Logic and Computation II

Part 7. Recursion-theoretic hierarchies

Kazuyuki Tanaka

BIMSA

May 29, 2025

北京雁栖湖应用数学研究院
Beijing Institute of Mathematical Sciences and Applications

Logic and Computation II

- **Part 4. Modal logic** (7 lectures)
- **Part 5. Modal μ-calculus** (5 lectures)
- **Part 6. Automata on infinite objects** (8 lectures)
- **Part 7. Recursion-theoretic hierarchies** (4 lectures)

Part 7. Schedule

- May 20, (1) Oracle computation and relativization
- May 22, (2) m-reducibility and simple sets
- May 27, (3) T-reducibility and Post's problem
- May 29, (4) Miscellaneous

Recap

- $A \leq_{\mathrm{m}} B$, if there exists a computable function f s.t. $x \in A \Leftrightarrow f(x) \in B$ for any x.
- $A \leq_{\mathrm{T}} B$, if A is computable in oracle B (i.e., recursive in χ_B).
- A CE set A is **(T-)complete** / **m-complete** if $B \leq_{\mathrm{T}} A$ / $B \leq_{\mathrm{m}} A$ for any CE set B.

Theorem 7.12 (Post's theorem, 1944)

There exists a CE set that is neither computable nor m-complete.

- **Post's problem**: Is there a CE set that is neither computable nor (T-)complete.
- To challenge this problem, various notions of CE set were introduced.
  A **simple** set is a CE set that has a nonempty intersection with any infinite CE set and whose complement is an infinite set. A simple set satisfies Post's theorem.
- A set A is a **low** set if $A' := \mathrm{K}^A \leq_{\mathrm{T}} \mathrm{K}$. A simple low set is a solution to Post's problem.

Lemma 7.21

$A <_{\mathrm{T}} \mathrm{K}$ if A is a low set.

Proof. If A is a low set, $A <_{\mathrm{T}} A' \leq_{\mathrm{T}} \mathrm{K}$, and so $A <_{\mathrm{T}} \mathrm{K}$. □

Lemma 7.22 (main lemma for Post's problem)

There exists a simple low set.

Proof

- In the finite injury priority argument, a desired CE set A is constructed as the infinite sum $\bigcup_s A_s$ of finite sets A_s, where $A_0 = \varnothing$ and A_s is "the (finite) set of numbers that are verified to be members of A within s step". Once an element is determined to be a member of A, it is never removed. Thus $A_s \subset A_{s+1}$ for each s.
- To ensure that A is low and simple, we construct A_s to satisfy several requirements. A **positive requirement** is satisfied by adding some elements to a desired set A and a **negative requirement** is by excluding some elements from A.
- Satisfying one requirement may **injure** another requirement that is already satisfied. So, **priorities** are set to all requirements, so that a requirement will be injured by only a finite number of requirements (with higher-priority).

- A is low and simple if all of the following are satisfied.
  (i) A is CE,
  (ii) A^c is infinite,
  (iii) A has a common element with each infinite CE set, and
  (iv) $\mathrm{K}^A \leq_{\mathrm{T}} \mathrm{K}$.

- In the above, condition (i) naturally holds from the inductive construction of A. Condition (ii) is also easily satisfied.

- The essential ones are the positive condition (iii) and the negative condition (iv). Rewriting these into *requirements* for each e, we have

$$\begin{array}{rcl} P_e & : & |W_e| = \infty \Rightarrow A \cap W_e \neq \varnothing \\ N_e & : & \exists^\infty s\ \varphi_{e,s}^{A_s}(e) \downarrow \Rightarrow \varphi_e^A(e) \downarrow . \end{array}$$

  Here, $\exists^\infty$ means "exists infinitely many". By "$\varphi_{e,s}^{A_s}(x) = y$", we denote the computation of $\varphi_e^A(x) = y$ will be completed within s steps, and if it exceeds s steps, we denote it as $\varphi_{e,s}^A(x) \uparrow$.

- It is clear that (iii) holds if P_e holds for each e.
- Next, we show that (iv) holds if N_e holds for each e. First, assume that $s \mapsto A_s$ is computable.
  If N_e holds, then

$$\exists^\infty s\ \varphi_{e,s}^{A_s}(e)\downarrow \Rightarrow\ \varphi_e^A(e)\downarrow \Rightarrow\ \exists t \forall s > t\ \varphi_{e,s}^{A_s}(e)\downarrow$$

$$\Rightarrow\ \forall t \exists s > t\ \varphi_{e,s}^{A_s}(e)\downarrow \equiv\ \exists^\infty s\ \varphi_{e,s}^{A_s}(e)\downarrow .$$

- Thus, $\mathrm{K}^A = \{e : \varphi_e^A(e)\downarrow\}$ is a Δ_2 set.

Corollary 7.11 (Revisited)

A is Δ_2 if and only if $A \leq_T \mathrm{K}$.

- By the above fact, we have $\mathrm{K}^A \leq_{\mathrm{T}} \mathrm{K}$.

- Now we explain why N_e is a negative requirement.
- We define the following computable function r as a tool to control N_e:

$$r(e, s) = u(A_s, e, e, s).$$

  Here, the right-hand side is called the **use function**, which is 1 + the maximum number used in the computation of $\varphi_{e,s}^{A_s}(e)$, and 0 if the computation never halts.
- Assuming $s \mapsto A_s$ is computable, r is also computable, called a **restraint function**.
- That is, given A_s such that $\varphi_{e,s}^{A_s}(e)\downarrow$, unless an element $x < r(e, s)$ is added to A, we have $A \upharpoonright r = A_s \upharpoonright r$, so $\varphi_e^{A}(e)\downarrow$, and thus N_e is fullfilled as a negative requirement.

- Among all P_e and N_e, set the priority as
$$P_0 > N_0 > P_1 > N_1 > P_2 > N_2 > \dots$$
- Note that for any requirement there are only a finitely many requirements with higher priorities. So, numbers below $r(e,s)$ may be added to A only for P_i with $i < e$.
- Now, we show the construction of A.
- Step $s = 0$: Set $A_0 = \varnothing$.
Step $s+1$: Assume that A_s is obtained.
If there is an $i \leq s$ which satisfies (i) $W_{i,s} \cap A_s = \varnothing$, and
(ii) $\exists x \in W_{i,s}(x > 2i \wedge \forall e \leq i\ r(e,s) < x)$,
then take the smallest such i and choose the smallest x that satisfies (ii) and set $A_{s+1} = A_s \cup \{x\}$.

  Then the requirement P_i is satisfied, and after $s+1$ it will never be injured.

  If there is no such $i \leq s$, put $A_{s+1} = A_s$.

- When $A_{s+1} = A_s \cup \{x\}$, for e such that $x \le r(e,s)$, N_e is **injured** by x at $s+1$. However, we have

Claim 1

For every e, N_e is injured at most finitely many times.

$(\because)$ N_e can be injured only by P_i for $i < e$.

Claim 2

For all e, $r(e) = \lim_s r(e,s)$ exists and hence N_e holds.

$(\because)$ Fix any e. From Claim 1, there exists a step s_e such that N_e is not injured after s_e. But if $\varphi^{A_s}_{e,s}(e)\downarrow$ for $s > s_e$, then for $t \ge s$, $r(e,t) = r(e,s)$ and so $r(e) = \lim_s r(e,s)$ exists. Hence $A_s \restriction r = A \restriction r$ and $\varphi^A_e(e)\downarrow$, which implies N_e holds.

Claim 3

P_i holds for all i.

$(\because)$ Suppose that W_i is an infinite set. From Claim 2, we take such an s that

$$\forall t \geq s \ \forall e \leq i \ r(e,t) = r(e).$$

We may assume that no P_j with $j < i$ receives attention after $s'(\geq s)$, In addition, take $t > s'$ such that

$$\exists x \in W_{i,t}(x > 2i \wedge \forall e \leq i \ r(e) < x).$$

Then we already have $W_{i,t} \cap A_t \neq \varnothing$ or P_i receives attention at $t+1$. In either case, $W_{i,t} \cap A_{t+1} \neq \varnothing$, and so P_i holds.

From the above, $A = \bigcup_{s \in \mathbb{N}} A_s$ is a simple low set. Also, A^c is infinite, since from condition (ii) that $x > 2i$, we have $| \{x \in A : x \leq 2i\} | \leq i$. $\square$

Friedberg and Mucinik actually proved the following assertion.

Theorem 7.23 (Friedberg, Muchnik)

There exist CE sets A, B such that $A \not\leq_{\mathrm{T}} B$ and $B \not\leq_{\mathrm{T}} A$.

It is clear that A, B in this theorem are neither computable nor complete. By the finite injury priority argument, these sets are constructed as $A = \bigcup_s A_s$ and $B = \bigcup_s B_s$ with the following requirements:

$$\begin{aligned} R_{2e} &: \quad A \neq W_e^B \\ R_{2e+1} &: \quad B \neq W_e^A \end{aligned}$$

Among many generalizations of the above theorem, the following theorem is particularly important.

Theorem 7.24 (G. E. Sacks)

Let C be an incomputable CE set.
(1) There is a simple set A such that $C \not\leq_{\mathrm{T}} A$.
(2) There exists low CE sets A, B s.t. $A \not\leq_{\mathrm{T}} B$ and $B \not\leq_{\mathrm{T}} A$ with $C = A \cup B$ and $A \cap B = \varnothing$.

Relativized arithmetical hierarchy

- $A = W_e^\xi$ is called ξ-CE if it is the domain of a partial recurisive function $\{e\}^\xi$ with oracle ξ. In particular when $\xi = \chi_B$, we say A is CE in B.
- A set A is computable in B if A is recursive in χ_B, written as $A \leq_T B$.

Then a **relativized arithmetical hierarchy** for subsets of $\mathbb{N}^k$ is defined as follows.

$$\begin{aligned}
\Sigma_1(\xi) &:= \{\xi\text{-CE sets}\},\\
\Delta_1(\xi) &:= \{\xi\text{-computable sets}\},\\
\Sigma_{n+1}(\xi) &:= \{A \mid A \text{ is CE in some } B \in \Sigma_n(\xi)\},\\
\Delta_{n+1}(\xi) &:= \{A \mid A \text{ is computable in some } B \in \Sigma_n(\xi)\},\\
&:= \{A \mid A \leq_T B \text{ for some } B \in \Sigma_n(\xi)\},\\
\Pi_n(\xi) &:= \{\text{the complement of sets in } \Sigma_n(\xi)\}
\end{aligned}$$

When ξ is a computable function, we omit to mention (ξ) or ξ, and classes $\Sigma_n, \Pi_n, \Delta_n$ are usual **arithmetical hierarchy** .

- We write $A \leq_{\mathrm{m}} B$ if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that for any $x \in \mathbb{N}$, $x \in A \Leftrightarrow f(x) \in B$.
- A set B is called **m-hard** if $A \leq_{\mathrm{m}} B$ for every CE set A.
  A set B is called **m-complete** if B is CE and m-hard.

In the following, such CE sets will be generalized to $\mathcal{C} = \Sigma_n$, etc.

- Let $\mathcal{C}$ be a class of sets.
  A set B is said to be **$\mathcal{C}$-hard** if for every $A \in \mathcal{C}$, $A \leq_{\mathrm{m}} B$.
  A set B is said to be **$\mathcal{C}$-complete** if B is $\mathcal{C}$-hard and $B \in \mathcal{C}$.
- Clearly, if $A \leq_{\mathrm{m}} B$ and $B \in \Sigma_n$ (Π_n, Δ_n), then so is A.
- A Σ_n-complete set is not Π_n, since arithmetical hierarchy is strict.

Now, the following are typical m-complete sets.

(i) K $= \{e : e \in W_e\}$ is Σ_1-complete.

(ii) MEM $= \{(e, x) : x \in W_e\}$ is Σ_1-complete.

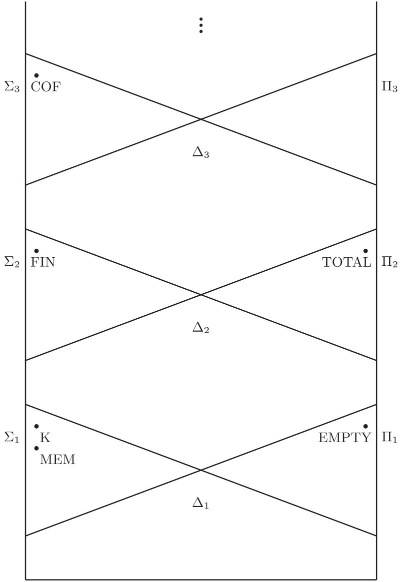
(iii) EMPTY $= \{e : W_e = \varnothing\}$ is Π_1-complete.

(iv) FIN $= \{e : W_e$ is finite$\}$ is Σ_2-complete.

(v) TOTAL $= \{e : \{e\}$ is a total function $\}$ is Π_2-complete.

(vi) COF $= \{e :$ the complement of W_e is finite$\}$ is Σ_3-complete.

(vii) REC $= \{e : W_e$ is recursive$\}$ is Σ_3-complete.



Polynomial-time reducibility

- Finally, we discuss the polynomial-time versions of m-reduction and T-reduction.
- A is **polynomial (time) reducible** to B ($A \leq_{\mathrm{P}} B$) if there exists a polynomial time computable function f and $x \in A \Leftrightarrow f(x) \in B$. This is a kind of m-reduciblity, which also written as $A \leq_{\mathrm{m}}^{\mathrm{P}} B$.
- On the other hand, A is said to be **polynomial-time Turing reducible** to B ($A \leq_{\mathrm{T}}^{\mathrm{P}} B$ or $A \in \mathrm{P}^B$) if there exists a polynomial q and a deterministic Turing machine M^B with oracle B that can decide whether $x \in A$ within $O(q(|x|))$ time.
- We will not consider how to measure the time required for querying the oracle ($n \in B$). We only treat it very naively as shown in the proof of the next theorem.
- Furthermore, making M^B nondeterministic, we also defines $A \in \mathrm{NP}^B$.
- If $A \leq_{\mathrm{m}}^{\mathrm{P}} B$ then $A \leq_{\mathrm{T}}^{\mathrm{P}} B$. The reverse does not hold over a large class such as EXP(TIME) (Ladner, Lynch, and Selman [1975]).

Theorem 7.25 (Baker, Gill, Solovay (1975))

(1) There exists a computable oracle A such that $\mathrm{P}^A = \mathrm{NP}^A$.
(2) There exists a computable oracle A such that $\mathrm{P}^A \neq \mathrm{NP}^A$.

Proof To show (1)

- Let A be a PSPACE complete problem such as TQBF (Lecture02-06). First, obviously $\mathrm{P}^A \subset \mathrm{NP}^A \subset \mathrm{PSPACE}^A$.
- Since A is PSPACE, one can compute PSPACE^A in PSPACE without using A as an oracle. That is, $\mathrm{PSPACE}^A \subset \mathrm{PSPACE}$.
- Finally, due to the PSPACE completeness of A, $\mathrm{PSPACE} \subset \mathrm{P}^A$.
- Therefore, $\mathrm{P}^A = \mathrm{NP}^A = \mathrm{PSPACE}^A$.

To show (2)

- For any $A \subset \{0,1\}^*$, $B = \{0^{|x|} : x \in A\}$ is in NP^A.
- So, we only need to construct a computable $A = \bigcup_s A_s$ such that $B \notin \mathrm{P}^A$.
- Let M_e enumerate deterministic machines (or sets accepted by such machines) running in polynomial p_e time.
- We want to prove $R_e : M_e^A \neq B$ for all e. That is, for each e, we guarantee the existence of n such that
$$M_e^A(0^n) \neq B(0^n).$$
- Assume that A_s is constructed at step $s = e$. Then, take n greater than any number used in the previous constructions and $2^n > p_e(n)$.
- When $M_e^{A_s}(0^n) = 1$, set $A_{s+1} = A_s$. Then, no words with length n are added to A and also will never be added, and hence we have $B(0^n) = 0$.
- Next assume $M_e^{A_s}(0^n) = 0$. This computation queries the oracle A_s at most $p_e(n)$ times, and so by the assumption $2^n > p_e(n)$, there is a word x of length n that is irrelevant to the oracle query. Then, by setting $A_{s+1} = A_s \cup \{x\}$, we have $M_e^{A_{s+1}}(0^n) = 0$ but $B(0^n) = 1$.

□

Polynomial time hierarchy

Finally, we introduce the polynomial-time version of arithmetical hierarchy.
We defined P^A and NP^A for the set $A \subset \Omega^*$. For a class $\mathcal{C}$ of sets,

$$\mathsf{P}(\mathcal{C}) = \bigcup_{A \in \mathcal{C}} \mathsf{P}^A, \quad \mathsf{NP}(\mathcal{C}) = \bigcup_{A \in \mathcal{C}} \mathsf{NP}^A.$$

Definition 7.26 (Polynomial time hierarchy)

The **polynomial-time hierarchy** (PH) is defined inductively defined as follows

- $\Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} = \mathsf{P}$,
- $\Sigma_{n+1}^{\mathrm{P}} = \mathsf{NP}(\Sigma_n^{\mathrm{P}})$,
- $\Pi_{n+1}^{\mathrm{P}} = \text{co-}\Sigma_{n+1}^{\mathrm{P}}$,
- $\Delta_{n+1}^{\mathrm{P}} = \mathsf{P}(\Sigma_n^{\mathrm{P}})$
- $\mathsf{PH} = \bigcup_n \Sigma_n^{\mathrm{P}}$

Then it is easy to see that:

Lemma 7.27

PH $\subset$ PSPACE

Proof. NP(PSPACE) $\subset$ PSPACE(PSPACE) $\subset$ PSPACE. $\square$

Lemma 7.28

If PH $=$ PSPACE, then $\Sigma_n^{\mathrm{P}} = \Sigma_{n+1}^{\mathrm{P}}$ for some n.

Proof. If TQBF $\in \Sigma_n^{\mathrm{P}}$ then PSPACE $\subset \Delta_{n+1}^{\mathrm{P}}$. $\square$

Homework

Given A as an NP-complete set, show the following.

(1) $\Sigma_1^{\mathrm{P}} = \{B : B \leq_{\mathrm{m}}^{\mathrm{P}} A\}$.

(2) $\Delta_2^{\mathrm{P}} = \{B : B \leq_{\mathrm{T}}^{\mathrm{P}} A\}$.

(3) $\Sigma_{n+1}^{\mathrm{P}} = \Sigma_n^{\mathrm{P}}(A)$.

Further Reading

- Kozen, D. C. (2006). *Theory of computation* (Vol. 170). Heidelberg: Springer.
- Soare, R. I. (2016). *Turing computability. Theory and Applications of Computability*. Springer.

Thank you for your attention!