談談庫克定理的證明

堵丁柱 • 葛可一

談到庫克定理, 凡是懂點計算複雜性理論的人都知道它。它是該理論中最重要, 最基礎的定理之一。因此, 凡是含有計算複雜性內容的專著和教科書都會講解庫克定理。各本書中的證明也不盡相同。其花樣翻新, 恰似八仙過海, 各顯其能。這中間溶入了許多人的智慧。不過, 其中也有不小心, 含有失誤者。讓我們就從這樣一個含有失誤的相當流行的證明開始, 到這座百花園中走一走, 學一學。學習不就是去僞存眞, 去粗取精的過程嗎?

首先, 讓我們簡略解釋一下什麼是庫克定理。如下是個趣味邏輯問題: 三個好朋友苗苗, 壯壯, 山山告訴人們, 他們三個之中誰大誰小。苗苗說:“如果山山不是最小的, 那麼我就是”。壯壯說: “如果我不是最小的, 那麼苗苗就是最大的”。你能知道誰大誰小嗎?

解它的方法之一是列邏輯方程。以 A, B, C 分別記苗苗, 壯壯, 山山三人。以 X_0 記命題 “X 是最大的”, 以 X_y 記命題 “X 是最小的”。這樣一來, 命題“如果山山不是最小的, 那麼苗苗就是”。可以表達爲 C_y+A_y, 而命題“如果壯壯不是最小的, 那麼苗苗就是最大的”。可以表達爲 B_y+A_o。由於這兩個命題爲眞, 因此我們得到兩個等式:

$$C_y + A_y = 1$$

$$B_y + A_o = 1$$

將這兩個等式相乘, 得

$$(C_y + A_y)(B_y + A_o) = 1.$$

亦即,

$$C_yB_y + C_yA_o + A_yB_y + A_yA_o = 1.$$

注意, 不可能有兩人最小, 也不可能有一人既最小也最大。因此,

$$C_yB_y = A_yB_y = A_yA_o = 0.$$

這樣, 我們得到

$$C_yA_o = 1,$$

也就是說, 苗苗最大, 山山最小, 壯壯居中。

給出一個邏輯方程, 判斷它是否有解。或者, 給出一個邏輯函數, 判斷它是否有使函數值等於1的變量賦值。這稱爲 SAT 問題。具有使函數值等於1的變量賦值的邏輯函數稱爲可滿足的, 該變量賦值稱爲眞賦值。

庫克定理: SAT問題是 NP 完全的。

現在, 來看看一個相當流行的教科書[4]中的證明。爲避免太繁瑣, 我們將概述該證明, 只將值得注意的地方詳細列出。尤其, 無法詳細解釋有關概念。誠請讀者諒解。如果有不懂的術語, 請對照原書。

首先證明, SAT 問題屬於 NP 類。這很容易, 對給出函數, 猜一組變量賦值, 檢驗它是否是眞賦值。這過程可以由 NTM (Non-deterministic Turing Machine) 在多項式時間內實現。

其次證明, NP 類中任何語言 L, 均有 $L \leq_m^p SAT$, 也就是說, 存在一個多項式時間可計算的映射 g 使得, $x \in L$ 若且唯若 $g[x](\cdot) \in SAT$, 其中 $g[x](\cdot) \in SAT$ 的意思是說, $g[x](\cdot)$ 是可滿足的邏輯函數。

語言 L 屬於 NP 類, 這意味著存在多項式時間單帶 NTM M 接受 L。(單帶 NTM 由三部份組成: 一條記憶帶, 帶上一個讀寫頭和與其連接的有限控制器。) 設多項式 $p(n)$ $(p(n) \geq n)$ 是 M 的時間上界。這意味著, $x \in L$ 若且唯若, 在輸入 x 之後, M 有一條計算道路, 最多經過 $p(n)$ 次移動, 就會進入終止狀態, 其中, n 是符號行 x 的長度, 即 $n = |x|$.

設 $\#\beta_0\#\beta_1\cdots\#\beta_{p(n)}$ 是 M 的一條計算道路。其中, 每個 β_i 是 M 的一個 ID。ID是瞬間像 (Instantaneous Description) 的縮寫。它含 M 在瞬間具有的如下三個數據: 帶上符號行, 讀寫頭位置, 有限控制器的狀態。由於讀寫頭最初停在帶的最左格處, 而每次只向左或右移動一格, 因此諸 β_i 最多有 $p(n)+1$ 格非空。讀寫頭位置也一定在最左邊的這 $p(n)+1$ 格之中。這樣一來, 每個 ID β_i 可以只含這個 $p(n)+1$ 格。爲使 ID 更簡捷, 不妨把有限控制器的狀態也放到讀寫頭所在的格里。這也就是說, 每個 ID β_i 含$p(n)+1$ 格。其中, $p(n)$ 格含 Γ 中符號, 恰有一格含 $Q\#\Gamma$ 中元素 (稱爲複合符合)。這裡, Γ 是M 的帶上可用符號字母表, Q 是 M 的有限控制器的狀態集合。值得注意, 當 $x \in L$ 時, M 可能只移動 $k < p(n)$ 次就進入終止狀態。這時, 我們令 $\beta_k = \cdots = \beta_{p(n)}$。

記 ID β_i 的第 j 格爲格 (i,j), $0 \leq i \leq p(n), 1 \leq j \leq p(n)+1$。對每個 $a \in \Gamma \cup Q\#\Gamma$ 和每個格 (i,j), 定義一個邏輯變量 $c_{i,j,a}$: $c_{i,j,a} = 1$ 若且唯若格 (i,j) 所含的是 a。

下面, 需要構造四個邏輯函數 g_i, $i = 1,2,3,4$, 使得, $g_i = 1$ 若且唯若下面的 (i) 眞。

(1) 對每格 (i,j), 有且僅有一 $a \in \Gamma \cup Q\#\Gamma$ 使得 $c_{i,j,a} = 1$。
(2) β_0 是 M 的初始 ID。
(3) $\beta_{p(n)}$ 含終止狀態。
(4) 對每個 i, β_{i+1} 是由 β_i 經一次移動而得到。

如果這四個邏輯函數構造好了, 那麼令 $g[x] = g_1g_2g_3g_4$, 就會有 $x \in L$ 若且唯若 $g[x]$ 可滿足。事實上, 若 $x \in L$, 則對輸入 x, M 有一條計算道路 $\#\beta_0\#\beta_1\cdots\#\beta_{p(n)}$ 滿足 (2)(3)(4)。考慮如下變量賦值

$$c_{i,j,a} = \begin{cases} 1 & \text{若 } \beta_i \text{ 的第 } j \text{格所含的是a} \\ 0 & \text{否則。} \end{cases}$$

這賦值可使 $g_i = 1$, $i = 1,2,3,4$, 因此使 $g[x] = 1$。反之, 若 $g[x]$ 可滿足, 則有變量賦值使 $g[x] = 1$。這意味著, (1)(2)(3)(4) 眞。由於 (1) 眞, 可以構造 $\#\beta_0\#\beta_1\cdots\beta_{p(n)}$ 使得, $c_{i,j,a} = 1$ 若且唯若格 (i,j) 所含的是 a。由於 (2)(3)(4) 眞, 可知 $\#\beta_0\#\beta_1\cdots\#\beta_{p(n)}$ 是接受 x 的一條計算道路。於是, $x \in L$。

現在, 所剩工作是構造四個邏輯函數 $g_i, i = 1,2,3,4$ 了。對三個邏輯函數 g_1, g_2 和 g_3 的構造, 我們沒有異議。可是, 對於 g_4 的構造, 我們就不敢苟同了。

事實上, 該書首先"定義一個邏輯判據 $f(W,X,Y,Z)$ 使得, $f(W,X,Y,Z) = 1$ 若且唯若, 當某個 ID 的第 $j-1, j, j+1$ 格所含的分別是 W,X,Y 時, Z 允許出現在緊跟其後的 ID 的第 j 格中 [若 $j = 1$, 則 $W = \#$; 若 $j = p(n)+1$, 則 $Y = \#$]。"然後定義 g_4 爲

$$\prod_{(i,j)}\Big(\sum_{f(w,x,y,z)=1}(c_{i,j-1,W}c_{i,j,X}c_{i,j+1,Y}c_{i+1,j,Z})\Big).$$

爲什麼這樣定義 g_4? 有一段解釋:

"爲了弄清怎樣寫第四個公式, 其表達每個 ID $\beta_i, i \geq 1$, 是由 β_{i-1} 通過 β_{i-1} 中複合符號的移動而得, 要注意, 我們基本上可以從 β_{i-1} 的相應符號以及它兩邊的符號 (其中之一可能是 $\#$) 導出 β_i 的符號。亦即, β_i 中符號與 β_{i-1} 中相應符號相同, 除非後者或者它的相鄰者是複合符號, 並且讀寫頭移動到了那個 β_i 的符號"。

這個聽起來挺順耳的定義眞能保證"若 $g_4 = 1$ 則 (4) 眞"嗎? 其實, 不能。讓我們舉一個反例, 讀者就會很快明白。

設 δ 是 M 的轉換函數。由於 M 是個 NTM, δ 從 $Q\times\Gamma$ 映射到 $2^{Q\times\Gamma\times\{R,L\}}$; 亦即, 對每個狀態 $q \in Q$ 和每個符號 $a \in \Gamma$, $\delta(q,a)$ 是 $Q\times\Gamma\times\{R,L\}$ 的一個子集合。$(p,b,R) \in \delta(q,a)((p,b,L) \in \delta(q,a))$ 的意思是說, 當 M 在狀態 q 下, 讀寫頭讀到符號 a 時, M 可以進入狀態 p, 同時, 讀寫頭將所讀格中符號 a 塗掉, 寫上符號 b 並且向右 (左) 移動一格。

現在, 讓我們把 $p(n)+1$ 個 ID 排列成一個 $(p(n)+1)\times(p(n)+1)$ 矩陣; 第 i 行是第 i 個 ID β_i。這時, g_4 的上述定義意味著, 我們用四個格 $(i,j-1),(i,j),(i,j+1),(i+1,j)$ 所形成的如下窗口 (記爲窗口 A) 來檢驗這矩陣。

$(i,j-1)$	(i,j)	$(i,j+1)$
	$(i+1,j)$	

考慮一個轉換函數$\delta(q,a)=\{(p,b,R),(p,b,L)\}$。對此轉換函數來說, 下面兩種移動都是合理的:

c	$q\#a$	d
c	b	$p\#d$

c	$q\#a$	d
$p\#c$	b	d

現在, 我們用四格窗口 A 來觀察如下所定義的 β_i 和 β_{i+1}。

c	$q\#a$	d
$p\#c$	b	$p\#d$

不難看出, 每個窗口景象都與前面的合理移動中的某個窗口景象相同。這就是說, 所定義之 β_i 和 β_{i+1} 滿足 $g_4 = 1$。可是, 顯然 (4) 不眞。β_{i+1} 含兩個複合符號, 不是 ID, 怎麼能自 β_i 通過一次移動而得到呢。

應該指出, 使用四個函數是較勇敢的做法, 多數作者在證明庫克定理中構造六個邏輯函數。除前述之四個以外, 還有兩個勿庸置疑、容易構造的 g_5 和 g_6 , 分別表述下面兩個條件:

(5) 每個 β_i 恰含一個狀態。

(6) 每個 β_i 恰含一個獨寫頭所讀之格。

這實質上來說, 每個 β_i 恰含一個複合符號。這種條件的簡單增加是否可以補救 [4]中的過失呢? 答案是否定的, 雖然所加的條件使先前反例不再成立, 可是我們可以再造一個稍微苦澀一點的。

考慮一下轉換函數$\delta(q,a)=\{(p,b,R),(r,e,R)\}$, 其中 $p \neq r$ 和 $b \neq e$。對此種轉換函數來說, 下面兩種移動都是合理的:

c	$q\#a$	d
c	b	$p\#d$

c	$q\#a$	d
c	e	$r\#d$

現在, 我們用四格窗口 A 來觀察如下所定義的 β_i 和 β_{i+1}。

c	$q\#a$	d
c	b	$r\#d$

不難看出, 每個窗口景象都與前面的合理移動中的某個窗口景象相同。這就是說, 所定義的 β_i 和 β_{i+1} 滿足 $g_4 = 1$。可是 (4) 不眞。事實上, $(r,b,R) \notin \delta(q,a)$。

那麼, 如何彌補這項失誤呢? 第一種是改變檢驗窗口。

先考慮像 [4]那樣用四個條件的情況。有兩種基本方法:

(a) 採用兩個窗口。除開由四格 $(i,j-1),(i,j),(i,j+1),(i+1,j)$ 組成的窗口 A 外, 再由四格 $(i,j),(i+1,j-1),(i+1,j),(i+1,j+1)$ 組成另外一個窗口 B。這就是說, 再定義一個邏輯判據 $h(W,X,Y,Z)$ 使得, $h(W,X,Y,Z) = 1$ 若且唯若, 當某個 ID 的第 $j-1, j, j+1$ 格所含的分別是 W,X,Y 時, Z 允許出現在緊排其前的 ID 的第 j 格中。然後定義

$$g_4 = \Bigg(\prod_{(i,j)}\Big(\sum_{f(W,X,Y,Z)=1} (c_{i,j-1,W}\, c_{i,j,X}\, c_{i,j+1,Y}\, c_{i+1,j,Z})\Big)\Bigg) \cdot \Bigg(\prod_{(i,j)}\Big(\sum_{h(W,X,Y,Z)=1} (c_{i+1,j-1,W}\, c_{i+1,j,X}\, c_{i+1,j+1,Y}\, c_{i,j,Z})\Big)\Bigg).$$

這實質上就是分別用兩個窗口檢察那個由 $g[x]$ 的眞賦値所造出的 $\#\beta_0\#\beta_1\cdots\#\beta_{p(n)}$ 是否是條計算道路。難道用兩個窗口就會充分了嗎? 沒錯, 如果它能通得過兩個窗口的檢查, 那麼我們就能證明它是條計算道路。

首先, 讓我們用數學歸納法證明, 每個 β_i 恰含一個複合符號。由條件 (2), β_0 是

M 的初始 ID, 因此它恰有一個複合符號。現在, 假設 β_i 恰有一個複合符號, 不妨說在格 (i,j) 之中, 那麼兩格 $(i+1,j-1)$ 和 $(i+1,j+1)$ 之中, 恰有一個含複合符號; 否則, 無法通過窗口 B 的檢查。β_{i+1} 在其餘之處沒有複合符號; 否則, 無法通過窗口 A 的檢查。

其次, 如果格 (i,j) 所含是複合符號 $q\#a$, 格 $(i+1,j)$ 所含符號是 b, 格 $(i+1,j-1)$ (格$(i+1,j+1)$) 所含複合符號是 $p\#c$, 那麼一定會有 $(p,b,L)\in\delta(q,a)((p,b,R)\in\delta(q,a))$; 否則, 無法通過窗口 B 的檢查。如果格 (i,j) 所含是符號 a, 那麼格 $(i+1,j)$ 所含一定是符號 a 或者複合符號 $p\#a$ 否則, 無法通過窗口 B 的檢查。這意味著, 條件 (4) 已滿足。

(b) 採用一個大窗口。最容易想到的就是把窗口 A 和窗口 B 合併成一個六格組成的大窗口。[6]和 [1]採用的就是這種窗口, 它由六格 $(i,j-1),(i,j),(i,j+1),(i+1,j-1),(i+1,j),(i+1,j+1)$ 所組成。不過, 這不是唯一的可以承擔檢驗責任的六格窗口。還有兩種, 讀者可以自己找找試試。有趣的是, 不存在少於六格的窗口有能力承擔檢驗責任。對每種少於六格的窗口, 舉個反例, 這是個挺好的練習題。

現在, 考慮像 [3]那樣用六個條件的情況。由於有了兩個額外條件, 因此, 我們不再需要證明, 每個 β_i 恰含一個複合符號。這樣一來, 只要通過窗口 B 的檢驗, 條件 (4) 已滿足。這就是說, 四格窗口 B 足有能力承擔檢驗責任。這就是 [2]採用的證明。有趣的是, 有能力承擔檢驗責任四格窗口只有這一種。這證明也是道挺好的的練習題。

第二種是改變圖靈機 (Turing Machine) 的定義。在書 [5]中, 讀寫頭不能同時改變所讀的符號和左右移動; 亦即, 在圖靈機的一次移動中, 讀寫頭若改變所讀符號就不能移動, 若移動就不能改變所讀符號。當採用這種圖靈機定義時, 如果使用六個條件, 那麼四格窗口 A 照樣能承擔責任。有興趣的讀者可以證證看。

第三種是將 NTM 改爲 DTM。這是相當有趣, 而技巧特異的方法, 只有在 [3]中可以看到。它利用 NP 類的一個性質: L 屬於 NP 類, 若且唯若存在一個多項式 q和一 P 類中語言 A, 使得

$$x\in L\Leftrightarrow\exists y(|y|\leq q(|x|)):x\#y\in A.$$

這性質將所討論的接受 L 的 NTM 轉化成了接受 A 的 DTM。由於 DTM 的轉移函數是單值的, 因此, 在採用六條件時, 無論使用四格窗口 A 還是四格窗口 B 都無關緊要。可是, 如果像 [3]那樣選用四格窗口 A, 那麼四個條件就夠了。如果選用四格窗口 B, 那麼四個條件不行。這事實的嚴格證明是非常好的練習題。需要說明, 這種技巧有一定的局限性。例如說, 如果像有些書籍 (例如 [4]) 那樣採用 log-space reduction, 那麼我們就得考慮 log-space NTM。對這種 NTM, 上述轉化就有點問題了。事實上, NTM 不能有足夠的記憶空間把所有猜測都執行完, 非得猜猜, 用用, 塗掉, 再猜不可。

最後指出, [2]是筆者近著。本文內容是該書的一個小部分。那裡匯集了許多筆者多年學習、研究、教書的心得、體會。如果你想

了解更多一些, 歡迎您將來讀原書, 並批評指正。

參考文獻

1. D. -Z. Du and Ker-I Ko, Theory of Computational Complexity, (John Wiley & Sons, New York, 2000).
2. D. -Z. Du and Ker-I Ko, Problem Solving in Automata and Languages, to appear.
3. M. R. Garey and D. S. Johnson, Computers and Intractability, a Guide to the Theory of NP-Computation, (W. H. Freeman, San Francisco, 1979).
4. J. E. Hopcraft and J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, (Addison-Wesley, 1979).
5. H. R. Lewis and C. H. Papadimitriou, Elements of the Theory of Computation (2nd Edition), (Prentical-Hall, 1998).
6. M. Sipser, Introduction to the Theory of Computation, (PWS, 1997).

—本文作者堵丁柱爲明尼蘇達大學計算機科學系教授, 葛可一爲紐約州立大學石溪分校計算機科學系教授—