Automata theory and its applications

Lecture 2: Chomsky hierarchy-Overview and Turing machine

Zhilin Wu

State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences

October 10, 2012

Outline

1. Chomsky hierarchy: An Overview

2. Turing machine
   - Definition
   - Equivalence with Type-0 grammar
   - Halting problem: Undecidability

Formal grammar

Definition

- A formal grammar $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$,
  - $\mathcal{N}$: nonterminals,
  - Σ: terminals,
  - $\mathcal{P}$: production rules $\alpha \to \beta$, where $\alpha \in (\mathcal{N} \cup \Sigma)^+, \beta \in (\mathcal{N} \cup \Sigma)^*$,
  - $S \in \mathcal{N}$: start symbol.
- Derivation relation: If $\alpha \to \beta$, then $w_1 \alpha w_2 \models w_1 \beta w_2$ for any $w_1, w_2 \in (\mathcal{N} \cup \Sigma)^*$
- The language generated by G (denoted by $L(G)$): $\{w \in \Sigma^* \mid S \models^* w\}$.

Grammars

- Type-0 (Phrase-structure): $\alpha \to \beta$ (no restrictions),
- Type-1(Context-sensitive): $\alpha A \beta \to \alpha \gamma \beta$ such that $\gamma \neq \varepsilon$,
- Type-2 (Context-free): $A \to \gamma$,
- Type-3 (Right linear): $A \to a$ and $A \to aB$.

Chomsky hierarchy

Grammar	Languages	Automata
Type-0	Recursively enumerable	Turing machine
Type-1	Context-sensitive	Linear-bounded nondet. Turing machine
Type-2	Context-free	nondet. pushdown automaton
Type-3	Regular	Finite state automaton

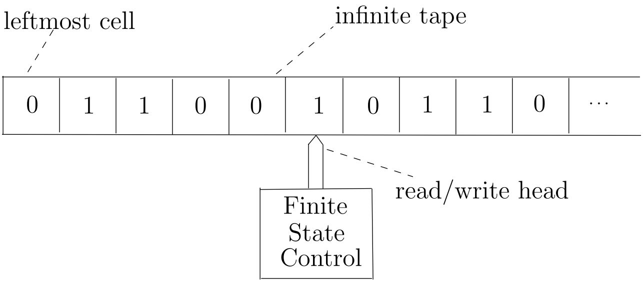
Strictness of the inclusion

- Context-sensitive $\subset$ Recursive $\subset$ Recursively enumerable,
- Context-sensitive and non-context-free: $\{a^n b^n c^n \mid n \in \mathbb{N}\}$,
- Context-free and non-regular: $\{a^n b^n \mid n \in \mathbb{N}\}$.

Outline

Outline

Informal definition



Informally, a Turing machine consists of
an infinite tape
with a read/write head
controlled by
a finite state device.

Formal definition

A Turing machine M is a tuple $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$, where

- Q: Finite set of *states*,
- Γ: Finite set of *tape alphabet*,
- B: a symbol in Γ, called *blank*,
- Σ: A subset of Γ, not including B, called the *input alphabet*,
- δ: *Next-move function*, a partial mapping from $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$,
- $q_0 \in Q$: *Initial state*,
- $F \subseteq Q$: Set of *final states*.

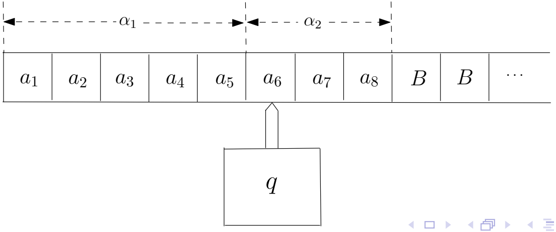
Formal definition (continued)

Instantaneous configuration

An instantaneous configuration of $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ is a sequence $\alpha_1 q \alpha_2$, where

- $q \in Q$: The current state,
- $\alpha_1 \in \Gamma^*$: The sequence of symbols from the leftmost cell to the head, with itself excluded,
- $\alpha_2 \in \Gamma^*$: The sequence of symbols from the head to the rightmost non-blank symbol, or ε if the head is scanning a blank.

Initial configuration: $q_0 w$ (w is the input).



Formal definition (continued)

A move $\vdash_M$

if $\delta(q, X_i) = (p, Y, L)$, then

$$X_1 \ldots X_{i-1}\ q\ X_i X_{i+1} \ldots X_n \vdash_M X_1 \ldots X_{i-2}\ p\ X_{i-1} Y X_i \ldots X_n.$$

Alternatively, if $\delta(q, X_i) = (p, Y, R)$, then

$$X_1 \ldots X_{i-1}\ q\ X_i X_{i+1} \ldots X_n \vdash_M X_1 \ldots X_{i-1} Y\ p\ X_{i+1} \ldots X_n,$$

in particular, in the case $i - 1 = n$, the string $X_i \ldots X_n$ is empty, then the righthand side is longer than the lefthand side.

Languages accepted by M (denoted by $L(M)$)

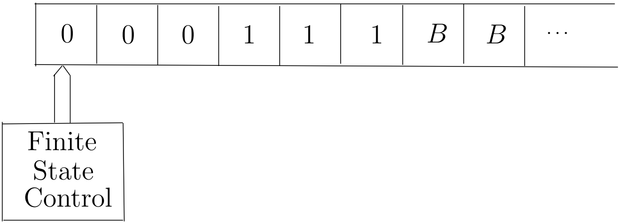
$$L(M) = \{w \mid w \in \Sigma^*, \exists p \in F, \alpha_1 \in \Gamma^*, \alpha_2 \in \Gamma^* \text{ s.t. } q_0 w \vdash_M^* \alpha_1 p \alpha_2\},$$

where $\vdash_M^*$: The reflexive and transitive closure of $\vdash_M$.

Example

A TM M accepting the language $\{0^n1^n \mid n \geqslant 1\}$:

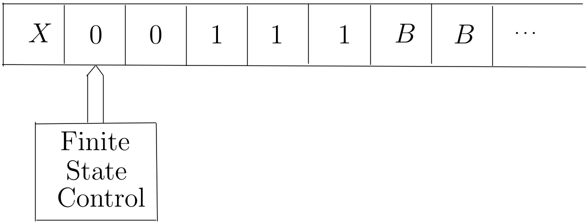
The intuition:



Example

A TM M accepting the language $\{0^n1^n \mid n \geqslant 1\}$:

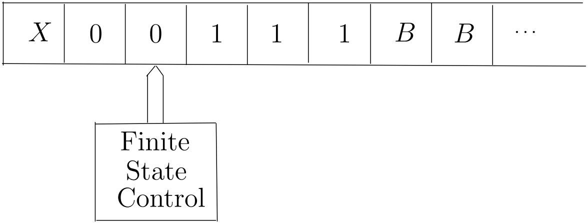
The intuition:



Example

A TM M accepting the language $\{0^n 1^n \mid n \geqslant 1\}$:

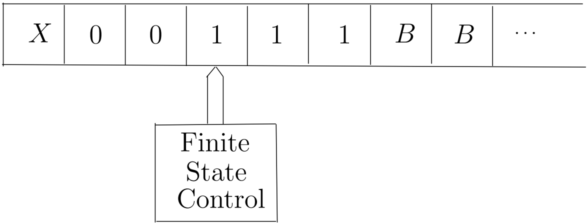
The intuition:



Example

A TM M accepting the language $\{0^n 1^n \mid n \geqslant 1\}$:

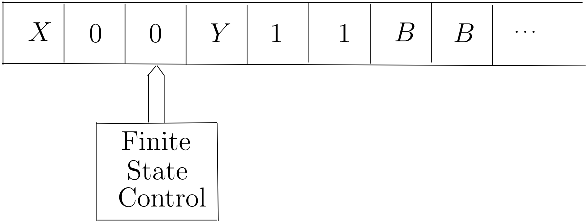
The intuition:



Example

A TM M accepting the language $\{0^n1^n \mid n \geqslant 1\}$:

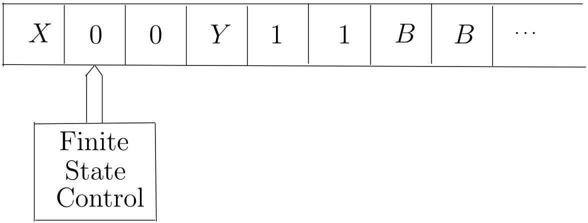
The intuition:



Example

A TM M accepting the language $\{0^n1^n \mid n \geqslant 1\}$:

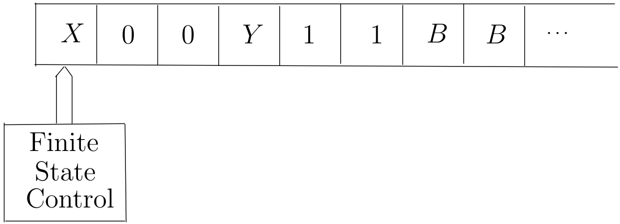
The intuition:



Example

A TM M accepting the language $\{0^n1^n \mid n \geqslant 1\}$:

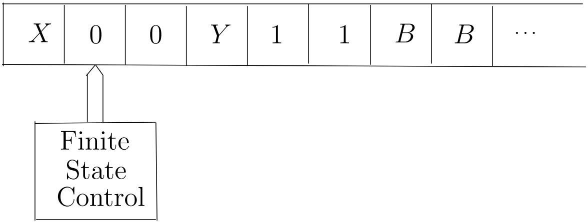
The intuition:



Example

A TM M accepting the language $\{0^n 1^n \mid n \geqslant 1\}$:

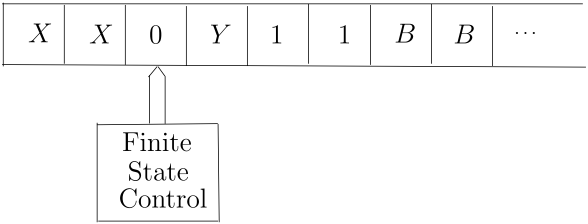
The intuition:



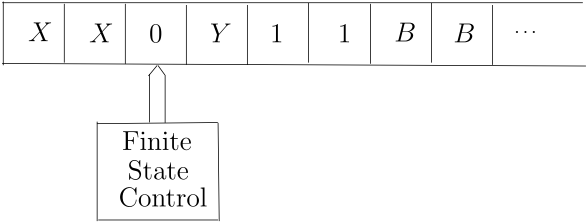
Example

A TM M accepting the language $\{0^n 1^n \mid n \geqslant 1\}$:

The intuition:



A TM M accepting the language $\{0^n1^n \mid n \geqslant 1\}$:

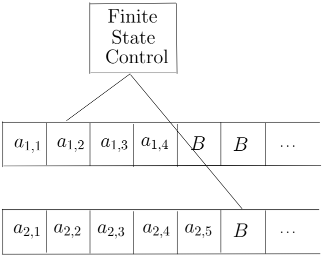


$Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, X, Y, B\}$, $F = \{q_4\}$,

δ	0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_4, B, L)

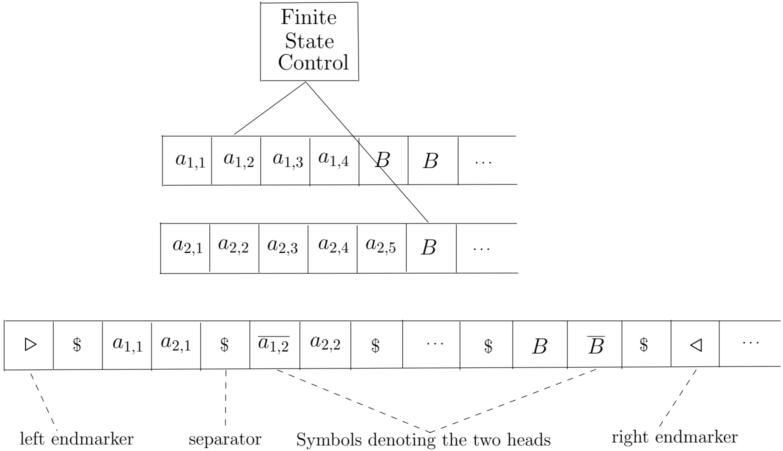
Mutitape Turing machine

- A finite state control with k tapes and k tape heads, one for each tape.
- In one move,
  depending on the state and the symbols scanned by the tape heads, the machine
  - changes the state,
  - changes the symbols scanned by the tape heads,
  - moves the tape heads left or right, independently for each tape.
- Initially, the input is in the first tape and the other tapes are blank.



Multitape TM ≡ TM

Theorem. Multitape TMs can be simulated by TMs.



Nondeterministic TM

The only difference from TM: δ is not a function anymore,

$$\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}.$$

In other words, $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$.

In each move, there are a finite number of (nondet.) choices.

Theorem. Nondet. TM $\equiv$ TM.

A nondet. TM M_1 can be simulated by a three-tape TM M_2 as follows. Suppose r is the maximum number of nondet. choices in each move of M_1.

- The input of M_1 is put on tape 1 of M_2.
- M_2 generates the sequences in $\{1, \ldots, r\}^+$ on tape 2 in the canonical order.
  - Shorter sequences are generated earlier;
  - the sequences of the same length are generated according to the numerical order.

Nondeterministic TM

The only difference from TM: δ is not a function anymore,

$$\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}.$$

In other words, $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$.

In each move, there are a finite number of (nondet.) choices.

Theorem. Nondet. TM $\equiv$ TM.

A nondet. TM M_1 can be simulated by a three-tape TM M_2 as follows. Suppose r is the maximum number of nondet. choices in each move of M_1.

- The input of M_1 is put on tape 1 of M_2.
- M_2 generates the sequences in $\{1, \ldots, r\}^+$ on tape 2 in the canonical order.
- For each sequence $k_1 \ldots k_n$ generated on tape 2, M_2 copies the input to tape 3, simulates the n moves of M_1 on the input, with the sequence generated on tape 2 as the nondet. choices of M_1.
- If for some sequence $k_1 \ldots k_n$, the simulation of M_1 on tape 3 accepts, then M_2 accepts.

Outline

From TM to Type-0 grammar

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a TM. The Type-0 grammar G

- first nondet. generates a finite word

$$\begin{pmatrix} a_1 \\ a_1 \end{pmatrix} \cdots \begin{pmatrix} a_n \\ a_n \end{pmatrix} \begin{pmatrix} \varepsilon \\ B \end{pmatrix} \cdots \begin{pmatrix} \varepsilon \\ B \end{pmatrix},$$

  with the intention that
  - $a_1 \ldots a_n$ is the input of M,
  - the blanks in the second component denote the space used by M.
- then G simulates the computation of M over $a_1 \ldots a_n$, by rewriting the second components of the generated word.

From TM to Type-0 grammar

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a TM. The Type-0 grammar G

- first nondet. generates a finite word

$$\begin{pmatrix} a_1 \\ a_1 \end{pmatrix} \cdots \begin{pmatrix} a_n \\ a_n \end{pmatrix} \begin{pmatrix} \varepsilon \\ B \end{pmatrix} \cdots \begin{pmatrix} \varepsilon \\ B \end{pmatrix},$$

- then G simulates the computation of M over $a_1 \ldots a_n$, by rewriting the second components of the generated word.

Formally, $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ is defined as follows.

- $\mathcal{N} = Q \bigcup (\Sigma \cup \{\varepsilon\}) \times \Gamma \bigcup \{A_1, A_2, A_3\}$, $\mathcal{T} = \Sigma$, $S = A_1$,
- $\mathcal{P}$ includes the following rules,
    - $A_1 \to q_0 A_2$, $A_2 \to [a, a]A_2$, $A_2 \to A_3$, $A_3 \to [\varepsilon, B]A_3$, $A_3 \to \varepsilon$,
    - $q[a, X] \to [a, Y]p$
      for each $a \in \Sigma \cup \{\varepsilon\}$ and $q, p, X, Y : \delta(q, X) = (p, Y, R)$,
    - $[a_1, Z]q[a_2, X] \to p[a_1, Z][a_2, Y]$
      for each $a_1, a_2 \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$, and $q, p, X, Y : \delta(q, X) = (p, Y, L)$,
    - $[a, X]q \to qaq$, $q[a, X] \to qaq$, $q \to \varepsilon$
      for each $a \in \Sigma$, $X \in \Gamma$, and $q \in F$.

From TM to Type-0 grammar (continued)

Example.
TM for $\{0^n 1^n \mid n \in \mathbb{N}\}$.
The grammar:

$$
\begin{aligned}
& A_1 \models q_0 A_2 \models q_0 [0,0] A_2 \models \dots \\
& \models q_0 [0,0][0,0][0,0][1,1][1,1][1,1] A_2 \\
& \models q_0 [0,0][0,0][0,0][1,1][1,1][1,1] A_3 \\
& \models q_0 [0,0][0,0][0,0][1,1][1,1][1,1][\varepsilon, B] A_3 \\
& \models q_0 [0,0][0,0][0,0][1,1][1,1][1,1][\varepsilon, B] \\
& \models [0,X] q_1 [0,0][0,0][1,1][1,1][1,1][\varepsilon, B] \\
& \models \dots \\
& \models [0,X][0,0][0,0] q_1 [1,1][1,1][1,1][\varepsilon, B] \\
& \models [0,X][0,0] q_2 [0,0][1,Y][1,1][1,1][\varepsilon, B] \\
& \models \dots \\
& \models [0,X][0,X][0,X] q_0 [1,Y][1,Y][1,Y][\varepsilon, B] \\
& \models [0,X][0,X][0,X][1,Y] q_3 [1,Y][1,Y][\varepsilon, B] \\
& \models [0,X][0,X][0,X][1,Y][1,Y] q_3 [1,Y][\varepsilon, B] \\
& \models [0,X][0,X][0,X][1,Y][1,Y][1,Y] q_3 [\varepsilon, B] \\
& \models [0,X][0,X][0,X][1,Y][1,Y][1,Y][\varepsilon, B] q_4 \\
& \models [0,X][0,X][0,X][1,Y][1,Y][1,Y] q_4 q_4 \\
& \models [0,X][0,X][0,X][1,Y][1,Y] q_4 1 q_4 q_4 \\
& \models \dots \\
& \models q_4 0 q_4 0 q_4 0 q_4 1 q_4 1 q_4 1 q_4 q_4 \\
& \models \dots \\
& \models 000111
\end{aligned}
$$

From Type-0 grammar to TM

Let $G = (\mathcal{N}, \Sigma, \mathcal{P}, S)$ be a Type-0 grammar.
Construct a nondet. TM M to recognize the language $L(G)$.

- M has two tapes.
- The input of M (say w) is in tape 1.
- M simulates the derivation relation of G in tape 2 by repeating the following procedure.
  1. It nondet. chooses a position i in tape 2 and a production rule $\alpha \to \beta$.

     If α appears from position i in tape 2, then α is replaced by β in tape 2. Some shifting over of the symbols on tape 2 should be done if $|\alpha| \neq |\beta|$.
  2. M compares the sequence of symbols in tape 2 with the sequence in tape 1, to see whether w has been generated by G. If so, then M accepts.

Outline

1. Chomsky hierarchy: An Overview

2. Turing machine
   - Definition
   - Equivalence with Type-0 grammar
   - Halting problem: Undecidability

Decision problems and problem instances

Language-theoretical viewpoint:

- Decision problems: A language over a finite alphabet Σ.
- Problem instances: A finite word over the alphabet Σ.

SAT: An example

SAT: Decide whether a given Boolean formula is satisfiable or not ?

- Instances of SAT problem: The Boolean formulas, e.g. $(x1 \vee x2 \vee x3) \wedge (\neg x1 \vee x2 \vee \neg x3) \wedge (x1 \vee \neg x2 \vee x3)$.
- SAT problem: The set of satisfiable Boolean formulas.

More formally,

- An instance of SAT problem: A finite word over the alphabet $\Sigma_{SAT} := \{\vee, \wedge, \neg, \ (\ ,\ )\} \cup \{x, 0, 1, \dots, 9\}$.
- SAT problem: A language over the alphabet Σ_{SAT}.

Recursively enumerable and recursive languages

Recursively enumerable (r.e.) languages

A language $L \subseteq \Sigma^*$ is recursively enumerable if $L = L(M)$ for some TM M.

- If $w \in L(M)$, then the computation of M over w halts with a final state.
- If $w \notin L(M)$, then the computation of M over w either halts with a non-final state or never halts.

Recursively enumerable and recursive languages

Recursively enumerable (r.e.) languages

A language $L \subseteq \Sigma^*$ is recursively enumerable if $L = L(M)$ for some TM M.

- If $w \in L(M)$, then the computation of M over w halts with a final state.
- If $w \notin L(M)$, then the computation of M over w either halts with a non-final state or never halts.

Recursive languages

A language $L \subseteq \Sigma^*$ is recursive if $L = L(M)$ for some TM M such that M halts over all inputs.

- If $w \in L(M)$, then the computation of M over w halts with a final state.
- If $w \notin L(M)$, then the computation of M over w halts with a non-final state.

Intuitively, for a recursive language L, $\exists$ an algorithm to tell for every input w, if $w \in L$, then the algorithm answers "yes", otherwise, it answers "no".

Basic properties of r.e. and recursive languages

Theorem. The following closure properties hold.

- R.e. languages are closed under union and intersection.
- Recursive languages are closed under all Boolean operations.

Proof sketch.

- Union and intersection:
  Simulate simultaneously the two TMs by a two-tape TM.
- Complementation for recursive languages: Replace F by $Q \backslash F$.

Basic properties of r.e. and recursive languages

Theorem. The following closure properties hold.

- R.e. languages are closed under union and intersection.
- Recursive languages are closed under all Boolean operations.

Theorem. Let $L \subseteq \Sigma^*$. If L and $\Sigma^* \backslash L$ are both r.e., then L is recursive.

Proof sketch.

Let $M_1, M_2 : L = L(M_1)$ and $\Sigma^* \backslash L = L(M_2)$.

Then a two-tape TM M is constructed to simulate simultaneously M_1 and M_2.

If M_1 accepts, then M accepts.
If M_2 accepts, then M rejects.

The termination of M over all inputs is guaranteed by the following fact.

For any $w \in \Sigma^*$,
either the computation of M_1 accepts,
or the computation of M_2 accepts,
but not both.

Decidable and undecidable problems

A problem is decidable if

the language corresponding to the problem is recursive.

Otherwise, the problem is undecidable.

Example

- Decidable problems: SAT problem, Primality problem, etc.
- Undecidable problems: Halting problem of TMs (defined later).

Encoding of TMs

W.l.o.g, we restrict our attention to TMs
with the input alphabet $\Sigma = \{0, 1\}$ and the tape alphabet $\Gamma = \{0, 1, B\}$.
Let $M = (Q, \Sigma, \Gamma, \delta, q_1, B, F)$ such that $Q = \{q_1, \ldots, q_n\}$.
Let X_1, X_2, X_3 denote $0, 1, B$ and D_1, D_2 denote L, R.

Binary encoding of M (denoted by $\langle M \rangle$)

- each transition $\delta(q_i, X_j) = (q_k, X_l, D_m)$ is encoded by the binary word

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m. \tag{1}$$

- A binary encoding of M is a word of the form

$$111 \text{ code}_1 \ 11 \text{ code}_2 \ 11 \ \ldots \ 11 \text{ code}_r \ 111,$$

where each code_i is a word of the form (1) and
each transition of M is encoded by one of the code_i's.

The notation $\langle M, w \rangle$ (where M is a TM and $w \in \{0, 1\}^*$):
The concatenation of $\langle M \rangle$ and w.

Encoding of TMs: Example

TM for $L = 10^*$

Let $M = (\{q_1, q_2, q_3\}, \{0,1\}, \{0,1,B\}, \delta, q_1, \{q_3\})$, where

- $\delta(q_1, 1) = (q_2, 1, R)$,
- $\delta(q_2, 0) = (q_2, 0, R)$,
- $\delta(q_2, B) = (q_3, B, L)$.

The encoding

Encoding of δ transitions,

- $\delta(q_1, 1) = (q_2, 1, R)$: 0100100100100,
- $\delta(q_2, 0) = (q_2, 0, R)$: 001010010100,
- $\delta(q_2, B) = (q_3, B, L)$: 0010001000100010.

Then $\langle M, 1000 \rangle$ is the following word,

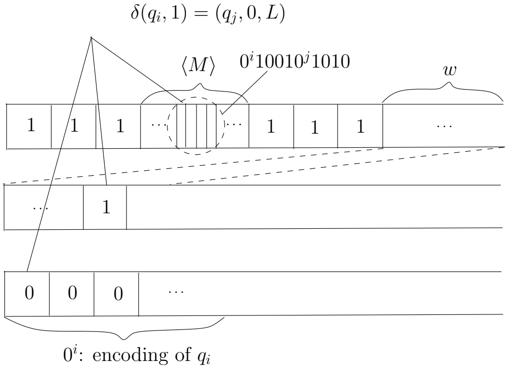
111010010010010011001010010100110010001000100010111000.

Halting problem.

Given an input $\langle M, w\rangle$, decide whether the computation of M on w halts.

$L_h = \{\langle M, w\rangle \mid \text{The comutation of } M \text{ over } w \text{ halts}\}$.

Proposition. The language L_h is recursively enumerable.



Halting problem of TMs (continued)

Theorem. The language L_h is not recursive.

Two notations

- M_i: The TM M whose binary encoding $\langle M \rangle$ is the binary encoding of the integer i.
- w_j: The j-th word in the list of the words in $\{0, 1\}^*$ according to the canonical order.

Halting problem of TMs (continued)

Proof sketch (Diagonalization argument).
To the contrary, suppose L_h is recursive.
Then $\exists$ a TM M_h deciding L_h.
Define M_d as follows:

> *Over an input $\langle M \rangle$,*
> *if M_h accepts $\langle M, \langle M \rangle \rangle$,*
> *then loop forever,*
> *otherwise accepts.*

Derivation of contradiction.

- If M_d halts over $\langle M_d \rangle$,
  then M_h does not accept $\langle M_d, \langle M_d \rangle \rangle$,
  so M_d does not halt over $\langle M_d \rangle$.
- If M_d does not halt over $\langle M_d \rangle$,
  then $\langle M_d, \langle M_d \rangle \rangle \notin L_h$,
  so M_h rejects $\langle M_d, \langle M_d \rangle \rangle$,
  therefore, over $\langle M_d \rangle$, M_d accepts and halts.

$w_j \longrightarrow$

$M_i \downarrow$

	1	2	3	4	5	6	$\cdots$
1	0	1	0	1	1	0	$\cdots$
2	1	1	0	0	1	1	$\cdots$
3	0	1	1	0	1	0	$\cdots$
4	1	0	1	0	1	1	$\cdots$
5	1	0	0	1	1	0	$\cdots$
6	1	0	1	0	1	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$