計算理論基礎：有限自動機、正則語言與圖靈機

完整教材（依據 5CCS2FC2 Foundations of Computing II 第一週內容編寫）

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 6 月

本週學習目標

1. 複習**確定性有限自動機（DFA）**與**非確定性有限自動機（NFA）**的形式定義、組態（configuration）與計算（computation）。
2. 理解**正則表達式**與**正則語言**，以及它們與有限自動機的等價性（Kleene 定理）。
3. 透過鴿籠原理證明 $L = \{a^n b^n\}$ **不是正則語言**，並掌握一般化工具——**泵引理（Pumping Lemma）**。
4. 認識**圖靈機（Turing Machine, TM）**的形式定義、接受準則，以及「機器可能不停機」帶來的根本差異。
5. 認識**非確定性圖靈機（NDTM）**及其與確定性圖靈機的等價性。
6. 理解 **Church–Turing 論題**的內容、地位與支持證據。

Contents

1 形式語言的基本概念

計算理論研究的核心問題是：「什麼問題可以被機器解決？需要多強的機器？」為了精確回答，我們先把「問題」形式化為**語言（language）**，把「機器」形式化為各種**自動機（automata）**。

定義 1.1 (字母表、字串、語言)**.**

- **字母表（alphabet）**：任意一個有限的符號集合，記作 $\Sigma = \{a_0, a_1, \ldots, a_k\}$。
- **字串（word / string）**：由 Σ 中符號組成的有限序列。所有有限字串的集合記作

$$\Sigma^* = \{\, \text{所有由} \Sigma \text{ 中符號構成的有限字串} \,\}.$$

- **空字串（empty string）**：長度為零的字串，記作 $\varepsilon \in \Sigma^*$。
- **語言（language）**：Σ^* 的任意子集 $L \subseteq \Sigma^*$（可以是有限或無限集），都稱為一個語言。特別地，**空語言** $\emptyset \subseteq \Sigma^*$ 也是語言。

例 1.2 (字母表的例子)**.** 字母表可以是任何有限符號集，例如：

- 二元字母表 $\Sigma = \{0, 1\}$；
- 英文字母 $\Sigma = \{a, b, c, \ldots, x, y, z\}$；
- 命題邏輯符號 $\Sigma = \{p, q, r, \wedge, \vee, \to, \neg, (,)\}$；
- 甚至表情符號、撲克牌花色等任何有限集合。

例 1.3 (語言的例子)**.**

- 所有英文單字構成的集合（有限語言）。
- 所有代表質數的二進位字串（無限語言）。
- $L = \{a^n b^n : n = 0, 1, 2, \ldots\} = \{\varepsilon,\ ab,\ aabb,\ aaabbb, \ldots\}$ ——本週的主角。

觀念：判定問題 = 語言

「判定一個輸入是否具有某性質」這類**判定問題（decision problem）**，可以等同於「判定字串 w 是否屬於語言 L」。例如「n 是否為 3 的倍數？」對應到語言 $L = \{\, w \in \{0,1\}^* : w \text{ 是3 的倍數的二進位表示} \,\}$。因此研究「機器能解決哪些問題」就是研究「機器能辨識哪些語言」。

2 確定性有限自動機（DFA）

2.1 形式定義

定義 2.1 (DFA)**.** 一個**確定性有限自動機（Deterministic Finite Automaton, DFA）**是一個五元組

$$\mathcal{A} = \langle\, \Sigma,\ Q,\ q_{\text{init}},\ F,\ \delta \,\rangle,$$

其中：

- $\Sigma = \{a_1, \ldots, a_k\}$：輸入**字母表**；
- $Q = \{q_1, \ldots, q_n\}$：有限的**控制狀態**集合；
- $q_{\text{init}} \in Q$：**初始狀態**；

- $F \subseteq Q$：**接受（最終）狀態**集合；
- $\delta : Q \times \Sigma \to Q$：**轉移函數**，

$$\delta(\text{目前狀態},\ \text{讀到的符號}) \ \mapsto\ \text{新狀態}.$$

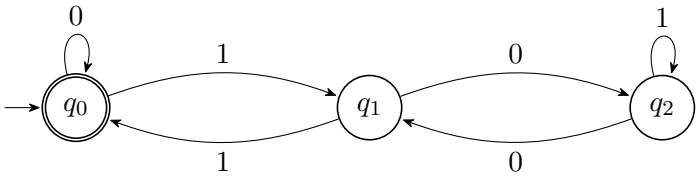
直觀上，DFA 是一台只有「有限記憶」的機器：它從左到右一次讀一個輸入符號，每讀一個符號就依 δ 換到下一個狀態；讀完整個字串後，看它停在哪個狀態來決定接受或拒絕。「確定性」的意思是：任一時刻、任一輸入符號，下一步都唯一確定。

2.2 範例：判定二進位數是否為 3 的倍數

考慮 $\Sigma = \{0, 1\}$，$Q = \{q_0, q_1, q_2\}$，$q_{\text{init}} = q_0$，$F = \{q_0\}$，轉移函數如下表：

目前狀態	讀入符號	新狀態
q_0	0	q_0
q_0	1	q_1
q_1	0	q_2
q_1	1	q_0
q_2	0	q_1
q_2	1	q_2

對應的**狀態圖**如下（雙圈表示接受狀態）：



這台 DFA 的設計思想：狀態 q_i 記住「目前為止讀到的二進位數值除以 3 的餘數是 i」。讀入一個新位元 b 時，數值由 v 變為 $2v + b$，故餘數由 i 變為 $(2i + b) \bmod 3$ ——正是上表的轉移。

2.3 組態、計算與接受準則

定義 2.2 (組態與計算)**.**

- **組態（configuration）**：機器在某一時刻的完整描述。對 DFA 而言是

$$(\ \text{目前狀態},\ \text{讀寫頭右側尚未讀取的子字串}\ ).$$

- **計算（computation）**：執行 DFA 所得到的組態序列。

例 2.3 (一次完整的計算)**.** 上述 DFA 讀入 $w = \texttt{1001}$（即十進位 9）的計算為：

$$(q_0,\ \texttt{1001}) \to (q_1,\ \texttt{001}) \to (q_2,\ \texttt{01}) \to (q_1,\ \texttt{1}) \to (q_0,\ \varepsilon).$$

最後停在 $q_0 \in F$，所以 1001 被**接受**（9 確實是 3 的倍數）。

定義 2.4 (DFA 的接受準則)**.**

$$\mathcal{A} \text{ 接受} w \iff \text{從} (q_{\text{init}}, w) \text{ 出發的唯一計算，其最終狀態屬於} F.$$

DFA 所**辨識的語言**為

$$\text{Language}(\mathcal{A}) = \{\, w \in \Sigma^* : \mathcal{A} \text{ 接受} w \,\}.$$

例 2.5 (把 DFA 當作判定程序)**.** 把每個輸入丟給上述 DFA，它就是一個永遠會停、且回答正確的「判定程序」：

$1111 = 15$ ✓	$10001 = 17$ ✗	$1001 = 9$ ✓	$111 = 7$ ✗
$11100 = 28$ ✗	$110 = 6$ ✓	$11 = 3$ ✓	$11111 = 31$ ✗
$101 = 5$ ✗	$0 = 0$ ✓	$10010 = 18$ ✓	

3 非確定性有限自動機（NFA）

3.1 形式定義

定義 3.1 (NFA)**.** 一個**非確定性有限自動機（Non-deterministic Finite Automaton, NFA）**也是五元組 $\mathcal{A} = \langle \Sigma, Q, q_{\text{init}}, F, \delta \rangle$，其中 $\Sigma, Q, q_{\text{init}}, F$ 與 DFA 完全相同，唯一的差別在轉移函數：

$$\delta : Q \times \Sigma \to \mathcal{P}(Q), \qquad \delta(\text{目前狀態},\ \text{讀到的符號}) \mapsto \{\, \text{所有可能的新狀態} \,\},$$

其中 $\mathcal{P}(Q)$ 表示 Q 的**冪集（powerset）**。更一般地，可允許 ε**-轉移**：

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q),$$

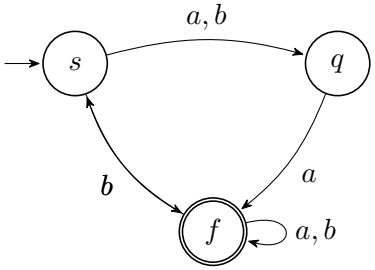
即機器可以在不讀任何符號的情況下換狀態。

關鍵差異：在 NFA 中，同一個（狀態, 符號）可能有多個後繼狀態（甚至零個——此時該分支「卡死」）。機器面對選擇時可以「分身」同時探索所有可能。

3.2 範例

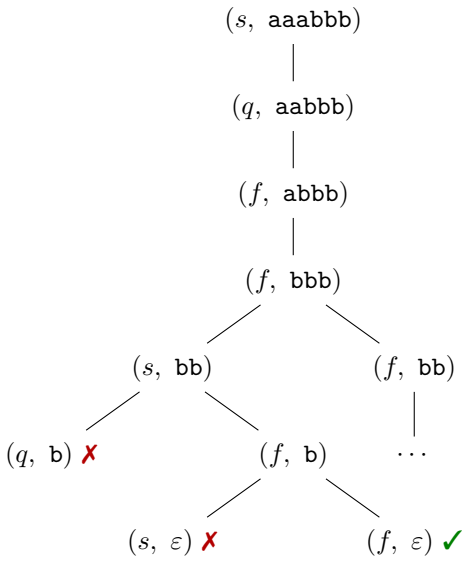
考慮 $\Sigma = \{a, b\}$，$Q = \{s, q, f\}$，$q_{\text{init}} = s$，$F = \{f\}$，轉移如下：

目前狀態	讀入符號	可能的新狀態
s	a	$\{q\}$
s	b	$\{q, f\}$
q	a	$\{f\}$
q	b	$\emptyset$
f	a	$\{f\}$
f	b	$\{s, f\}$



3.3 計算樹與接受準則

由於每一步可能有多個選擇，NFA 在輸入 w 上的所有計算構成一棵**計算樹**。例如輸入 $w =$ `aaabbb` 時（部分）計算樹如下：



定義 3.2 (NFA 的接受準則)**.**

$$\mathcal{A} \text{ 接受} w \iff \text{從} (q_{\text{init}}, w) \text{ 出發的任何一條（ANY）計算，其最終狀態屬於} F.$$

也就是說：只要計算樹中**存在**一條到達接受狀態的分支，整個輸入就被接受；全部分支都失敗才算拒絕。

3.4 NFA 與 DFA 的等價性

定理 3.3 (子集構造，Rabin–Scott)**.** 每一個 *NFA* 都可以轉換成一個接受相同語言的 *DFA*。

證明概要. 使用**冪集構造（powerset / subset construction）**：給定 NFA $\mathcal{A}$，建造 DFA $\mathcal{A}'$，其狀態為 $\mathcal{P}(Q)$ 中的元素，即「NFA 目前可能所在的狀態集合」。轉移定義為

$$\delta'(S, a) = \bigcup_{q \in S} \delta(q, a),$$

初始狀態為 $\{q_{\text{init}}\}$（含 ε-閉包），接受狀態為所有與 F 有交集的集合。可驗證 $\text{Language}(\mathcal{A}') = \text{Language}(\mathcal{A})$。 □

備註 3.4. 冪集構造最壞情況下會使狀態數從 n 暴增為 2^n ——非確定性不增加能力，但可能大幅增加簡潔性。

4 正則表達式與正則語言

定義 4.1 (正則表達式)**. 正則表達式（regular expression）**是只用以下三種運算、由字母表符號（以及 ε、$\emptyset$）建構出的式子：

- **連接（concatenation）**：wv ——先匹配 w 再匹配 v；
- **選擇（alternation / choice）**：$(w \cup v)$ ——匹配 w 或 v（亦常寫作 $w \mid v$、$w + v$）；
- **Kleene 星號（Kleene star）**：w^* ——把 w 重複零次或任意有限多次。

定義 4.2 (正則語言)**.** 能被某個正則表達式「表示」的字串集合，稱為**正則語言（regular language）**。

例 4.3 (正則表達式的例子)**.**

- 所有以 1 開頭、以 0 結尾的二進位字串：

$$1\,(1 \cup 0)^* 0$$

- 所有只含符號 $2, 3$ 且長度為偶數的字串：

$$\big((2 \cup 3)(2 \cup 3)\big)^*$$

- （較不顯然）所有代表 3 的倍數的二進位字串：

$$\big(0 \ \cup \ 1\,(01^*0)^*\,1\big)^*$$

這個表達式正好對應第 2 節那台 DFA ——把狀態圖中所有「從 q_0 回到 q_0」的迴路寫成表達式。

定理 4.4 (Kleene 定理)**.** 一個語言是正則語言 $\iff$ 它能被某個 *DFA*（等價地，*NFA*）辨識：

正則 $\iff$ 可被某 *DFA/NFA* 辨識

證明概要. ($\Rightarrow$) 對正則表達式的結構做歸納：單一符號、ε、$\emptyset$ 顯然有對應 NFA；再證明 NFA 可辨識的語言在連接、選擇、星號三種運算下封閉（用 ε-轉移把小自動機串接起來）。

($\Leftarrow$) 給定 DFA，可用「狀態消去法」或動態規劃（Kleene 演算法）把所有路徑彙整成一個正則表達式。 □

於是自然出現本週的核心問題：

所有語言都是正則的嗎？

答案是否定的，下一節給出經典反例。

5 並非所有語言都是正則的：$\{a^n b^n\}$

定理 5.1. 語言 $L = \{\, a^n b^n : n = 0, 1, 2, \dots \}$ **不是**正則語言。

直觀理由：要驗證 a 與 b 的個數相等，機器必須「數」出前段有幾個 a；但 n 可以任意大，而 DFA 只有固定有限個狀態，記不住任意大的計數。以下用鴿籠原理把這個直觀變成嚴格證明。

Proof. 採**反證法**，依投影片的八個步驟進行。

步驟 1）假設存在某 DFA $\mathcal{A}^?$ 使得

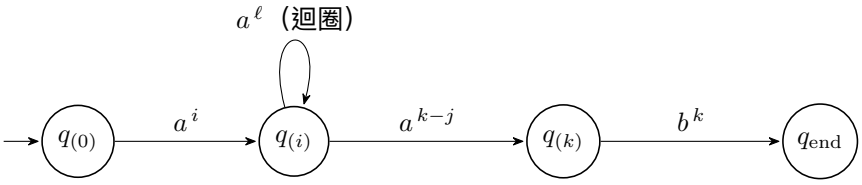
$$\text{Language}(\mathcal{A}^?) = \{a^n b^n : n = 0, 1, 2, 3, \dots\}.$$

步驟 2）令 p 為 $\mathcal{A}^?$ 的狀態個數。

步驟 3）取 $k > p$，觀察 $\mathcal{A}^?$ 在輸入 $a^k b^k$ 上的計算。在讀完前 k 個 a 的過程中，機器依序經過 $k+1$ 個狀態

$$q_{(0)} \xrightarrow{a} q_{(1)} \xrightarrow{a} q_{(2)} \xrightarrow{a} \cdots \xrightarrow{a} q_{(k)},$$

但機器總共只有 $p < k+1$ 個狀態。由**鴿籠原理**，必有某狀態在讀 a 的階段被重複造訪：存在 $0 \le i < j \le k$ 使 $q_{(i)} = q_{(j)}$。令 $\ell = j - i > 0$，即機器在讀了某 ℓ 個 a 之後繞了一個**迴圈**回到同一狀態。



步驟 4）現在比較兩個輸入的計算：

$$w_1 = a^k b^k \qquad \text{與} \qquad w_2 = a^{(k-\ell)} b^k.$$

w_2 就是把 w_1 中「迴圈那一段 a^{ℓ}」刪掉。由於迴圈起點與終點是同一個狀態，刪掉迴圈不影響其後的狀態走勢，因此**兩個計算讀完整個字串後停在完全相同的最終狀態。**

步驟 5）若該最終狀態是接受狀態，則兩個計算**都接受**；

步驟 6）若不是接受狀態，則兩個計算**都拒絕**。

步驟 7）但我們假設 $\mathcal{A}^?$ 恰好辨識 $\{a^n b^n\}$：它必須**接受** $w_1 = a^k b^k \in L$，同時**拒絕** $w_2 = a^{k-\ell} b^k \notin L$（因為 $\ell > 0$，a 的個數已不等於 b）。這與步驟 5/6「兩者命運相同」矛盾。

步驟 8）故由反證法，不存在接受 $L = \{a^n b^n\}$ 的 DFA。由 Kleene 定理，L 不是正則語言。 □

這個證明的本質

有限自動機 = 有限記憶。任何需要「無界計數」或「無界配對」的語言（如 a、b 個數相等），有限記憶都辦不到。證明的技術核心是**鴿籠原理** ⇒ **必有迴圈** ⇒ **迴圈可刪可重複** ⇒ **產生矛盾**。把這個論證抽象化，就得到下一節的泵引理。

6 泵引理（The Pumping Lemma）

6.1 敘述與證明

定理 6.1 (泵引理)**.** 設 $L \subseteq \Sigma^*$ 為 (無限的) **正則**語言。則存在某個 $p > 0$ (稱為**泵長度，*pumping length***)，使得每個長度 $|w| \ge p$ 的字 $w \in L$ 都可以寫成

$$w = xyz, \qquad x, y, z \in \Sigma^*,$$

並滿足：

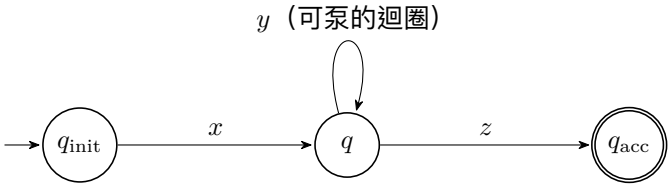
(i) $|xy| \le p$ 且 $|y| > 0$；

(ii) 對**所有** $n \ge 0$，$xy^n z \in L$。

Proof. 設 L 正則，取辨識 L 的 DFA，令 p 為其狀態數。任取 $w \in L$，$|w| \ge p$。機器讀 w 的前 p 個符號時造訪了 $p+1$ 個狀態，由鴿籠原理必有重複狀態 q。令：

- $x =$ 第一次抵達 q 之前讀的前綴；
- $y =$ 從 q 繞迴圈回到 q 之間讀的子字串（故 $|y| > 0$，且 $|xy| \le p$）；
- $z =$ 其餘部分。

由於 y 對應一個從 q 回到 q 的迴圈，把 y 重複 n 次（$n \ge 0$，含刪除）機器仍停在同一最終狀態，故 $xy^n z \in L$。 □



6.2 如何用泵引理證明語言不正則

泵引理的用途幾乎都是它的**逆否命題**：若語言 L 違反泵引理的結論，則 L 不是正則語言。標準的「四步驟範本」（可想成與對手的博弈）：

泵引理證明範本（反證法）

1. 假設 L 正則，於是存在泵長度 p（p 由對手指定，我們不能挑）。
2. **我們挑**一個字 $w \in L$，$|w| \ge p$（挑得好證明就容易）。
3. 考慮**所有**滿足 $|xy| \le p,\ |y| > 0$ 的分解 $w = xyz$（分解由對手指定，必須全部討論）。
4. **我們挑**某個 $n \ge 0$，證明 $xy^n z \notin L$，得到矛盾。故 L 不正則。

例 6.2 ($\{a^n b^n\}$ 再證一次)**.** 設 $L = \{a^n b^n : n \ge 0\}$ 正則，泵長度 p。挑 $w = a^p b^p \in L$。任何合法分解 $w = xyz$ 因 $|xy| \le p$，x, y 必全落在前段的 a 區，故 $y = a^m$，$m > 0$。取 $n = 2$：

$$xy^2 z = a^{p+m} b^p, \qquad m > 0,$$

a 比 b 多，不在 L 中，矛盾。故 L 不正則。

例 6.3 (投影片例題：$L = \{a^n b^m : n < m\}$)**.** 設 L 正則，泵長度 p。挑

$$w = a^p b^{p+1} \in L.$$

任何合法分解 $w = xyz$（$|xy| \le p$）中，y 必全為 a，即 $y = a^m$，$m > 0$。這次「往上泵」：取 $n = 2$，

$$xy^2 z = a^{p+m}\, b^{p+1},$$

由於 $m \ge 1$，有 $p + m \ge p + 1$，即 a 的個數不再嚴格小於 b 的個數，所以 $xy^2 z \notin L$，矛盾。故 $L = \{a^n b^m : n < m\}$ 不是正則語言。

例 6.4 (更多經典的非正則語言)**.** 用同樣的範本可以證明下列語言都不是正則的（建議當作練習）：

- $\{\, w \in \{0,1\}^* : w$ 中0 與1 的個數相等 $\}$　　（挑 $w = 0^p 1^p$）；
- 迴文語言 $\{\, w : w = w^R \,\}$　　（挑 $w = a^p b a^p$）；

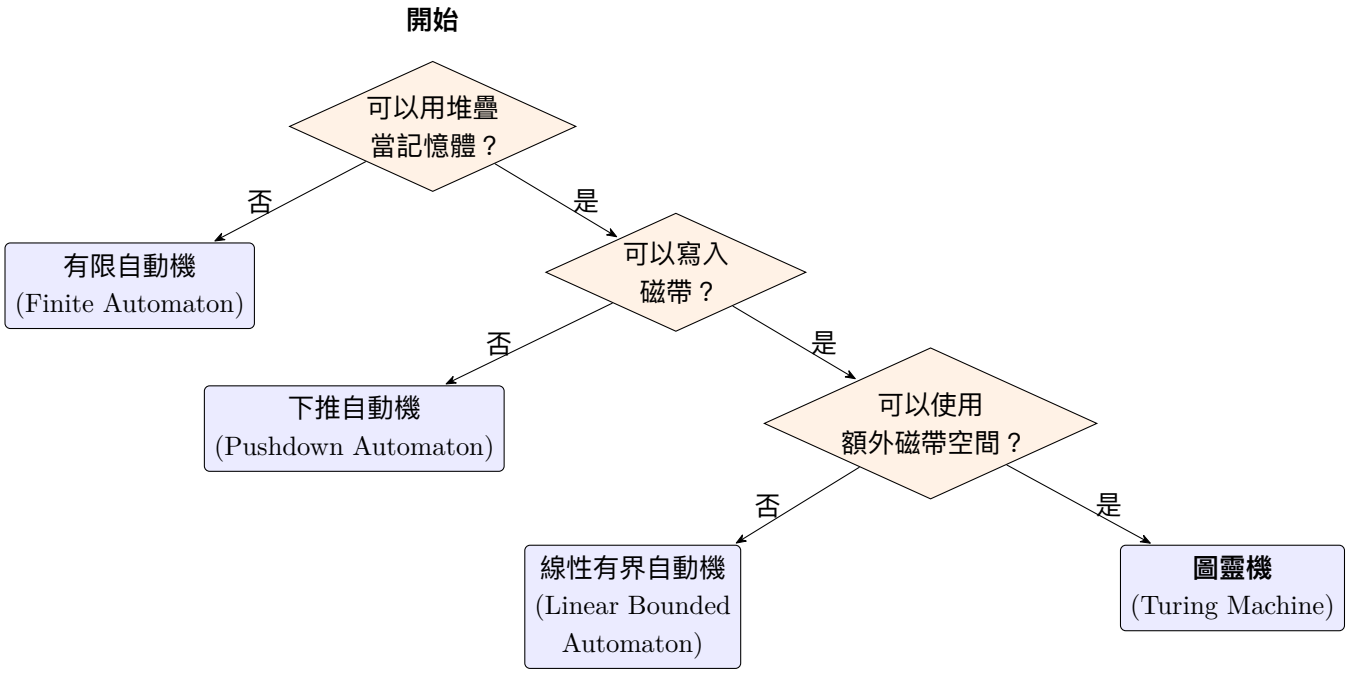
- $\{\, 1^{n^2} : n \geq 0 \,\}$　（完全平方數的間距會超過 $|y|$）；
- $\{\, 1^n : n$ 為質數 $\,\}$　（泵出合數長度）。

備註 6.5 (使用上的注意事項).
- 泵引理是**必要條件而非充分條件**：滿足泵引理的語言不一定正則。要證「正則」請直接構造 DFA/NFA 或正則表達式。
- 另一個判定工具（補充資料）：**Myhill–Nerode 定理**——語言正則 $\iff$ 它的「左商」$x^{-1}L = \{w : xw \in L\}$ 只有有限多種。以 $L = \{a^n b^n\}$ 為例，a^k 產生的商 $\{a^{n-k}b^n : n \geq k\}$ 兩兩不同，有無限多種，故不正則。

7 圖靈機（Turing Machines）

7.1 從有限自動機到圖靈機：機器的層級

有限自動機記憶力不足，自然的想法是給機器更多配備。下圖（依投影片）依「配備」對機器分類：



圖靈機（Alan Turing, 1936）擁有一條**無限長的磁帶**：可讀、可寫、讀寫頭可左右移動。它是本課程中最強的計算模型。

7.2 形式定義

定義 7.1 (圖靈機)**.** 一個**圖靈機（Turing Machine, TM）**是六元組

$$M = \langle\, \Sigma,\ Q,\ q_{\text{init}},\ q_{\text{accept}},\ q_{\text{reject}},\ \delta \,\rangle,$$

其中：

- $\Sigma = \{a_1, \ldots, a_k\}$：字母表（磁帶上另有**空白符號** $\sqcup \notin \Sigma$）；

- $Q = \{q_1, \ldots, q_n\}$：有限控制狀態集合；
- $q_{\text{init}} \in Q$：初始狀態；
- $q_{\text{accept}}, q_{\text{reject}} \in Q$：**接受狀態**與**拒絕狀態**（一旦進入立即停機）；
- 轉移函數

$$\delta : \big(Q \times (\Sigma \cup \{\sqcup\})\big) \to \big(Q \times (\Sigma \cup \{\sqcup\}) \times \{\leftarrow, \rightarrow\}\big),$$

$$\delta(\text{目前狀態},\ \text{讀到的符號}) \ \mapsto\ (\text{新狀態},\ \text{寫入符號},\ \text{移動方向}).$$

每一步，機器依目前狀態與讀寫頭下的符號：(1) 換到新狀態；(2) 在當前格**覆寫**一個符號；(3) 讀寫頭向左（$\leftarrow$）或向右（$\rightarrow$）移動一格。

定義 7.2 (TM 的組態與計算)**.** • **組態**：機器某時刻的完整描述

$$(\text{目前狀態},\ \text{讀寫頭左側的磁帶內容},\ \text{讀寫頭（含）右側的磁帶內容}).$$

- **計算**：執行 TM 得到的組態序列。

定義 7.3 (TM 的接受準則)**.**

$$M \text{ 接受} w \iff M \text{ **會停機**，且從} (q_{\text{init}},\ \varepsilon,\ w) \text{ 出發的計算最終停在} q_{\text{accept}}.$$

注意與 DFA 的根本差異：圖靈機**不保證停機**——它可能永遠跑下去。

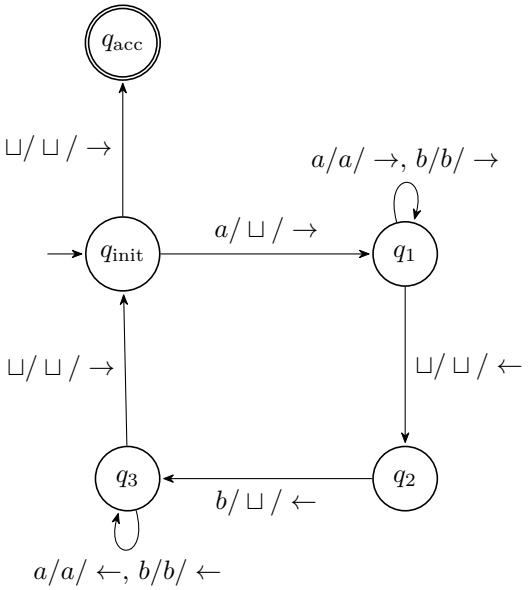
7.3 範例：辨識 $\{a^n b^n\}$ 的圖靈機

DFA 辦不到的事，圖靈機輕鬆做到。策略：反覆「劃掉最左邊的一個 a 與最右邊的一個 b」，若每次都恰好成對劃完則接受。

轉移表（$\sqcup$ 表空白）：

目前狀態	讀入	新狀態	寫入	移動
q_{init}	a	q_1	$\sqcup$	$\rightarrow$
q_{init}	$\sqcup$	q_{accept}	$\sqcup$	$\rightarrow$
q_1	a	q_1	a	$\rightarrow$
q_1	b	q_1	b	$\rightarrow$
q_1	$\sqcup$	q_2	$\sqcup$	$\leftarrow$
q_2	b	q_3	$\sqcup$	$\leftarrow$
q_3	a	q_3	a	$\leftarrow$
q_3	b	q_3	b	$\leftarrow$
q_3	$\sqcup$	q_{init}	$\sqcup$	$\rightarrow$

（未列出的組合，如 q_{init} 讀到 b、q_2 讀到 a 或 $\sqcup$，一律轉移到 q_{reject}。）



各狀態的意義：

- q_{init}：站在剩餘字串最左端。若是 a，劃掉它（寫 $\sqcup$）並往右出發；若已是空白，表示 a, b 已成對劃完——**接受**。
- q_1：一路向右走到字串**最右端**（遇到第一個空白）。
- q_2：退一格站在最後一個符號上。它必須是 b，劃掉它；若是 a 則拒絕。
- q_3：一路向左走回最左端，再回到 q_{init} 開始下一輪。

例 7.4 ($\texttt{aabb}$)**.**

$$
\begin{aligned}
&(q_{\text{init}}, \varepsilon, \texttt{aabb}) && \to (q_1, \sqcup, \texttt{abb}) && \to \cdots \to (q_1, \sqcup\texttt{abb}, \varepsilon)\\
&\to (q_2, \sqcup\texttt{ab}, \texttt{b}) && \to (q_3, \sqcup\texttt{a}, \texttt{b}\sqcup) && \to \cdots \to (q_{\text{init}}, \sqcup, \texttt{ab}\sqcup)\\
&\to (q_1, \sqcup\sqcup, \texttt{b}\sqcup) && \to (q_2, \sqcup\sqcup, \texttt{b}\sqcup) && \to (q_3, \sqcup\sqcup, \sqcup\sqcup)\\
&\to (q_{\text{init}}, \sqcup\sqcup\sqcup, \varepsilon) && \to (q_{\text{accept}}, \dots) && \checkmark \text{ 接受}
\end{aligned}
$$

反之，輸入 $\texttt{aaab}$ 會在第二輪於 q_2 讀到 a（或在 q_{init} 讀到殘留的 b 之前耗盡 b），轉入 q_{reject}，✗ 拒絕。

7.4 圖靈機可能不停機

投影片給了一台「追著磁帶上的方向指令跑」的機器：磁帶上寫滿 R（往右）與 L（往左），機器讀到 R 就右移、讀到 L 就左移並擦除符號。對某些輸入（如 RLRLLRL…），機器會接受或拒絕；但對另一些輸入（如 RLRRLRL…），機器會陷入**永無止境的左右徘徊**——既不接受也不拒絕，永遠「未定（Undecided?）」。

這引出 TM 行為的三分法：

定義 7.5 ($M(w)$ 記號)**.** 對輸入 w，定義

$$
M(w) := \begin{cases} 1 & \text{若} M \text{ 接受} w, \\ 0 & \text{若} M \text{ 拒絕} w, \\ \uparrow & \text{若} M \text{ 在} w \text{ 上不停機}. \end{cases}
$$

定義 7.6 (判定一個語言)**.** 機器 M **判定（decide）**語言 L，若它同時滿足：

- **健全性（Sound）**：若 $M(w)=1$ 則 $w\in L$（不誤收）；
- **完備性（Complete）**：若 $w\in L$ 則 $M(w)=1$（不漏收）；
- **停機性（Terminating）**：對所有 $w\in\Sigma^*$，M 在 w 上的執行時間 $T_M(w)<\infty$。

M 所辨識的語言記作

$$\text{Language}(M)=\{\,w\in\Sigma^*:M\text{ 接受}w\,\}.$$

辨識 vs. 判定（補充）

只滿足健全與完備（但可能在 $w\notin L$ 時不停機）的機器稱為**辨識（recognise）**L；對應的語言類是**遞迴可枚舉語言（recursively enumerable）**。三條件全滿足才叫**判定**，對應**可判定（遞迴）語言**。兩者的差距正是之後課程的主題（停機問題、不可判定性）。

8 非確定性圖靈機（NDTM）

8.1 定義

定義 8.1 (NDTM)**. 非確定性圖靈機（Non-deterministic Turing Machine, NDTM）**是

$$M=\langle\,\Sigma,\ Q,\ q_{\text{init}},\ q_{\text{accept}},\ q_{\text{reject}},\ \delta\,\rangle,$$

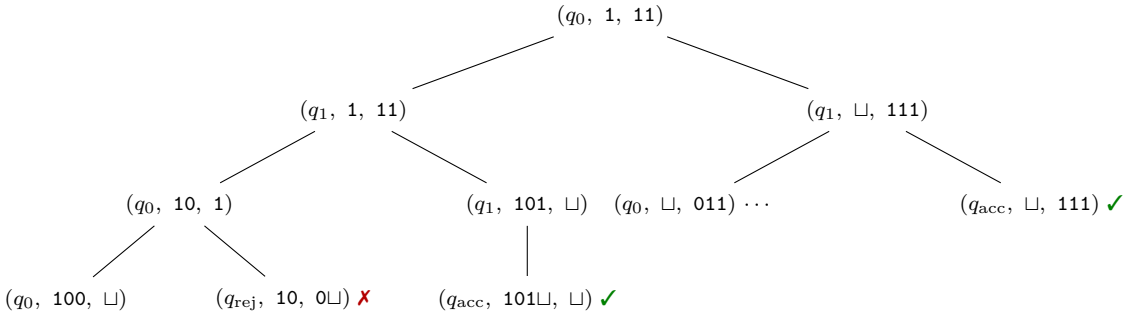
與 TM 唯一的差別是轉移函數改為

$$\delta:\big(Q\times(\Sigma\cup\{\sqcup\})\big)\ \to\ \mathcal{P}\Big(Q\times(\Sigma\cup\{\sqcup\})\times\{\leftarrow,\rightarrow\}\Big),$$

$$\delta(\text{目前狀態},\ \text{讀到的符號})\ \mapsto\ \big\{\,\text{所有可能的指令(新狀態},\ \text{寫入},\ \text{移動)}\,\big\}.$$

8.2 組態、計算樹與接受準則

- NDTM 的**組態**與確定性 TM 完全相同：(目前狀態, 左子字串, 右子字串)。
- NDTM 在輸入 w 上的**計算**是所有可能的組態序列；它們可以整理成一棵**分支計算樹（branching computation tree）**，每條分支對應一條可能的計算。



定義 8.2 (NDTM 的接受準則)**.**

$$M\text{ 接受}w\iff\text{從}(q_{\text{init}},\,\varepsilon,\,w)\text{ 出發的任何一條（ANY）計算最終停在接受狀態.}$$

即：計算樹中**存在**一條接受分支即可。

8.3 NDTM 不比 TM 更強

定理 8.3. 每個能被非確定性圖靈機辨識的語言，都能被某個確定性圖靈機辨識。

證明概要. 給定 NDTM M，建造確定性 TM M'，讓它有系統地**寫出 M 的分支計算樹中出現的所有組態**：

- M' 逐層（廣度優先，BFS）枚舉計算樹：先列出深度 1 的所有組態，再列出深度 2 的所有組態……。每個組態的子節點數有限（由 δ 決定的常數上界），故每一層都能在有限步內列完。
- 一旦 M' 寫出某個**狀態為 M 之接受狀態**的組態，M' 立即進入自己的接受狀態。

若 M 有接受分支（深度 d），BFS 必在第 d 層發現它，故 M' 接受相同的字串。 □

備註 8.4 (為何要用廣度優先？(補充)). 若改用深度優先（DFS），M' 可能一頭栽進某條不停機的分支，永遠出不來，從而錯過別條分支上的接受組態。BFS 保證：只要接受組態存在於有限深度，就一定會被找到。標準教科書（如 Sipser）用「三條磁帶」實作：輸入帶、模擬帶、位址帶（按字典序枚舉樹中路徑）。代價是時間可能**指數爆炸**：若 NDTM 用時 $f(n)$，模擬可能需要 $O(c^{f(n)})$ 時間——「N(D)TM 是否能被有效率地模擬」正是著名的 P vs. NP 問題的核心。

9 Church–Turing 論題

Church–Turing 論題（The Church–Turing Thesis）

任何能用某個**有限程序（finite process）**有效計算（effectively computed）的語言，都能被圖靈機辨識。

它在說什麼？ 「有效計算」指的是直觀上的演算法概念：一個由有限條明確規則構成、可機械地一步步執行的程序（紙筆即可，不需要靈感或無限資源）。論題主張：圖靈機已經完全捕捉了這個直觀概念——不存在「直觀上可計算、但圖靈機算不出來」的東西。

它是定理嗎？ 不是。「有效計算」是非形式的直觀概念，沒有數學定義，所以論題**無法被形式證明或否證**；它的地位介於「定義」與「自然定律」之間。但它得到壓倒性的支持證據：

- **殊途同歸**：1930 年代各自獨立提出的計算模型——Church 的 λ-演算、Gödel–Kleene 的一般遞迴函數、Turing 的圖靈機——被證明計算能力**完全相同**；其後的暫存器機（register machine）、現代程式語言等也全部等價。
- **穩健性**：對圖靈機加料（多條磁帶、雙向無限磁帶、非確定性——如定理 8.3）都不增加可辨識的語言類。
- **Turing 的分析**：Turing 1936 年的論文論證任何「人類計算員」按規則紙筆計算的過程，原則上都可由圖靈機逐步模仿。

歷史脈絡（補充） ：此論題源自 Hilbert 的判定問題（*Entscheidungsproblem*）——「是否存在能判定任何數學命題可否證明的通用程序？」Church 與 Turing 在 1936 年分別給出否定答案，而否定答案的前提正是先精確定義「程序」是什麼——這就是圖靈機誕生的原因。

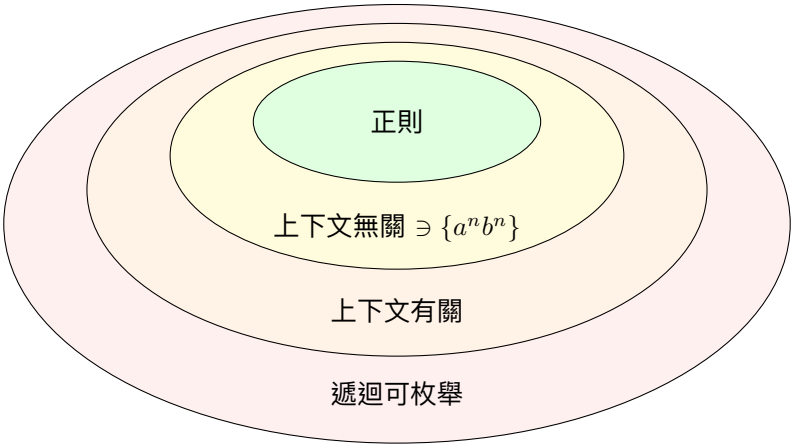
對本課程的意義

之後課程證明「某問題**無法被圖靈機**解決」（如停機問題）時，藉由 Church–Turing 論題，可以直接解讀為「該問題**無法被任何演算法**解決」——不管用什麼程式語言、什麼電腦。

10 全景：Chomsky 階層（補充）

把本週出現的機器與語言類整理起來，就是著名的 **Chomsky 階層**（Noam Chomsky, 1956）：

類型	語言類	對應的機器	例子
Type 3	正則語言	有限自動機（DFA/NFA）	$\{a^n : n > 0\}$
Type 2	上下文無關語言	（非確定性）下推自動機	$\{a^n b^n : n > 0\}$
Type 1	上下文有關語言	線性有界自動機（LBA）	$\{a^n b^n c^n : n > 0\}$
Type 0	遞迴可枚舉語言	圖靈機	停機問題的補集之外的一切



每一層都是上一層的**真子集**。本週證明了 $\{a^n b^n\}$ 落在「正則」之外（定理 5.1），又示範了圖靈機能輕鬆判定它——這正是階層中「機器越強、語言類越大」的一個具體切片。

11 本週重點整理

模型	轉移函數	接受準則
DFA	$\delta : Q \times \Sigma \to Q$	唯一計算的最終狀態 $\in F$
NFA	$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$	**任一**計算的最終狀態 $\in F$
TM	$\delta : Q \times (\Sigma \cup \{\sqcup\}) \to$ $Q \times (\Sigma \cup \{\sqcup\}) \times \{\leftarrow, \to\}$	停機且停在 q_{accept}（可能不停機 !）
NDTM	$\delta : Q \times (\Sigma \cup \{\sqcup\}) \to$ $\mathcal{P}(Q \times (\Sigma \cup \{\sqcup\}) \times \{\leftarrow, \to\})$	計算樹中**存在**接受分支

- NFA = DFA（冪集構造）；NDTM = TM（BFS 模擬）。**非確定性不增加計算能力**（但可能指數加速 / 縮小機器）。
- 正則 $\iff$ 可被 DFA/NFA 辨識 $\iff$ 可被正則表達式表示（Kleene 定理）。

- $\{a^n b^n\}$ 不正則：鴿籠原理 $\Rightarrow$ 迴圈 $\Rightarrow$ 刪/重複迴圈後接受結果不變 $\Rightarrow$ 矛盾。一般化即**泵引理**。
- 圖靈機與 DFA 的本質差異：可寫、可雙向移動、磁帶無限——以及**可能不停機**，因此「判定」需要健全、完備、停機三條件。
- Church–Turing 論題：圖靈機 $=$「演算法」的數學定義。

12 練習題（附提示與解答）

練習 1. 寫出一個 DFA，接受所有「含有偶數個 1」的二進位字串。再寫出對應的正則表達式。

解答

兩個狀態 $q_{\text{偶}}$（初始、接受）、$q_{\text{奇}}$；讀 1 互換、讀 0 不動。正則表達式：$\big(0 \cup 1\,0^*\,1\big)^*$。

練習 2. 用冪集構造，將第 3 節的 NFA（狀態 s, q, f）轉換為 DFA。需要列出哪些「狀態集合」？

解答（概要）

從 $\{s\}$ 開始依轉移閉包展開：$\{s\} \xrightarrow{a} \{q\}$、$\{s\} \xrightarrow{b} \{q,f\}$、$\{q\} \xrightarrow{a} \{f\}$、$\{q\} \xrightarrow{b} \emptyset$、$\{q,f\} \xrightarrow{a} \{f\}$、$\{q,f\} \xrightarrow{b} \{s,f\}$、$\{f\} \xrightarrow{a} \{f\}$、$\{f\} \xrightarrow{b} \{s,f\}$、$\{s,f\} \xrightarrow{a} \{q,f\}$、$\{s,f\} \xrightarrow{b} \{q,s,f\}$、$\{q,s,f\} \xrightarrow{a} \{q,f\}$、$\{q,s,f\} \xrightarrow{b} \{q,s,f\}$，加上死狀態 $\emptyset$ 共 7 個可達狀態；凡含 f 者為接受狀態。

練習 3. 用泵引理證明 $L = \{\, w \in \{a,b\}^* : w$ 中a, b 個數相等 $\}$ 不是正則語言。

解答

設 L 正則、泵長度 p，挑 $w = a^p b^p \in L$。任何合法分解中 $y = a^m$（$m > 0$）。取 $n = 2$ 得 $a^{p+m} b^p \notin L$，矛盾。(也可以由「$L \cap a^* b^* = \{a^n b^n\}$ 且正則語言對交集封閉」直接導出。)

練習 4. $L = \{a^n b^m : n < m\}$（例 6.3）中，為什麼挑 $n = 0$（把 y 刪掉）不一定成功？挑 $n = 2$ 為什麼一定成功？

解答

取 $w = a^p b^{p+1}$、$y = a^m$。刪掉 y 得 $a^{p-m} b^{p+1}$，仍滿足 $p - m < p + 1$，**還在** L 裡，得不到矛盾。往上泵一次得 $a^{p+m} b^{p+1}$，因 $m \geq 1$ 使 $p + m \geq p + 1$，破壞嚴格不等式 $n < m$，必不在 L。教訓：泵的方向要朝「會破壞語言條件」的那一邊。

練習 5. 修改第 7 節的圖靈機，使其判定 $L = \{a^n b^n c^n : n \geq 0\}$（這個語言連下推自動機都辦不到）。

解答（概要）

每一輪改為「劃掉一個 a、一個 b、一個 c」：從左端找第一個 a 改寫成 $\sqcup$（或標記 X），向右找第一個 b 標記，繼續向右找第一個 c 標記，再回到左端。若某輪三者無法湊齊、或符號順序錯誤則拒絕；若磁帶上只剩標記/空白則接受。每輪至少劃掉三個符號，必定停機，故此機器**判定**該語言。

練習 6 (思考題)**.** NDTM 的 BFS 模擬中，如果 NDTM 的某條分支不停機，確定性模擬機 M' 在「$w \notin \text{Language}(M)$」時會發生什麼事？這說明 M' 是「辨識」還是「判定」$\text{Language}(M)$？

解答

若沒有任何接受分支且存在不停機分支，BFS 會永遠枚舉下去，M' 不停機。因此 M' 一般而言只**辨識**（recognise）$\text{Language}(M)$；只有當 NDTM 的所有分支都保證停機時，模擬才成為**判定**程序。

參考資料

1. C. Hampson, *5CCS2FC2 Foundations of Computing II, Week 1* 投影片（recap / anbn / turing / ndtm），King's College London.
2. M. Sipser, *Introduction to the Theory of Computation*, 3rd ed., Cengage, 2012.（圖靈機形式定義、NDTM 等價性定理 3.16）
3. J. E. Hopcroft, R. Motwani, J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd ed., Pearson, 2006.
4. Wikipedia: *Pumping lemma for regular languages*; *Church–Turing thesis*; *Chomsky hierarchy*.
5. Stanford Encyclopedia of Philosophy: *The Church-Turing Thesis*(https://plato.stanford.edu/entries/church-turing/) .
6. J. Watrous, *Introduction to the Theory of Computing* 講義第 12 章（University of Waterloo）.
7. A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proc. London Math. Soc.*, 1936.