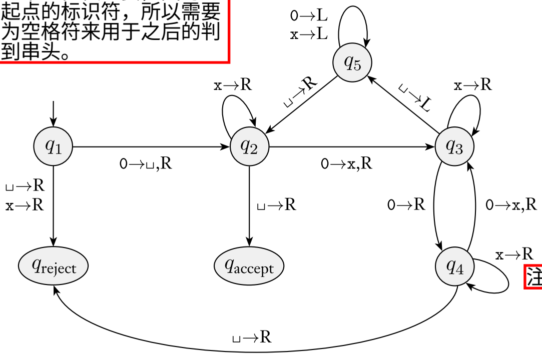
Example: $A = \{0^{2^n} \mid n \geq 0\}$

$Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$, where q_1 is the start state.

$\Sigma = \{0\}$ and $\Gamma = \{0, x, \sqcup\}$.

The transition function δ:

此处从q1出来的第一步很重要，因为此图灵机并没有起点的标识符，所以需要使得首字符变为空格符来用于之后的判断是否已经回到串头。



注意此处的q4

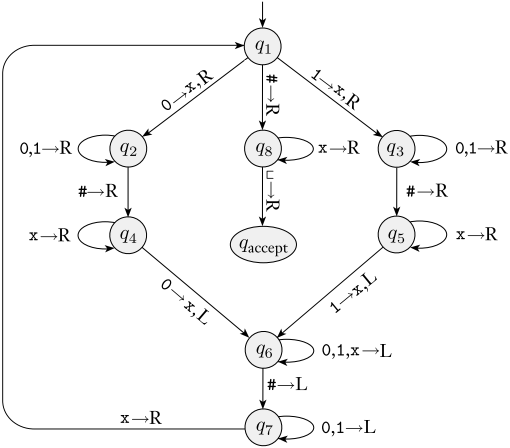
Example: $B = \{w\sharp w \mid w \in \{0,1\}^*\}$

Example: $B = \{w\sharp w \mid w \in \{0,1\}^*\}$

$Q = \{q_1, q_2, \ldots, q_8, q_{accept}, q_{reject}\}$, where q_1 is the start state.

$\Sigma = \{0, 1, \sharp\}$ and $\Gamma = \{0, 1, \sharp, x, \sqcup\}$.

The transition function δ:



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

0 1 1 0 0 0 # 0 1 1 0 0 0 $\sqcup$ . . .

x 1 1 0 0 0 # 0 1 1 0 0 0 $\sqcup$ . . .

x 1 1 0 0 0 # x 1 1 0 0 0 $\sqcup$ . . .

x 1 1 0 0 0 # x 1 1 0 0 0 $\sqcup$ . . .

x x 1 0 0 0 # x 1 1 0 0 0 $\sqcup$ . . .

x x x x x x # x x x x x x $\sqcup$ . . .

accept

Example: $C = \{a^i b^j c^k \mid i, j, k \geq 1$ and $i \times j = k\}$

Example: $C = \{a^i b^j c^k \mid i, j, k \geq 1$ and $i \times j = k\}$

On input string w:

1. Scan the input from left to right to determine whether it is a member of $a^+ b^+ c^+$ and reject if it is not.

Example: $C = \{a^i b^j c^k \mid i, j, k \geq 1$ **and** $i \times j = k\}$

On input string w:

1. Scan the input from left to right to determine whether it is a member of $a^+ b^+ c^+$ and reject if it is not.
2. Return the head to the left-hand end of the tape.

Example: $C = \{a^i b^j c^k \mid i, j, k \geq 1$ and $i \times j = k\}$

On input string w:

1. Scan the input from left to right to determine whether it is a member of $a^+ b^+ c^+$ and reject if it is not.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and the c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, reject.

Example: $C = \{a^i b^j c^k \mid i, j, k \geq 1 \text{ and } i \times j = k\}$

On input string w:

1. Scan the input from left to right to determine whether it is a member of $a^+ b^+ c^+$ and reject if it is not.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and the c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, reject.
4. Restore the crossed off b's and repeat stage 3 if there is another a to cross off. If all a's have been crossed off, determine whether all c's also have been crossed off. If yes, accept; otherwise, reject.

此问题是典型的抽象语言的证明，及直接通过自然语言来描述一个复杂的自动机的实现过程。

本问题的核心思想：因为i*j = k;
故对于每找到的一个a，都应该有|b|=|c|

Example: $E = \{\sharp x_1 \sharp \dots \sharp x_\ell \mid$ each $x_i \in \{0,1\}^*$ and $x_i \neq x_j$ for each $i \neq j\}$

Example: $E = \{\sharp x_1 \sharp \ldots \sharp x_\ell \mid$ each $x_i \in \{0,1\}^*$ and $x_i \neq x_j$ for each $i \neq j\}$

On input string w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, accept. If that symbol was a $\sharp$, continue with the next stage. Otherwise, reject.

Example: $E = \{\sharp x_1 \sharp \ldots \sharp x_\ell \mid$ each $x_i \in \{0,1\}^*$ and $x_i \neq x_j$ for each $i \neq j\}$

On input string w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, accept. If that symbol was a $\sharp$, continue with the next stage. Otherwise, reject.
2. Scan right to the next $\sharp$ and place a second mark on top of it. If no $\sharp$ is encountered before a blank symbol, only x_1 was present, so accept.

Example: $E = \{\sharp x_1 \sharp \ldots \sharp x_\ell \mid$ each $x_i \in \{0,1\}^*$ and $x_i \neq x_j$ for each $i \neq j\}$

On input string w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, accept. If that symbol was a $\sharp$, continue with the next stage. Otherwise, reject.
2. Scan right to the next $\sharp$ and place a second mark on top of it. If no $\sharp$ is encountered before a blank symbol, only x_1 was present, so accept.
3. By zig-zagging, compare the two strings to the right of the marked $\sharp$s. If they are equal, reject.

Example: $E = \{\sharp x_1 \sharp \ldots \sharp x_\ell \mid$ each $x_i \in \{0,1\}^*$ and $x_i \neq x_j$ for each $i \neq j\}$

On input string w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, accept. If that symbol was a $\sharp$, continue with the next stage. Otherwise, reject.
2. Scan right to the next $\sharp$ and place a second mark on top of it. If no $\sharp$ is encountered before a blank symbol, only x_1 was present, so accept.
3. By zig-zagging, compare the two strings to the right of the marked $\sharp$s. If they are equal, reject.
4. Move the rightmost of the two marks to the next $\sharp$ symbol to the right. If no $\sharp$ symbol is encountered before a blank symbol, move the leftmost mark to the next $\sharp$ to its right and the rightmost mark to the $\sharp$ after that. This time, if no $\sharp$ is available for the rightmost mark, all the strings have been compared, so accept.

Example: $E = \{\sharp x_1 \sharp \ldots \sharp x_\ell \mid$ each $x_i \in \{0,1\}^*$ and $x_i \neq x_j$ for each $i \neq j\}$

On input string w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, accept. If that symbol was a $\sharp$, continue with the next stage. Otherwise, reject.
2. Scan right to the next $\sharp$ and place a second mark on top of it. If no $\sharp$ is encountered before a blank symbol, only x_1 was present, so accept.
3. By zig-zagging, compare the two strings to the right of the marked $\sharp$s. If they are equal, reject.
4. Move the rightmost of the two marks to the next $\sharp$ symbol to the right. If no $\sharp$ symbol is encountered before a blank symbol, move the leftmost mark to the next $\sharp$ to its right and the rightmost mark to the $\sharp$ after that. This time, if no $\sharp$ is available for the rightmost mark, all the strings have been compared, so accept.
5. Go to stage 3.

Summery

Languages A, B, C, and E are decidable.

All decidable languages are Turing-recognizable, so these languages are also Turing-recognizable.

Demonstrating a language that is Turing-recognizable but undecidable is more difficult.

Variants of Turing Machines

Multitape Turing machines

多带图灵机：

A multitape Turing machine M has several tapes:

- Each tape has its own head for reading and writing.
- The input is initially on tape 1, with all the other tapes being blank.
- The transition function is

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, RS\}^k$$

where k is the number of tapes. 多带TM可以选择在某个位置stay保持不动

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, L, R, \ldots, L)$$

means that if M is in state q_i and heads 1 through k are reading symbols a_1 through a_k, the machine goes to state q_j, writes symbols b_1 through b_k, and directs each head to move left or right, or to stay put, as specified.

Equivalence

Theorem

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

单带图灵机就是一种特殊的多带图灵机

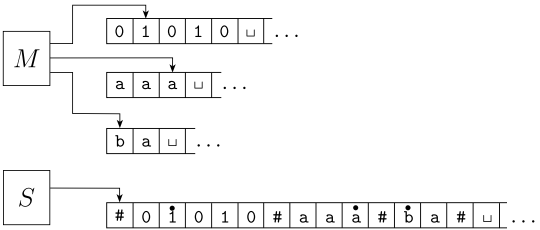
Proof

We simulate an M with k tapes by a single-tape S.

Proof

We simulate an M with k tapes by a single-tape S.

- S uses $\sharp$ to separate the contents of the different tapes.
- S keeps track of the locations of the heads by writing a tape symbol with a dot above it to mark the place where the head on that tape would be.



Proof

On input $w = w_1 \dots w_n$:

1. First S puts its tape into the format that represents all k tapes of M:

$$\sharp \dot{w}_1 w_2 \dots w_n \sharp \dot{\sqcup} \sharp \dot{\sqcup} \sharp \dots \sharp$$

通过#号将每个tape上面的内容连接起来，初始时第二条tape开始的纸带都为空格符。并且最后一个tape的最后也以#结尾作为结束的标识符。

Proof

On input $w = w_1 \ldots w_n$:

1. First S puts its tape into the format that represents all k tapes of M:

$$\sharp \dot{w}_1 w_2 \ldots w_n \sharp \dot{\sqcup} \sharp \dot{\sqcup} \sharp \ldots \sharp$$

2. To determine the symbols under the virtual heads, S scans its tape from the first $\sharp$, which marks the left-hand end, to the $k+1$st $\sharp$, which marks the right-hand end.

Proof

On input $w = w_1 \dots w_n$:

1. First S puts its tape into the format that represents all k tapes of M:

$$\sharp \dot{w}_1 w_2 \dots w_n \sharp \dot{\sqcup} \sharp \dot{\sqcup} \sharp \dots \sharp$$

2. To determine the symbols under the virtual heads, S scans its tape from the first $\sharp$, which marks the left-hand end, to the $k+1$st $\sharp$, which marks the right-hand end.
3. Then S makes a second pass to update the tapes according to the way that Ms transition function dictates.

Proof

On input $w = w_1 \ldots w_n$:

1. First S puts its tape into the format that represents all k tapes of M:

$$\sharp \dot{w}_1 w_2 \ldots w_n \sharp \dot{\sqcup} \sharp \dot{\sqcup} \sharp \ldots \sharp$$

2. To determine the symbols under the virtual heads, S scans its tape from the first $\sharp$, which marks the left-hand end, to the $k+1$st $\sharp$, which marks the right-hand end.
3. Then S makes a second pass to update the tapes according to the way that Ms transition function dictates.
4. If S moves one of the virtual heads to the right onto a $\sharp$, i.e., M has moved the corresponding head onto the previously unread blank portion of that tape. So S writes on this tape cell and shifts the tape contents, from this cell until the rightmost $\sharp$, one unit to the right.

Proof

On input $w = w_1 \dots w_n$:

1. First S puts its tape into the format that represents all k tapes of M:

$$\sharp \dot{w}_1 w_2 \dots w_n \sharp \dot{\sqcup} \sharp \dot{\sqcup} \sharp \dots \sharp$$

2. To determine the symbols under the virtual heads, S scans its tape from the first $\sharp$, which marks the left-hand end, to the $k+1$st $\sharp$, which marks the right-hand end.
3. Then S makes a second pass to update the tapes according to the way that Ms transition function dictates.
4. If S moves one of the virtual heads to the right onto a $\sharp$, i.e., M has moved the corresponding head onto the previously unread blank portion of that tape. So S writes on this tape cell and shifts the tape contents, from this cell until the rightmost $\sharp$, one unit to the right.
5. Go back to 2.

Corollary

Corollary

A language is Turing-recognizable if and only if some multitape Turing machine recognizes it.

Nondeterministic Turing Machines

$\boxed{\text{非确定图灵机：}}$

The transition function for a nondeterministic Turing machine has the form

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

The computation of a nondeterministic Turing machine is a tree whose branches correspond to different possibilities for the machine.

If some branch of the computation leads to the accept state, the machine accepts its input.

Equivalence

Theorem

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Proof

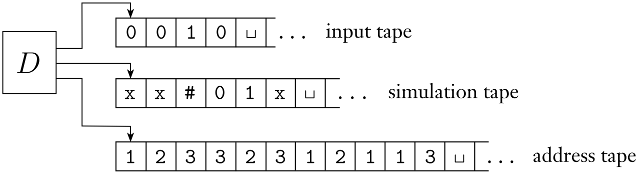
We simulate a nondeterministic N by a deterministic D.

Proof

We simulate a nondeterministic N by a deterministic D.

- D try all possible branches of N's nondeterministic computation.
- If D ever finds the accept state on one of these branches, it accepts.
- Otherwise, D's simulation will not terminate.

类似于广度优先遍历



Proof

1. Initially, tape 1 contains the input w, and tapes 2 and 3 are empty.

Proof

1. Initially, tape 1 contains the input w, and tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2 and initialize the string on tape 3 to be ε.

Proof

1. Initially, tape 1 contains the input w, and tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2 and initialize the string on tape 3 to be ε.
3. Use tape 2 to simulate N with input w on one branch of its nondeterministic computation.
   3.1 Before each step of N, consult the next symbol on tape 3 to determine which choice to make among those allowed by N's transition function.
   3.2 If no more symbols remain on tape 3 or if this nondeterministic choice is invalid, abort this branch by going to stage 4.
   3.3 Also go to stage 4 if a rejecting configuration is encountered. If an accepting configuration is encountered, accept the input.

Proof

1. Initially, tape 1 contains the input w, and tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2 and initialize the string on tape 3 to be ε.
3. Use tape 2 to simulate N with input w on one branch of its nondeterministic computation.
   3.1 Before each step of N, consult the next symbol on tape 3 to determine which choice to make among those allowed by N's transition function.
   3.2 If no more symbols remain on tape 3 or if this nondeterministic choice is invalid, abort this branch by going to stage 4.
   3.3 Also go to stage 4 if a rejecting configuration is encountered. If an accepting configuration is encountered, accept the input.
4. Replace the string on tape 3 with the next string in the string ordering. Simulate the next branch of Ns computation by going to stage 2.

Corollary

Corollary

A language is Turing-recognizable if and only if some nondeterministic Turing machine recognizes it.

Corollary

A language is decidable if and only if some nondeterministic Turing machine decides it.

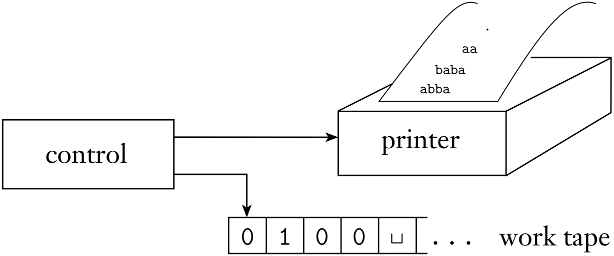
Enumerators

An enumerator is a Turing machine with an attached printer.

The Turing machine can use that printer as an output device to print strings.

Every time the Turing machine wants to add a string to the list, it sends the string to the printer.

Schematic of an enumerator



Equivalence

Theorem

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

Proof

Let E be an enumerator E that enumerates a language A. The desired M on input w:

Proof

Let E be an enumerator E that enumerates a language A. The desired M on input w:

- Run E. Every time that E outputs a string, compare it with w.
- If w ever appears in the output of E, then accept.

Proof

If M recognizes a language A, we can construct the following enumerator E for A. Let $s_1, s_2, s_3, \ldots,$, be a list of all possible strings in Σ^*.

1. Repeat the following for $i = 1, 2, 3, \ldots$
2. Run M for i steps on each input, $s_1, s_2, \ldots, s_i$.
3. If any computations accept, print out the corresponding s_j.

$\boxed{\text{广度优先遍历}}$

The Definition of Algorithm

Polynomials and their roots

A polynomial is a sum of terms, where each term is a product of certain variables and a constant, i.e., coefficient. For example,

系数

$$6 \cdot x \cdot x \cdot x \cdot y \cdot z \cdot z = 6x^3yz^2$$

is a term with coefficient 6, and

$$6x^3yz^2 + 3xy^2 - x^3 - 10$$

is a polynomial with four terms, over the variables x, y, and z.

Polynomials and their roots

A polynomial is a sum of terms, where each term is a product of certain variables and a constant, i.e., coefficient. For example,

$$6 \cdot x \cdot x \cdot x \cdot y \cdot z \cdot z = 6x^3yz^2$$

is a term with coefficient 6, and

$$6x^3yz^2 + 3xy^2 - x^3 - 10$$

is a polynomial with four terms, over the variables x, y, and z.

A root of a polynomial is an assignment of values to its variables so that the value of the polynomial is 0. This root is an integral root because all the variables are assigned integer values. Some polynomials have an integral root and some do not.

Hilbert's problems

Hilbert's tenth problem was to devise an algorithm that tests whether a polynomial has an integral root. He did not use the term algorithm but rather

> *a process according to which it can be determined by a finite number of operations.*

Church-Turing Thesis

In 1936 to formalize the definition of an algorithm:

- Alonzo Church proposed λ-calculus;
- Alan Turing proposed Turing machines,

which were shown to be equivalent.

Church-Turing Thesis

In 1936 to formalize the definition of an algorithm:

- Alonzo Church proposed λ-calculus;
- Alan Turing proposed Turing machines,

which were shown to be equivalent.

So we have the Church-Turing Thesis:

Intuitive notion of algorithms = Turing machine algorithms

Hilbert's Tenth Problem

$D = \{p \mid p$ is a polynomial with integer coefficients and with an integral root$\}$

Hilbert's Tenth Problem

$D = \{p \mid p \text{ is a polynomial with integer coefficients and with an integral root}\}$

Theorem

(Yuri Matijasevič, Martin Davis, Hilary Putnam, and Julia Robinson, 1970)
D *is not decidable.*

A simple variant

$$D_1 = \left\{ p \;\middle|\; \begin{array}{l} p \text{ is a polynomial on a single variable } x \text{ with integer} \\ \text{coefficients and with an integral root} \end{array} \right\}$$

A simple variant

$$D_1 = \left\{ p \;\middle|\; \begin{array}{l} p \text{ is a polynomial on a single variable } x \text{ with integer} \\ \text{coefficients and with an integral root} \end{array} \right\}$$

Lemma

Both D and D_1 are Turing-recognizable.

A simple variant

$$D_1 = \left\{ p \;\middle|\; \begin{array}{l} p \text{ is a polynomial on a single variable } x \text{ with integer} \\ \text{coefficients and with an integral root} \end{array} \right\}$$

Lemma

Both D and D_1 are Turing-recognizable.

Proof.

A simple variant

$$D_1 = \left\{ p \;\middle|\; \begin{array}{l} p \text{ is a polynomial on a single variable } x \text{ with integer} \\ \text{coefficients and with an integral root} \end{array} \right\}$$

Lemma

Both D and D_1 are Turing-recognizable.

Proof.

On input $p(x)$

evaluate p with x set successively to the values $0, 1, -1, 2, -2, 3, -3, \ldots$. If at any point the polynomial evaluates to 0, then accept.

A simple variant

Lemma

Let

$$p(x) = c_1 x^n + c_2 x^{n-1} + \ldots + c_n x + c_{n+1}$$

with $c_1 \neq 0$ *and* $p(x_0) = 0$. *Define*

$$c_{max} = \max\{|c_i|\}_{i \in [n+1]}$$

Then

$$|x_0| < \frac{c_{max} \cdot (n+1)}{|c_1|}$$

A simple variant

Lemma

Let

$$p(x) = c_1 x^n + c_2 x^{n-1} + \ldots + c_n x + c_{n+1}$$

with $c_1 \neq 0$ *and* $p(x_0) = 0$. *Define*

$$c_{max} = \max\{|c_i|\}_{i \in [n+1]}$$

Then

$$|x_0| < \frac{c_{max} \cdot (n+1)}{|c_1|}$$

Corollary

D_1 *is decidable.*

Relationship among classes of languages