第三章　程序代數(Process Algebra)之建構步驟[1]

程序代數源於 R.Milner 之 CCS(Calculus of Communicating Systems)[1]，其構想是希望能把並行程式(concurrent programs)中的程序抽象化為代數的表現形式，然後賦予嚴謹的代數結構。一旦抽象化完成之後，其最大的野心是希望能夠運用代數的數學工具，從程序代數中求解，瞭解並行程序同步與互動之後的複雜行為。使用代數方法研究程序的優點為：1.可使用常數、變數以及結合它們的函數(operator)表示程序規格(process specification)及行為(process behavior)．2.所有數學邏輯可為我們使用。

欲把程序代數化，首先將程序(process)會用到的最基本元件抽象化，例如程序最基本的要素為 action 以及選擇、循序結合運算子，以 a、b、c 等代表不同的 action 稱為常數，以+、・分別表示選擇運算子、循序結合運算子稱為函數，然後根據程序運作的模式建立一群公設。接著訂定一套規格(Σ,E)，Σ包含所有抽象化基本元件之集合，E 為所有公設之集合。因為函數運算必需具有封閉性(closure)，因此Σ中常數與函數運算會產生許多式子，我們稱之為項(term)，剛開始我們並未將變數包含在Σ中，因此產生的項(term)都不含變數稱之為 closed 項，往後進一步的應用時，Σ包含變數，因此產生的項(term)若含變數稱之為 open 項。

由(Σ,E)規格，我們可以制定一個代數(algebra)如(T,Op,C)，其中 T 為所有Σ產生的closed項之集合、Op為算子的集合、C為所有常數的集合，並使得(T,Op,C)□E。(T,Op,C) 並沒有定義任何等價關係，惟一能將 T 分割成等價類的是 E。分割完成後，我們可以把一個等價類視為項(term)，也就是一個元素。(T,Op,C)為(Σ,E)之 initial algebra，E 為(T,Op,C)之 complete axiomation。然後我們可以從這

[1] 本章內容摘自[2]

個代數發展出許多程序模型(model)，使得每一公設對程序模型之運作均為正確無誤，且不會使不應相等的程序變成相等。因(Σ,E)與(T,Op,C)極為相像，我們將(T,Op,C)看成是(Σ,E)的實作，所以本章我們著重在(Σ,E)討論與擴充。

為了有系統的介紹程序代數理論，首先我們介紹所有程序模型都適用的核心代數—基本程序代數(Basic Process Algebra 簡稱 BPA)。我們以 BPA 為基礎核心結構找出一組完整的代數公設。以 BPA 為基礎出發，每當加入一性質，即對應地增加新的公設，來界定及描述此新性質之行為。例如，假設我們要加入死結(deadlock)性質到代數結構中。令死結的代表符號為δ。我們則必須新增一條與死結有關的代數公設：如 $x+\delta=x$。此公設表示只要有別的選擇可進行，死結就不會發生。另外，每當加入一運算子(operator)，即利用對此運算子所新增的公設，將其展開。例如，假設公設中還沒有加入"$\between$"的運算子(本章第五節說明$\between$與$\lfloor\!\lfloor$之意義)。加入"$\between$"後，新增公設為

1. $x \between y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x$
2. $ax \lfloor\!\lfloor y = a(x \between y)$

所以利用 1、2 可將 ab$\between$c 展開成 a(b.c+c.b)+cab 消除$\between$，只有+、・)，使其相等於模型內已存在運算子之表示式(expression)。如此一來新的運算子即可以原模型內之運算子來表示，使得原有的公設系統因為新的運算子加入而不需要進行更改。不過值得注意的是要考慮新的運算子加入是否會產生新的性質，以及運算結果是否符合封閉性(closure)。例如在沒有 deadlock δ 的代數結構中，若加入空程序(empty process)ε，因而必需新增

$$\varepsilon \lfloor\!\lfloor x = \delta$$

這樣的公設。此公設產生δ，所以又需對δ性質加以界定。以此方式我們就可以很容易的將 BPA 規格擴充，以適應各種不同的應用需求。

因我們要將規格應用在程序方面，所以在架構一個規格時，我們需考慮四種基本組成：

1. 規格之公設

2. 基本項形成方式(definition and formations of basic terms)

3. 每個常數的 action relation(常數 action 的語意)

4. initial algebra

以下數節即以這四點描述代數結構。

因我們所要研究的對象為程序，我們給予規格中每一元素實際意義為：

- 常數代表不可分割之行動(action)，稱為原子 action
- +代表選擇算子：x + y 表示只可執行 x、y 兩者之一
- ・代表循序算子：x・y 表示 x 執行完成 y 才可開始執行

基本項代表程序，且・算子優先於+算子，如果沒有語意問題時通常都省略・及小括號。

本章的第一小節從基本程序代數(Basic Process Algebra)談起，然後以基本程序代數為基礎，逐漸增加新的運算子，常數（如 deadlock δ）。每增加一個新的運算子或常數，我們就建構了一個新的代數。針對每一個代數，我們以一個小節來描述它。每個小節，我們都依序探討此代數的上述四個基本組成。

第一節　基本程序代數(Basic Process Algebra)

本節 BPA=($\sum_{BPA}$, E_{BPA})，稱之為基本程序代數系統。BPA 實際上為一代數結構(也就是規格)，$\sum_{BPA}$ 為此結構中使用到的任何符號，包含常數、函數(operator)，常數記成 a、b、c 等，至於要多少常數視應用而定，我們用 A 來表示所有常數的集合，兩個二元算子，+及・。E_{BPA} 包含此代數結構所有公設的集合。

1. BPA 的公設

E_{BPA} 包含表 3-1 之五個公設。若忽略 A1、A2 出現的次序，A3 至 A5 可將基本項化成唯一形式(unique norm form)的 TRS(Term Rewrite System)。若一組公設

系統可以把基本項化成唯一的形式，則一個等價類就可以以一個基本項作代表[2]。

A1	x + y = y + x
A2	(x + y) + z = x + (y + z)
A3	x + x = x
A4	(x + y)z = xz + yz
A5	(xy)z = x(yz)

表 3-1　BPA 公設

依照+、・之語意這些公設應該都很明顯為真，其中 A4 乘對加左分配成立，但 process algebra 通常不包含 x(y + z)=xy + xz 這樣的公設，因左右兩式在選擇 y 或 z 的的那一刻是不同的，也就是分支結構(branching structure)不同。目前我們在架構通用的規格通常不加入此公設，將來也許在架構不同的應用模型時可自由加入。現舉一例說明其差異：door 表示升降機門打開、risk 表示危險升降機、safe 表示安全升降機，door・risk + door・safe 表示當你按下按鈕呼叫升降機時，可能來一台升降機接著門打開，你看不到另一台與已來的這一台有何差異，而 door(risk + safe)表示兩台同時到且門打開，你可同時看到兩台裡面(或許一台有警示燈)，所以你可依判斷選一台去乘。

2. BPA 之基本項(basic term)

BPA 中之基本項(basic term)定義如下：

- 每一個常數為一基本項
- 若 t 為一基本項，且 a 為一常數，則 a・t 為一基本項
- 若 t、s 為基本項，則 t + s 為基本項

公設中之變數被取代為任意基本項均正確。由基本項定義可推出任何基本項 p 皆可表示為下列形式：

$p=\Sigma a_i t_i+\Sigma b_j$　a_i, b_j 為常數，t_i 為基本項，i<n,j<k　(*)

[2]若此 TRS 可以把基本項化成唯一的形式，表示此公設系統比較嚴謹，不會推導出 critical pairs

任何 closed term 經由 A4 (x+y)z=xz+yz、A5 (xy)z=x(yz)都可化成基本項，也就是(*)形式，這也是我們要定義基本項的原因之一。另一個原因為如果我們要證明某些敘述(statement)對 closed term 是正確的，我們只需證明這個敘述對基本項定義的三種形式均為正確即可，這種證明的方式稱*結構性歸納證明(structural induction)*。

3. BPA 的 action relation

對於 BPA 之 action relation，就是 operational semantics，我們定義一個二元關係 $\xrightarrow{a}$、一個單元關係 $\xrightarrow{a}\surd$，$\forall a \in A$ 意義如下：

$$a \xrightarrow{a} \surd$$

$$x \xrightarrow{a} x' \Rightarrow x + y \xrightarrow{a} x' \text{ and } y + x \xrightarrow{a} x'$$

$$x \xrightarrow{a} \surd \Rightarrow x + y \xrightarrow{a} \surd \text{ and } y + x \xrightarrow{a} \surd$$

$$x \xrightarrow{a} x' \Rightarrow xy \xrightarrow{a} x'y$$

$$x \xrightarrow{a} \surd \Rightarrow x\,y \xrightarrow{a} y$$

表 3-2　BPA 的 Action relation

- $t \xrightarrow{a} s$ 表示作一步 a action 由狀態 t 轉移到狀態 s，也就是 t process 執行 a action 變成 s process；
- $t \xrightarrow{a} \surd$ 表示 t 執行 a action 可以成功的結束，$\surd$ 表示成功的結束(CCS 沒有成功的結束，結束以 0 表示)。

我們也以 $\xrightarrow{\sigma}$ 表示 generalized action，σ 表示一連串來自 A 的 action。$t \xrightarrow{\sigma} s$ 表示 t 經執行一連串的 actions 可達 s，$t \xrightarrow{\sigma} \surd$ 表示 t 經執行一連串的 actions 可以成功的結束，不論是 $t \xrightarrow{\sigma} s$ 或 $t \xrightarrow{\sigma} \surd$，$\sigma$ 都稱為 t 的一個 trace。

表 3-2、表 3-3 列出 BPA 所有 action relation 及 generalized action。

$$t \xrightarrow{a} s \quad \Rightarrow \quad t \xrightarrow{a}\!\!\rightarrow s$$

$$t \xrightarrow{\sigma}\!\!\rightarrow s \text{ and } s \xrightarrow{\rho}\!\!\rightarrow r \quad \Rightarrow \quad t \xrightarrow{\sigma\rho}\!\!\rightarrow r$$

$$t \xrightarrow{a} \surd \quad \Rightarrow \quad t \xrightarrow{a}\!\!\rightarrow \surd$$

$$t \xrightarrow{\sigma}\!\!\rightarrow s \text{ and } \xrightarrow{\rho}\!\!\rightarrow \surd \quad \Rightarrow \quad t \xrightarrow{\sigma\rho}\!\!\rightarrow \surd$$

表 3-3　Generalized relation

有了 action relation 就可定義 summand 及 BPA-summand 如下：

i. 如果 s 為 t 之 summand 則∃ r ∋ A1 及 A2 □ t = s + r 或 A1,A2 □ t = s

ii. 如果 BPA □ t = s + t 則我們稱 s 為 t 之 BPA-summand 記成 s≤t

從 i，ii 定義可直接有下面兩個結果：

i. $t \xrightarrow{a} s \Leftrightarrow$ t 有一 summand **as**

ii. $t \xrightarrow{a} \surd \Leftrightarrow$ t 有一 summand **a**

4. Initial algebra

我們說兩個 closed term s 和 t 是等價的記成 $s \equiv_{BPA} t$，它的意義為 BPA 的公設可以導出 s=t，也就是說 $s \equiv_{BPA} t$　if BPA □ s = t。若 BPA 之 initial algebra 為 A, A 由 BPA 所有 closed term 的等價類所組成, 每一等價類都包含一個基本項, 例如(a+b)c 與 ac+bc 在同一等價類(用 A4 公設)、a(b+c) 與 ab+ac 不屬於同一等價類。此等價關係具有 congruence 性質，因 $s \equiv_{BPA} s'$(s，s'同一等價類)、$t \equiv_{BPA} t'$(t，t'同一等價類)則 BPA□s=s'、BPA□t=t' 所以 BPA □ s + t = s' + t'且 BPA □ s・t = s'・t'。

若 p,q∈A，p、q 均為等價類，它的 action relation 如下：

i. $p \xrightarrow{a} q \Leftrightarrow \exists$一個 term $s \in p$，$\exists$一個 term $t \in q$ ∋ $s \xrightarrow{a} t$

ii. 同樣 $p \xrightarrow{a} \surd$，$p \xrightarrow{\sigma} \rightarrow q$ 與 $p \xrightarrow{\sigma} \rightarrow \surd$都與上式同。

第二節　Deadlock 加入 BPA

考慮 x・y 這個程序，只有在 x 成功完成後，y 才有機會執行。若在某種情況下，執行 x 時發生 deadlock，一般的情形都不允許 y 繼續執行，所以我們有必要將成功完成與未能成功完成區分開來。δ 加入 BPA 後我們以 BPA_δ 表示。

1. BPA_δ 的公設

δ 的加入需增加的公設列於表 3-4，BPA_δ 的公設等於 BPA+A6-A7，A6 表示只要有別的選擇可進行，死結就不會發生。A7 只要發生 deadlock 任何行為都不可能繼續執行。

A6	$x + \delta = x$
A7	$\delta x = \delta$

表 3-4　因應δ加入增加的公設

2. BPA_δ 之基本項(basic term)

BPA_δ 中之基本項(basic term)如下：

・每一個常數及δ均為一基本項

・若 t 為一基本項，且 a 為一常數，則 a・t 為一基本項

・若 t、s 為基本項，則 t + s 為基本項

由基本項定義可推出在 BPA_δ 中任意 closed term p 及任何基本項皆可表示為下列形式：

$p=\Sigma a_i t_i + \Sigma b_j$　a_i, b_j 為常數，t_i 為基本項，i<n, j<k，若 i<0 且 j<0 即為δ

3. BPA_δ 的 action relation

δ 為一特殊常數，且$\delta \notin A$，所以 action relation 與 BPA 沒有差別如表 3-2。

4. BPA_δ的 Initial algebra

BPA_δ的 initial algebra 記成A_δ，它比 BPA 之 Initial algebra 多了一些等價類，這些等價類中每一項都與δ有關。

第三節　空程序(Empty process)加入 BPA

空程序以ε表示，與δ相反，意義為成功結束，加入 BPA 後我們以 BPAε表示。ε會改變程序的行為方式，例如 a +ε ≠ a，因為 a +ε既可執行 a 也可立刻成功結束。

1. BPAε的公設

A8	$\varepsilon x = x$
A9	$x\varepsilon = x$

表 3-5　因應ε加入增加的公設

ε的加入需增加的公設列於表 3-5，BPAε的公設等於 BPA+A8-A9，A8、A9 表示ε對・就像單位元素(identity)。

2. BPAε之基本項(basic term)

BPAε中之基本項(basic term)如下：

・ε為一基本項

・若 t 為一基本項，且 a 為一常數，則 a・t 為一基本項

・若 t、s 為基本項，則 t + s 為基本項

由基本項定義可推出，常數亦為一基本項：由第二條 t=ε可得，在 BPAε中任意 closed term p 及任何基本項皆可表示為下列形式：

p=$\Sigma a_i t_i + \Sigma\varepsilon = \Sigma a_i t_i + \varepsilon$　a 為常數，t_i為基本項，i<n

3. BPAε的 action relation

$$a \xrightarrow{a} \varepsilon$$

$$x \xrightarrow{a} x' \Rightarrow x + y \xrightarrow{a} x' \text{ and } y + x \xrightarrow{a} x'$$

$$x \xrightarrow{a} x' \Rightarrow xy \xrightarrow{a} x'y$$

$$x\downarrow \text{ and } y \xrightarrow{a} y' \Rightarrow xy \xrightarrow{a} y'$$

$$\varepsilon\downarrow$$

$$x\downarrow \Rightarrow (x + y)\downarrow \text{ and } (y + x)\downarrow$$

$$x\downarrow \text{ and } y\downarrow \Rightarrow (xy)\downarrow$$

表 3-6　BPAε之 action relation

ε為一特殊常數，且$\varepsilon \notin A$，在 BPA 中已有√表示成功結束，但√不是 BPA 常數只用來表達 action relation，現在有ε就可以取代√，所以 BPAε的 action relation 需改寫如表 3-6。同時增加一符號↓表示程序有ε為其子項(summand)，例如 p=abc+ε則 p↓為真。(p↓表示 p 有ε子項)

因ε加入 $t \xrightarrow{a} \surd$ ⇔ t 有一 summand a 改為 t↓ ⇔ t 有一 summand ε

4. BPAε的 Initial Algebra

BPAε的 initial algebra 記成$A\varepsilon$，它比 BPA 之 Initial algebra 多了是由ε組成的等價類以及ε為子項的等價類，這個等價類包含所有經公設 A8 及 A9 可化成基本項ε的 closed term。

第四節　δ與ε加入 BPA

當δ與ε同時加入 BPA 時我們記成 $BPA_{\delta\varepsilon}$，$BPA_{\delta\varepsilon}$的公設等於 BPA+A6-A9。$BPA_{\delta\varepsilon}$中之基本項(basic term)如下：

・δ與ε均為基本項

・若 t 為一基本項，且 a 為一常數，則 a・t 為一基本項

・若 t、s 為基本項，則 t + s 為基本項

基本項一般表示式如 BPAε，action relation 則與 BPAε同，$BPA_{\delta\varepsilon}$的 initial

algebra 記成 $A_{\delta\varepsilon}$，$A_{\delta\varepsilon}$ 比 $A\varepsilon$ 多一個 δ 等價類，因此 $A_{\delta\varepsilon}=[\delta]\cup A\varepsilon = A_{\delta}\cup A\varepsilon$，同樣的對任何屬於 $BPA_{\delta\varepsilon}$ 之 closed term 都可以使用 $BPA_{\delta\varepsilon}$ 的公設化成一個基本項。

第五節　並行式程序代數(Algebra of Concurrent Processes)

為了以程序代數模擬並行式程序，必需引入新的二元函數 ∥，稱為合併(merge)算子，程序 x、y 經合併成新的程序 x∥y，其意義為程序 x 的原子 action 與程序 y 的原子 action 可以交錯(interleaved)且 handshaking 的執行。

因為想循序漸進的延伸 BPA，所以在本節我們只處理單純的交錯執行，並沒有 handshaking 情況發生，下一節再介紹含入 handshaking 的情況。為能正確的敘述 ∥的執行我們作下列的假設：

- 原子 action(atomic action)的執行沒有持續時間
- 兩個原子 action 不能同時開始執行
- 不涉及時間

在這些假設下，a∥b(a,b∈A)表示 a 先執行再輪到 b 或 b 先執行再輪到 a，因此

$$a \between b = ab + ba。$$

為了能使∥很容易展開，我們需要一個輔助算子 ||_ ，它的意義與∥相同，只是必需先執行左邊第一個原子 action，所以 a∥b = a||_ b + b||_ a= ab + ba，其餘的例子參考表 3-7 及表 3-8：

(ab)∥c = (ab)||_ c + c||_ (ab) =a(b∥c) + cab =a(b||_ c+c||_ b) + cab =a(bc +cb) + cab

表 3-7 ∥例 1

(a + b)∥c = (a + b)||_ c + c||_ (a + b) =ac + bc + c(a + b)

表 3-8 ∥例 2

現在我們開始介紹 PA(Process Algebra)的規格，PA 包含・、+、〃、||_ 四種運算子及常數集合 A，A 中元素以 a,b,c,…表示。

1. PA 的公設

PA 有表 3-9 的九個公設，公設 M2、M3 之 a，可為 A 之任一元素，我們稱具有如此性質的公設為公設 schemes。公設 A1-5 為 BPA 公設，M1-4 為因應合併算子而增加的公設，M1 定義〃算子以輔助算子 ||_ 展開的情形，M2-4 定義不同型式的左邊程序執行第一步後的情形。我們並規定算子運作的先後次序為・> 〃、||_ >+，〃、||_ 同時出現時由左至右運算。

A1	$x + y = y + x$
A2	$(x + y) + z = x + (y + z)$
A3	$x + x = x$
A4	$(x + y)z = xz + yz$
A5	$(xy)z = x(yz)$
M1	$x \between y = x \between\!\!\!\_\ y + y \between\!\!\!\_\ x$
M2	$a \between\!\!\!\_\ x = ax$
M3	$ax \between\!\!\!\_\ y = a(x \between y)$
M4	$(x + y) \between\!\!\!\_\ z = x \between\!\!\!\_\ z + y \between\!\!\!\_\ z$

表 3-9　PA 的公設

2. PA 之基本項(basic term)

PA 的基本項就是 BPA 的基本項，因(1).對任何一個 PA 的 closed term 經不斷的由公設 M1 至 M4 的取代，會將含有〃與 ||_ 的 closed term 化成只含・或+的 closed term。(2).對於不含〃與 ||_ 的 PA 的兩 closed term s,t 若 PA□s=t 則 BPA □s=t，原因是不含〃與 ||_ 兩 closed term 的相等都經由相同公設 A3 至 A5 的取代來證明的。由(1).及(2).可知 PA 的基本項就是 BPA 的基本項。

3. PA 的 action relation

對於新增加的運算子，必需定義其運算的語意，表 3-1 最後四條定義有關〃與 ||_ 的運算的語意，其餘均與 BPA 相同。

$$a \xrightarrow{a} \surd$$

$$x \xrightarrow{a} x' \Rightarrow x + y \xrightarrow{a} x' \text{ and } y + x \xrightarrow{a} x'$$

$$x \xrightarrow{a} \surd \Rightarrow x + y \xrightarrow{a} \surd \text{ and } y + x \xrightarrow{a} \surd$$

$$x \xrightarrow{a} x' \Rightarrow xy \xrightarrow{a} x'y$$

$$x \xrightarrow{a} \surd \Rightarrow x\,y \xrightarrow{a} y$$

$$x \xrightarrow{a} x' \Rightarrow x /\!/ y \xrightarrow{a} x' /\!/ y \text{ and } y /\!/ x \xrightarrow{a} y /\!/ x'$$

$$x \xrightarrow{a} \surd \Rightarrow x /\!/ y \xrightarrow{a} y \text{ and } y /\!/ x \xrightarrow{a} y$$

$$x \xrightarrow{a} x' \Rightarrow x \between y \xrightarrow{a} x' /\!/ y$$

$$x \xrightarrow{a} \surd \Rightarrow x \between y \xrightarrow{a} y$$

表 3-10 Action relation for PA

4. PA 的 Initial Algebra

由 2. PA 之基本項可知，對 PA 的 closed term 而言都可經公設 M1-M4 化成只含・與+的 closed term，也就是 BPA 的 closed term，而 BPA 的 closed term 都可化成 BPA 基本項之形式，所以 PA 的 Initial algebra 與 BPA 的 Initial algebra 是一樣的，唯一的差別在 PA 的等價類比 BPA 的等價類包含更多的原元素，這一點可作以下推論得知，因 PA 多了〃與 ||_，所以 closed term 除了・和+外尚有〃與 ||_ 可用，但如前所述可利用公設化成只含・與+的等價 closed term，例如表 3-7 (ab)〃c 與 a(bc +cb) + cab 在同一個等價類中，相對於 BPA 而言 a(bc +cb) + cab 這個等價類中多了(ab)〃c 表示的方法，也就是說對 closed term 而言 PA 比 BPA 擁有較多的表示方法(expressive)。有更多的表示方法的好處為：在 PA 以一個等式就可以表達的程序規格，在 BPA 要表達同樣的程序規格可能就需許多等式。

第六節　δ與ε加入 PA

PA 中加入δ比較簡單，僅需新增有關δ的公設如將表 3-4，因此 PA = PA_δ + A6-A7，另外更改 PA 之 M2、M3 公設 schemes 的範圍，原先 $a\in A$ 改為 $a\in A\cup\{\delta\}$。Action relation 與 PA 一樣。根據上節討論 PA_δ的基本項就是 BPA_δ的基本項，initial algebra 也與 BPA_δ一樣，對 closed term PA_δ比 BPA_δ擁有較多的表示方法。

至於空程序ε的加入就有一點複雜了。首先我們討論ε ||_ x，ε ||_ x 的第一步需執行ε，但ε不是一個可執行的行為，所以我們主張ε ||_ x=δ。根據 x // y = x ||_ y + y ||_ x 以及ε ||_ x=δ可推論出ε // ε = ε ||_ ε + ε ||_ ε=δ+δ=δ，這個結果ε // ε =δ又與我們的直觀不合，因為既然左右兩邊都可成功結束，merge 後也應該成功結束，也就是我們認為ε // ε = ε較合理，為了解決這個問題引入一個單元算子$\surd$，$\surd$(x)指出程序 x 是否有成功結束的選項，若有成功結束的選項$\surd$(x)=ε否則為δ。

1. PAε的公設

空程序ε的加入自然產生δ，所以在 PA 加入ε時就必需同時包含δ以及δ的公設。PAε的公設如表 3-11，前 9 個公設與 $BPA_{\delta\varepsilon}$相同，為了使ε // ε = ε成立而改變 x // y = x ||_ y + y ||_ x 成為 x // y = x ||_ y + y ||_ x+$\surd$(x)・$\surd$(y)，PA 的 M2 a ||_ x=ax 改變為 TM2 ε ||_ x=δ，a ||_ x=ax 可由 A9 與 TM3 推出，新增 TE1-TE4 為定義$\surd$算子。

A1	$x+y=y+x$	A6	$x+\delta=x$
A2	$(x+y)+z=x+(y+z)$	A7	$\delta x=\delta$
A3	$x+x=x$	A8	$\varepsilon x=x$
A4	$(x+y)z=xz+yz$	A9	$x\varepsilon=x$
A5	$(xy)z=x(yz)$		
TM1	$x\between y=x\lfloor y+y\lfloor x+\surd(x)\cdot\surd(y)$	TM3	$ax\lfloor y=a(x\between y)$
TM2	$\varepsilon\lfloor x=\delta$	TM4	$(x+y)\lfloor z=x\lfloor z+y\lfloor z$
TE1	$\surd(\varepsilon)=\varepsilon$	TE3	$\surd(x+y)=\surd(x)+\surd(y)$
TE2	$\surd(a)=\delta$	TE4	$\surd(xy)=\surd(x)\cdot\surd(y)$

表 3-11　PAε的公設

2. PAε 之基本項(basic term)

PAε 的基本項就是 BPA$_{\delta\varepsilon}$ 的基本項，理由請看 PA 之基本項。

3. PAε 的 action relation

PAε 的 action relation 與表 3-6 BPAε 的 action relation 是一樣的，因δ不屬於可執行的行為，故不影響 action relation。

4. PAε 的 Initial Algebra

因為每一個 PAε 的 closed term 都可經由 PAε 的公設化成為 BPA$_{\delta\varepsilon}$ 的基本項，同樣的對 closed term PAε 比 BPA$_{\delta\varepsilon}$ 擁有較多的表示方法。

第七節　傳輸式程序代數(Algebra of Communicating Process)

模擬程序間的傳輸(communication)是並行式理論重要課題之一，在本節我們討論的是同步傳輸(synchronous communication)，就是程序間的傳輸發生在同時執行相對應的 actions，例如一個程序執行「送」action，另一個程序執行「收」action，兩個 actions 同時發生在同樣的通道(port 或 channel)上，謂之產生同步傳輸。

讓 A 代表所有原子 actions 的集合, 在 A 上定義一個傳輸函數(communication function) f，f 為一個部分二元函數，滿足下列兩個條件：

1. $\forall$ a,b$\in$A ：f(a,b)=f(b,a) 傳輸函數可交換。
2. $\forall$ a,b,c$\in$A ：f(f(a,b),c)=f(a,f(b,c)) 傳輸函數可結合。

若 f(a,b)未定義或 f 之參數不屬於 A，就不能發生傳輸行為。f 定義為二元函數表

示只允許二元傳輸，也就是只允許 handshaking，f(a,b,c)則未定義。

一般都有標準符號表示程序間的傳輸，例如圖 1：A、B、C 表示程序，1、2、3、4 表示傳輸通道，2、3 為內部(internal)傳輸通道，1、4 為外部(external)傳輸通道，如 A 在通道 2「送」資料 d 以 $s_2(d)$ 表示，如 B 在通道 2「收」資料 d 以 $r_2(d)$ 表示，結果為傳輸 d 在通道 2 以 $c_2(d)$表示，用傳輸函數表示為 $f(s_2(d), r_2(d))= c_2(d)$。

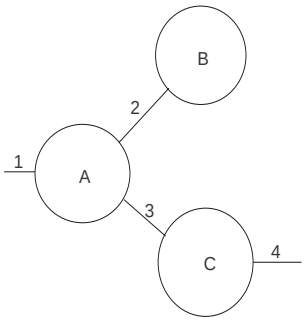

圖 3-1　程序間的傳輸

接下來我們以代數方式來模擬傳輸行為，稱為 ACP(Algebra of Communicating Process)，在第五節討論合併(merge)算子時，只討論交錯執行，並沒有傳輸情況發生，現在加入一個新的二元運算子來模擬傳輸行為，我們把它記成 | 稱為 communication merge。所以合併算子 // 除了 PA 所定義的兩種情況外，新增一種情況為傳輸，因此 $x // y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x | y$。

x | y 表示程序 x、y 的第一步必需是傳輸行為，若它們的第一步不是傳輸行為，我們定義為 δ，所以 ACP 包含 δ。根據前面數節的討論，ACP 就是 BPA_δ 的延伸。

ACP 的規格包含以下元素：

- 一群常數的集合 A 及定義在 A 上的一個部分二元函數 f，f 具有交換及結合兩種性質
- 一個特別常數 δ 代表死結但 $\delta \notin A$
- 五種二元運算分別為・、+、//、$\lfloor\!\lfloor$ 及 |。

1. ACP 的公設

A1	$x + y = y + x$	A5	$(xy)z = x(yz)$
A2	$(x + y) + z = x + (y + z)$	A6	$x + \delta = x$
A3	$x + x = x$	A7	$\delta x = \delta$
A4	$(x + y)z = xz + yz$		
CF1	$a \mid b = f(a,b)$ 若 f(a,b)已定義	CF2	$a \mid b = \delta$ 若 f(a,b)未定義
CM1	$x \between y = x \between y + y \between x + x \mid y$	CM6	$a \mid bx = (a \mid b) \cdot x$
CM2	$a \between x = ax$	CM7	$ax \mid by = (a \mid b) \cdot (x \between y)$
CM3	$ax \between y = a(x \between y)$	CM8	$(x + y) \mid z = x \mid z + y \mid z$
CM4	$(x + y) \between z = x \between z + y \between z$	CM9	$x \mid (y + z) = x \mid y + x \mid z$
CM5	$ax \mid b = (a \mid b) \cdot x$		

表 3-12　ACP 的公設

ACP 的公設列於表 3-12，A1-A7 為 BPA_δ 的公設，CF1 定義 ∣，因 $\delta \notin A$ 且 f 是定義在 A 上的所以 δ 不可以是 f 之參數，因此 CF2 已暗示 $\forall a \in A, a \mid \delta = \delta \mid a = \delta$，CM1 為 PA 的 M1 加上本節討論的傳輸行為，CM2-4 為 PA 的 M2-4，CM5-9 為因應 ∣ 之加入而新增的公設。最後如果只允許 handshaking 也可以另加一條公設 $a \mid b \mid c = \delta$。

2. ACP 之基本項(basic term)

ACP 的基本項就是 BPA_δ 的基本項，每一個 ACP 的 closed term 都可經由公設化成不含 ⫽、⌊ 及 ∣ 之 BPA_δ 基本項。

3. ACP 的 action relation

首先七個規則來自表 3-10 為 PA 的 action relation，最後三個規則定義傳輸行為的 action relation。

$$a \xrightarrow{a} \surd$$

$$x \xrightarrow{a} x' \Rightarrow x + y \xrightarrow{a} x' \text{ and } y + x \xrightarrow{a} x'$$

$$x \xrightarrow{a} \surd \Rightarrow x + y \xrightarrow{a} \surd \text{ and } y + x \xrightarrow{a} \surd$$

$$x \xrightarrow{a} x' \Rightarrow xy \xrightarrow{a} x'y$$

$$x \xrightarrow{a} \surd \Rightarrow x\,y \xrightarrow{a} y$$

$$x \xrightarrow{a} x' \Rightarrow x /\!\!/ y \xrightarrow{a} x' /\!\!/ y,\ y /\!\!/ x \xrightarrow{a} y /\!\!/ x' \text{ and } x \between y \xrightarrow{a} x' /\!\!/ y$$

$$x \xrightarrow{a} \surd \Rightarrow x /\!\!/ y \xrightarrow{a} y,\ y /\!\!/ x \xrightarrow{a} y \text{ and } x \between y \xrightarrow{a} y$$

$$x \xrightarrow{a} x', y \xrightarrow{b} y' \text{ and } f(a,b) = c \Rightarrow x /\!\!/ y \xrightarrow{c} x' /\!\!/ y', x \mid y \xrightarrow{c} x' /\!\!/ y'$$

$$x \xrightarrow{a} x', y \xrightarrow{b} \surd \text{ and } f(a,b) = c \Rightarrow x /\!\!/ y \xrightarrow{c} x', x \mid y \xrightarrow{c} x',$$
$$y /\!\!/ x \xrightarrow{c} x' \text{ and } y \mid x \xrightarrow{c} x'$$

$$x \xrightarrow{a} \surd, y \xrightarrow{b} \surd \text{ and } f(a,b) = c \Rightarrow x /\!\!/ y \xrightarrow{c} \surd, x \mid y \xrightarrow{c} \surd$$

表 3-13　ACP 的 action relation

4. ACP 的 Initial Algebra

因 ACP 的基本項可以化成 BPA_δ 的基本項，所以 ACP 的 Initial Algebra 即為 BPA_δ 的 Initial Algebra, 差異在於 ACP 的每一個等價類包含較多的 closed term。

第八節　ε 加入 ACP

當 ε 加入 ACP 時我們記成 ACPε。在 PAε 我們已討論過 ε 加入時應增加的公設及對 action relation 的影響，因 ACP 比 PA 只多了傳輸算子，所以必需新增有關 ε 與 $\mid$ 運算的公設，但傳輸的行為只發生在兩個原子 action，因此公設 $\varepsilon \mid x = \delta$、$x \mid \varepsilon = \delta$ 就非常容易瞭解了，ACPε 的公設列於表 3-14。

A1	$x + y = y + x$	A6	$x + \delta = x$
A2	$(x + y) + z = x + (y + z)$	A7	$\delta x = \delta$
A3	$x + x = x$	A8	$\varepsilon x = x$
A4	$(x + y)z = xz + yz$	A9	$x\varepsilon = x$
A5	$(xy)z = x(yz)$		
CTM1	$x \between y = x \between y + y \between x + x \mid y + \surd(x) \cdot \surd(y)$		
TM2	$\varepsilon \between x = \delta$	TM5	$\varepsilon \mid x = \delta$
TM3	$ax \between y = a(x \between y)$	TM6	$x \mid \varepsilon = \delta$
TM4	$(x + y) \between z = x \between z + yvz$	CM7	$ax \mid by = (a \mid b)(x \between y)$
CF1	$a \mid b = f(a,b)$ 若 f(a,b)已定義	CM8	$(x + y) \mid z = x \mid z + y \mid z$
CF2	$a \mid b = \delta$ 若 f(a,b)未定義	CM9	$x \mid (y + z) = x \mid y + x \mid z$
CM5	$ax \mid b = (a \mid b) \cdot x$		
CM6	$a \mid bx = (a \mid b) \cdot x$		
TE1	$\surd(\varepsilon) = \varepsilon$	TE3	$\surd(x + y) = \surd(x) + \surd(y)$
TE2	$\surd(a) = \delta$	TE4	$\surd(xy) = \surd(x) \cdot \surd(y)$

表 3-14　ACPε 的公設

至於 ACPε 的基本項則與 BPA$_{\delta\varepsilon}$ 基本項相同, 因為對於 ACPε 中任何一個 closed term 都可以經由 ACPε 的公設將它化成 BPA$_{\delta\varepsilon}$ 中的基本項。

ACPε 的 action relation 因ε加入而擁有成功結束，因此必需將 ACP 的$\surd$符號改為ε，ACPε 的 action relation 列於表 3-15：

$a \xrightarrow{a} \varepsilon$

$\varepsilon \downarrow$

$x \xrightarrow{a} x' \Rightarrow x + y \xrightarrow{a} x'$, $y + x \xrightarrow{a} x'$ and $xy \xrightarrow{a} x'y$

$x \downarrow$ and $y \xrightarrow{a} y' \Rightarrow xy \xrightarrow{a} y'$

$x \xrightarrow{a} x' \Rightarrow x \between y \xrightarrow{a} x' \between y$, $y \between x \xrightarrow{a} y \between x'$ and $x \between y \xrightarrow{a} x' \between y$

$x \xrightarrow{a} x'$, $y \xrightarrow{b} y'$ and $f(a,b) = c \Rightarrow x \between y \xrightarrow{c} x'$, $x \mid y \xrightarrow{c} x'$, $y \between x \xrightarrow{c} x'$ and $y \mid x \xrightarrow{c} x'$

$x \downarrow$ and $y \downarrow \Rightarrow (xy) \downarrow$ and $(x \between y) \downarrow$

$x \downarrow \Rightarrow \surd(x) \downarrow$

表 3-15　ACPε的 action relation

ACPε的 initial algebra 與 BPA$_{\delta\varepsilon}$相同，差異在於 ACPε的每一個等價類包含較多的 closed term。

第九節　τ程序代數(abstract process algebra)

首先我們介紹一個新的函數叫重新命名(renaming function)，令 A 為所有原子 action 之集合，C 為特別常數的集合，例如 C 包含δ、ε等，一個重新命名函數 f，定義為

f：A$\rightarrow$A$\cup$C

它將程序中每一個 a$\in$A 發生的地方取代為 f(a)，因我們只考慮重新命名到特別常數 C，因此將其操作的語意列於表 3-16，其中 a$\in$A、$\gamma\in$C、f。g 表示兩個重新命名函數的合成函數。

RN0	$f(\gamma) = \gamma$	C 中的特別常數不允許重新命名
RN1	$f(x + y) = f(x) + f(y)$	
RN2	$f(xy) = f(x) \cdot f(y)$	
RN3	$f_{id}(x) = x$	f_{id}為 identity 函數
RN4	$f(g(x)) = f_{\circ}\ g(x)$	

表 3-16　重新命名函數

在實際應用時常常使用更簡單的符號來表示重新命名函數，如 $t \in C$、$H \subseteq A$ 我們以 $t_H(a)$ 表示如果 $a \in H$ 時 $t_H(a)=t$，否則 $t_H(a)=a$。

重新命名在實際應用方面有其存在的價值，假如在某一個環境下因某些因素不能執行某種 action 時，我們只需將每個程序使用重新命名函數取代不能執行的 action 為 δ，即可模擬此環境下執行程序的情形。例如某一環境不能執行 c，執行程序 a(b+c) 。我們可以用一個重新命名函數 $\delta_{\{c\}}(x)$ 把 x 中的 c 帶換成 δ。其過程如下：

$\delta_{\{c\}}(a(b+c))=\delta_{\{c\}}(a)\cdot\delta_{\{c\}}(b+c)=a(\delta_{\{c\}}(b)+\delta_{\{c\}}(c))=a(b+\delta)=ab$。

使用重新命名函數時需考慮是否原程序保有的性質，在使用重新命名函數後仍然存在。例如程序 $i\cdot\delta+a$ 有 deadlock 性質，使用重新命名函數 $\varepsilon_{\{i\}}$，將 i 變成空程序，我們得到 $\varepsilon_{\{i\}}(i\cdot\delta+a)=\varepsilon\cdot\delta+a=\delta+a=a$，程序 a 變成沒有 deadlock 性質。

抽象化(abstraction)實際上就是一種重新命名函數，它將程序內某些內在 action 抽象化成為程序外不可觀察的、安靜的 action，一般在程序代數的文獻上都把它記成 τ，τ 與 ε 最大的差異在 τ 不像 ε 能任意的移除。例如程序 $a+b\delta$，若將 b 抽象化後程序變成 $a+\tau\cdot\delta$，δ 之前的 τ 就不可隨意移除，若移除則抽象化前的程序有 δ 的性質，抽象化後的程序沒有 δ 的性質，又如程序 abcd，假設 c 為程序內在 action，將 c 抽象化為 τ，程序變成 $ab\tau d$，此時 τ 可移除並不影響外界的觀察。

τ 還可以解釋為具有搶先(preemptive)執行的特性，例如 $\tau a+b$，一當程序開始執行時就到達 a 狀態，所以我們有 $\tau a+b \neq a+b$、$\forall b \in A$，也就是 $\tau a \neq a$。因為 τ 的

特殊性，因此τ的移除與否端視τ所在的兩個狀態，若移除τ，也就是將此兩個狀態視為同一狀態時，是否同時也減少了原先可執行的 action，若如此τ就不可移除，否則τ可移除。例如 aτ，在觀察 a 執行後過一會兒就安全的結束，在移除τ後成為 a，移除τ之前與之後的狀態並沒有不同，所以 aτ = a。又如圖 3-2，左右兩邊的程序經相同的步驟都到達 P 狀態，從 P 開始右程序經由不可觀察的τ可以執行 y 或 z，同樣的左程序也是可以執行 y 或 z，所以對一個觀察者而言，並不能區分左右程序的不同，因此我們可以將圖 3-2 歸納成一個公式為 x(τ(y+z)+y)=x(y+z)。經上面的討論，我們將有關τ的行為形成表 3-17τ的兩個公設

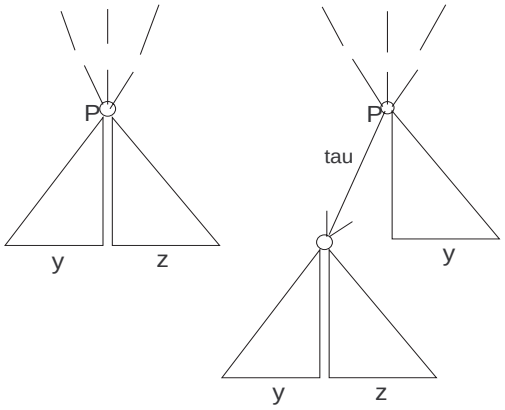

圖 3-2　τ公設的圖示說明

瞭解抽象化即為重新命名函數以後，現在我們定義抽象化算子τ_I，I 為某些內部 action 的集合，τ_I將 I 中每一個 action 轉換成τ，τ_I的公設如表 3-18

B1　xτ　= x

B2　x(τ(y+z)+y) = x(y+z)

表 3-17　τ 的兩個公設

TI1	$\tau_I(a) = a \quad \text{if } a \notin I$
TI2	$\tau_I(a) = \tau \quad \text{if } a \in I$
TI3	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$
TI4	$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$

表 3-18 抽象化算子 τ_I

現在我們將 τ 這個特別常數加到程序代數中，加到 BPA 或 BPA_δ 中幾乎是一樣的，我們討論加到 BPA_δ 中，以便下一節順利的延伸到 ACP^τ。當 τ 加到 BPA_δ 中我們記成 BPA_δ^τ。

BPA_δ^τ 的公設為 BPA_δ 公設再加上表 3-17 及表 3-18，它的 action relation 如表 3-2 就是 BPA 的 action relation(δ 加入時不影響 action relation)，不過 $a \in A \cup \{\tau\}$。BPA_δ^τ 的基本項定義如下：

- τ 及 δ 都是基本項
- 若 t 為一基本項，且 $a \in A \cup \{\tau\}$，則 a・t 為一基本項
- 若 t、s 為基本項，則 t + s 為基本項

從基本項的定義可看出每一個基本項 t 都可寫成下列形式

$t = \tau$ 或 $t = \sum_{i<n} a_i t_i$。

同時每一個 BPA_δ^τ 的 closed term 都可經由公設化成基本項的形式。

至於 BPA_δ^τ 的 initial algebra 則比 BPA_δ 大許多, 因為許多情況 τ 並不能移除, 因此每一等價類中的元素不但比 BPA_δ 多，且比 BPA_δ 有更多的等價類。

第十節　τ 加入 ACP

當 τ 加到 ACP 中我們記成 ACP^{τ}。從第七節的討論我們知道 ACP 是 BPA_{δ} 的延伸，且 ACP 的 closed term 都可以化成 BPA_{δ} 的基本項，同樣的 ACP^{τ} 也就是 BPA_{δ}^{τ} 的延伸，因為它們都只是多加了特殊常數 τ。

ACP^{τ} 的 signature 分類如下：

・常數	a、b、c、d….	以 A 集合表示
・特殊常數	δ、τ	$\delta \notin A$ 且 $\tau \notin A$
・二元算子	+	選舉算子
	・	循序組合算子
	//	並行組合算子
	⌊⌊	左 merge 算子
	|	傳輸組合算子
・單元算子	τ_I	抽象化算子，$I \subseteq A$

1. ACP^{τ} 的公設

ACP^{τ} 的公設為 ACP 公設加上表 3-17 τ 的兩個公設及表 3-18 抽象化算子 τ_I 的公設，現在重新整理在表 3-19 中，其中 a、b、$c \in A \cup \{\delta, \tau\}$，$I \subseteq A$，x、y、z 代表任意程序。因為 $\tau \notin A$，所以我們有 $\tau \mid a = a \mid \tau = \delta, \forall\ a \in A \cup \{\delta, \tau\}$。

2. ACP^{τ} 的基本項

ACP^{τ} 的基本項與 closed term 都可經由公設化成 BPA_{δ}^{τ} 的基本項形式，所以 ACP^{τ} 的基本項與 BPA_{δ}^{τ} 的基本項是一樣的。

3. ACP^{τ} 的 action relation

ACP^{τ} 的 action relation 與 ACP 的 action relation 是一樣的，只是

$a \in A \cup \{\tau\}$。

A1	$x + y = y + x$	B1	$x\tau = x$
A2	$(x + y) + z = x + (y + z)$	B2	$x(\tau(y+z)+y) = x(y+z)$
A3	$x + x = x$		
A4	$(x + y)z = xz + yz$		
A5	$(xy)z = x(yz)$		
A6	$x + \delta = x$	CF1	$a \triangleleft b = f(a,b)$ 若 f(a,b)已定義
A7	$\delta x = \delta$	CF2	$a \triangleleft b = \delta$ 若 f(a,b)未定義
CM1	$x \between y = x \between y + y \between x + x \triangleleft y$	CM5	$ax \triangleleft b = (a \triangleleft b) \cdot x$
CM2	$a \between x = ax$	CM6	$a \triangleleft bx = (a \triangleleft b) \cdot x$
CM3	$ax \between y = a(x \between y)$	CM7	$ax \triangleleft by = (a \triangleleft b) \cdot (x \between y)$
CM4	$(x + y) \between z = x \between z + y \between z$	CM8	$(x + y) \triangleleft z = x \triangleleft z + y \triangleleft z$
		CM9	$x \triangleleft (y + z) = x \triangleleft y + x \triangleleft z$
TI1	$\tau_I(a) = a$ if $a \notin I$	TI3	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$
TI2	$\tau_I(a) = \tau$ if $a \in I$	TI4	$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$

表 3-19　ACP^{τ}的公設

4. ACP^{τ}的 initial algebra

ACP^{τ}的 initial algebra 與 BPA^{τ}_{δ}的 initial algebra 是一樣的記成 A^{τ}_{δ}，而 BPA^{τ}_{δ}的 initial algebra 比 BPA_{δ}多一個等價類$[\tau]$。所以 $A^{\tau}_{\delta}=[\tau]\cup A_{\delta}$。

第十一節　ε加入 ACP^{τ}

ACP^{τ}_{ε}是由第八節的 $ACP\varepsilon$以及第十節的 ACP^{τ}組合形成的，它的 signature 是以 ACP^{τ}為基礎，再加入一個特殊常數ε以及一個能偵測是否能成功結束的單元算子$\surd$。

1. ACP^{τ}_{ε}的公設

A1	$x + y = y + x$	A6	$x + \delta = x$
A2	$(x + y) + z = x + (y + z)$	A7	$\delta x = \delta$
A3	$x + x = x$	A8	$\varepsilon x = x$
A4	$(x + y)z = xz + yz$	A9	$x\varepsilon = x$
A5	$(xy)z = x(yz)$	BE	$a(\tau(x + y) + x) = a(x + y)$
CTM1	$x \between y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x \mid y + \surd(x)\surd(y)$		
TM2	$\varepsilon \lfloor\!\lfloor x = \delta$	TM5	$\varepsilon \mid x = \delta$
TM3	$ax \lfloor\!\lfloor y = a(x \between y)$	TM6	$x \mid \varepsilon = \delta$
TM4	$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$		
CM7	$ax \mid by = (a \mid b) \cdot (x \between y)$	CF1	$a \mid b = f(a,b)$ 若 f(a,b)已定義
CM8	$(x + y) \mid z = x \mid z + y \mid z$	CF2	$a \mid b = \delta$ 若 f(a,b)未定義
CM9	$x \mid (y + z) = x \mid y + x \mid z$	TI0	$\tau_I(\varepsilon) = \varepsilon$
TI1	$\tau_I(a) = a$ if $a \notin I$	TI3	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$
TI2	$\tau_I(a) = \tau$ if $a \in I$	TI4	$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$
TE1	$\surd(\varepsilon) = \varepsilon$	TE3	$\surd(x + y) = \surd(x) + \surd(y)$
TE2	$\surd(a) = \delta$	TE4	$\surd(xy) = \surd(x) \cdot \surd(y)$

表 3-20　ACP_ε^τ 的公設

ACP_ε^τ 的公設如表 3-20，表 3-20 沒有表 3-17 τ 的第一個公設--$x\tau = x$，如果包含 $x\tau = x$，令 $x = \varepsilon$ 配合使用 A8 產生 $\tau = \varepsilon\ \tau = \varepsilon$，這個結果不是我們希望的，所以捨去第一個 τ 公設，雖然沒有第一個 τ 公設還是可以推出 $a\tau = a$(見下文)。至於第二個 τ 公設— $x(\tau(y+z)+y) = x(y+z)$，如果令 $x = \varepsilon$、$y = \delta$ 左邊產生 $\varepsilon(\tau(\delta+z)+\delta)) = \tau(\delta+z)+\delta) = \tau z$，右邊產生 $\varepsilon(\delta+z) = \delta+z = z$，結果 $\tau z = z$ 也非我們希望的，因此我們更改第二個 τ 公設為 $a(\tau(x + y) + x) = a(x + y), a \in A \cup \{\delta, \tau\}$，$a \neq \varepsilon$ 就不會產生我們不希望的結果。由更改後的第二個 τ 公設，令 $x = \delta$、$y = \varepsilon$ 我們立刻可得到 $a\tau = a$。

2. ACP_ε^τ 的基本項

從第八節我們知道 ACP_ε 與 $BPA_{\delta\varepsilon}$ 有相同的基本項，所以 ACP_ε^τ 和 $BPA_{\delta\varepsilon}^\tau$ 應該也有相同的的基本項，或者說 $BPA_{\delta\varepsilon}^\tau$ 的基本項可以表示 ACP_ε^τ 的任何一個 closed term。前面數節未曾討論過 $BPA_{\delta\varepsilon}^\tau$，所以需要定義 $BPA_{\delta\varepsilon}^\tau$ 的基本項，$BPA_{\delta\varepsilon}^\tau$ 的基本項定義如下：

・ε 及 δ 都是基本項

・若 t 為一基本項，且 $a \in A \cup \{\tau\}$，則 a・t 為一基本項

・若 t、s 為基本項，則 t + s 為基本項

由基本項的定義可推出每一個基本項都可以寫成下列形式：

$t=\varepsilon$ 或 $t = \sum_{i<n} a_i t_i$ $a_i \in A \cup \{\tau\}$, t_i 為基本項 $i < n$。

3. ACP_ε^τ 的 action relation

因 ACP_ε^τ 和 $BPA_{\delta\varepsilon}^\tau$ 有相同的的基本項，而 δ 不影響 action relation 的運作，所以 $BPA_{\delta\varepsilon}^\tau$ 的 action relation 與 BPA_ε 的 action relation 是相同的如表 3-6，僅需將 a 的範圍擴大含 τ，即 $a \in A \cup \{\tau\}$。

4. ACP_ε^τ 的 initial algebra

對於 ACP_ε^τ 的任何一個 closed term 都可經由 ACP_ε^τ 的公設化為 $BPA_{\delta\varepsilon}^\tau$ 的基本項，所以 ACP_ε^τ 的 initial algebra 與 $BPA_{\delta\varepsilon}^\tau$ 的 initial algebra 是相同的，我們把它記成 $A_{\delta\varepsilon}^\tau$，它們之間的差異在 ACP_ε^τ 中項(term)有較多的表示方式。

第十二節　各種 Initial Algebra 之關係

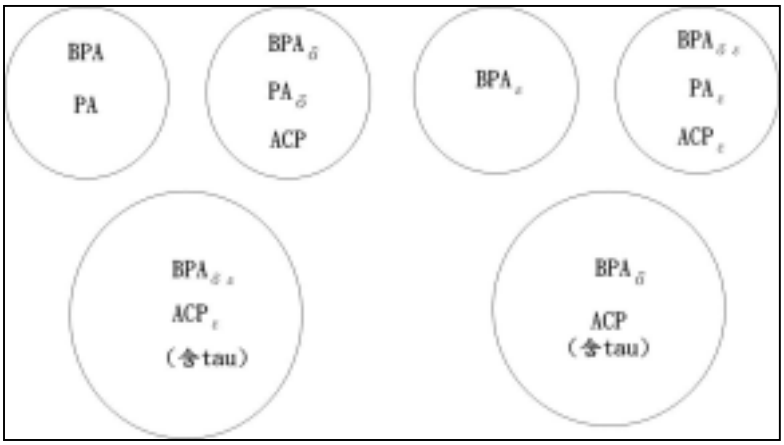

圖 3-3 Initial Algebra 之關係

圖 3-3 說明每一種代數規格之 Initial Algebra 的關係。第一列第二個圓圈裡面有 ACP、PA_δ 及 BPA_δ，表示這三種規格的 Initial Algebra 都一樣，也就是說它們的等價類一樣的個數，不過 ACP 比 PA_δ 有更多表示 closed term 的方式，同樣 PA_δ 比 BPA_δ 有更多表示 closed term 的方式。其他的圓圈也是如此。

第十三節　其他有用的公設

下面列出一些有用的公設，這些公設稱為 *standard concurreny* 公設：

1. $x|y = y|x$
2. $x /\!/ y = y /\!/ x$
3. $(x|y)|z = x|(y|z)$
4. $(x \between y) \between z = x \between (y /\!/ z)$
5. $x|(y \between z) = (x|y) \between z$
6. $x /\!/ (y /\!/ z) = (x /\!/ y) /\!/ z$

另外還有一個展開定理如下：

$$x_1 /\!/ x_2 \cdots /\!/ x_n = \sum_{1\le i\le n} x_i \between (\mathop{\Vert}_{1\le j\le n, j\ne i} x_j) + \sum_{1\le i\le j\le n} (x_i \mid x_j) \between (\mathop{\Vert}_{1\le k\le n, k\ne i,j} x_k) \quad n\ge 3$$

第十四節　程序代數規格與程序代數之關係

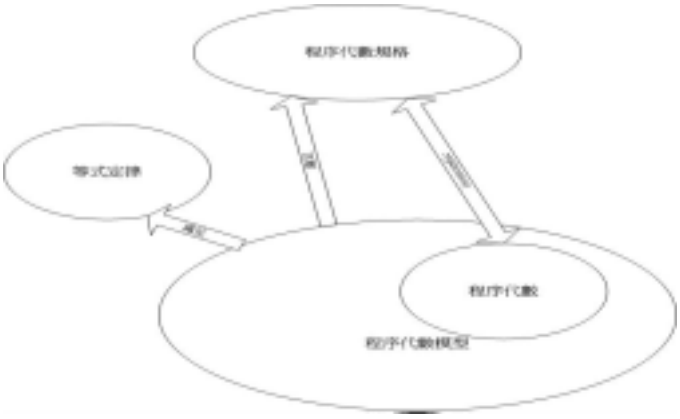

圖 3-4 程序代數規格與程序代數之關係

圖 3-4 說明程序代數規格與程序代數(closed term 組成)之間是同構的(isomorphic)，程序代數與程序代數模型都滿足程序代數規格，但程序代數模型另外會滿足一些定律，這些定律程序代數規格的公設導不出來。