Foundations of Computability Theory (4)

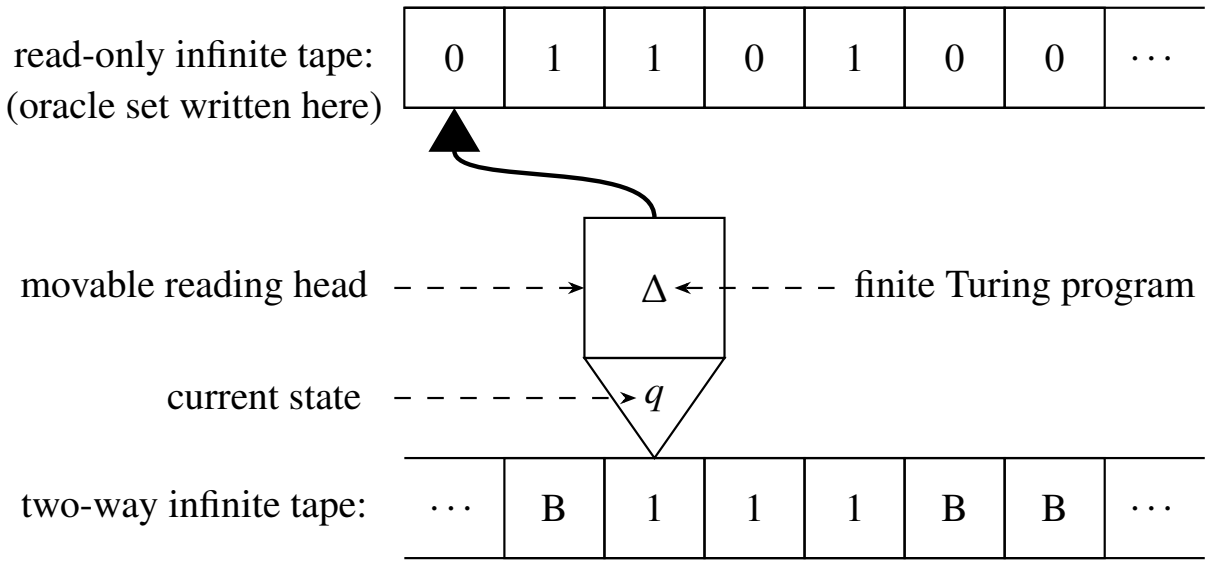
Nan Fang

Institute of Software, Chinese Academy of Sciences

Peking University

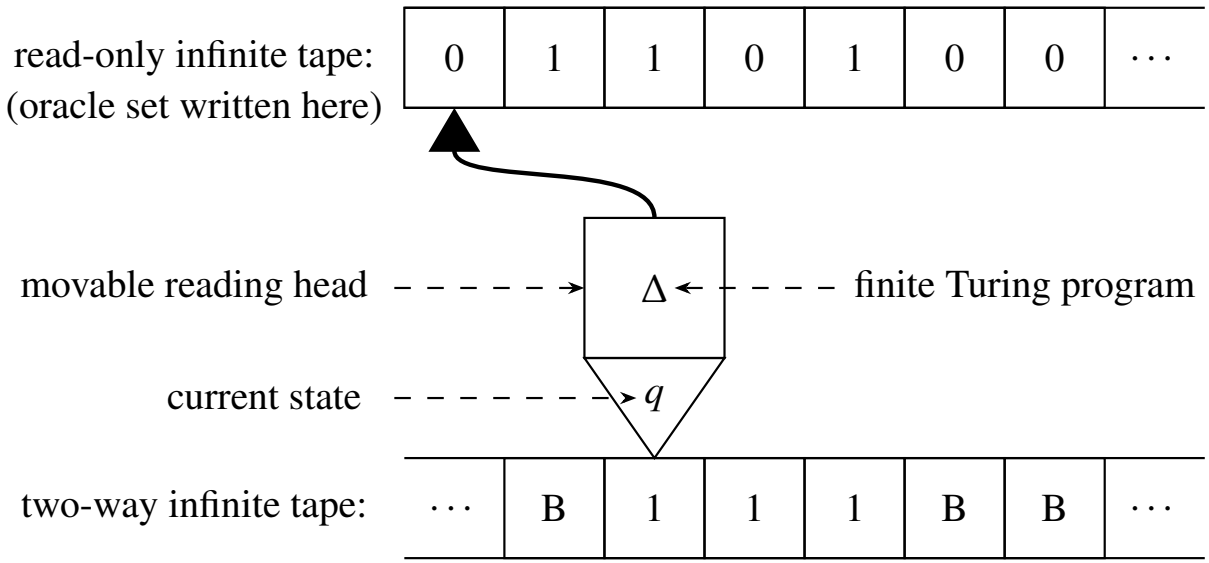
2025 Autumn Semester

oracle Turing machine



- An *oracle Turing machine* (*o*-machine) is a Turing machine with the usual work tape and an extra "read only" tape, called the *oracle tape*, upon which is written the characteristic function of some set A, called the *oracle*, whose symbols $\{0, 1\}$ cannot be printed over. Two reading heads move along these two tapes simultaneously.
- Now each instruction $\delta \in \Delta$ is a 7-tuple of the form $\delta = (q_i, s, t, q_j, s', D, F)$, where $q_i, q_j \in Q$ are states, $s, s' \in S = \{B, 1\}$ are symbols in the work tape, $t \in T = \{0, 1\}$ is a symbol in the oracle tape, and $D, F \in \{L, R\}$ are directions (left or right). It means that if the machine is in state q_i with the heads reading symbols s and t, the machine will change to state q_j, write s' on the work tape, and then move each reading head one cell in direction D, F independently.

oracle Turing machine



- Each oracle Turing machine is determined by the finite set of instructions $\Delta \subset Q \times S \times T \times Q \times S \times \{L, R\} \times \{L, R\}$. Fix an effective coding of all set of instructions for o-machines. Let $\tilde{P}_e$ denote the eth such oracle Turing machine under this effective coding.

- If the oracle machine halts, let u be 1 + the maximum cell on the oracle tape scanned during the computation, i.e., the maximum integer tested for membership in A. We define u to be the *use of the computation*. We say that the elements $z < u$ are *used* in the computation. If no element is scanned we say the use of the computation is 0.

Turing functional and use function

- If the oracle program $\tilde{P}_e$ with A on the oracle tape and input x halts with output y and if u is the use of the computation, then we write,

$$\Phi_e^A(x) = y \quad \text{and} \quad \varphi_e^A(x) = u.$$

- If this happens in $< s$ steps and if e, x, y, and $u < s$, then we write,

$$\Phi_{e,s}^A(x) = y \quad \text{and} \quad \varphi_{e,s}^A(x) = u$$

  Note that by this convention we always have $\varphi_{e,s}^A(x) < s$.

- If $\sigma \in 2^{<\omega}$ and this happens with σ on the oracle tape and $u \leq |\sigma|$ (therefore only σ is scanned) then we write,

$$\Phi_e^\sigma(x) = y, \quad \varphi_e^\sigma(x) = u, \quad \Phi_{e,s}^\sigma(x) = y, \quad \text{and} \quad \varphi_{e,s}^\sigma(x) = u.$$

  In this case, if during the computation the head on the oracle tape tries to move right of $|\sigma| - 1$, then the computation terminate without output, i.e., $\Phi_e^\sigma(x) \uparrow$.

- If $\Phi_{e,s}^A(x)$ diverges, we write $\Phi_{e,s}^A(x) \uparrow$ and $\varphi_{e,s}^A(x) \uparrow$. Let $W_e^A = \mathrm{dom}(\Phi_e^A)$, and likewise for $W_{e,s}^A$, W_e^σ, and $W_{e,s}^\sigma$.

Turing functional and use function

- A partial function θ is *Turing computable in A* (*A-Turing computable*), written $\theta \leq_T A$, if there is an e such that $\Phi_e^A(x) \downarrow = y$ iff $\theta(x) = y$. A set B is *Turing reducible* to A ($B \leq_T A$) if the characteristic function $\chi_B \leq_T A$.

- We also allow functions f as oracles by defining Φ_e^f to be Φ_e^A where $A = \{\langle x, y\rangle : f(x) = y\}$.

- We refer to Φ_e^A as a *Turing functional* (*Turing reduction*), and we call φ_e^A the corresponding *use function*. The functional is simply determined by the program $\tilde{P}_e$ and may be partial or total. However, for every x we have $\Phi_e^A(x)$ defined iff $\varphi_e^A(x)$ is defined.

- We regard Φ_e as a partial function from 2^ω to 2^ω, which takes A to B if $\Phi_e^A = B$. We call this map a *functional* (type 2 object).

- A functional Ψ on Cantor space 2^ω is *total* if $\Psi^A(x)$ is defined for every $A \subseteq \omega$ and $x \in \omega$, i.e., $(\forall A)(\exists B)(\forall x)[\Psi^A(x) = B(x)]$.

use principle

Theorem 4.1 (Use Principle)

(i) $\Phi_e^A(x) = y \quad \Rightarrow \quad (\exists s)(\exists \sigma \prec A)\ [\Phi_{e,s}^{\sigma}(x) = y]$;

(ii) $\Phi_{e,s}^{\sigma}(x) = y \quad \Rightarrow \quad (\forall t \geq s)(\forall \tau \succeq \sigma)\ [\Phi_{e,t}^{\tau}(x) = y]$;

(iii) $\Phi_e^{\sigma}(x) = y \quad \Rightarrow \quad (\forall A \succ \sigma)\ [\Phi_e^A(x) = y]$.

Proof.

For (i), any computation which converges does so at some finite stage, having used only finitely many elements. We may take $\sigma = A \upharpoonright \varphi_e^A(x)$. (ii) and (iii) follows from the definition. □

Notation: $A \upharpoonright n = A(0)A(1)\ldots A(n-1)$ is the characteristic function of A restricted to $\{0, 1, \ldots, n-1\}$.

Corollary 4.2

For any $A, B \subseteq \mathbb{N}$, if $\varphi_{e,s}^A(x) \downarrow = u$, then

$$[\Phi_{e,s}^A(x) = y \quad \& \quad A \upharpoonright u = B \upharpoonright u] \quad \Rightarrow \quad \Phi_{e,s}^B(x) = y,$$

relativization

Theorem 4.3 (Relativized Enumeration Theorem)

There exists $i \in \omega$ such that for all sets $A \subseteq \omega$ and all $e, x \in \omega$, the partial A-computable function $\Phi_i^A(e,x)$ satisfies $\Phi_i^A(e,x) = \Phi_e^A(x)$.

Theorem 4.4 (Relativized *s-m-n* Theorem)

There is a 1:1 computable (not merely A-computable) function $s(x,y)$ such that for all sets $A \subseteq \omega$ and for all x, y, and z,

$$\Phi_{s(x,y)}^A(z) = \Phi_x^A(y,z).$$

Proof sketch.

The new program $\tilde{P}_{s(x,y)}$ on input z consists of applying program $\tilde{P}_x$ to input $\langle y, z\rangle$. This defines $s(x,y)$ as a computable function since the program $\tilde{P}_{s(x,y)}$ is independent of the oracle A. s can be made 1:1 by padding Lemma. □

Theorem 4.5 (Relativized Recursion Theorem)

(i) *For all sets $A \subseteq \omega$ and all $x, y \in \omega$, if $f(x,y)$ is an A-computable function, then there is a computable function $n(y)$ such that $\Phi^A_{n(y)} = \Phi^A_{f(n(y),y)}$.*

(ii) *Furthermore, $n(y)$ does not depend upon the oracle A, i.e., if*

$$f(x,y) = \Phi^A_e(x,y)$$

then the computable function $n(y)$ can be found uniformly in e.

Apply the proof of the Recursion Theorem with Parameters but notice that $n(y)$ is a computable function (not merely A-computable) because the function $s(x,y)$ in the *s-m-n* Theorem is computable, and hence the function $d(x,y)$ is computable, where $d(x,y)$ is obtained as in the proof of Recursion Theorem with Parameters but with all partial functions relativized to A.

relativization

Definition 4.6

B is *computably enumerable (c.e.) in* A (or A-c.e.) if $B = W_e^A$ for some e.

All previous results on c.e. sets relativize to A-c.e. sets by virtually the same proofs, where we replace "c.e." and "computable" by "A-c.e." and "A-computable." For example:

Theorem 4.7

$B \leq_T A$ iff B and $\overline{B}$ are A-c.e.

Theorem 4.8

An nonempty set B is A-c.e. iff it is the range of an A-computable total function.

properties of Turing reducibility

Proposition 4.9

For any set A, $A \equiv_{\mathrm{T}} A$ and $\overline{A} \equiv_{\mathrm{T}} A$.

Proposition 4.10

If $A \leq_{\mathrm{T}} B$ and $B \leq_{\mathrm{T}} C$ then $A \leq_{\mathrm{T}} C$.

Proposition 4.11

For any sets $A, B \subseteq \mathbb{N}$,

$$A \leq_1 B \quad \Rightarrow \quad A \leq_{\mathrm{m}} B \quad \Rightarrow \quad A \leq_{\mathrm{T}} B.$$

Turing degrees

Definition 4.12

1. $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$. (Note that $\leq_T$ is reflexive and transitive, so $\equiv_T$ is an equivalence relation.)
2. The *Turing degree* (also called *degree of unsolvability*) of A is the equivalence class $\deg(A) = \{B : B \equiv_T A\}$.
3. Lower-case boldface letters $\mathbf{a}, \mathbf{b}, \mathbf{c}$ denote degrees, and $\mathcal{D}$ denotes the class of all degrees.
4. The degrees $\mathcal{D}$ form a partially ordered set $(\mathcal{D}, \leq)$ under the relation $\deg(A) \leq \deg(B)$ iff $A \leq_T B$. We write $\deg(A) < \deg(B)$ if $A <_T B$, i.e., if $A \leq_T B$ and $B \not\leq_T A$.
5. $\deg(A) \vee \deg(B) = \deg(A \oplus B)$. With $\vee$ the degrees $(\mathcal{D}, <, \vee)$ form an *upper semi-lattice* with a supremum but not an infimum.
6. A degree $\mathbf{a}$ is *computably enumerable* if it contains a c.e. set. Let $\mathcal{E}$ denote the class of c.e. degrees with the same ordering as for $\mathcal{D}$.

Proposition 4.13

Any Turing degree is countably infinite.

Turing jump

Definition 4.14

1. Let $K^A = \{x : \Phi_x^A(x)\downarrow\} = \{x : x \in W_x^A\}$. K^A is called the *jump* of A and is denoted by A'.
2. $A^{(n)}$, the nth jump of A, is obtained by iterating the jump n times, namely $A^{(0)} = A$, $A^{(n+1)} = (A^{(n)})'$. Note that $A^{(1)} = A'$.

It follows from relativization that $K^A \equiv K_0^A$ where $K_0^A = \{\langle x, y\rangle : \Phi_x^A(y)\downarrow\}$.

Turing jump

Theorem 4.15 (Jump Theorem)

(i) A' *is* A*-c.e.*

(ii) $A' \not\leq_T A$.

(iii) B *is* A*-c.e. iff* $B \leq_1 A'$.

(iv) $A \leq_T A'$.

Proof.

(i)-(iii) follows by relativizing previous results. Note that $A' = K^A$.

(i) follows by relativizing that K is c.e.;

(ii) follows by relativizing that K is not computable.

(iii) "$\Rightarrow$:" follows by relativizing that K is 1-complete.
"$\Leftarrow$:" Suppose $B \leq_1 A'$. By (i) A' is A-c.e., then B is A-c.e.

(iv) follows by (iii) and the fact that A is A-c.e. and the fact that 1-reducibility implies Turing reducibility.

□

Turing jump

Theorem 4.15 (Jump Theorem)

(v) If A is c.e. in B and $B \leq_T C$ then A is c.e. in C.

(vi) $B \leq_T A$ iff $B' \leq_1 A'$.

(vii) If $B \equiv_T A$ then $B' \equiv_1 A'$ (and therefore $B' \equiv_T A'$).

(viii) A is c.e. in B iff A is c.e. in $\overline{B}$.

Proof.

(v) If $A \neq \emptyset$, then A is the range of some B-computable function, and hence of some C-computable function, since $B \leq_T C$.

(vi) "$\Rightarrow$:" Suppose $B \leq_T A$. By (i) B' is B-c.e. Then by (v) B' is A-c.e. Hence, $B' \leq_1 A'$ by (iii).
"$\Leftarrow$:" Suppose $B' \leq_1 A'$. As both B and $\overline{B}$ are B-c.e., by (iii), $B, \overline{B} \leq_1 B'$. Then $B, \overline{B} \leq_1 A'$. By (iii) again, both B and $\overline{B}$ are A-c.e. Hence, $B \leq_T A$.

(vii) follows immediately from (vi).

(viii) follows immediately from (v).

□

Turing jump

Let $\mathbf{a}' = \deg(A')$ for $A \in \mathbf{a}$. Note that $\mathbf{a}' > \mathbf{a}$ and $\mathbf{a}'$ is c.e. in $\mathbf{a}$. By the Jump Theorem the jump is well defined on degrees. Let $\mathbf{0}^{(n)} = \deg(\emptyset^{(n)})$. Thus, we have an infinite hierarchy of degrees,

$$\mathbf{0} < \mathbf{0}' < \mathbf{0}'' < \cdots < \mathbf{0}^{(n)} < \ldots$$

The first few degrees in this hierarchy are of special importance.

$$\begin{aligned}
\mathbf{0} &= \deg(\emptyset) = \{B : B \text{ is computable}\};\\
\mathbf{0}' &= \deg(\emptyset'), \text{ where } \emptyset' := K^{\emptyset} \equiv K \equiv K_0 \equiv K_1;\\
\mathbf{0}'' &= \deg(\emptyset'') = \deg(\mathrm{Fin}) = \deg(\mathrm{Tot}) = \deg(\mathrm{Inf});\\
\mathbf{0}''' &= \deg(\emptyset''') = \deg(\mathrm{Cof}) = \deg(\mathrm{Rec}) = \deg(\mathrm{Ext}).
\end{aligned}$$

By definition, degree $\mathbf{0}$ is the least degree and consists precisely of the computable sets. Degree $\mathbf{0}'$ is the degree of the halting problem.