中文字根的貯存和中文字的合成

謝清俊 杜敏文 戚樹紅
（交通大學工學院）

（1972年12月16日收到）

摘要——在本篇中，主要討論的是中文字根的貯存，由中文字根合成中文字的軟體設計，以及提出一個使合成的字美觀的方法——定字根比重法。

一、簡　　介

現代電子計算機的發明使得歐美的資料處理速率有了驚人的增進（約一百萬倍）。而中文資料處理由於輸入及輸出的種種困難，至今仍不能對計算機加以有效地運用。我們知道，人類能有今天的文明，主要是能够運用積年累月傳留下來的知識。所以那一個國家能更有效地，更快更好地處理資料，那一國便可在未來發展中凌駕於他國之上。處在這種情況下，研究如何使中文資料也能用計算機處理，使我們在這方面超上甚且超過先進的國家，便是我輩不可旁貸的責任了。

中文資料輸入輸出計算機的障碍導源於中國文字是兩向性（two-dimension）的結構，不像西方文字是線性的（linear）且多由僅26個字母組成。目前國外製的中文計算機輸入時每個字都得從上萬個字鍵中找出鍵按下（或者利用選擇鍵，則一個字須按兩下鍵），輸出時則計算機中必須存有每個字的字形資料。若我們可能用到的中文字以10,000個計，每個字用16×16的點矩陣（dot matrix，參考圖一）貯存，則共須 2.56×10^6 筆（bit）的記憶單位。這樣做出來的機器不但輸入資料很不方便，每部機器的造價高昂也是可以想見的。

輸入輸出的方法，由於研究分析字根的結果而有了很大的改進〔1〕。在〔1〕中，8532個中文電腦用字〔2〕（另加 597 個異體字，共 9129 字）分析成了496個字根。（再據統計，用這 496 個字根可組合成 48655 個可能用得到的中文字）。如此輸入鍵盤上的字鍵節省了十八倍以上，鍵盤的製作費自然降低。同時，若每個字根以16×16的點矩陣去貯存，則用以貯存字形的記憶空間（memory space）也將大幅度地減小。本文要討論的，就是如何去貯存字根使能更節省需用的記憶空間，如何由字根組合成字的軟體設計，以及提出一個使合成的字美觀的方法——定字根的比重法。

二、輸入字的結構

分析字根的功用主要是利用文字組成的重複性來解決輸入和輸出的困難〔1〕。每一個字都可分成字根和定位符號的組合（詳見〔1〕）。在〔1〕中分字根時使用的定位符號總共只有三種，即是橫連（⿻），直連（⿱）和包含（⿴）。這三種定位符號由字義即可明瞭其作用。爲了更清楚起見，舉例如下：

例一：

肯＝止⿱月，表示止字根直連月字根形成肯字。

例二：

調＝言⿻（冂⿴(士⿱口)）表示冂字根包含士和口的直連再與言橫連形成調字。

括號在一個組合中的功用是指示那一種定位符號應當先行運作。爲了減少括號的使用次數，我們可規定⿴，⿱，⿻之間有優先次序（Precedence）存在〔1〕，即⿴＞⿱＞⿻。任意兩個相鄰的定

位符號A，B，若A≥B，則A先運作，與算術四則中先乘除後加減方法相同。

除了上述的⿴，⿱，⿻，（，）等符號外，由於同一字根又常會在同一字中成某種組合出現，我們又可增加些方便符號如8，8，∞，∞，等，以期使用者能更加方便。其中小圓圈代表字根的位置。例如，匿字可分成匚⿴口8。在這表示法中方便符號是給予較⿴更高的優先次序〔1〕。

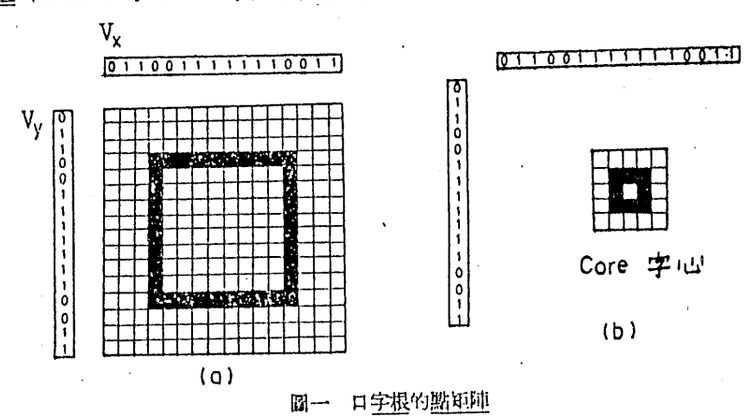
綜合前述，我們知道在用字根作中文資料輸入時，每個字的字式（expression）是由字根，定位符號，括號，和方便符號所組成。實際上⿴可以省略，因爲除了少數特例外，一個字根是否爲包含根可由字根的本身來判別。而⿱和⿻兩者又可省略其中的一個，因爲字根和字根中間必定有定位符號，若是發現少了定位符號，便是省略去的⿱或是⿻了。省略去的定位符號可在使字式標準化時自動加入。

三、字根的貯存

用〔1〕中發展出來的這組字根去做成輸出用的字形，需要的記憶空間可節省到40倍以上。能够節省如此多倍的空間是經由兩個步驟得來。

1. 9,129個字的點矩陣，減成496個字根的點矩陣，記憶空間需要量減少了18倍。〔註*〕
2. 由於字根的結構較字爲簡單，在字根的點矩陣中，橫向和直向都有許多重複處。在貯存時重複的地方便可省略掉。以圖一(a)的口字根點矩陣爲例，第二，三行與第一行雷同，第六至第十二行與第五行雷同等。所以圖一(a)的點矩陣可濃縮成圖一(b)的字心（core）形態，加上 V_x, V_y 兩個表示何處重複的向量即構成口字根的濃縮資料。在需要時以字心照 V_x, V_y 所標示向直向和橫向擴展即可得口的點矩陣原形。據大略估計這樣表示法記憶空間又可節省二到三倍左右。

由1,2兩階段下來，9129 字的字形只需 496×16×16/2≃8 千排（kilo byte, 1byte=8bits）的記憶空間即足。普通小型計算機的主要記憶區（main memory）已足够容納得下。或者用只可讀記憶裝置（read only memory）貯存也所費無多。



圖一　口字根的點矩陣

因爲在字式中省去了包含的定位符號，一個字根是否爲包含根，包含於那個位置，多少大小，也得要存在字根資料中。據分析包含根包含位置可大別爲七大類〔1〕，所以只要指明是那一種即可。此外，爲了使合成的字形美觀起見，還需要指明各字根直向橫向的比重，這點將在第五節中討論。

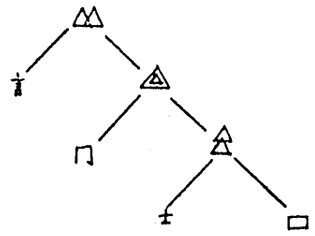
四、結　構　樹

一個字可分析成由字根和定位符號組合起來的標準字式（即是不包含方便符號的字式）。這在實際上就是一個有優先序的文法結構（Precedence Grammar）〔3〕。任何字的構造可用一個結構樹（Parsing Tree）來表示。

〔註*〕：參考資料〔1〕中之中文模式 496 個字根可產生 48,655個中國字，然中文常用字不過四千，計算機若存九千字一般應用可以足够，是故本文僅以9,129個字，做比較之基準，若以48,655字比較則更可再縮小5.33倍。

例三：

圖二就是例二中調字的結構樹。

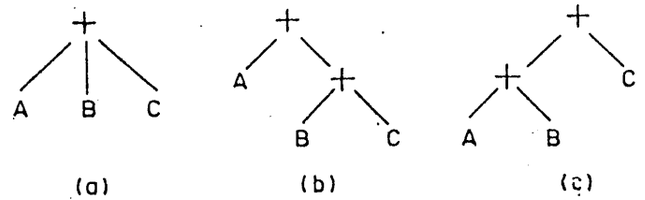


圖二 調字的結構樹

給了一個字的字式，如何去造該字的結構樹可參考〔3〕中提供的方法。在〔3〕中造的是數學算式的結構樹。因爲同是有優先序的文法結構，〔3〕中的方法自然可以引用來造中文字的結構樹。

例四：

給一算式A＋B＋C，它的結構樹可有三種形式，如圖三。

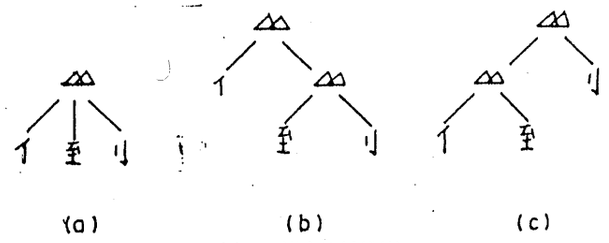


圖三 A＋B＋C的結構樹

例五：

倒字可分爲亻亾至亾刂。

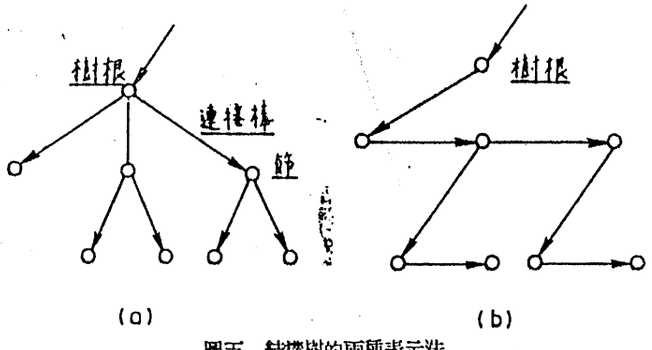
對應於圖三，倒字也有三種結構樹，如圖四。



圖四 倒字的結構樹

於圖四的 (a), (b), (c) 三種結構樹中，我們採取的是 (a)，以便利往後的運算。

一個結構樹是由節 (node) 和連接棒 (link) 組成，這從結構樹的圖上即可看出。每個結構樹

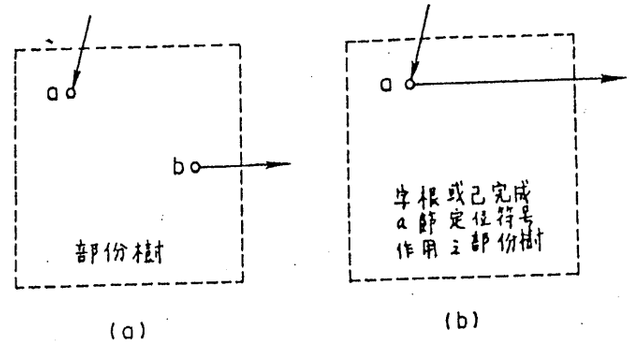


圖五 結構樹的兩種表示法

可有兩種連接法〔4〕，如圖五 (a) (b) 所示。圖中圓圈爲節，節爲節中間的連接棒。事實上 (a) (b) 兩種連接法作用完全一樣，即兩種連接法有一對一的對應〔4〕。

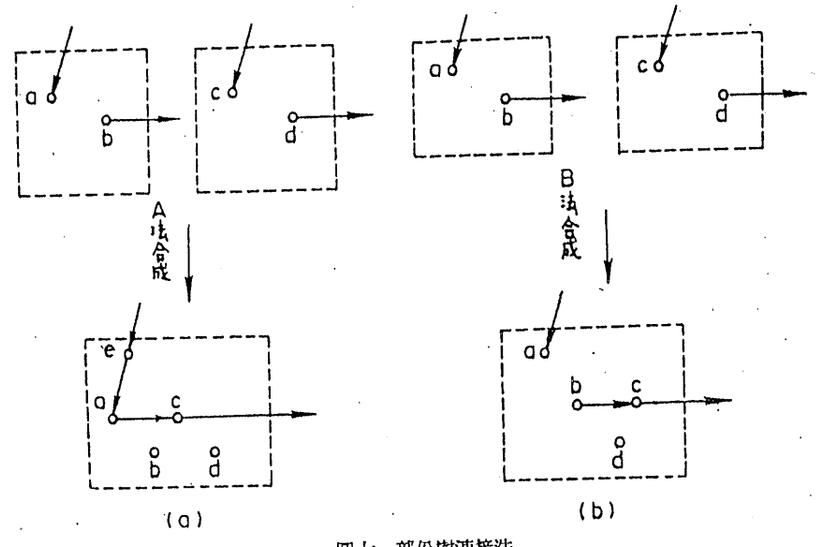
在 (a) 和 (b) 中我們將採取後者，因爲後者每個節向外最多只有兩根連接棒，即向下與向右，處理起來較爲方便。

由於一個字式爲有優先序的文法結構，在從左向右掃過 (Scan) 字式時，有些部份的字式可在掃過過程中造好相對應的部份樹 (Subtree)。這樣的部份樹與整棵樹其他部份連接只能經由兩處，*a* 節(頭)和 *b* 節(尾)，見圖六。圖中當 a 節爲字根或 a 節之定位符號已無另一節 *b* 向右連出時，則 a=b。



圖六 部份樹的頭和尾

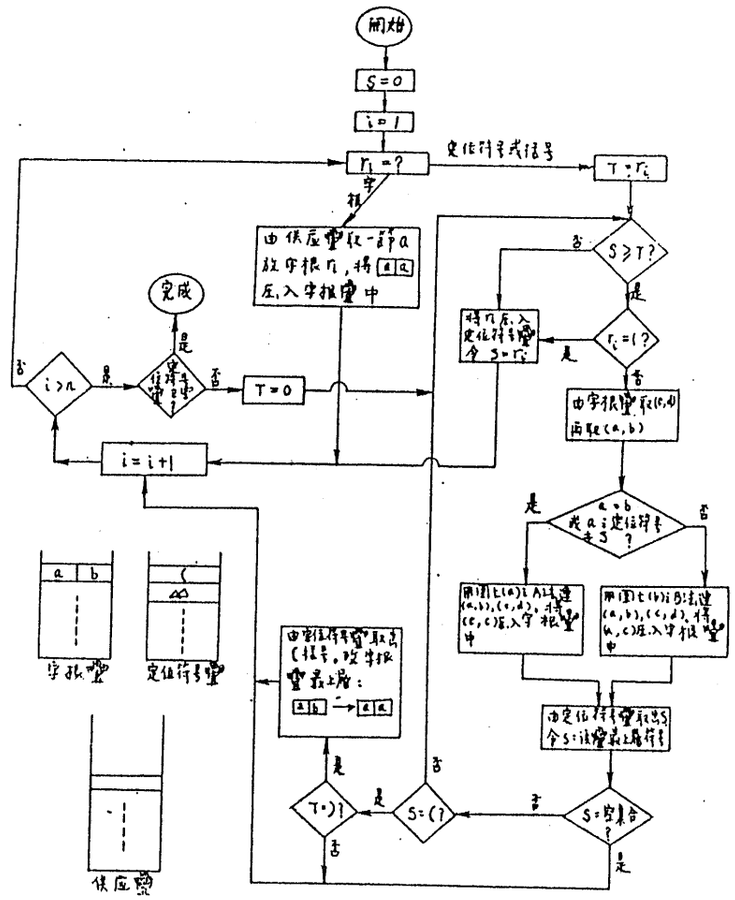
兩個如上述的部份樹 (a, b), (c, d) 若由定位符號P連在一起時可有兩種不同的連法，若是 a=b 或P與 a 的定位符號不等，則由圖七 (a) 的A法，其中 *e* 爲新節。否則用圖七 (b) 的B法，a 的定位符號還可繼續向右作用。



圖七 部份樹連接法

用圖八的流程圖就可由一個字式造出該字的結構樹。整個流程中用了兩個壓下疊 (pushdown store)〔4〕。壓下疊的作用是後放進的物件先取出。兩個疊裏一個存放還未處理的定位符號或括號，叫做定位符號疊，另一個存放字根或已合成的部份樹，叫做字根疊。除壓下疊外另需一供應疊 (available list) 以供應所需用的節。流程開始時假定整個字式爲 $r_1\ r_2 \cdots\cdots r_n$，其中每個 r_i 爲字根或定位符號或括號。定位符號和括號的優先序數則定爲 (=5，▲=4，△=3，亾=2，)=1。

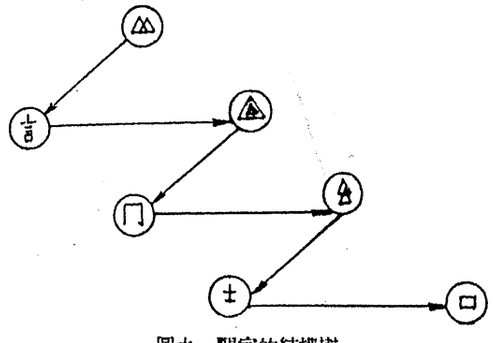
字根疊每一層存的是字根或部份樹的頭和尾（a, b）。



圖八　造結構樹流程圖

例六：

調的字式爲言𠙵冂⿰（士𠙵口）。經由圖八的流程即可造出如圖九的結構樹。



圖九　調字的結構樹

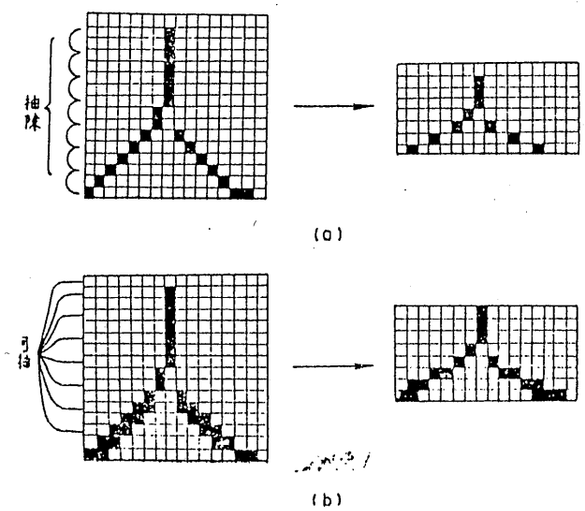
五、字　的　合　成

從一個字式造出結構樹，已完成了合成字的初步工作。下一步即是如何把兩個或多個點矩陣壓成一個點矩陣的問題了。要把一個點矩陣由大壓小，可以採取兩種方法。第一種是抽取法，抽取點

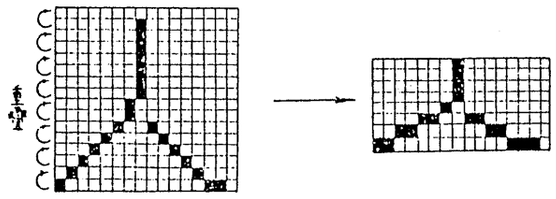
矩陣中的部份行或列。再把剩餘的行或列合併在一起。當然，這樣做應該抽取那幾行或列而使字的形式不致於大變就有了問題。如圖十（a），要把人字壓成一半，每隔一行抽取一行，得到了斜線處已不連的人字。解決這個問題的一個途徑是先費心設計好點矩陣的圖案，再加上直橫兩個向量，如 V_x, V_y 然，以標示何處可抽何處不可抽，如圖十（b）所示。這樣做的壞處是每個字根需要較多的記憶空間來貯存，殊不經濟。

第二種壓縮的方法是疊合法，即是把要縮的行全部重疊在一起。如在圖十一中，要把人字壓扁成一半，我們把相臨的每兩行疊合起來。

由於中文字構造複雜的部份需要疊合成二分之一以下的機會很小，簡單的部份（如口，日等）重疊多次亦不會變形，所以疊合壓縮法當較抽取壓縮法爲適用。

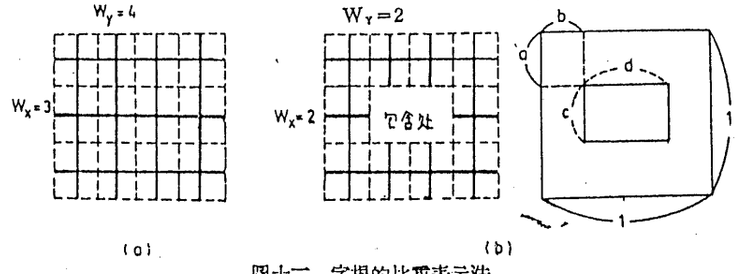


圖十　抽取法壓縮點矩陣



圖十一　疊合法壓縮點矩陣

在進行造字的時候，並不須眞把一個字的各組成部份一步步壓縮而合成字，而只需循結構樹照比例把各字根所佔位置，大小計算出來，再把各字根分別壓縮放進算出的所在即可。這種方法可稱爲虛壓法（Pseudo-compression），免除了一個字根可能同一方向壓縮好幾次的浪費。以下解釋的是字根的位置和大小是如何計算出來的。



圖十二　字根的比重表示法

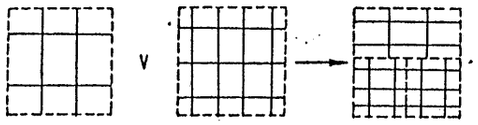
每一個中文字根，我們都給一個長的比重 W_x 和寬的比重 W_y。所以一個字根可以一個方柵來表，如圖十二 (a)，橫的柵數爲 W_x，直的柵數爲 W_y。

圖十二 (b) 則爲包含根的長寬比重和包含處的位置及大小，包含位置和大小可由 (a, b, c, d) 的向量表。

每個字根暫時看成一個可拉長或壓縮的方柵。在拉長或壓縮時直柵間和橫柵間的距離也成比例增大或減小。在由字根合成部份字體時，字根佔據部份字體的大小和合成部份字體的長寬比重是由下面的方法來計算。

設一個部份字體由 k 個部份直連而來，即該部份爲 $L_1 \triangleq L_2 \triangleq \cdots\cdots \triangleq L_k$，其中 L_i 各爲字根或已合成的部份字體，各有長寬比重 W_{xi}, W_{yi}。則每個 L_i 長度爲合成全長的 $\frac{W_{xi}}{\Sigma W_{xj}}$ 倍。而每個 L_i 寬度皆爲合成後的寬度。合成後部份字體長比重爲 ΣW_{xj}，寬比重爲 $MAX\{W_{yj}\}$。

圖十三說明兩個長寬比重各爲 (2, 3) (3, 4) 的方柵合成了一個長寬比重爲 (5, 4) 的方柵。在合成後各橫柵仍然均勻分佈在整個方塊中。



圖十三 方柵的合成

若部份字體爲 $L_1 \triangleq L_2 \triangleq \cdots\cdots \triangleq L_k$，則每個 L_i 長度皆與合成字體長度等，寬度爲全寬的 $\frac{W_{yi}}{\Sigma W_{yj}}$。合成後部份字體長比重爲 $MAX(W_{xj})$，寬比重爲 ΣW_{yj}。

若部份字體由包含而成，即 $L_1 \triangleq L_2$, L_1 參考圖十二 (b)，則 L_1 長寬與合成字長寬等。L_2 長佔合成字體長 a 倍，寬佔 b 倍。合成後長比重爲 $W_{x1}+W_{x2}$，寬比重爲 $W_{y1}+W_{y2}$。

定字根或合成字體長寬比重的目的在求合成後的字筆劃均勻。一個字根的長寬比重大致上可定爲該字根直向和橫向疊合壓縮至仍不影響字的外形時橫的行數和直的行數。

結構樹上的一個部份樹代表的部份字體在一個點矩陣中佔的位置和大小可以四個數 (A, B, C, D) 表示。(A, B) 表此部份字體的左上角原點坐標，從頂往下第A列，從左往右第B行。C表長度，D表寬度。

設有一個直連的節 $L_1 \triangleq L_2 \triangleq \cdots\cdots \triangleq L_k$ 它的位置和大小爲 (A, B, C, D)，各個 L_i 長寬比重爲 W_{xi}, W_{yi} 則 L_i 的位置和大小 (A_i, B_i, C_i, D_i) 可計算如下：

$$A_1 = A$$

$$A_i = A + \sum_{1}^{i-1} C_j \quad \text{當 } 1 < i \leq k \cdots\cdots (V1)$$

$$B_i = B \cdots\cdots (V2)$$

$$C_i = \left[\frac{W_{xi}}{\Sigma W_{xj}} \times C\right] \text{ 當 } i < k$$

$$= C - \sum_{1}^{i-1} C_j \quad \text{當 } i = k \cdots\cdots (V3)$$

$$D_i = D \cdots\cdots (V4)$$

(V3)式中方括號表最接近括號內數的整數。

若節爲橫連 $= L_1 \triangleq L_2 \triangleq \cdots\cdots \triangleq L_k$，則 (A_i, B_i, C_i, D_i) 爲

$$A_i = A \cdots\cdots (H1)$$

$$B_1 = B$$

$$B_i = B + \sum_{1}^{i-1} D_j \quad \text{當 } 1 < i \leq k \cdots\cdots (H2)$$

$$C_i = C \cdots\cdots (H3)$$

$$D_i = \left[\frac{W_{yi}}{\Sigma W_{yj}} \times D\right] \text{ 當 } i < k$$

$$= D - \sum_{1}^{i-1} D_j \quad \text{當 } i = k \cdots\cdots (H4)$$

若一個節是包含，$L_1 \triangleq L_2$，則需先查出 L_1 屬於那種包含，得到如圖十二 (b) 的資料。L_1 和 L_2 的位置和大小可計算如下：

$$A_1 = A \cdots\cdots (C11)$$

$$B_1 = B \cdots\cdots (C12)$$

$$C_1 = C \cdots\cdots (C13)$$

$$D_1 = D \cdots\cdots (C14)$$

$$A_2 = A + \lceil a \times C \rceil \cdots\cdots (C21)$$

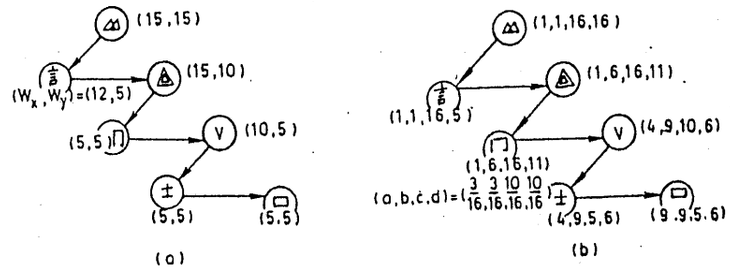
$$B_2 = B + \lceil b \times C \rceil \cdots\cdots (C22)$$

$$C_2 = \lfloor c \times C \rfloor \cdots\cdots (C23)$$

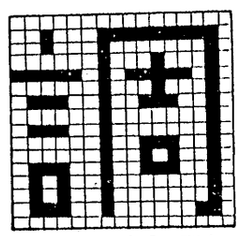
$$D_2 = \lfloor d \times D \rfloor \cdots\cdots (C24)$$

這個計算可先循結構樹由下層往上照先處理部份樹 (endorder traversal)〔4〕的次序依前述求直橫比重法求出各節的直橫比重。再由結構樹最上層算起，令最高一節 (即樹根 root) 位置和大小爲 (1, 1, 16, 16)。然後輪迴地照先處理節 (Preorder traversal)〔4〕的次序，用 V, H 或 C 的公式計算每個節的位置和大小，每個字根的位置和大小最後便可得出。

例七：調字的結構樹每個節的直橫比重由圖十四 (a) 算出。依這種比重，每個節的位置和大小即可算出如圖十四 (b)。圖十五的調字形即是用算出的位置和大小把各字根放入而得到的。



圖十四 調字字根位置和大小的計算



圖十五 合成的調字

整個由一個字的字式造出該字的字形的步驟爲：

1. 化字式爲標準形式，即加入省去的定位符號。除去&等的方便符號。
2. 造出字的結構樹。
3. 提取字根的資料。

4. 算出各字根的位置和大小。
5. 放入字根。
6. 完成字形。

六、討　　論

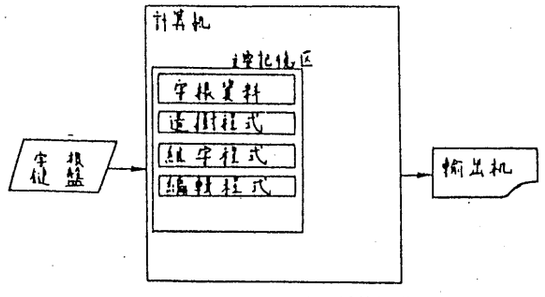
用由〔1〕發展出的這組字根來做中文的輸出字形使所需記憶空間節省了四十倍以上。用這組字根來做資料輸入的鍵盤字鍵數也可省到十八倍以上。其實分析中文字成字根在早期已有許多人做過研究〔詳見5〕，但都因種種困難而捨棄使用。用字根來造字最大的困難在於造出的字外形不易美觀，甚至於完全變形而不能辨認。這是由於一般由字根合成字時沒有考慮到每個字中筆劃的勻稱是中文字的一個特點。中文字不論字中筆劃多寡，大多均勻地分佈在同樣大的一個方塊裏，如此才會有四平八穩的感覺。在本文中，我們提出的字根加比重就是爲了解決這個問題。由於定筆劃簡單的字根比重小，複雜的比重大，合成字的筆劃將均勻地分佈在全方塊中，免除了頭重脚輕，左肥右瘦，腹大中空等等弊病。我們除了正在用計算機進行試驗外，用手試的一些字，由字根合成與直接點出的字形外觀上大都非常接近。

文中提出的字心貯存字根法，使貯存字根用的記憶空間與點矩陣一邊邊長成正比，而非與其面積成正比。原因是許多字根的字心對大的點矩陣和小的點矩陣完全相同。例如口，在用32×32的點矩陣貯存時，只須把 V_x, V_y 擴張到32單位長即可。

雖然在一個字式中大部份的包含和所有橫連(或直連)定位符號可以省略去，有少許的包含根則必須在其後加▲符號表示其確爲包含。例如戶字根，又可作包含根，如在房字裏，又可作非包含根，如在啓字裏。如字式資料爲由鍵盤輸入，則這類包含根必須特別處理以免有作用不明確 (ambiguity) 的情況發生。解決這個問題的一個方法是把這些可兩用包含根的字鍵定位在一起，用異色標出。打鍵者便可注意使用。如房字中，戶爲包含根，整個字式應是戶▲方，包含符號▲應隨著戶送入。

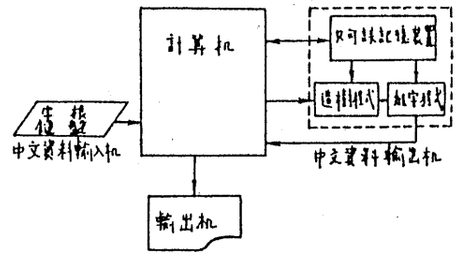
由於貯存字形所需記憶空間的大幅度減少，本文討論的字根資料至少可由以下三種方法來使用。

1, 在一計算機要做中文輸出時，可把字根資料放進主要記憶區中，再需一個造樹程式使字的字式標準化及造出結構樹，一個組字程式提取字根資料並循結構樹合成字，及一個編輯程式做整個裁剪工作，便可從計算機輸出中文資料，如圖十六所示。



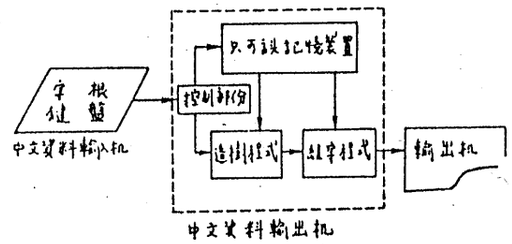
圖十六　字根資料第一種使用法

2. 如圖十七，整個字根資料和一些使字式標準化的資料可放於只可讀記憶裝置中。造樹程式和組字程式因都是固定程式，可由硬體 (Hardware) 直接製造。直接製造的好處，第一是整個中文資料處理用硬體構成處理速率可以大增，而且形成單獨的個體可連接於任何計算機上。



圖十七　字根資料第二種使用法

3. 在圖十七中的鍵盤若直接連到中文資料輸出機，就構成一個中文打字機了，如圖十八。



圖十八　字根資料第三種使用法

上述第一種字根使用法中，在造一個字形時，須先用計算機計算每個字根的位置和大小，再算 $16\times16=256$ 個數，看每個是否爲 0 或 1（即點矩陣每個點白或黑）。

由於一個字平均約 1.9個字根〔1〕，算字根位置和大小所費計算時間很少，整個造字需用時間幾乎即爲求256個數的時間。以每數平均使用40個機器指令 (machine instruction)，每個指令費時 10^{-6}秒計，一秒鐘約可造 100 個字。這種速率在需要大量資料輸出時可能還嫌不够快。所以在第二及第三種使用法裏，把字根資料提出後放在一個兩向的傳移記憶器 (Shift register) 中，依該字根 V_x, V_y 兩向量作所需的壓縮或擴張的運作，整個造字形程序便可在不到 10^{-5} 秒內完成。足够來做大量資料輸出了。

誌　　謝

本文所討論的字根點矩陣 (24×24) 及其對應的數字碼 (Code) 經袁正春、李正芳、張慧雲、周翠萍、祁德慧諸位小姐協助製成。對整個研究工作進行有莫大的幫助，謹此致謝。

參 考 資 料

1. 謝清俊、黃永文、林　樹，"中文字根之分析" 見交大學刊本期。
2. 林　樹，中文電腦基本用字研究，國立交通大學科技研究報告CC-601。
3. Wegner P., *Programming languages, information structures and machine organization*, McGraw-Hill, New York, U.S.A., 1968
4. Knuth, D. E., *The art of computer programming I: Fundamental Algorithms*, Addison-Wesley, Mass., 1969.
5. 倪　耿，中國字之結構模式及其分析，碩士論文，國立交通大學工學院電子研究所，1972.