計算的極限: 不可判定性

完整教材 (依據 5CCS2FC2 Foundations of Computing II 第十週內容編寫)

通用圖靈機、對角線論證、停機問題與 Entscheidungsproblem

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**可判定語言**的三要件: 健全 (sound)、完備 (complete)、必停機 (terminating)。
2. 掌握**字串編碼**與**通用圖靈機 (Universal Turing Machine)** 的概念。
3. 用**對角線論證**證明: 不可判定的語言必然存在。
4. 完整掌握**停機問題** HALT_{TM} 不可判定的證明 (自我指涉與矛盾)。
5. 認識其他不可判定問題:A_{TM}、R_{TM}、E_{TM}、EQ_{TM}、$\text{REGULAR}_{\text{TM}}$, 與**歸約**技巧、Rice 定理。
6. 理解希爾伯特的 **Entscheidungsproblem** 為何不可判定, 及其歷史意義 (Turing 1936、Church)。

Contents

1 可判定與不可判定語言

1.1 Church–Turing 論題 (回顧)

Church–Turing 論題

任何能以某種**有限程序**有效計算的語言, 都能被圖靈機辨識。

這是「演算法」這個非形式概念與「圖靈機」這個數學模型之間的橋樑: 凡是直覺上「可以機械化計算」的東西, 圖靈機都做得到。因此, 圖靈機做不到的事, 任何演算法都做不到——本週要探索的正是這條界線。

1.2 記號與可判定性

定義 1.1 (機器的三種行為)**.** 對圖靈機 M 與輸入 w, 記

$$M(w) := \begin{cases} 1 & \text{若} M \text{ 接受} w \\ 0 & \text{若} M \text{ 拒絕} w \\ \uparrow & \text{若} M \text{ 在} w \text{ 上不停機 (卡住/無窮迴圈)} \end{cases}$$

定義 1.2 (可判定語言)**.** 語言 L 是**可判定的 (decidable)**, 若存在圖靈機 M 同時滿足:

- **健全 (Sound)**: 若 $M(w) = 1$, 則 $w \in L$(說「是」不會錯);
- **完備 (Complete)**: 若 $w \in L$, 則 $M(w) = 1$(該說「是」時必說「是」);
- **必停機 (Terminating)**: 對所有 $w \in \Sigma^*$,$T_M(w) < \infty$(永不卡住)。

核心問題: **是否每個語言都可判定?** 本週的答案是否定的——而且反例並非人造珍品, 而是「我的程式會不會當機」這種再實際不過的問題。

2 通用圖靈機 (Universal Turing Machines)

2.1 把一切編碼成字串

- **元組 (tuple)**: 用擴充字母表 $\Sigma \cup \{\#\}$ 編碼字串元組:

$$\langle w_0, w_1, \ldots, w_n \rangle := \# \, w_0 \, \# \, w_1 \, \# \cdots \# \, w_n \, \#.$$

(也可以再把每個字元轉成二進位碼, 如 $0 \mapsto 00$、$1 \mapsto 01$、$\# \mapsto 11$, 把一切壓回 $\{0,1\}^*$。)

- **有限函數**:$f : A \to B$(A, B 有限) 可編碼為「元組的元組」:

$$\langle f \rangle := \big\langle \langle a_1, f(a_1) \rangle, \langle a_2, f(a_2) \rangle, \ldots, \langle a_n, f(a_n) \rangle \big\rangle$$

(實際上就是一張 $2 \times n$ 的查表)。

- **圖靈機本身**: 轉移函數 δ 是有限函數, 故整台機器可編碼為字串:

$$\mathrm{code}(M) := \big\langle q_{\mathrm{init}}, \, q_{\mathrm{accept}}, \, q_{\mathrm{reject}}, \, \langle \delta \rangle \big\rangle.$$

備註 2.1. 「程式即資料」——這是整週 (也是整個計算理論) 最重要的觀念: 一台圖靈機可以被寫成字串, 因此可以餵給另一台圖靈機當輸入。現代電腦的「儲存程式」架構 (von Neumann) 正是這個想法的工程實現。

2.2 通用圖靈機

定義 2.2 (通用圖靈機)**. 通用圖靈機** M_u 接受一對輸入 $\langle \mathrm{code}(M), w\rangle$, 並滿足

$$M_u(\langle \mathrm{code}(M), w\rangle) = M(w),$$

亦即:M_u 接受 $\iff$ M 接受 w;M_u 拒絕 $\iff$ M 拒絕 w;M_u 不停機 $\iff$ M 在 w 上不停機。

備註 2.3. 通用圖靈機就像**編譯器/直譯器**: 讀入「軟體」$\mathrm{code}(M)$, 在輸入 w 上模擬 M 的運作。M 接受它就接受、M 拒絕它就拒絕、M 卡住它也跟著卡住。一台機器可以執行所有機器——這是「可程式化電腦」的理論原型。

3 不可判定語言的存在性: 對角線論證

定理 3.1. 存在不可判定的語言 L。

Proof. **步驟 1(枚舉所有字串).** 選定字母表後, 所有字串可依長度、再依字典序枚舉:

$$\varepsilon,\ 0,\ 1,\ 00,\ 01,\ 10,\ 11,\ 000,\ 001,\ldots \quad \rightsquigarrow \quad w_0, w_1, w_2, w_3, \ldots$$

步驟 2(枚舉所有圖靈機). 每台圖靈機都有字串編碼 $\mathrm{code}(M)$, 而字串可枚舉, 故圖靈機也可枚舉 (機器的數量不會多於編碼的數量):

$$M_0, M_1, M_2, M_3, \ldots$$

步驟 3(對角線表). 把「M_i 是否接受 w_j」排成一張無窮表格 (✓= 接受、✗= 不接受), 並注視**對角線**:

	w_0	w_1	w_2	w_3	w_4	w_5	$\cdots$
M_0	✓	✗	✓	✓	✗	✓	$\cdots$
M_1	✗	✗	✓	✗	✗	✓	$\cdots$
M_2	✓	✗	✗	✓	✓	✗	$\cdots$
M_3	✗	✓	✓	✓	✗	✓	$\cdots$
M_4	✗	✓	✗	✓	✗	✗	$\cdots$
M_5	✓	✗	✗	✓	✗	✗	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

定義 L 為「把對角線**翻轉**」得到的語言:

$$L = \{w_i \in \Sigma^* : M_i \text{ 不接受 } w_i\}, \qquad \text{即} \quad w_i \in L \iff M_i \text{ 不接受 } w_i.$$

以上表為例:$L = \{w_1, w_2, w_4, w_5, \ldots\}$(取對角線上為✗的字)。

步驟 4(矛盾). 若 L 可判定, 則 L 是清單中某台必停機機器 M_j 的語言:$L = \text{Language}(M_j)$。但看第 j 個對角線格子:

$$w_j \in \text{Language}(M_j) \iff M_j \text{ 接受 } w_j \iff w_j \notin L.$$

故 $\text{Language}(M_j)$ 與 L 在 w_j 上必不一致,$\text{Language}(M_j) \neq L$——矛盾! 所以 L 不可判定。 □

備註 3.2 (數量論證). 更粗略但更快的看法: 圖靈機只有**可數**多台 (每台是一個有限字串), 但語言有**不可數**多個 ($\mathcal{P}(\Sigma^*)$ 與實數等勢,Cantor)。機器根本不夠用——幾乎所有語言都不可判定! 對角線論證的價值在於下一節: 它能指出一個具體、自然的不可判定問題。

4 停機問題 (The Halting Problem)

定義 4.1 (停機問題)**.**

$$\textsc{Halt}_{\text{TM}} = \{\langle \text{code}(M), w\rangle : M \text{ 在輸入 } w \text{ 上會停機}\}.$$

輸入 機器的編碼 $\text{code}(M)$ 與輸入字 $w \in \Sigma^*$。

輸出 **True** 若且唯若 M 在 w 上終止 (接受或拒絕皆可)。

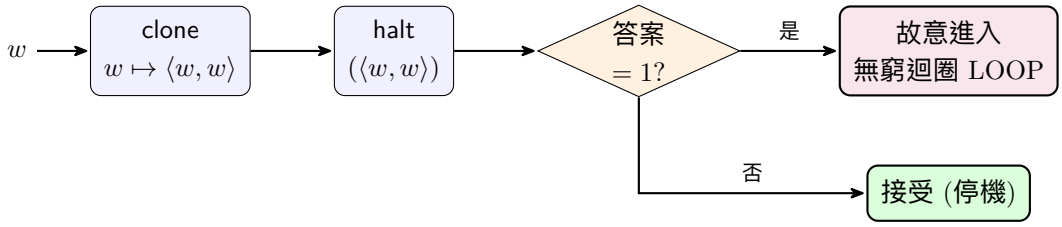
定理 4.2. 停機問題 $\textsc{Halt}_{\text{TM}}$ 不可判定。

Proof. **步驟 1(反證假設).** 假設存在健全、完備且必停機的演算法 halt, 使得

$$\text{halt}(\langle \text{code}(M), w\rangle) = \begin{cases} 1 & \text{若} M(w) = 1 \text{ 或 } M(w) = 0 \quad (\text{會停}) \\ 0 & \text{若} M(w) = \uparrow \quad (\text{不停}) \end{cases}$$

步驟 2(複製器). 先造一個顯然可計算的函數 clone: 輸入單一字串 w, 輸出一對 $\langle w, w\rangle$(把輸入複製兩份)。

步驟 3(惡魔機器 $M^\star$). 把 clone、halt 與一個「唱反調」的開關串起來:



用一句話描述 $M^\star$:「問 *halt:* 我 *(*輸入 w*)* 套在 w 自己身上會不會停*?* 如果說會停*,* 我就故意不停*;* 如果說不停*,* 我就立刻停。」形式上:

$$M^\star(w) = 1 \iff \text{halt}(\langle w, w\rangle) = 0, \qquad M^\star(w) = \uparrow \iff \text{halt}(\langle w, w\rangle) = 1.$$

步驟 4(自我指涉). $M^\star$ 也是一台機器, 有自己的編碼 $\text{code}(M^\star)$。把它餵給自己: 取 $w = \text{code}(M^\star)$。則

$$M^\star(\text{code}(M^\star)) = 1 \iff \text{halt}(\langle \text{code}(M^\star), \text{code}(M^\star)\rangle) = 0 \iff M^\star(\text{code}(M^\star)) = \uparrow.$$

步驟 5(矛盾). 「$M^\star$ 套在自己身上停機 $\iff$ 它不停機」——荒謬! 唯一的出路是: 一開始假設的 halt 根本不存在。 □

備註 4.3 (與對角線論證的關係). 步驟 4 正是對角線論證的化身:$M^\star$ 沿著「機器 $\times$ 自身編碼」這條對角線, 逐格唱反調。理髮師悖論 (「只幫不自己刮鬍子的人刮鬍子的理髮師, 要不要幫自己刮?」)、羅素悖論, 與此同構。

實務意義

停機問題不可判定, 意味著**不存在完美的通用除錯器**: 沒有任何工具能對「任意程式 + 任意輸入」百分之百正確地回答「它會不會當機/無窮迴圈」。靜態分析器、模型檢查器都只能做保守近似 (可能回答「不知道」) 或針對受限的程式類別。

5 其他不可判定問題與歸約

5.1 一批不可判定的語言

問題	定義
接受問題 A_{TM}	$\{\langle \mathrm{code}(M), w\rangle : M(w) = 1\}$——$M$ 是否接受 w?
拒絕問題 R_{TM}	$\{\langle \mathrm{code}(M), w\rangle : M(w) = 0\}$——$M$ 是否拒絕 w?
空語言問題 E_{TM}	$\{\mathrm{code}(M) : \mathrm{Language}(M) = \emptyset\}$——$M$ 是否什麼都不接受?
等價問題 EQ_{TM}	$\{\langle \mathrm{code}(M_1), \mathrm{code}(M_2)\rangle : \mathrm{Language}(M_1) = \mathrm{Language}(M_2)\}$——兩台機器語言是否相同?
正規性問題 $\textsc{Regular}_{\mathrm{TM}}$	$\{\mathrm{code}(M) : \mathrm{Language}(M)$ 是正規語言$\}$——M 的語言是否正規?

以上**全部不可判定**。

5.2 歸約: 證明不可判定的標準武器

歸約的邏輯

若能把已知不可判定的問題 A **歸約**到問題 B(即: 有了解 B 的演算法, 就能造出解 A 的演算法), 則 B 也不可判定。記作 $A \le_m B$。

方向千萬別搞反:「A 難 $\Rightarrow$ B 難」需要的是A 歸約到 B。

例 5.1 ($\textsc{Halt}_{\mathrm{TM}} \le_m A_{\mathrm{TM}}$: 接受問題不可判定)**.** 假設有判定 A_{TM} 的演算法 accept。給定停機問題實例 $\langle \mathrm{code}(M), w\rangle$, 造 M': 先模擬 $M(w)$, 一旦 M 停機 (不論接受或拒絕)M' 就接受。則

$$M \text{ 在 } w \text{ 上停機} \iff M'(w) = 1 \iff \mathsf{accept}(\langle \mathrm{code}(M'), w\rangle) = 1,$$

即可判定停機問題——矛盾。故 A_{TM} 不可判定。(同法可證 R_{TM}。)

例 5.2 ($A_{\mathrm{TM}} \leq_m E_{\mathrm{TM}}$: 空語言問題不可判定)**.** 假設有判定 E_{TM} 的演算法 empty。給定 $\langle \mathrm{code}(M), w\rangle$, 造機器 M_w:「輸入 x: 若 $x \neq w$ 直接拒絕; 若 $x = w$, 模擬 $M(w)$ 並在其接受時接受。」則

$$\mathrm{Language}(M_w) = \begin{cases} \{w\} & \text{若} M(w) = 1 \\ \emptyset & \text{否則} \end{cases}$$

故 $M(w) = 1 \iff \mathsf{empty}(\mathrm{code}(M_w)) = 0$, 即可判定 A_{TM}——矛盾。

例 5.3 ($E_{\mathrm{TM}} \leq_m EQ_{\mathrm{TM}}$: 等價問題不可判定)**.** 取一台固定的機器 $M_\emptyset$(對所有輸入都拒絕,$\mathrm{Language}(M_\emptyset) = \emptyset$)。則

$$\mathrm{code}(M) \in E_{\mathrm{TM}} \iff \langle \mathrm{code}(M), \mathrm{code}(M_\emptyset)\rangle \in EQ_{\mathrm{TM}},$$

故判定 EQ_{TM} 就能判定 E_{TM}——矛盾。

5.3 Rice 定理: 一網打盡

定理 5.4 (Rice, 1953)**.** 圖靈機語言的任何**非平凡語意性質**都不可判定。「性質 P 非平凡」意指*:* 有些圖靈機的語言滿足 P、有些不滿足*;*「語意」意指 P 只取決於 $\mathrm{Language}(M)$*,* 與 M 的內部構造無關。

備註 5.5. E_{TM}(語言是否為空)、$\textsc{Regular}_{\mathrm{TM}}$(語言是否正規)、「語言是否包含 ε」、「語言是否有限」……全是非平凡語意性質, 由 Rice 定理一次全部不可判定。注意:「M 是否有超過 100 個狀態」是語法性質 (可判定);「M 是否在 100 步內接受 w」也可判定 (直接模擬 100 步)。不可判定性來自「對所有輸入/無限時間的行為」提問。

6 Entscheidungsproblem(判定問題)

6.1 希爾伯特之夢

1928 年, 希爾伯特 (D. Hilbert) 與阿克曼 (W. Ackermann) 提出 **Entscheidungsproblem**(德文「判定問題」):

定義 6.1 (Entscheidungsproblem)**. 輸入** 一條述詞邏輯 (一階邏輯) 公式 F。

輸出 True 若且唯若 F 是恆真式 (tautology)。

若這個問題有演算法, 數學將被「機械化」: 任何猜想只要寫成一階公式, 交給機器即可判定真偽。1936 年,Church(用 λ 演算) 與 Turing(用圖靈機) 分別獨立粉碎了這個夢。

定理 6.2. *Entscheidungsproblem* 不可判定。

Proof. **步驟 1(回顧 Cook–Levin)**. 在 Cook–Levin 定理的證明中, 對給定的機器 M 與輸入 w, 我們造過一條命題邏輯公式 $F_{M,w}$, 使得

$$F_{M,w} \text{ 可滿足} \iff M \text{ 在多項式步數內接受 } w.$$

步驟 2(命題邏輯的極限). 可惜,「M 終究會接受」這類**全域**性質, 用命題邏輯需要無窮長的「析取」——寫不出來:

$$S_{q,0} \vee S_{q,1} \vee S_{q,2} \vee S_{q,3} \vee S_{q,4} \vee \cdots \qquad (q = q_{\mathrm{accept}}).$$

(命題變數 $S_{q,t}$=「時刻 t 機器處於狀態 q」, 但 t 沒有上界!)

步驟 3(述詞邏輯的威力). 換用適當的**述詞符號**, 把「時間」變成量詞可及的變數:

$$\begin{aligned} C_a(i,t) &= \text{「時刻}t\text{時, 帶子第}i\text{格的符號是}a\text{」} \\ H(i,t) &= \text{「時刻}t\text{時, 讀寫頭在位置}i\text{」} \\ S_q(t) &= \text{「時刻}t\text{時, 機器處於狀態}q\text{」} \end{aligned}$$

那條無窮析取就濃縮成一條有限公式:

$$S_q(0) \vee S_q(1) \vee S_q(2) \vee S_q(3) \vee \cdots \quad \rightsquigarrow \quad \exists t\, S_q(t).$$

步驟 4(歸約完成). 依 Cook–Levin 的方式把 M 的整個轉移規則寫成述詞公式 (初始組態、每一步怎麼動、接受條件 $\exists t\, S_{q_{\text{accept}}}(t)$), 得到一階公式 $F_{M,w}$ 使得

$$F_{M,w} \text{ 可滿足} \iff M \text{ 終究會接受 } w.$$

這是從接受問題 A_{TM} 到「Entscheidungsproblem 之補」的歸約 (可滿足性與恆真性互為對偶:F 可滿足 $\iff \neg F$ 不是恆真式)。A_{TM} 不可判定, 故 Entscheidungsproblem 也不可判定。 □

歷史注記:Turing 1936

Turing 的論文《On Computable Numbers, with an Application to the Entscheidungsproblem》(1936 年 5 月 28 日投稿) 一口氣完成了三件事:

1. **定義了圖靈機**——第一個令人信服的「演算法」數學模型, 並提出**通用機**的構造;
2. 用**對角線論證**證明存在不可計算的實數與不可判定的問題 (他的「circle-free」問題, 即現代停機問題的前身;「halting problem」一詞是 1950 年代才出現的);
3. 把不可判定性**歸約**到一階邏輯, 證明 Hilbert 的 Entscheidungsproblem 無解。

Church 稍早用 λ 演算獨立得到同樣結論; 兩個模型後來被證明等價——這正是 Church–Turing 論題的由來。可以說, 這篇論文同時發明了「電腦科學」與證明了它的極限。

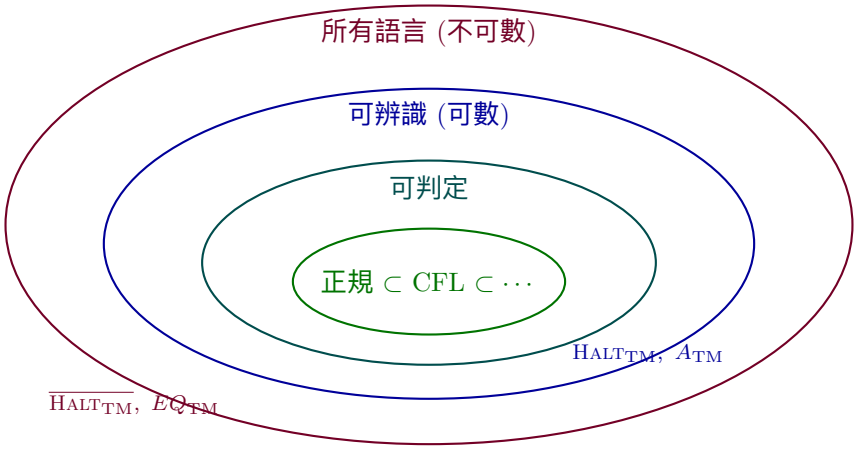
7 補充: 可辨識 vs. 可判定

定義 7.1 (可辨識語言)**.** 語言 L 是**圖靈可辨識的 (Turing-recognizable / semi-decidable)**, 若存在健全且完備的機器 M($w \in L \iff M(w) = 1$), 但 M 在 $w \notin L$ 時允許不停機。

定理 7.2. L 可判定 $\iff$ L 與其補集 $\overline{L}$ 都可辨識。

證明概要. ($\Rightarrow$) 顯然。($\Leftarrow$) 設 M_1 辨識 L、M_2 辨識 $\overline{L}$。**平行執行**兩者 (輪流各走一步): 任何 w 必屬於 L 或 $\overline{L}$ 之一, 故必有一台終將接受——先接受者決定答案。這台新機器必停機且正確。 □

例 7.3. HALT_{TM} 與 A_{TM} 可辨識 (用通用機模擬, 若停/接受就接受) 但不可判定; 因此由定理 7.2, 其補集 $\overline{\text{HALT}_{\text{TM}}}$、$\overline{A_{\text{TM}}}$ 連可辨識都不是。EQ_{TM} 更極端: 它與補集都不可辨識。



8 本週重點整理

主題	重點
可判定	健全 + 完備 + 必停機;$M(w) \in \{1, 0, \uparrow\}$
編碼與通用機	$\text{code}(M)$ 把機器變成字串;$M_u(\langle \text{code}(M), w\rangle) = M(w)$(直譯器)
對角線論證	枚舉 w_i、M_i;$L = \{w_i : M_i$ 不接受$w_i\}$; 若 $L = \text{Language}(M_j)$ 則在 w_j 矛盾
停機問題	HALT_{TM} 不可判定:halt→clone→ 唱反調機 $M^\star$;$M^\star(\text{code}(M^\star))$ 停 $\iff$ 不停
歸約	$A \le_m B$ 且 A 不可判定 $\Rightarrow$ B 不可判定;$\text{HALT}_{\text{TM}} \le_m A_{\text{TM}} \le_m E_{\text{TM}} \le_m EQ_{\text{TM}}$
Rice 定理	語言的任何非平凡語意性質都不可判定
Entscheidungsproblem	一階恆真性不可判定: 述詞 $C_a(i,t), H(i,t), S_q(t)$+$\exists t\, S_q(t)$ 把 A_{TM} 寫進邏輯
可辨識	可判定 $\iff$ L 與 $\overline{L}$ 皆可辨識;HALT_{TM} 可辨識不可判定;$\overline{\text{HALT}_{\text{TM}}}$ 不可辨識

9 練習題 (附解答)

練習 1. 判斷下列問題是否可判定, 並簡述理由: (a) 「M 在輸入 w 上於 1000 步內停機?」(b) 「M 的狀態數是否為偶數?」(c) 「$\text{Language}(M)$ 是否包含 ε?」

解答

(a) **可判定**: 用通用機模擬 1000 步即可, 必停機。(b) **可判定**: 讀編碼數狀態即可——這是語法性質,Rice 定理不適用。(c) **不可判定**:「$\varepsilon \in \text{Language}(M)$」是非平凡語意性質 (有的語言含 ε、有的不含), 由 Rice 定理不可判定。

練習 2. 在停機問題的證明中, 為什麼需要 clone 這一步? 直接把 w 餵給 halt 不行嗎?

解答

halt 的輸入格式是一對 $\langle \mathrm{code}(M), w\rangle$。$M^\star$ 只收到單一字串 w, 而我們想問的是「w 所編碼的機器跑在 w 自己身上」——即 $\langle w, w\rangle$。clone 把單輸入複製成對, 才能形成「機器套用到自身編碼」的對角線; 沒有它就無法自我指涉。

練習 3. 設計歸約證明:「M 是否接受某個以 0 開頭的字串?」不可判定。

解答

由 A_{TM} 歸約。給定 $\langle \mathrm{code}(M), w\rangle$, 造 M':「輸入 x: 若 $x = 0$, 模擬 $M(w)$, 接受若其接受; 否則拒絕。」則 $M(w) = 1 \iff \mathrm{Language}(M') = \{0\} \ni$ 某個以 0 開頭的字 $\iff M'$ 接受某個以 0 開頭的字串。若後者可判定, 則 A_{TM} 可判定——矛盾。(或直接引用 Rice 定理: 這是非平凡語意性質。)

練習 4. 為什麼「命題邏輯的可滿足性 (SAT)」可判定 (甚至 NP-complete), 而「一階邏輯的恆真性」不可判定? 差別出在哪裡?

解答

命題公式只有有限個變數, 真值指派共 2^n 種——暴力枚舉即可 (可判定; 快不快是另一回事)。一階公式的量詞 $\exists t$ 跑遍無窮論域, 無法枚舉所有結構; 正是 $\exists t\, S_q(t)$ 這種「無界搜尋」讓圖靈機的「終究會接受」可以被表達, 把 A_{TM} 嵌進了邏輯。表達力越強, 判定越難——這是邏輯裡反覆出現的權衡。

練習 5 (思考題)**.** 朋友宣稱寫出了一個工具, 能檢查任何 Python 程式是否含無窮迴圈。用本週的定理反駁他, 並說明現實中的靜態分析工具為何仍有價值。

解答

Python 圖靈完備, 故「任意程式 + 任意輸入是否停機」就是 $\textsc{Halt}_{\mathrm{TM}}$——不可判定 (定理 4.2); 對「所有輸入都停機」的版本更難。所以完美工具不存在。但現實工具仍有價值: 它們可以 (1) 對受限的程式類別 (如無遞迴、有界迴圈) 給出完整答案;(2) 做保守近似——回答「安全/可能有問題/不知道」三值;(3) 抓出常見模式的錯誤。不可判定性排除的是「對所有程式都對」的萬靈丹, 不是「對很多程式都有用」的工程工具。

參考資料

1. C. Hampson, *5CCS2FC2 Foundations of Computing II, Week 10* 投影片 (undecidable / halting / entscheidungsproblem),King's College London.
2. A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proc. London Math. Soc.*, s2-42:230–265, 1936.
3. A. Church, "A Note on the Entscheidungsproblem," *J. Symbolic Logic*, 1:40–41, 1936.
4. H. G. Rice, "Classes of recursively enumerable sets and their decision problems," *Trans. AMS*, 74:358–366, 1953.
5. M. Sipser, *Introduction to the Theory of Computation*, 3rd ed., Cengage, 2013.(第 4、5 章)

6. Stanford Encyclopedia of Philosophy: *The Church-Turing Thesis* — *The Rise and Fall of the Entscheidungsproblem*.

7. Wikipedia: *Halting problem*; *Entscheidungsproblem*; *Rice's theorem*; *Universal Turing machine*; *Turing's proof*.