Introduction to Tree Language Theory

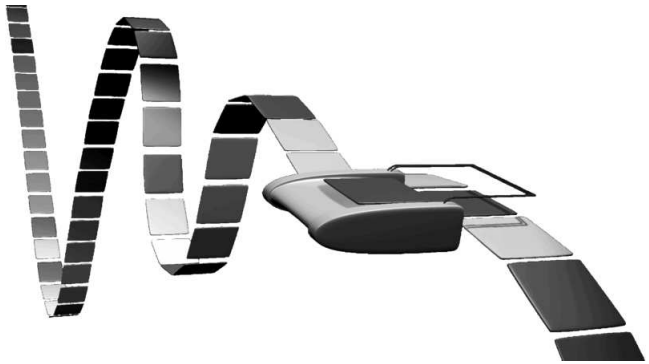
Hitoshi Ohsaki

 National Institute of
Advanced Industrial Science and Technology (AIST)

seminar talk (5/10)

2009

V. Turing Machine



Turing Machine (TM)

(Deterministic) Turing machine : $(\Gamma, \Sigma, Q, \mathsf{q}_0, Q_{\mathsf{fin}}, \Delta)$

- Γ : alphabet containing special symbol $\sharp$ (called *blank*)
- Σ : input tape symbols such that $\Sigma \subseteq \Gamma - \{\sharp\}$
- Q : finite set of state symbols
- q_0 : start symbol such that $\mathsf{q}_0 \in Q$
- Q_{fin} : final states such that $Q_{\mathsf{fin}} \subseteq Q$
- Δ : transition function, which is a partial function such that

  $$\langle p, a\rangle \mapsto \langle q, b, x\rangle \quad (p, q \in Q,\ a, b \in \Gamma,\ x \in \{\mathsf{L}, \mathsf{R}\})$$

  where, e.g. $\langle p, a\rangle \mapsto \langle q, b, \mathsf{L}\rangle$ means "a state p is changed to q after reading a character a and rewriting it to b, and then the tape-head moves on the tape to the left." If $\langle q, b, \mathsf{R}\rangle$ on the right-hand side in the above assignment, the tape-head moves to the right.

Turing Machine (cont'd)

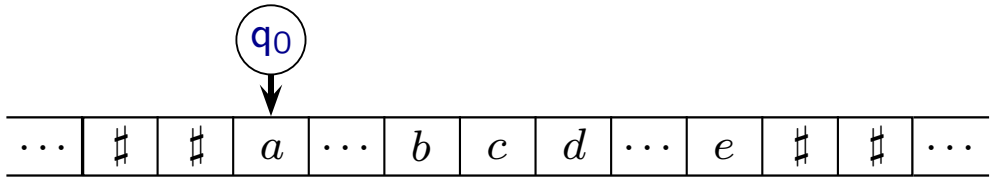
Every TM $(\Gamma, \Sigma, Q, \mathsf{q}_0, Q_{\mathsf{fin}}, \Delta)$ is equipped with a single input tape which is divided into infinitely many *cells* such that a character in Γ can be stored in each cell.

TM has a single *tape-head* which reads a character in the cell, rewrites the character to some character, indicates the current state from Q, and move the head on the tape to the left or right, according to Δ.

Initially, the input tape contains finite sequence of characters from Σ in the cells, and the remaining cells contains blanks ($\sharp$).

Besides, the tape-head indicates start symbol q_0 and is located at the leftmost cell which contains a character. See below for example.

– initial configuration :



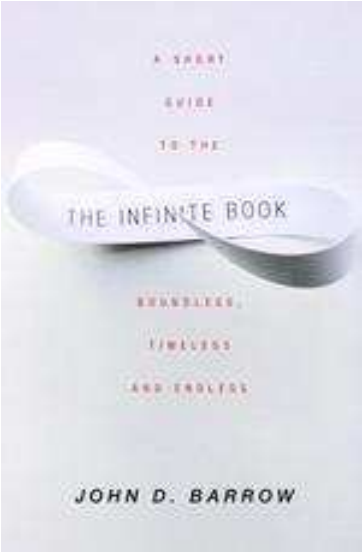
Infinity

Countable infinity ($\aleph_0$) : $\exists$ one-to-one correspondence with the list of natural numbers 1, 2, 3, ...

Georg Cantor (1845–1918)

... you turn up at the check-in counter of the Hotel Infinity only to find that the infinite number of rooms (numbered 1, 2, 3, 4, ... and so on, forever) are all occupied. The receptionist is perplexed — the Hotel is full — but the manager is unperturbed. No problem, he says: move the guest in room 1 to room 2, the guest in room 2 to room 3, and so on, forever. This leaves room 1 vacant for you to take and everyone still has a room!

("Welcome to the Hotel Infinity", chap. 3, by J.D. Barrow)



The Infinite Book
© Vintage books

Note

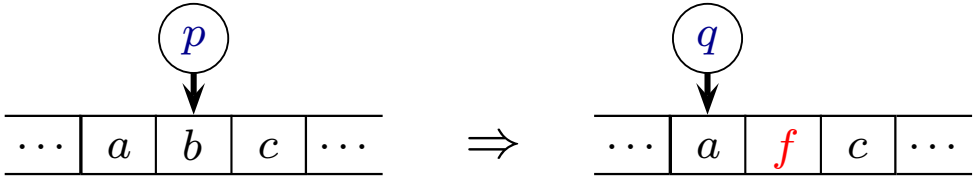
Infinite hierarchy of infinity : $\aleph_0 \subsetneq \aleph_1 \subsetneq \aleph_2 \subsetneq \aleph_3 \subsetneq \ldots$

Accepted input tape

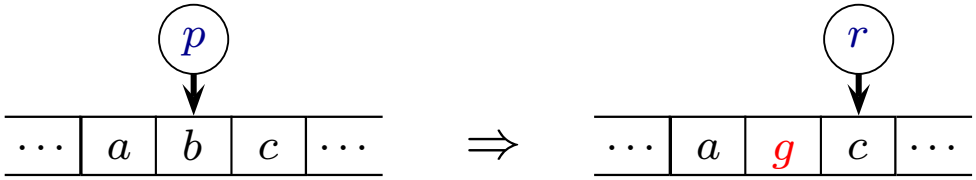
TM operates move by move, according to the transition function Δ.

The machine either operates forever, or it halts when the tape-head with state p reads character a but there is no transition that matches $\langle p, a \rangle$.

– If $\langle p, b \rangle \mapsto \langle q, f, \mathsf{L} \rangle$ in Δ :



– If $\langle p, b \rangle \mapsto \langle r, g, \mathsf{R} \rangle$ in Δ :



For input-tape with a finite sequence w of characters, if TM $\mathcal{M}$ halts with final state p in Q_{fin}, we say w is accepted by $\mathcal{M}$. The set of words accepted by $\mathcal{M}$ is denoted by $\mathcal{L}(\mathcal{M})$.

A language L is called recursively enumerable if there exists TM $\mathcal{M}$ that halts on each word in L, and it never halts on any word in $\Sigma^* - L$.

(In this definition, the set Q_{fin} of final states does not play an essential role to define this class of languages)

Multi-tape TM (MTM)

Multi-tape TM is TM with n tapes ($n \geqslant 1$) and n tape-heads (one head for each tape) but one state indicator.

Initially, the first tape contains the input (that is input symbols in the finite sequence of cells and blanks in the remaining cells), and the remaining tapes contain blanks.

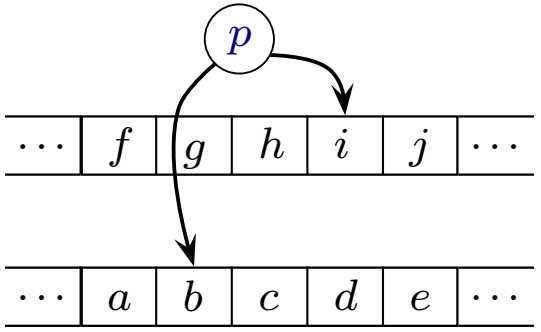
And tape-head on the first tape is located at the leftmost of the input, and the other heads can be placed at arbitrary cells.

Transition function Δ is the mapping :

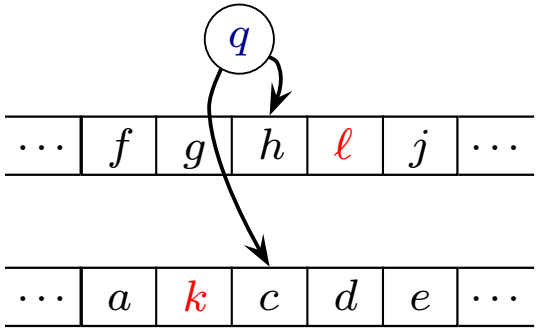
$$\langle p, a_1, \ldots, a_n \rangle \mapsto \langle q, b_1, \ldots, b_n, X_1, \ldots, X_n \rangle$$

so that one transition admits simultaneous move of the multiple tape-heads.

2-tape TM



$\langle p, b, i \rangle \mapsto \langle q, k, \ell, \mathsf{R}, \mathsf{L} \rangle$



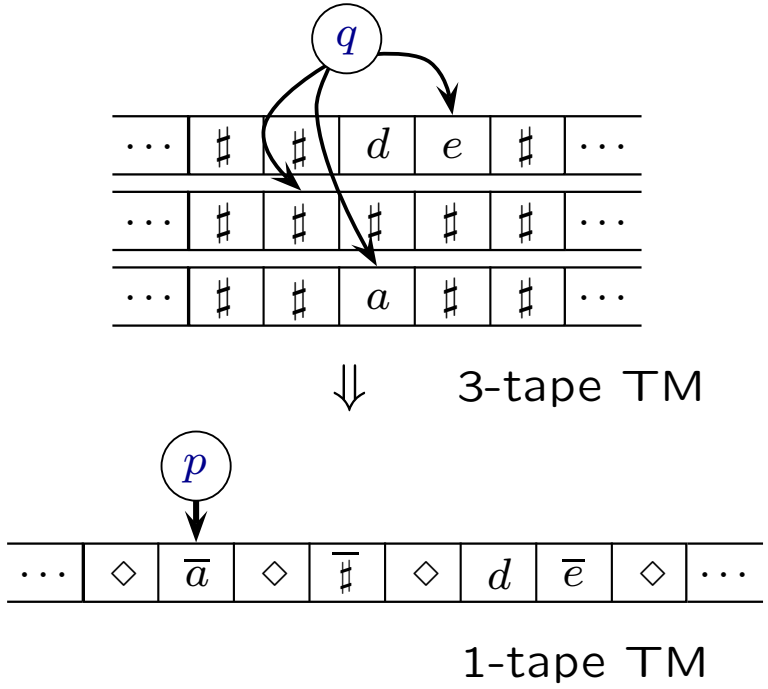
TM = MTM

Every language accepted by MTM is recursively enumerable

Proof

Let $\mathcal{M}$ be MTM with n-tapes whose set of tape symbols is Γ. Suppose $\diamond$ is a fresh symbol. Then, define a single tape TM $\mathcal{M}_{\mathsf{S}}$ whose tape symbols are $\Gamma \cup \{\diamond\} \cup \{\,\overline{a} \mid a \in \Gamma\,\}$. Consider $n = 3$ below, but the proof can be generalized for every $n \geqslant 2$:

1. For the input $a_1\, a_2 \cdots a_k$, $\mathcal{M}_{\mathsf{S}}$ creates the word $\diamond\, \overline{a}_1\, a_2 \cdots a_k \diamond \overline{\sharp} \diamond \overline{\sharp} \diamond$, meaning that 1st head is placed on the leftmost character a_1, and the other heads are placed on the blank and cells on those tapes do not contain any character.
2. Simultaneous move on 3 tapes is simulated by the single tape that contains three finite segments (separated by $\diamond$). When one of heads rewrites a to b and moves to the right cell with c, the corresponding word $\overline{a}\, c$ on the single tape is rewritten to $b\, \overline{c}$. If $c = \diamond$, shift c and all symbols after c one cell to the right, and then rewrite $\overline{a}\, \sharp \diamond$ to $b\, \overline{\sharp} \diamond$. Similar for the left. $\square$



Non-deterministic TM (NTM)

Non-deterministic TM $\mathcal{M} = (\Gamma, \Sigma, Q, \mathsf{q}_0, Q_{\mathsf{fin}}, \Delta)$ where

Δ : finite set of transition rules in the form of

$$\langle p, a \rangle \rightarrow \langle q, b, x \rangle$$

$$p, q \in Q,\ a, b \in \Gamma,\ x \in \{\mathsf{L}, \mathsf{R}\}$$

$\mathcal{M}$ accepts word w if and only if there exists a sequence of moves of the tape head that leads from the initial configuration to the situation where the head halts with a final state (a final configuration).

Proposition

TM $\subseteq$ NTM

Proof

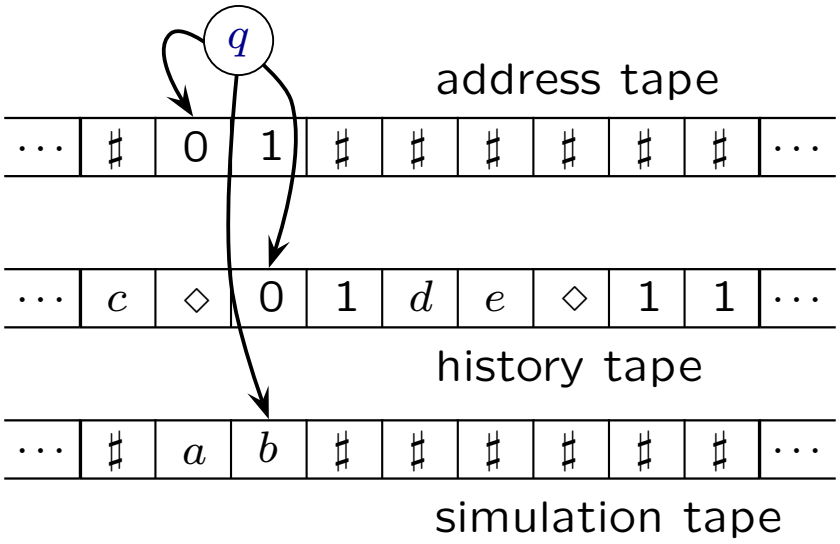
Because every TM is NTM, languages accepted by TM are accepted by NTM. □

TM = NTM

Every language accepted by NTM is recursive enumerable

Proof

For NTM $\mathcal{M}$ with Γ, define 3-tape TM $\mathcal{M}_3$ with $\Gamma \cup \{\diamond, 0, 1\}$: Tape 1 is a simulation tape, which means this tape contains a configuration (a copy of tape) on $\mathcal{M}$'s computation. Tape 2 contains nodes of $\mathcal{M}$'s computation tree which are already or will be visited by simulation. Nodes on tape 2 are separated by $\diamond$. Tape 3 contains the next address or the empty word (which means no address left to visit for next). The simulation proceeds in breadth-first manner. If $\mathcal{M}$ accepts the input on tape 1, $\mathcal{M}_3$ enter a final state. If $\mathcal{M}$ halts on a non-final state, $\mathcal{M}_3$ traverses $\mathcal{M}$'s computation tree. If $\mathcal{M}$ meets a non-deterministic choice in the current computation on tape 1, keep in tape 2 this address (denoted by a word in $\{0,1\}^*$) and the current configuration, and then follow the way of backtracking computation. If $\mathcal{M}$ halts on a non-final state and tape 3 is the empty word, $\mathcal{M}_3$ halts on a non-final state. $\square$



Recursive languages

Given an alphabet Σ

A language L over Σ is recursive if there exists a TM $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = L$ and $\mathcal{M}$ halts on any input in Σ^* ($\mathcal{M}$ is called halting TM).

Note 1

Let $\mathcal{M}$ be halting TM $(\Gamma, \Sigma, Q, \mathsf{q}_0, Q_{\mathsf{fin}}, \Delta)$, then $\mathcal{M}_{\mathsf{C}} = (\Gamma, \Sigma, Q, \mathsf{q}_0, Q - Q_{\mathsf{fin}}, \Delta)$ accepts the complement of $\mathcal{L}(\mathcal{M})$, because TM is deterministic. Therefore, recursive languages are closed under complement. Since recursive languages are also closed under union, by de Morgan's law, this class is closed under Boolean operations.

Note 2

One can transform halting TM $\mathcal{M}$ to $\mathcal{M}'$ such that for input w, $\mathcal{M}'$ halts if $\mathcal{M}$ halts with a state in Q_{fin} ; otherwise (if $\mathcal{M}$ halts with a state in $Q - Q_{\mathsf{fin}}$), $\mathcal{M}'$ does not halt. Hence, recursive languages are recursive enumerable.

Question

Are recursive enumerable languages recursive?

Binary encoding

C(TM) : set of TM's for languages over $\{0,1\}$ defined as follows

$\mathcal{M} \in$ C(TM) if $\mathcal{M} = (\{0,1,\sharp\}, \{0,1\}, Q, 10, Q_{\mathsf{fin}}, \Delta)$ where

Q is a finite subset of $\{1 \cdot 0^n \mid n \geqslant 1\}$

Q_{fin} contains 100 only

Ψ : mapping from Δ to $\{0,1\}^*$ such that

$$\Psi(\langle p, a\rangle \mapsto \langle q, b, X\rangle) = p\,\psi(a)\,q\,\psi(b)\,\psi(X)$$

where

$$\psi(x) = \begin{cases} 10 & \text{if } x = 0 \text{ or } x = \mathsf{L} \\ 100 & \text{if } x = 1 \text{ or } x = \mathsf{R} \\ 1000 & \text{if } x = \sharp \end{cases}$$

E.g. $\Psi(\langle 10, \sharp\rangle \mapsto \langle 100, 0, \mathsf{L}\rangle) = \underline{10}\ \underline{1000}\ \underline{100}\ \underline{100}\ \underline{10}$

Binary encoding (cont'd)

Let $\mathcal{M} = (\{0, 1, \sharp\}, \{0, 1\}, Q, 10, Q_{\text{fin}}, \Delta)$ with the mappings in Δ :

$$\langle p_1, a_1 \rangle \mapsto \langle q_1, b_1, X_1 \rangle \ \ldots \ \langle p_n, a_n \rangle \mapsto \langle q_n, b_n, X_n \rangle$$

Define the word for $\mathcal{M}$:

$$\Psi(\langle p_1, a_1 \rangle \mapsto \langle q_1, b_1, X_1 \rangle) 110 \cdots 110 \Psi(\langle p_n, a_n \rangle \mapsto \langle q_n, b_n, X_n \rangle)$$

Note

Let w be the above word,

- w can be defined in $|\Delta|!$ different ways (one-to-many mapping)
- w does not contain any 1^n for $n \geqslant 3$

Moreover,

- TM $\mathcal{M}$ can be retrieved from w

Diagonalization language

Let $\mathcal{M}_\varnothing = (\{0, 1, \sharp\}, \{0, 1\}, \{10\}, 10, \varnothing, \varnothing)$

Define

$$L_{\mathsf{DL}} = \{ w \in \{0, 1\}^* \mid w \text{ is not accepted by } \mathsf{decode}(w) \}$$

where $\mathsf{decode}(w) = \mathcal{M}$ if w represents TM $\mathcal{M}$; otherwise, $\mathcal{M}_\varnothing$

Proposition

No TM accepts L_{DL}

Proof

Suppose, by way of contradiction, that TM $\mathcal{M}$ accepts L_{DL}. One can modify $\mathcal{M}$ to $\mathcal{M}'$ whose initial state is 10 and final state is 100 only. And all state symbols of $\mathcal{M}'$ are renamed with symbols from $\{10^n \mid n \geqslant 1\}$. Then $\mathcal{M}'$ can be translated by Ψ to a word w over $\{0,1\}$. If $w \in L_{\mathsf{DL}}$, w is accepted by $\mathcal{M}'$, but then $w \notin L_{\mathsf{DL}}$. If $w \notin L_{\mathsf{DL}}$, that means w is not accepted by $\mathcal{M}'$, then $w \in L_{\mathsf{DL}}$. We conclude by this contradiction, that $\mathcal{M}'$ does not exist, and hence, $\mathcal{M}$ does not exist either. □

Complement of L_{DL}

Take the complement of L_{DL}, that is,

$$(L_{\mathsf{DL}})^{\mathsf{c}} = \{\, w \in \{\,0,\,1\,\}^* \mid w \text{ is accepted by } \mathsf{decode}(w)\,\}$$

Proposition

There exists TM that accepts $(L_{\mathsf{DL}})^{\mathsf{c}}$

Proof (sketch)

Define TM $\mathcal{M}$ such that (1) for input w, first $\mathcal{M}$ duplicates it as the pair of word w and TM $\mathcal{M}_w\,(= w)$, and (2) verifies whether $\mathcal{M}_w$ is TM defined by binary encoding. (3) If yes, $\mathcal{M}$ starts to simulate the move of $\mathcal{M}_w$. (4) When $\mathcal{M}_w$ enters the final state and halts, $\mathcal{M}$ accepts w ; when $\mathcal{M}_w$ halts with a non-final state, $\mathcal{M}$ also halts without accepting w. □

Corollary

No halting TM accepts $(L_{\mathsf{DL}})^{\mathsf{c}}$

Universal TM (UTM)

There exists TM $\mathcal{M}_{\mathsf{U}}$ that simulates on input $\langle x, y \rangle$, TM $\mathsf{decode}(x)$ on input y such that $\mathcal{L}(\mathcal{M}_{\mathsf{U}}) = \{ \langle x, y \rangle \mid y \in \mathcal{L}(\mathsf{decode}(x)) \}$

Proof

Similar to the previous proof, but to be precise, first define 3-tape TM $\mathcal{M}$. For the input $\langle x, y \rangle$, x is copied to tape 2, and verify whether x is (deterministic, single-tape) TM defined by binary encoding. If yes, $\mathcal{M}$ simulates x's computation on tape 1. The current state of x and other information needed for simulation are stored in tape 3. Because the above MTM $\mathcal{M}$ can be simulated by a single-tape TM, take that TM for $\mathcal{M}_{\mathsf{U}}$. □

The smallest UTM over $\{0, 1\}$ known so far needs 22 state symbols [1]. Other small UTM's, in order of $|Q|$ (the number of state symbols) $\times$ $|\Gamma - \{\sharp\}|$ (the number of tape symbols), known so far are : 2×18, 3×10, 4×6, 5×5, 7×4, 10×3 [2].

---

[1] Y. Rogozhin: *A Universal Turing Machine with 22 States and 2 Symbols*, Romanian Journal of Information Science and Technology 1, pp.259-265, 1998.

[2] Y. Rogozhin: *Small Universal Turing Machines*, TCS 168, pp. 215-240, 1996.

Computability and Undecidability

Where does undecidability come from? :

Proposition

The set of functions from $\{0,1\}^*$ to $\{0,1\}$ is not countable

Proof

Suppose, by way of contradiction, that the above set is represented as $\{f_i \mid i \geqslant 0\}$. Let $a_0, a_1, \ldots, a_i, \ldots$ be a sequence of words over $\{0,1\}$ in the lexicographic order. Define a function f from $\{0,1\}^*$ to $\{0,1\}$ such that for each a_i, $f(a_i) = 1$ if $f_i(a_i) = 0$; $f(a_i) = 0$ if $f_i(a_i) = 1$. Then, f is different from any f_i $(n \geqslant 0)$, leading to the contradiction. □

Definition

A function f from Σ^* to $\{0,1\}$ is computable if there is a halting TM $\mathcal{M}$ such that $w \in \mathcal{L}(\mathcal{M})$ if $f(w) = 1$; $w \notin \mathcal{L}(\mathcal{M})$ if $f(w) = 0$

Only proper subset of functions from $\{0,1\}^*$ to $\{0,1\}$ is computable

Rice's theorem

Every non-trivial property of recursively enumerable languages is undecidable

A property of recursively enumerable languages is a subset of recursively enumerable languages. A property is non-trivial if it is not the empty set and not the set of all recursively enumerable languages. The above theorem states that for every non-trivial property P, the question if, for a given TM $\mathcal{M}$, $\mathcal{L}(\mathcal{M}) \in P$ is undecidable.

Proof

Suppose, by way of contradiction, that P is decidable, i.e. that there is a halting TM $\mathcal{M}_1$ accepting binary codes of TM's that accept languages in P. If $\varnothing \notin P$, let $\mathcal{M}_L$ be TM accepting a non-empty language L in P. There must be such L in P, since P is non-trivial. Next, define TM $\mathcal{M}_2$ such that, depending on a given TM $\mathcal{M}$ and input w, $\mathcal{M}_2$ accepts L or $\varnothing$: If $\mathcal{M}$ does not accept w, $\mathcal{M}_2$ does not accept any input, so $\mathcal{M}_1$ does not accept binary code of $\mathcal{M}_2$. If $\mathcal{M}$ accepts w, $\mathcal{M}_2$ simulates TM $\mathcal{M}_L$, so $\mathcal{M}_1$ accepts binary code of $\mathcal{M}_2$. Then, $\mathcal{M}$ accepts w if and only if $\mathcal{M}_1$ accepts binary code of $\mathcal{M}_2$. If $\varnothing \in P$, take the complement of P, and then apply the above argument. Because P is a recursive language, $(P)^{\mathsf{c}}$ is also decidable. However, our assumption leads to the contradiction, since the halting problem whether $\mathcal{M}$ accepts w is undecidable. Hence, P is not decidable. □

Trakhtenbrot's theorem

The problem deciding if a first-order sentence is finitely satisfiable (i.e. has a finite model) is undecidable.

Corollary 1

The set of finitely valid sentences (i.e. valid sentences, each of which has a finite model) is not recursively enumerable.

Proof of Corollary 1

It is not difficult to see that the set of finitely satisfiable sentences (i.e. satisfiable sentences, each of which has a finite model) is recursively enumerable. Now we suppose, for leading to a contradiction, that the set of finitely valid sentences, denoted S, is also recursively enumerable. Since a sentence ϕ ($\in S$) is finitely valid iff $\neg\phi$ is not finitely satisfiable, S and $(S)^{\mathsf{C}}$ are recursive, because $(S)^{\mathsf{C}}$ is recursively enumerable and S is so by assumption. Note that $(S)^{\mathsf{C}}$ is the set of finitely satisfiable sentences. However, it contradicts to the above Trakhtenbrot's undecidability theorem. □

Corollary 2

An effective axiomatization of first-order logic over finite models is not possible.

Proof of Trakhtenbrot's theorem

We encode a TM $\mathcal{M}$ to a first-order sentence $\Psi_{\mathcal{M}}$ such that: $\mathcal{M}$ halts on the empty input iff $\Psi_{\mathcal{M}}$ is finitely satisfiable. Let $\mathcal{M} = (\Gamma, \Sigma, Q, \mathsf{q}_0, Q_{\mathsf{fin}}, \Delta)$ where $\Gamma = \{\,0, 1, \sharp\,\}$. We take the set σ of function symbols $(0, \mathsf{s}, \mathsf{p})$ and predicate symbols

$$0 \quad \mathsf{p}(\_) \quad \mathsf{s}(\_) \quad \_ \prec \_ \quad \mathsf{T}_0(\_,\_) \quad \mathsf{T}_1(\_,\_) \quad \mathsf{T}_\sharp(\_,\_) \quad \mathsf{H}_q(\_,\_) \text{ for all } q \in Q.$$

Here $\mathsf{T}_c(p,t)$ means a cell at position p at time t contains a character c ($c \in \{0,1,\sharp\}$); $\mathsf{H}_q(p,t)$ means the tape-head is located at position p and in state q at time t. The predicate $\prec$ is a linear order of $\cdots \prec \mathsf{p}(\mathsf{p}(0)) \prec \mathsf{p}(0) \prec 0 \prec \mathsf{s}(0) \prec \mathsf{s}(\mathsf{s}(0)) \prec \cdots$, which is definable by a first-order sentence, denoted $\Psi_{\prec}$. Define eight other first-order sentences as follows :

$$\Psi_{\mathsf{tape}} : \forall p,t\,[\bigvee_{c\in\Gamma}(\,\mathsf{T}_c(p,t) \;\Leftrightarrow\; \bigwedge_{d\in\Gamma-\{c\}}\neg\mathsf{T}_d(p,t)\,)\,]$$

$$\Psi_{\mathsf{head}} : \forall t\,\exists p\,\bigvee_{q\in Q}\mathsf{H}_q(p,t)\;\wedge$$
$$\forall t,p_1,p_2,q_1,q_2\,(\,\mathsf{H}_{q1}(p_1,t)\wedge\mathsf{H}_{q2}(p_2,t)\;\Rightarrow\; p_1=p_2\wedge q_1=q_2\,)$$

$$\Psi_{\mathsf{init}} : \mathsf{H}_{\mathsf{q0}}(0,0)\;\wedge\;\forall p\,\mathsf{T}_\sharp(p,0)$$

$$\Psi_{\mathsf{tran\text{-}r1}} : \bigwedge_{\langle q1,c1\rangle\mapsto\langle q2,c2,\mathsf{R}\rangle}[\,\forall p,t\,\{\,(0\prec p\vee p=0)\wedge\mathsf{H}_{q1}(p,t)\wedge\mathsf{T}_{c1}(p,t)\;\Rightarrow$$
$$\mathsf{H}_{q2}(\mathsf{s}(p),\mathsf{s}(t))\wedge\mathsf{T}_{c2}(p,\mathsf{s}(t))\wedge$$
$$\forall r\,(\,p\neq r\;\Rightarrow\;\bigwedge_{c=0,1,\sharp}\mathsf{T}_c(r,t)\Leftrightarrow\mathsf{T}_c(r,\mathsf{s}(t))\,)\,\}\,]$$

(Proof cont'd)

Proof cont'd

$$\Psi_{\text{tran-r2}} : \bigwedge_{\langle q1,c1\rangle\mapsto\langle q2,c2,\mathsf{R}\rangle} [\,\forall p,t\,\{\,\mathsf{p}(p)\prec 0 \wedge \mathsf{H}_{q1}(\mathsf{p}(p),t)\wedge \mathsf{T}_{c1}(\mathsf{p}(p),t)\;\Rightarrow \mathsf{H}_{q2}(p,\mathsf{s}(t))\wedge \mathsf{T}_{c2}(\mathsf{p}(p),\mathsf{s}(t))\wedge \forall r\,(\,\mathsf{p}(p)\neq r\;\Rightarrow\; \textstyle\bigwedge_{c=0,1,\sharp}\mathsf{T}_c(r,t)\Leftrightarrow \mathsf{T}_c(r,\mathsf{s}(t))\,\}\,]$$

$$\Psi_{\text{tran-l1}} : \bigwedge_{\langle q1,c1\rangle\mapsto\langle q2,c2,\mathsf{L}\rangle} [\,\forall p,t\,\{\,(p\prec 0\vee p=0)\wedge \mathsf{H}_{q1}(p,t)\wedge \mathsf{T}_{c1}(p,t)\;\Rightarrow \mathsf{H}_{q2}(\mathsf{p}(p),\mathsf{s}(t))\wedge \mathsf{T}_{c2}(p,\mathsf{s}(t))\wedge \forall r\,(\,p\neq r\;\Rightarrow\; \textstyle\bigwedge_{c=0,1,\sharp}\mathsf{T}_c(r,t)\Leftrightarrow \mathsf{T}_c(r,\mathsf{s}(t))\,\}\,]$$

$$\Psi_{\text{tran-l2}} : \bigwedge_{\langle q1,c1\rangle\mapsto\langle q2,c2,\mathsf{L}\rangle} [\,\forall p,t\,\{\,0\prec \mathsf{s}(p)\wedge \mathsf{H}_{q1}(\mathsf{s}(p),t)\wedge \mathsf{T}_{c1}(\mathsf{s}(p),t)\;\Rightarrow \mathsf{H}_{q2}(p,\mathsf{s}(t))\wedge \mathsf{T}_{c2}(\mathsf{s}(p),\mathsf{s}(t))\wedge \forall r\,(\,\mathsf{s}(p)\neq r\;\Rightarrow\; \textstyle\bigwedge_{c=0,1,\sharp}\mathsf{T}_c(r,t)\Leftrightarrow \mathsf{T}_c(r,\mathsf{s}(t)))))$$

$$\Psi_{\text{halt}} : \exists p,t\,(\textstyle\bigvee_{q\in Q_{\text{fin}}}\mathsf{H}_q(p,t)\;\vee\;\bigwedge_{\langle q1,c1\rangle\mapsto\langle q2,c2,X\rangle\in\Delta}\neg(\mathsf{H}_{q1}(p,t)\wedge\mathsf{T}_{c1}(p,t))\,\}\,]$$

Let $\Psi_{\mathcal{M}}$ be the conjunction of all the previously defined sentences. By construction, $\Psi_{\mathcal{M}}$ has a finite model $\mathcal{A}$ iff $\mathcal{A}$ represents a computation of $\mathcal{M}$, in which $\mathcal{M}$ halts on the empty input. Since the halting problem of TM even with the empty input is undecidable (Rice's theorem), so is the finite satisfiability. □

Exercise

1. Show that language L is recursively enumerable if and only if there exists TM which accepts L.
2. Show that languages accepted by non-deterministic halting TM are recursive.
3. Show that $\mathcal{L}(\mathcal{M}_{\mathsf{U}})$ is recursively enumerable, but it is not recursive. (Cf. Halting problem)
4. Show that halting TM's are strictly more powerful than LBA.

   E.g., use binary encoding. Suppose $\sharp_{\mathsf{L}}, \sharp_{\mathsf{R}} \in \Sigma$, each of which is left-end and right-end of LBA, respectively. One can construct a halting TM that accepts binary codes of LBA, because a given code of non-deterministic TM is LBA if and only if Δ does not contain a transition rule $\langle p, \sharp_{\mathsf{L}} \rangle \to \langle q, a, \mathsf{L} \rangle$ or $\langle p, \sharp_{\mathsf{R}} \rangle \to \langle q, a, \mathsf{R} \rangle$ for some states $p, q \in Q$ and tape symbol a. Then, define the language $L = \{\, w \in \{0,1\}^* \mid w \text{ is not accepted by decode}(w) \,\}$. Show that L is recursive, and L is not accepted by any LBA.
5. Is the property $P = \{\varnothing\}$ of recursively enumerable languages decidable or undecidable? Explain the reason why.

Appendix : Counter Machines

Counter machine [1 ; also Lambek, Melzak 1961] consists of

- R : finite set of registers $r_0, \ldots, r_n$ $(n \geqslant 0)$, each of which can store any non-negative integer
- Δ : finite set of labeled instructions with the following forms
  - ℓ_i, INC(r_i) : Increment $[r_i]$ by 1 and go to ℓ_{i+1}
  - ℓ_i, DEC(r_i) : Decrement $[r_i]$ by 1 if r_i is not 0, and go to ℓ_{i+1}
  - ℓ_i, JZ(r_i, ℓ_k) : Jump to the instruction labeled ℓ_k if $[r_i]$ equals 0, otherwise go to ℓ_{i+1}

    where $[r_i]$ means the content of register r_i

Initially, non-negative integers as input are stored in registers, and the machine executes first the instruction labelled ℓ_0. The machine halts if the next instruction does not exit, and then the input is accepted.

The class of 2-counter machines is as expressive as TM's.

---

[1] M.L. Minsky: *Recursive Unsolvability of Post's Problem of 'Tag' and Other Topics in the Theory of Turing Machine*, Annual of Mathematics 74, pp.437–455, 1961.

National Institute of Advanced Industrial Science and Technology (AIST) – Senri-site, AIST Kansai.

Office: Shin-Senri Nishi 1–2–14 (MSK bldg. 5th floor), Toyonaka, Osaka 560–0083, Japan

URL: `http://staff.aist.go.jp/hitoshi.ohsaki/`