Ch11
計算理論

李官陵 彭勝龍 羅壽之

- 所謂計算理論就是在探討計算機可以解決問題的極限，及問題難易度的分類，主要訓練資訊相關人員對於問題的掌握度。

決策問題

- 一般來說，問題可以分成二類：最佳化 (optimization) 問題和決策 (decision) 問題。
- 最佳化問題就是該問題的答案必須是最佳值。
- 問題 1：最大值問題

  輸入：n 個數字

  輸出：最大的數字
- 問題 2：最短路徑問題

  輸入：一個圖形 (點代表城市，邊代表二城市間的距離)，二個點 A 和 B

  輸出：A 到 B 的最短路徑距離

決策問題 (續)

- 問題 3：背包問題

  輸入：一個背包，載重量為 K，n 個物品，每個物品有其重量與價值

  輸出：背包可以承載下 (總重 $\leq K$)，可以獲得的最大價值

- 決策問題，就是答案只有二種可能：「是」或「否」。
- 問題 4：滿足問題

  輸入： m 個皆以 or 連結的布林式子，總共含 n 個布林變數

  輸出：是否存在一組解滿足 m 個布林式子皆為真

決策問題 (續)

- 很容易的就可以把最佳化問題改為決策問題。
- 問題 1A：決策版的最大值問題

  輸入：一個數 K 和 n 個數字

  輸出：這 n 個數字裡，是否存在一個數字 $\geq K$
- 問題 2A：決策版的最短路徑問題

  輸入：一個數 K，一個圖形 (點代表城市，邊代表二城市間的距離)，二個點 A 和 B

  輸出：是否存在一條 A 到 B 的路徑，使得它的長度 $\leq K$

決策問題(續)

隨堂練習

- 請寫出問題3的決策版本。
- 解答：
- 輸入：一個背包，載重量為 K，n 個物品，每個物品有其重量與價值，一個值 V
- 輸出：是否存在數個物品，其重量和小於等於 K，且其價值和大於等於 V

決策問題 (續)

- 我們先來看問題 1 和 1A，很顯然的，如果我們能夠解決問題 1，也就是說能在 *n* 個數字裡找出最大值，令這個值為 *max*，那麼我們就能回答問題 1A 了，只要比較 *max* 和 *K* 的大小，如果問題 1 可以解決，那麼問題 1A 也可以被解決。
- 假設我們能夠解決問題 1A，也就是說，在 *n* 個數字裡，我們任意給一個 *K*，都能得到是或否的答案，為了解決問題 1，我們首先任意給一個 *K* 丟給問題 1A 的解答者，若答案為「是」，我們再繼續試 2*K*，若答案為「否」，則再試試 *K* / 2。

決策問題(續)

- 一個問題的最佳化版本，可以藉由它的決策版本解答的幫忙，而找出最佳解，時間複雜度或許不同，但是一個版本能解，另外一個版本也絕對可以解。
- 決策問題的答案只有「是」或「否」二種可能，相對單純多了，所以在計算理論裡，我們只考慮決策問題。

不能決定的問題

- 一個決策問題可以被決定 (decidable)，表示我們可以在有限的時間內，回答該問題「是」或「否」。
- 「決定」只適合用在決策問題上，而「解決」可以適用在所有問題上。

不能決定的問題(續)

- 有一個理髮師，他除了幫人理髮之外，有時也會幫顧客刮鬍子，但是他幫人家刮鬍子的原則是「只幫那些不會自己刮鬍子的人刮鬍子」，這個原則也高高的樹立在他的理髮店招牌之下，因此有路人就好奇了，理髮師他到底會不會幫自己刮鬍子呢？
- 其實這也是一個決策問題，因為它的答案是「會」或「不會」。那到底會不會呢？首先，我們先將所有的人分成二類，A類是會幫自己刮鬍子的人，另一類是不會幫自己刮鬍子的人，歸為B類。按照題意，理髮師只幫B類的人刮鬍子。我們從下面二種情況來分析：

不能決定的問題(續)

1. 理髮師會幫自己刮鬍子

   在這種情況下，理髮師是屬於A類的人，因為他只幫B類的人刮鬍子，所以理髮師他不會幫自己刮鬍子，這跟前提「理髮師會幫自己刮鬍子」矛盾，所以此情況不存在。

2. 理髮師不會幫自己刮鬍子

   在這種情況下，理髮師是屬於B類的人，所以符合他刮鬍子的原則，因此他會幫自己刮鬍子，這個結論又跟前提「理髮師不會幫自己刮鬍子」矛盾，所以此情況也不存在。

不能決定的問題 (續)

- 給定一個程式 A 和它的一個輸入 x，我們用 A(x) 表示執行 A，並且以 x 作為輸入，因為 A 是一個可執行程式，但是它存在於檔案時，是一個二位元碼 (binary code) 檔，因此我們用 [A] 表示該檔案，可以作為參數傳遞。
- 問題 5：停止 (Halting) 問題

  輸入：一個程式 A 和它執行時需要的輸入參數 x

  輸出：A(x) 是否會停止？
- 停止問題是計算理論裡很重要的一個問題。
- 問題 5 是一個決策問題。

不能決定的問題 (續)

$$H([A],x)=\begin{cases}Y & A(x)\text{會停}\\ N & A(x)\text{不會停}\end{cases}$$

- 因為我們假設停止問題是可以被決定的，所以我們上面所訂的 H 是存在的。把一個程式當作另外一個程式的副程式來執行，也是程式設計裡常見的情況，因此使用 H 當副程式，我們可以設計另一個程式 D，它以另一個程式 B 作為輸入，而執行下面二個步驟：

1. 執行 H([B], [B])
2. 如果 H([B], [B]) = Y，進入無窮迴圈
   如果 H([B], [B]) = N，程式結束

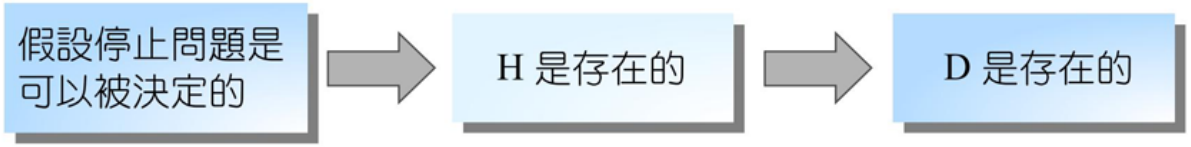

圖 11.1　程式 D 的推論

不能決定的問題 (續)

- D 也可以像 H 一樣，用下面的表示式來表示：

$$D([B]) = \begin{cases} \text{會停} & B([B]) \text{不會停} \\ \text{不停} & B([B]) \text{會停} \end{cases}$$

- 因為 H 會在 D 中被呼叫，所以 B([B]) 會不會停，是可以由 H 來決定，也就是說「B([B]) 不會停」的意思表示「H([B], [B]) = N」，而「B([B]) 會停」的意思表示「H([B], [B]) = Y」。

不能決定的問題 (續)

1. D([D]) 會停

   一執行 D([D])，就會先執行 H([D], [D])，在 H 裡面會執行 D([D]) 判斷會不會停，因為 D([D]) 會停，所以 H 會傳回 Y，但在 D 的第二個步驟裡，因為 H 傳回 Y，所以 D 就進入無窮迴圈了，表示D([D]) 不會停，和假設矛盾，因此這個假設「(D([D]) 會停」是錯的。

2. D([D]) 不會停

   同樣的，執行 D([D])，就會先執行 H([D], [D]) 來判斷 D([D]) 會不會停，因為 D([D]) 不會停，所以 H 會傳回 N，於是 D 就結束了，也就是說 D([D]) 停了，和假設矛盾，也就是說這個假設「D([D]) 不會停」也是錯的。

不能決定的問題 (續)

- 二種假設都是錯的，所以 D 是不存在的，「停止問題是可以被決定的」是錯的。

  停止問題是不能被決定的

- 「停止問題是不能被決定的」這個事實證明了並不是所有的決策問題都可以被決定，也可以說並不是所有的問題都可以被解決，問題是無法解的 (unsolvable)。
- 問題 6：執行時間問題

  輸入：一個程式 A 和它執行時需要的輸入參數 x

  輸出：A(x) 的執行時間
- 問題 6 之所以不能解，是因為如果我們可以解問題 6，那麼就知道 A(x) 會不會停了。

可決定的問題

- 圖靈機，它是一部計算的機器，因為藉由它的原理，而發明了計算機。基本上，若一個決策問題是可以被決定的，那麼一定存在有一台圖靈機來決定它。廣義的說，一個問題若是可解的話，一定存在一台圖靈機來解決它。

邱奇—圖靈論題 (Church-Turing Thesis)

- 演算法的能力和圖靈機的能力相等。

- 這個論題的意思就是說，任何在演算法上可計算的問題，同樣也可由圖靈機計算，而任何可以被圖靈機計算的問題，也能被演算法計算。

可決定的問題 (續)

- 『什麼問題是可以解決 (決定) 的呢？』，答案就是該問題存在一個演算法可以算出答案來。
- 舉例來說，我們想證明問題 4 是可以被決定的，下面是一個問題 4 的輸入例子：

$$x_1 + x_2 + x_3$$
$$\overline{x_1} + \overline{x_2} + x_3$$
$$x_1 + \overline{x_2} + \overline{x_3}$$

可決定的問題 (續)

- 這個例子裡有 3 個布林式子，也有 3 個布林變數，要知道這三個式子是否能被滿足？我們可以檢查這三個變數的所有可能組合，例如 $x_1 = \mathrm{T}, x_2 = \mathrm{F}, x_3 = \mathrm{F}$ 就可以滿足這三個式子，讓每一個式子皆為真 (T 為真，F 為假)。以上我們說明了一個演算法，那就是檢查所有變數可能值的組合，如果有一個組合滿足所有的式子，那就回答「是」，反之，就回答「否」。因為 n 個變數的組合有 2^n 組，這是有限的，因此演算法一定可以完成，所以證明了問題 4 是可以被決定的。

可決定的問題 (續)

- 要找一個演算法來決定問題 1A 是很簡單的，只要將這 n 個數字依序跟 K 來比較，只要比到一個數 $\geq K$，就回答「是」，如果沒有，就回答「否」。
- 二個演算法來決定問題 4 和問題 1A，統一用該問題的輸入大小來做比較，假設輸入大小設為 I。在問題 4 裡，此 I 值就是所有布林式子的總合，因為每種變數組合都要檢查一遍，所以全部都要檢查的量為 $2^n I$。而問題 1A 的輸入大小為 n 個數字，因此這 n 個數字全部都拿來比較，也只不過是 1 個 I。所以這二個演算法的時間複雜度差了 2^n 倍。可解決的問題裡，又可以分為容易 (easy) 解的問題和困難 (hard) 解的問題。
- 一般來說多項式時間代表簡單，指數代表複雜，因此目前來說，問題 4 是困難的，而問題 1A 是簡單的。

可決定的問題(續)

- 在計算理論裡，我們首先要了解哪些問題是可決定的，哪些問題是不可決定的。然後在那些可決定的問題裡，再區分出哪些問題是簡單的，那些問題是複雜的。可是問題的難易分類並不是僅由目前演算法的複雜度來決定。
- 我們可以定義困難的問題是那些至少需要花指數時間來決定的問題，但是我們無法證明現在需要花指數時間的問題，將來不可能被改進到只花多項式時間。
- 目前的想法是將那些可以在多項式時間內決定的問題收集在一個集合 P 裡，例如問題 1A 和問題 2A 都是屬於 P，問題 3 的決策版和問題 4 都還無法被證明屬於 P。但我們可以定義另一個集合 NP，它包含的元素更廣，前面所說的四個問題都被包含在裡面，那什麼是 NP 呢？

可決定的問題 (續)

- NP- 演算法 (Nondeterministic Polynomial-time Algorithms)，也就是多項式時間的不確定演算法。這種演算法允許不確定的步驟，但整體必須在多項式時間內完成。我們可以去猜答案 (屬於不確定步驟)，然後去驗證這個猜到的答案，整個猜加上驗證的步驟必須在多項式時間內完成。

可決定的問題 (續)

演算法 1

- 輸入：一個數 K 和 n 個數字
- 輸出：這 n 個數字裡，是否存在一個數字 $\geq K$
- 步驟：
- (1) 1 到 n 猜一個數字，設為 i
- (2) 比較第 i 個值和 K 的大小，若大於等於，則回答「是」，若小於，則回答「否」

可決定的問題 (續)

- 演算法 1 的步驟 1 是猜一個數字，只要 $O(1)$ 就可以完成，而步驟2 是依題意去比較大小，也是花 $O(1)$ 的時間，所以整個時間是 $O(1)$。而我們提到需要一個一個比較的明確 (deterministic) 演算法則需 $O(n)$ 的時間。
- 不確定演算法之所以會正確，是基於一個假設：如果有解，就一定會被猜到；如果無解，就無所謂。
- 問題 4 的 NP-演算法。

可決定的問題 (續)

演算法 2

- 輸入：m 個皆以 or 連結的布林式子，總共含 n 個布林變數
- 輸出：是否存在一組解滿足 m 個布林式子皆為真
- 步驟：
- (1) 分別對每一個變數猜出一個真假值
- (2) 用猜出的那組真假值，一一檢查 m 個布林式子，若皆滿足，則回答「是」，若有一式不滿足，則回答「否」

可決定的問題 (續)

- 套用之前的分析概念，假設輸入大小是 I (I 的最大值是 mn，因為有 m 個式子，每個式子可以有 n 個變數)，所以步驟 1 要花 $O(n)$ 的時間，而步驟 2 是 $O(I)$，所以整個時間是 $O(I)$，很明顯的是線性時間，和之前所述的 $O(2^n I)$ 指數時間的明確演算法差異很大。
- 集合 NP 就是那些具有 NP-演算法的決策問題所匯聚而成的集合。
- 很明顯的 $P \subseteq NP$，但是，是不是 $P \subset NP$？也就是 $P \neq NP$，則還沒有被證明出來，雖然大部分的人都選擇相信 $P \neq NP$。因此美國克雷數學研究所 (Clay Mathematics Institute) 於 2000 年 5 月24 日，將下面的問題收錄在七個千禧年難題中，並懸賞每題 100 萬美金。
- P = NP?