計算極限之外: 可計算枚舉性與映射歸約

完整教材 (依據 5CCS2FC2 Foundations of Computing II 第十一週內容編寫)

半可判定性、交錯模擬、co-C.E. 與映射歸約

整理自 Dr. Christopher Hampson(King's College London) 之投影片
並參考補充文獻擴充編寫

2026 年 7 月

本週學習目標

1. 理解**可計算枚舉 (C.E.)** 語言: 放棄「必停機」、只保留健全與完備。
2. 證明 HALT_{TM} 與非空問題 $\overline{E}_{\text{TM}}$ 是 C.E., 並掌握**交錯模擬 (dovetailing)** 技巧。
3. 理解 **co-C.E.** 與核心定理:L 與 $\overline{L}$ 皆 C.E. $\Rightarrow$ L 可判定。
4. 由此推出 $\overline{\text{HALT}}_{\text{TM}}$、$E_{\text{TM}}$ 不是 C.E.——存在「連半個演算法都沒有」的問題。
5. 掌握**映射歸約** $A \leq_m B$ 的正式定義、基本定理與使用方法。
6. 完整走過兩個歸約:$\text{HALT}_{\text{TM}} \leq_m A_{\text{TM}}$ 與 $A_{\text{TM}} \leq_m \overline{E}_{\text{TM}}$。

Contents

1 可計算枚舉性 (Computable Enumerability)

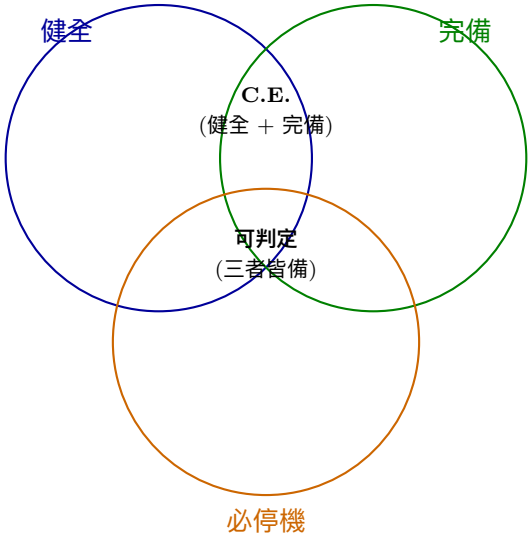
1.1 定義: 放棄「必停機」

上週看到: 可判定 = 健全 + 完備 + 必停機, 而 HALT_{TM} 不可判定。如果放棄第三個條件呢?

定義 1.1 (可計算枚舉 C.E.)**.** 語言 L 是**可計算枚舉的 (computably enumerable, C.E.; 也稱遞迴可枚舉 recursively enumerable、圖靈可辨識、半可判定)**, 若存在圖靈機 M 滿足:

- **健全 (Sound)**: 若 $M(w) = 1$, 則 $w \in L$;
- **完備 (Complete)**: 若 $w \in L$, 則 $M(w) = 1$。

(不要求必停機: 當 $w \notin L$ 時,M 可以拒絕、也可以永遠跑下去。)

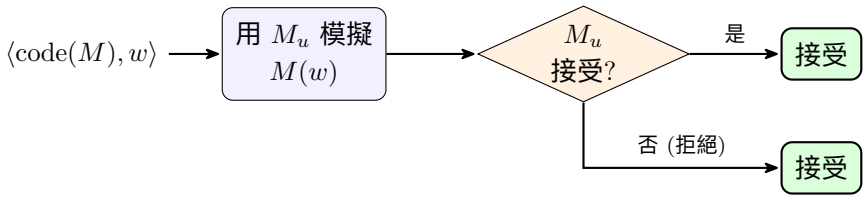


備註 1.2 (為什麼叫「枚舉」?). 等價的觀點:L 是 C.E. $\iff$ 存在一台機器能把 L 的成員一個個列印出來 (L 的「產生器」)。給定辨識機 M, 用下一節的交錯模擬跑遍所有字, 誰被接受就印誰; 反之, 給定產生器, 辨識 w 時只要盯著清單、看到 w 就接受。「可以列出清單」與「成員可被確認」是同一件事。

1.2 停機問題是 C.E.

定理 1.3. 停機問題 HALT_{TM} 是可計算枚舉的。

Proof. **步驟 1(建構).** 用通用圖靈機 M_u 造出健全且完備的機器 halt$^-$:



亦即: 模擬 $M(w)$; 只要模擬停了 (不論 M 接受或拒絕),halt$^-$ 都接受。若 $M(w)$ 不停機,halt$^-$ 也跟著永遠跑下去——沒關係,C.E. 允許!

步驟 2–4(驗證).

$$\mathsf{halt}^{-}(\langle \mathrm{code}(M), w\rangle) = 1 \iff M_u(\langle \mathrm{code}(M), w\rangle) \in \{1, 0\} \iff M(w) \in \{1, 0\}$$
$$\iff \langle \mathrm{code}(M), w\rangle \in \textsc{Halt}_{\mathrm{TM}}.$$

「⇐」是完備、「⇒」是健全。 □

備註 1.4. 直觀: 停機問題有**單向驗證**的結構——「會停」這件事, 等它停下來就驗證完畢 (有限時間內看得到證據);「不會停」卻永遠等不到證據。C.E. 正是「Yes 端有有限證據」的類。

1.3 非空問題是 C.E.: 交錯模擬 (Dovetailing)

定義 1.5 (非空問題). $\overline{E}_{\mathrm{TM}} = \{\mathrm{code}(M) : \mathrm{Language}(M) \neq \emptyset\}$——$M$ 是否至少接受一個字串?

定理 1.6. 非空問題 $\overline{E}_{\mathrm{TM}}$ 是可計算枚舉的。

Proof. **步驟 1.** 把所有輸入字枚舉為 $w_0, w_1, w_2, \ldots$。

步驟 2(第一次嘗試——失敗). 直覺的演算法:

Algorithm 1 $\mathsf{empty}^{-}(\mathrm{code}(M))$——天真版

```
for i = 0, 1, 2, 3, ... do
    在 ⟨code(M), w_i⟩ 上執行 M_u
    if M_u 接受 then return True
```

健全 ✓: 回傳 True 時確實找到了被接受的字。**完備** ✗: 若 M 在 w_0 上不停機, 演算法卡在 $i = 0$, 即使 M 其實接受 w_5 也永遠問不到——漏報!

步驟 2(第二次嘗試——交錯模擬). 給每一輪限定步數, 逐步放寬:

Algorithm 2 $\mathsf{empty}^{-}(\mathrm{code}(M))$——交錯模擬版

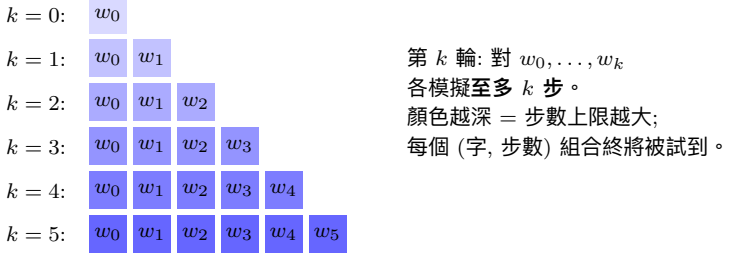
```
for k = 0, 1, 2, 3, ... do
    for i = 0, 1, 2, ..., k do
        在 ⟨code(M), w_i⟩ 上執行 M_u 至多 k 步
        if M_u 接受 then return True
```

健全 ✓ 且**完備** ✓: 若 M 接受某個 w_i, 設它花 s 步; 當 $k \geq \max(i, s)$ 時, 該回合必然模擬到 w_i 且步數足夠, 演算法回傳 True。任何單一的無窮模擬都不再堵路——因為每輪都限時。

步驟 3–5(驗證).

$$\mathsf{empty}^{-}(\mathrm{code}(M)) = 1 \iff \exists\, w_i :\ M(w_i) = 1 \iff \mathrm{Language}(M) \neq \emptyset \iff \mathrm{code}(M) \in \overline{E}_{\mathrm{TM}}.$$

□



備註 1.7 (Dovetailing). 這個「所有計算各跑一點、輪流推進」的技巧稱為**交錯模擬 (dovetailing, 原意為木工的鳩尾榫)**。它是 C.E. 理論的萬用工具: 要「平行」執行無窮多個可能不停機的模擬, 就把 (對象, 步數) 配對排成表格逐格推進——任何終將發生的接受, 都會在有限時間內被看到。

2 co-C.E. 與「連半個演算法都沒有」的問題

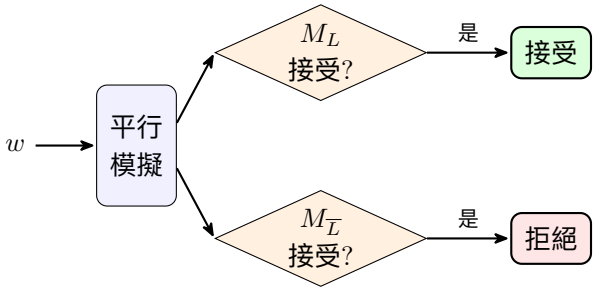
2.1 co-C.E. 與核心定理

定義 2.1 (co-C.E.)**.** 語言 L 是 **co-C.E.** 的, 若其補集 $\overline{L}$ 是 C.E.(即:「No 端」有有限證據)。

定理 2.2. 若 L 與 $\overline{L}$ 都是 *C.E.*, 則 L 可判定。

Proof. **步驟 1–2.** 設 M_L 對 L 健全完備、$M_{\overline{L}}$ 對 $\overline{L}$ 健全完備。

步驟 3(平行執行). 建構 M': 輪流各推進一步地**平行執行** $M_L(w)$ 與 $M_{\overline{L}}(w)$:



任何 w 必屬 L 或 $\overline{L}$ 之一; 由完備性, 對應的那台機器終將接受。故 M' 必停機, 且由健全性答案正確——M' 是 L 的判定器。 □

2.2 兩個「不是 C.E.」的問題

定理 2.3. 非停機問題 $\overline{HALT}_{\mathrm{TM}} = \{\langle \mathrm{code}(M), w\rangle : M(w) = \uparrow\}$ **不是** *C.E.*。

Proof. 若 $\overline{\textsc{Halt}}_{\mathrm{TM}}$ 是 C.E., 又已知 $\textsc{Halt}_{\mathrm{TM}}$ 是 C.E.(定理 1.3), 則由定理 2.2,$\textsc{Halt}_{\mathrm{TM}}$ 可判定——與上週的不可判定性矛盾。 □

定理 2.4. 空語言問題 $E_{\mathrm{TM}} = \{\mathrm{code}(M) : \mathrm{Language}(M) = \emptyset\}$ **不是** *C.E.*。

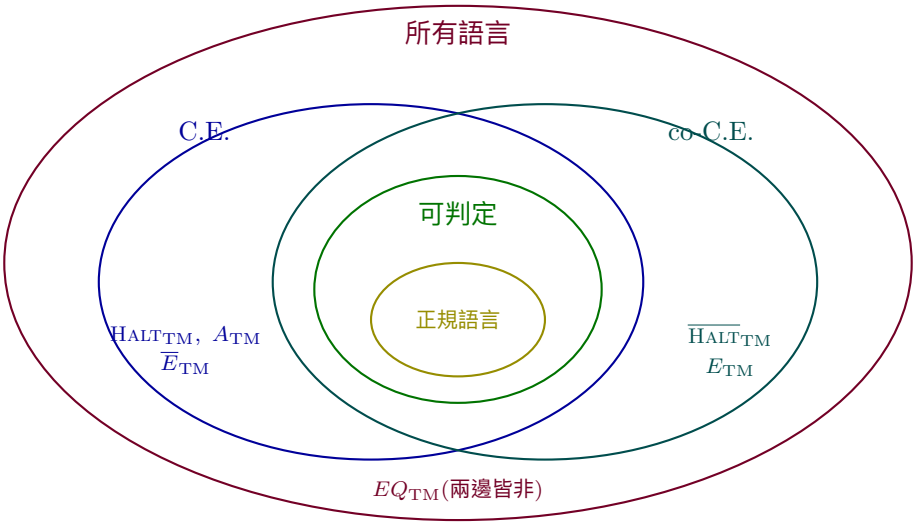
Proof. E_{TM} 的補集 $\overline{E}_{\mathrm{TM}}$ 是 C.E.(定理 1.6)。若 E_{TM} 也是 C.E., 則 E_{TM} 可判定 (定理 2.2), 與上週 ($A_{\mathrm{TM}} \leq_m E_{\mathrm{TM}}$) 的不可判定性矛盾。 □

備註 2.5 (三個層次的「難」).

- **可判定**: 有完美演算法 (如: 質數測試)。
- **C.E. 但不可判定**: 有「半個演算法」——Yes 能確認、No 會空等 (如 $\textsc{Halt}_{\mathrm{TM}}$、$A_{\mathrm{TM}}$、$\overline{E}_{\mathrm{TM}}$)。
- **連 C.E. 都不是**: 連半個演算法都沒有,Yes 端連有限證據都不存在 (如 $\overline{\textsc{Halt}}_{\mathrm{TM}}$、$E_{\mathrm{TM}}$;$EQ_{\mathrm{TM}}$ 更是兩邊都不是)。

「不會停機」沒有有限證據: 看一百萬步沒停, 不代表下一步不停——這正是 $\overline{\textsc{Halt}}_{\mathrm{TM}}$ 無法半判定的直觀原因。

2.3 完整的語言階層



可判定 = C.E. ∩ co-C.E.(定理 2.2); 左右兩瓣不對稱地各自延伸, 而絕大多數語言落在三者之外。

3 映射歸約 (Mapping Reductions)

上週的歸約是「敘述式」的; 本週把它正式化, 成為可以精確操作的數學物件。

定義 3.1 (映射歸約)**.** 從問題 A 到問題 B 的**映射歸約**是一個可計算函數 $f : \Sigma^* \to \Sigma^*$, 使得對所有 w:

$$w \in A \iff f(w) \in B.$$

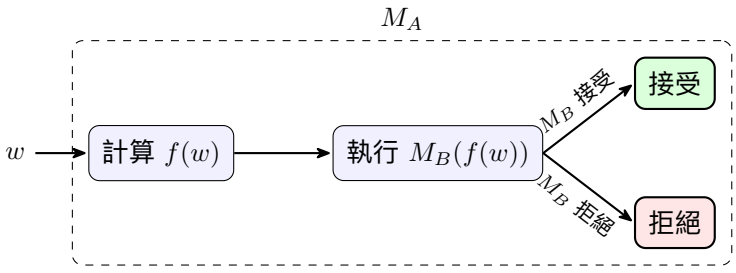
記作 $A \leq_m B$(A 可映射歸約到 B; 讀作「A 不比 B 難」)。

備註 3.2. 兩個要件缺一不可:(1) f **可計算** (有必停機的圖靈機算出 $f(w)$);(2) **雙向**等價——$w \in A \Rightarrow f(w) \in B$ 且 $w \notin A \Rightarrow f(w) \notin B$。$f$ 不必是單射或滿射。

定理 3.3. 若 $A \leq_m B$ 且 B 可判定, 則 A 可判定。

Proof. **步驟 1–3.** 設 f 是歸約、M_B 是 B 的判定器 ($u \in B \iff M_B(u) = 1$, 必停機)。

步驟 4(組合). 定義 $M_A(w) = M_B(f(w))$:



f 可計算且 M_B 必停機, 故 M_A 必停機; 且

$$w \in A \iff f(w) \in B \iff M_B(f(w)) = 1 \iff M_A(w) = 1.$$

□

推論 3.4 (實戰形式)**.** 若 $A \leq_m B$ 且 A 不可判定, 則 B 不可判定。

映射歸約的基本性質

1. **遞移性**:$A \le_m B$ 且 $B \le_m C \Rightarrow A \le_m C$(合成 $g \circ f$ 仍可計算)。
2. **保持 C.E.**:$A \le_m B$ 且 B 是 C.E. $\Rightarrow A$ 是 C.E.; 逆否:A 非 C.E. $\Rightarrow B$ 非 C.E.。
3. **補集**:$A \le_m B \iff \overline{A} \le_m \overline{B}$(同一個 f)。只能兩邊同時取補, 不能單邊!
4. 方向莫反: 要證 B 難, 需要「難題 $\le_m B$」, 不是「$B \le_m$ 難題」。

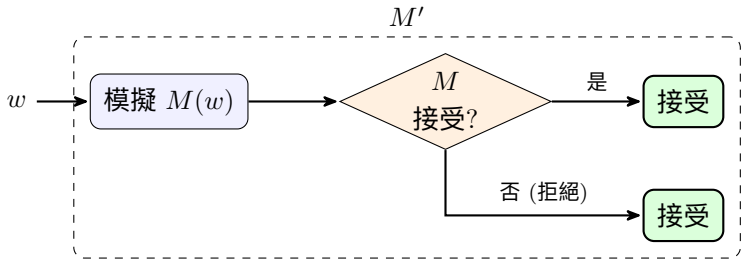
4 應用一:$\mathbf{Halt}_{\mathrm{TM}} \le_m A_{\mathrm{TM}}$

定理 4.1. 接受問題 $A_{\mathrm{TM}} = \{\langle \mathrm{code}(M), w\rangle : M(w) = 1\}$ 不可判定。

Proof. **步驟 1(目標).** $\textsc{Halt}_{\mathrm{TM}}$ 不可判定, 故只需給出 $\textsc{Halt}_{\mathrm{TM}} \le_m A_{\mathrm{TM}}$, 即找可計算的 f 使

$$\langle \mathrm{code}(M), w\rangle \in \textsc{Halt}_{\mathrm{TM}} \iff f(\langle \mathrm{code}(M), w\rangle) \in A_{\mathrm{TM}}.$$

步驟 2(改造機器). 對任意 M, 建構 M':「把拒絕改成接受」——



$$M'(w) = 1 \iff M(w) = 1 \text{ 或 } M(w) = 0 \iff M \text{ 在 } w \text{ 上停機.}$$

(若 $M(w)$ 不停機, 模擬也不停,$M'(w) = \uparrow$。)

步驟 3(歸約函數). 令

$$f(u) = \begin{cases} \langle \mathrm{code}(M'), w\rangle & \text{若} u = \langle \mathrm{code}(M), w\rangle \text{ 格式正確} \\ \varepsilon & \text{否則 (垃圾輸入送到} A_{\mathrm{TM}} \text{外的定點)} \end{cases}$$

f 只是對機器編碼做語法改寫 (把指向 q_{reject} 的轉移改指 q_{accept}), 顯然可計算。且

$$\langle \mathrm{code}(M), w\rangle \in \textsc{Halt}_{\mathrm{TM}} \iff M'(w) = 1 \iff \langle \mathrm{code}(M'), w\rangle \in A_{\mathrm{TM}}.$$

由推論 3.4,A_{TM} 不可判定。 □

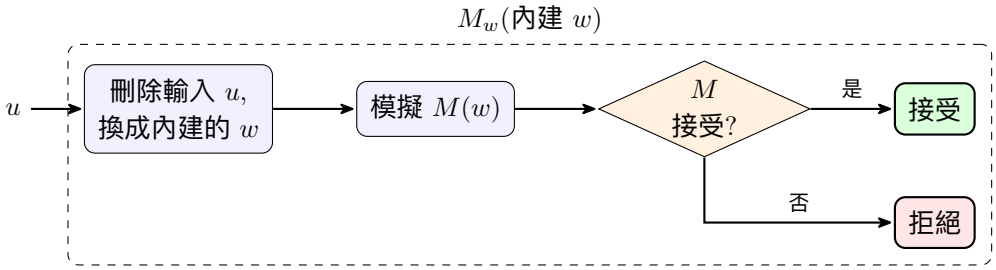
5 應用二:$A_{\mathrm{TM}} \le_m \overline{E}_{\mathrm{TM}}$

定理 5.1. 非空問題 $\overline{E}_{\mathrm{TM}}$ 不可判定 *(*從而 E_{TM} 也不可判定*)*。

Proof. **步驟 1(目標).** 給出 $A_{\mathrm{TM}} \le_m \overline{E}_{\mathrm{TM}}$: 找可計算的 f 使

$$\langle \mathrm{code}(M), w\rangle \in A_{\mathrm{TM}} \iff f(\langle \mathrm{code}(M), w\rangle) \in \overline{E}_{\mathrm{TM}}.$$

步驟 2(改造機器). 對給定的 M 與 w, 建構 M_w(把 w 寫死在機器裡):



M_w 無視自己的輸入 u, 永遠模擬同一件事 $M(w)$。故其語言只有兩種可能:

$$M_w(u) = 1 \iff M(w) = 1 \quad (\text{對所有} u) \qquad \Longrightarrow \qquad \text{Language}(M_w) = \begin{cases} \Sigma^* & \text{若} M(w) = 1 \\ \emptyset & \text{否則} \end{cases}$$

於是 $\text{Language}(M_w) \neq \emptyset \iff M(w) = 1$。

步驟 3(歸約函數). 令

$$f(u) = \begin{cases} \text{code}(M_w) & \text{若} u = \langle \text{code}(M), w \rangle \\ \varepsilon & \text{否則} \end{cases}$$

f 可計算 (把 w 寫進 M 的編碼前面加一段「清空帶子、抄入 w」的固定程式)。且

$$\langle \text{code}(M), w \rangle \in A_{\text{TM}} \iff \text{Language}(M_w) \neq \emptyset \iff \text{code}(M_w) \in \overline{E}_{\text{TM}}.$$

由推論 3.4,$\overline{E}_{\text{TM}}$ 不可判定; 可判定語言對補集封閉, 故 E_{TM} 亦不可判定。 □

備註 5.2 (「寫死輸入」是歸約的萬用招). M_w 的技巧——把資料燒進程式——是無數歸約的核心: 它把「一對 $\langle$機器, 輸入$\rangle$ 的問題」轉成「單一機器的性質問題」, 是連接 A_{TM} 與各種語意性質 (空性、正規性、等價性……, 參見 Rice 定理) 的橋樑。

6 補充: 映射歸約 vs. 圖靈歸約、RE-完全性

- **圖靈歸約** $A \leq_T B$: 解 A 時可把 B 的判定器當神諭 *(oracle)* 反覆呼叫、還能翻轉答案。映射歸約是其特例: 恰好呼叫一次、答案原封不動。因此 $\leq_m$ 更精細: 圖靈歸約下 $\text{HALT}_{\text{TM}} \leq_T \overline{\text{HALT}}_{\text{TM}}$(問完取反即可), 但映射歸約下 $\text{HALT}_{\text{TM}} \not\leq_m \overline{\text{HALT}}_{\text{TM}}$(否則 $\overline{\text{HALT}}_{\text{TM}}$ 會是 C.E.)——這正是為什麼「證明非 C.E.」必須用映射歸約。
- **RE-完全性**:A_{TM} 與 HALT_{TM} 是 C.E. 類的「最難者」——所有 C.E. 語言都可映射歸約到它們 (用通用機模擬即得), 地位如同 NP 中的 SAT。
- **算術階層一瞥**:C.E. $= \Sigma_1$(「$\exists$ 步數 t: 可判定條件」)、co-C.E. $= \Pi_1$(「$\forall t$:……」)、可判定 $= \Delta_1 = \Sigma_1 \cap \Pi_1$。往上還有 $\Sigma_2, \Pi_2, \ldots$(如 $EQ_{\text{TM}} \in \Pi_2$)——不可判定的世界內部還有無窮多層。

7 本週重點整理

主題	重點
C.E.	健全 + 完備、不要求停機;「Yes 端有有限證據」; 等價於可枚舉成員清單
HALT_{TM} 是 C.E.	halt$^-$: 模擬 $M(w)$, 一停 (接受或拒絕) 就接受
交錯模擬	天真逐字模擬會卡在不停機的字; 改為「第 k 輪: 前 k 個字各跑 k 步」——$\overline{E}_{\text{TM}}$ 是 C.E.
核心定理	L 與 $\overline{L}$ 皆 C.E. $\Rightarrow L$ 可判定 (平行執行, 先接受者定案)
非 C.E. 問題	$\overline{\text{HALT}_{\text{TM}}}$、$E_{\text{TM}}$ 不是 C.E.(EQ_{TM} 連 co-C.E. 都不是)
映射歸約	$A \le_m B$: 可計算 f,$w \in A \iff f(w) \in B$;B 可判定 $\Rightarrow A$ 可判定; 遞移; 補集須兩邊同取
$\text{HALT}_{\text{TM}} \le_m A_{\text{TM}}$	M' =「把 M 的拒絕改成接受」;M' 接受 $w \iff M$ 停機
$A_{\text{TM}} \le_m \overline{E}_{\text{TM}}$	M_w =「無視輸入, 寫死 w, 模擬 $M(w)$」;Language$(M_w) \in \{\Sigma^*, \emptyset\}$
階層	正規 $\subset$ 可判定 =C.E.$\cap$co-C.E.; 兩瓣之外還有 $\Sigma_2, \Pi_2, \ldots$

8 練習題 (附解答)

練習 1. 判斷下列語言是否 C.E.、co-C.E.、可判定 (可複選): (a) A_{TM};(b) $\overline{\text{HALT}_{\text{TM}}}$;(c) $\{\text{code}(M) : M$ 在空輸入上於10^6 步內停機$\}$。

解答

(a) C.E.(通用機模擬, 接受就接受) 但不可判定, 故不是 co-C.E.。(b) co-C.E.(補集 HALT_{TM} 是 C.E.) 但不可判定, 故不是 C.E.。(c) **可判定** (模擬 10^6 步即可), 因此同時是 C.E. 與 co-C.E.。

練習 2. 在 $\overline{E}_{\text{TM}}$ 的 C.E. 證明中, 為什麼天真版演算法「對每個 i 依序模擬 $M(w_i)$ 直到接受」不完備? 交錯模擬如何修復?

解答

天真版在 $i = 0$ 的模擬若不停機, 便永遠輪不到後面的字; 若 M 其實接受 w_5, 演算法卻永遠不回傳 True——完備性失敗。交錯模擬把每輪的模擬限定 k 步, 第 k 輪覆蓋 $w_0, \ldots, w_k$; 若 M 以 s 步接受 w_i, 則第 $\max(i, s)$ 輪必發現。任何單一模擬都無法再壟斷時間。

練習 3. 證明: 若 $A \le_m B$ 且 $B \le_m C$, 則 $A \le_m C$。

解答

設 f 實現 $A \le_m B$、g 實現 $B \le_m C$。令 $h = g \circ f$。兩台計算 f, g 的必停機機器串接, 仍必停機, 故 h 可計算; 且 $w \in A \iff f(w) \in B \iff g(f(w)) \in C \iff h(w) \in C$。故 h 實現 $A \le_m C$。

練習 4. 為什麼「$A \le_m B \Rightarrow \overline{A} \le_m \overline{B}$」成立, 但不能由 $A \le_m B$ 推出 $A \le_m \overline{B}$? 用 HALT_{TM} 舉例。

解答

同一個 f 滿足 $w \in A \iff f(w) \in B$, 取逆否即 $w \in \overline{A} \iff f(w) \in \overline{B}$——兩邊同取補, 免費成立。但單邊取補需要「翻轉答案」, 映射歸約不允許。反例:$\text{HALT}_{\text{TM}} \le_m \text{HALT}_{\text{TM}}$(恆等函數) 成立; 若 $\text{HALT}_{\text{TM}} \le_m \overline{\text{HALT}}_{\text{TM}}$ 也成立, 則由「$\overline{\text{HALT}}_{\text{TM}}$ 是 co-C.E.、映射歸約保持 C.E.」……實際上會推出 HALT_{TM} 是 co-C.E., 結合其 C.E. 性得 HALT_{TM} 可判定——矛盾。

練習 5 (思考題)**.** 用「寫死輸入」技巧與 Rice 定理的精神, 證明 $\text{ALL}_{\text{TM}} = \{\text{code}(M) : \text{Language}(M) = \Sigma^*\}$ 不可判定。

解答

沿用第 5 節的 M_w:$\text{Language}(M_w) = \Sigma^*$ 若 $M(w) = 1$, 否則 $= \emptyset$。故 $f(\langle \text{code}(M), w\rangle) = \text{code}(M_w)$ 實現 $A_{\text{TM}} \le_m \text{ALL}_{\text{TM}}(M(w) = 1 \iff \text{Language}(M_w) = \Sigma^*)$。$A_{\text{TM}}$ 不可判定, 故 ALL_{TM} 不可判定。(它其實連 C.E. 都不是——事實上 $\text{ALL}_{\text{TM}} \in \Pi_2$。)

參考資料

1. C. Hampson, *5CCS2FC2 Foundations of Computing II, Week 11* 投影片 (ce / mapping),King's College London.
2. M. Sipser, *Introduction to the Theory of Computation*, 3rd ed., Cengage, 2013.(第 4、5 章: 可辨識性、映射歸約)
3. E. Post, "Recursively enumerable sets of positive integers and their decision problems," *Bull. AMS*, 50:284–316, 1944.
4. H. Rogers, *Theory of Recursive Functions and Effective Computability*, MIT Press, 1987.
5. S. Arora, B. Barak, *Computational Complexity: A Modern Approach*, Cambridge Univ. Press, 2009.(第 1 章)
6. Wikipedia: *Computably enumerable set*; *Recursively enumerable language*; *Many-one reduction*; *Dovetailing (computer science)*; *Arithmetical hierarchy.*