第四章　程序代數(Process Algebra)的模型(Model)[1]

在第三章我們所討論的程序代數其適用的對象為 closed term，closed term 在程序代數中被定義為有限程序，有限程序不能完全的代表實際應用方面的程序。例如考慮一個簡單自動販賣機(V)，假設只有投十元(d)、出咖啡(c)兩個行為，如果以有限程序表示為 V=d・c，但自動販賣機遞出咖啡後又回到初始狀態，因此以 V=d・c 並不能正確表示自動販賣機能回到初始狀態，若以 V=d・c・V 即能正確的表達。

為能更正確的模擬程序，我們必需擴充程序代數規格的 signature，使它包含變數，一旦含有變數對不同的代數規格就會產生所謂的 open term，這些 open term 連同 closed term 的集合即為所有程序的集合，為要使這個程序的集合成為代數規格的模型，就必需定義一種具有 congruence 的等價關係，將這個程序的集合分割成不同的等價類，等價類視為新架構的元素，如此建立的新架構才是代數規格的模型。

在本章我們首先定義一些會用到的術語，然後討論不同的代數規格產生的模型，從這些模型中尋找適合我們在實際應用方面的模型。

第一節　遞迴程序的定義

若我們定義一個程序 x 為 ax，正式的寫法為 $X \overset{\text{def}}{=} a \cdot X$，或寫成 X≡a・X，在程序代數中簡寫成 X=aX，它的意義為程序 X 執行 a 原子 action 後又回到程序 X，也就是程序 X 可以一再的執行 a，所以 X=aaaa……，以 $X=a^{\omega}$ 表示(在代數中指數為 ω 表示重覆無限多次，指數為*表示重覆有限多次)，所以 X 為一無限程序，無限程序在 initial algebra 中是不存在的，initial algebra 僅包含 closed term，closed term 是有限的。

以程序本身來定義自己，我們都知道這叫作遞迴定義，如 X=aX 便是。現在我們

[1] 本章內容摘自[2]

就以遞迴等式(recursive equation)來定義遞迴程序：

i. 一個遞迴等式有下列形式

X = s(X)

s(X)是 BPA 中只包含變數 X，不包含其他變數的項(term)

ii. 令 V 是一群有限變數的集合、E 為一群在 BPA 上之遞迴等式的集合稱為遞迴規格，E 中每一個遞迴等式都有下列形式

$X_i = s_i(X_1, X_2, , , X_n)$ $\forall i=1,2,\cdots.n$，$n \geq 1$ 且 $X_i \in V$

在定義 ii 的等式並非每一個變數都必須出現。

所謂遞迴等式的解(solution)，就是在模型中可找到滿足遞迴等式的程序。例如遞迴等式 X=X，任何程序都滿足 X=X，因此任何程序都是 X=X 的解，也就是說遞迴等式 X=X 有無限多的解。再例如 X=a.b.X 有唯一解 $(ab)^{\omega}$。一個 recursive 等式有唯一解代表其行為被轉換成 state graph 時，可以用有限的狀態來表示。也就是第二節中的可定義的。一般而言遞迴等式本身就是一個解(需判斷這個解在所屬模型中是否存在)。是否有甚麼準則能讓我們判斷遞迴等式有唯一解，那就是遞迴等式是否具有 guarded 性質。

所謂 guarded 性質即變數在遞迴等式中，經過公設的簡化都是以 a・t 子項的形式出現，a 為原子 action(所以 a 不可以是 ε、τ，但 $\delta X=\delta$ 是 guarded)，t 則包含變數。完全的(completely)guarded 則指每一變數均為 guarded。例如 X=aX+Xa 不是 guarded，因 Xa 非 a・t 形式，又如遞迴規格{X=Y，Y=aX}，可導出 X=aX，所以 X 為 guarded，因此遞迴規格{X=Y，Y=aX}為 guarded。

為方便在程序代數中使用遞迴等式所定義的程序，就必需使用符號表示這類程序，同時也使得遞迴程序的 action relation 容易表達。若 E 表示遞迴等式的集合，根變數 X(root variable X)指 E 所定義的遞迴程序，它通常都是第一個遞迴等式之左邊變數，則程序<X|E>表示 X 為 E 之一解(CCS 使用 $\mu X.E$ 表示<X|E>)，若 t 為 BPA 中某一 open term，<t|E>表示 t 中的變數為 E 之一解，例如<aX|{X=bX}>代表的程序為 ab^{ω}。有了遞迴程序的表示方法，我們立刻有下列兩個結果：

1. <a・t|E> = a・<t|E>

2. <s+t|E> = <s|E> + <t|E>。

在程序代數中使用遞迴程序，也必需定義遞迴程序的 action relation。表 4-1 中 t_X 代表遞迴等式右邊的式子、Y 代表<Y|E>、a 代表原子 action，將表 4-1 加到表 3-2(第三章)就成為 BPA 含有遞迴程序的 action relation。

$$<t_X|E> \xrightarrow{a} \surd \quad \Rightarrow \quad <X|E> \xrightarrow{a} \surd$$

$$<t_X|E> \xrightarrow{a} Y \quad \Rightarrow \quad <X|E> \xrightarrow{a} Y$$

表 4-1　遞迴程序之 action relation

第二節　Projection and Bounded Non-determinism

在第一節我們提到遞迴等式的 guarded 性質，若遞迴等式具有 guarded 性質則它有唯一的解，也就是說只有一個程序能滿足具有 guarded 性質的遞迴規格。在本節我們介紹 Projection 運算子及 Bounded Non-determinism 性質，它們能讓我們瞭解遞迴程序的實質意義。

Projection 運算子主要目的是獲得無限程序之有限表示，Projection 運算子是一個單元算子，它可以隨時被加到任何一個程序代數裡面而不需變更已有的定義，我們以 π_n 表示 Projection 運算子，它的定義列於表 4-2，表 4-2 沒有 ε，a 為原子 action 且 $n \geq 1$。當包含 ε 時為表 4-3，此時 $n \geq 0$。當包含 τ 時，τ 要特別處理，因為 $a=a\tau=\pi_2(a\tau b)=\pi_2(ab)=ab$，這不是我們希望的結果，所以 τ 被視為沒有深度且非原子 action，表 4-4 為因應 τ 而增加的公設。因此表 4-3 與表 4-4 合起來就是有 ε 又有 τ 的 projection 公設。

PR1	$\pi_n(a) = a$	PR3	$\pi_{n+1}(ax) = a \cdot \pi_n(x)$
PR2	$\pi_1(ax) = a$	PR4	$\pi_n(x + y) = \pi_n(x) + \pi_n(y)$

表 4-2　Projection 公設

PRE1	$\pi_n(\varepsilon) = \varepsilon$
PRE2	$\pi_0(x) = \varepsilon$
PRE3	$\pi_{n+1}(ax) = a \cdot \pi_n(x)$
PRE4	$\pi_n(x + y) = \pi_n(x) + \pi_n(y)$

表 4-3　Projection 公設及 ε

PRT1	$\pi_n(\tau) = \tau$
PRT2	$\pi_n(\tau x) = \tau \pi_n(x)$

表 4-4　Projection 公設及 τ

有了 projection 算子，我們可以很容易的推出，對任何一個 closed term t 都可以找到一個自然數 k≥1，滿足 $\pi_k(t) = t$，因為 close term t 是有限程序，必定有一個這樣的 k 存在。也就是我們定義有限程序為某一 closed term 的解釋。

在介紹 Bounded Non-determinism 性質之前，我們先定義一些會用到的術語：

i. 一個程序 p 具有 **head normal form(HNF)**，則 p 可以化成下列形式：

$p=\sum_{i<n} a_i p_i + \sum_{j<m} b_j$ 其中 a,b 為原子 action，p_i 為程序，n+m>0，n,m∈N

ii. 一個**程序 p 是可定義的(definable)**，則 p 可以由 BPA(或 PA、ACP)中常數、運算子或 guarded 之遞迴等式架構而成。

iii. 一個**程序 p 是有限可定義的**(finitely definable)，其架構方式同 ii，不過只

有有限個遞迴等式。

由第三章及 i 的定義可知每一個 closed term 都可以化成 HNF。Guarded 遞迴等式也可以化成 HNF。所以每一個可定義的程序都可以化成 HNF，而且每個 p_i 都是可定義的。因 HNF 的每一個 projection 都分別相等於一個 closed term，因此可定義的程序的每一個 projection 也都會分別相等於一個 closed term。最後因 Guarded 遞迴等式可以化成 HNF，假如它有兩個解 p 及 q，則 p、q 都必需滿足這個 HNF，所以 $\pi_n(p)=\pi_n(q)$，$\forall n\geq 1$ 都會成立，這就是所謂的 **projection theorem**。假設有兩個無限的程序它們的每一個 projection 都相等，還符合下面所定義的 Bouinded Non-determinism 則這兩個程序一定相等。

Bounded Non-determinism(縮寫為 BND)是一種性質，它的定義為：執行一連串有限的原子 action，程序可以到達的狀態集合是有限的。如果一個程序具有 BND 性質就表示這個程序只有有限的分支(branch)結構。一般將 BND 性質記成 $B_N(p)$，表示程序 p 經過 n 個原子 action 後，可到達的狀態集合是有限的。所以如果我們只對可定義的程序有興趣，就可以有表 4-5 的公設。

$B_1(x)$
$B_n(a)$
$B_n(x) \Rightarrow B_{n+1}(ax)$
$B_n(x)$ and $B_n(y) \Rightarrow B_n(x+y)$

表 4-5　Bounded Non-determinism 的公設

現在我們已經討論過 Guarded 遞迴等式、projection 算子以及 BND，由這些定義可以組成適用在不同程序代數模型之原理(假設)，我們先將這些原理一一列出，並舉些例子說明。

· Recursive Definition Principle(RDP)

每個遞迴等式規格有一解(every recursive specification has a solution)。

- Restricted Recursive Definition Principle(RRDP)

  每個 guarded 遞迴等式規格有一解(every guarded recursive specification has a solution)。

- Approximation Induction Principle(AIP)

  任何程序可以由它的有限個 projection 來決定(a process is determined by its finite projections)。也就是 $\forall n \geq 1\ \pi_n(x) = \pi_n(y) \Rightarrow x = y$。

- Restricted Approximation Induction Principle(RAIP)

  $\forall n \geq 1\ \pi_n(x) = \pi_n(y)$ and $B_n(x) \Rightarrow x = y$。

  若兩個程序 x，y 它們的有限 projection 都相等，且其中之一的程序具有 BND 性質則這兩個程序是一樣的。

- Recursive Specification Principle(RSP)

  每個 guarded 遞迴等式規格最多有一解(a guarded recursive specification has at most one solution)。

上述的幾個原理，由上到下，限制愈來愈強，所能描述的程序愈來愈少。

例如對 BPA 之 initial algebra A 而言，$A \not\models$RRDP：因 A 中沒有無限程序、$A \models$AIP：因 A 中都是有限程序、$A \models$RSP：因最多有一解可以無解。當然滿足 AIP 一定滿足 RAIP、不滿足 RRDP 一定不滿足 RDP。最後***可定義程序都可以寫成 HNF 且具有 BND 性質***因此滿足 RAIP，所以只有唯一解，因 guarded 遞迴等式都可以寫成 HNF 形式，每一個 projection 對此 HNF 形式的結果都相等，若有兩個解時，每一個解必需滿足此唯一的 HNF 形式，所以這兩個解是相同的。

第三節　一個 Guarded 遞迴規格的例子

我們用這個例子來解釋 1.以 Guarded 遞迴規格表示實際應用的程序 2.同樣的程序在 PA 中以有限規格即可表達，在 BPA 中卻需用無限規格 3.任何在 PA 中的程序都

可以化成 BPA 中的程序。

這個例子為一個沒有限制容量的袋子，它可輸入 0 或 1，然後輸出 0 或 1，因此它有四種原子 action --in(0)、in(1)、out(0)、out(1)，分別表示 0、1 放入袋子以及 0、1 自袋子中移出。這個例子不在乎 0、1 的輸入輸出的次序，著重在袋子含 0、1 的數量。

$$B_{0,0} = in(0) \cdot B_{1,0} + in(1) \cdot B_{0,1}$$

$$B_{0,m+1} = in(0) \cdot B_{1,m+1} + in(1) \cdot B_{0,m+2} + out(1) \cdot B_{0,m}$$

$$B_{n+1,0} = in(0) \cdot B_{n+2,0} + in(1) \cdot B_{n+1,1} + out(0) \cdot B_{n,0}$$

$$B_{n+1,m+1} = in(0) \cdot B_{n+2,m+1} + in(1) \cdot B_{n+1,m+2} + out(0) \cdot B_{n,m+1} + out(1) \cdot B_{n+1,m}$$

表 4-6　袋子程序的無限規格

首先以 BPA 之無限規格表示袋子的程序，再以 PA 之有限規格表示，然後證明這兩種規格是一樣的。無限規格表示在表 4-6，表 4-6 中每一等式都是 guarded 遞迴等式，用到變數 $B_{n,m}$，$B_{n,m}$表示袋中有 n 個 0、m 個 1(n,m≥0)。以 PA 之有限規格表示在表 4-7，它也是一個 guarded 遞迴等式，僅用到一個變數 B 表示袋子的程序。

$$B = in(0) \cdot (B \parallel out(0)) + in(1) \cdot (B \parallel out(1))$$

表 4-7　袋子程序的有限規格

現在我們來證明 $B = B_{0,0}$，證明的過程是這樣的，首先定義一個新的程序 $D_{n,m}$，$D_{0,0}$就是袋子程序，然後證明 $D_{n,m}=B_{n,m}$，也就是證明了 $D_{n,m}$滿足表 4-6 的規格。$D_{n,m}$的定義如下：

i.　$D_{0,0} = B$

ii. $D_{0,m+1} = B \| out(1)^{m+1}$

iii. $D_{n+1,0} = B \| out(0)^{n+1}$

iv. $D_{n+1,m+1} = B \| out(1)^{m+1} \| out(0)^{n+1}$

我們只證明 i. $D_{0,0}$ 滿足 $B_{0,0}$ 的規格、ii. $D_{0,m+1}$ 滿足 $B_{0,m+1}$ 的規格，$D_{n+1,0}$ 的證明與 ii.相同，$D_{n+1,m+1}$ 的證明原理是一樣的。

i. $D_{0,0} = B = in(0) \cdot (B \| out(0)) + in(1) \cdot (B \| out(1)) = in(0) \cdot D_{1,0} + in(1) \cdot D_{0,1}$

(與表 4-6 第一個式子比較)

ii. $D_{0,m+1} = B \| out(1)^{m+1} = B \|\_ out(1)^{m+1} + out(1)^{m+1} \|\_ B$

$=(in(0) \cdot D_{1,0} + in(1) \cdot D_{0,1}) \|\_ out(1)^{m+1} + out(1)^{m+1} \|\_ B$

$= in(0) \cdot (D_{1,0} \| out(1)^{m+1}) + in(1) \cdot (D_{0,1} \| out(1)^{m+1}) + out(1) \cdot (out(1)^{m} \| B)$

$= in(0) \cdot ((B \| out(0)) \| out(1)^{m+1}) + in(1) \cdot ((B \| out(1)) \| out(1)^{m+1}) + out(1) \cdot D_{0,m}$

$= in(0) \cdot D_{1,m+1} + in(1) \cdot D_{0,m+2} + out(1) \cdot D_{0,m}$

所以 $D_{n,m}$ 為滿足表 4-6 規格的一個解，根據 RSP，表 4-6 最多一個解，因此 $D_{n,m}=B_{n,m}$ 且 $D_{0,0}= B_{0,0}=B$。這個例子說明了在 **PA(ACP)中一個可定義的程序一定可以**

化成 BPA 中的可定義的程序，反之亦成立，不過 PA 所包含之有限可定義的程序就比 BPA 多，以這個例子來講，只要可輸入的元素多於一個，BPA 都無法以有限可定義的程序表示。

第四節　BPA 的模型

令 *P* 為所有程序表示式(process expression)的集合，*P* 由下列方式架構起來：

i. 每一個原子 action 是一個程序表示式

ii. E 為遞迴等式的集合，X 為 E 中的一個變數，則<X|E>是一個程序表示式

iii.若 p,q 為程序表示式，則 p+q 及 p・q 都是程序表示式

所以 *P* 包含所有 BPA 之 signature 組成的任何一個 term，以及定義於 BPA 上的遞迴程序。因為 P 包含遞迴等式所代表的無限程序，我們僅可以證明 *P* 中所有程序都滿足 BPA 的公設，或許還可以滿足 BPA 的公設所導不出來的一些性質(或定理)，因此在 *P* 上定義了 congruence 關係後，*P* 的新結構僅能稱為 BPA 的模型，也就是說 BPA 的公設為 *P* 之 sound 公設。

為了讓 *P* 轉成為 BPA 的模型，我們必需在 *P* 上定義一個 congruence 關係，將相同的程序歸到同一個等價類，我們應該很容易就想到是 strong bisimulation，原因是 BPA 沒有特別的 action，所以根據 BPA 的 action relation，我們定義 strong bisimulation 如下：

Strong bisimulation 是一個二元等價關係 R，滿足下列條件：

i. 如果 R(p,q)　$p,q\in P$ 且 $p\xrightarrow{a}p'$，則存在一個 q' 滿足 $q\xrightarrow{a}q'$且 R(p',q')

ii. 如果 R(p,q)　$p,q\in P$ 且 $q\xrightarrow{a}q'$，則存在一個 p' 滿足 $p\xrightarrow{a}p'$且 R(p',q')

iii.如果 R(p,q)　$p,q\in P$ 則 $p\xrightarrow{a}\surd \Leftrightarrow q\xrightarrow{a}\surd$

如果 p 和 q 有 strong bisimulation 關係記成 $p\underline{\leftrightarrow}q$。$\underline{\leftrightarrow}$將 *P* 分割成等價類，我們把一個等價類視為一個元素，這樣由等價類所構成的集合記成 $P/\underline{\leftrightarrow}$，$\underline{\leftrightarrow}$對 BPA 的+、・

及π_n(如果加入)都有取代的等價關係(congruence)，所以$P/\underline{\leftrightarrow}$就成為 BPA 的模型。

$P/\underline{\leftrightarrow}$模型滿足 RDP、RAIP 及 RSP 但不滿足 AIP，滿足 RDP 之原因為遞迴等式本身即為一個解且此解在$P/\underline{\leftrightarrow}$中，滿足 RAIP 第二節已解釋過，至於不滿足 AIP 之原因可舉一個例子來說明：令程序 p≡<X|X=Xa + a>，若 p 為其解則 p = pa + a，a≤p(a 為 p 之子項)，若 a≤p ⇒ aa≤pa≤ pa + a = p，導出 aa 也是 p 之子項，繼續這樣推下去 a^n 也是 p 的子項，因此對每一個 n ，a^n 都是 p 的子項，所以 p=$\sum a^n$ (無限個和)為其解。令程序 q ≡ <X|X=aX>，我們知道 $q=a^\omega$ 為其解，因此π_n(p) $\underline{\leftrightarrow}$ π_n(p+q) $\underline{\leftrightarrow}$ ($\sum a^n$)，但 p$\underline{\leftrightarrow}$p+q 不成立，因為 p$\xrightarrow{a}$ a^n ∀n≥1 而 p+q$\xrightarrow{a}$ a^ω，由此得之 p 與 p+q 的每一個 projection 都相等但 p 與 p+q 卻不相等，它們不能相等的原因在於 p 沒有 BND 性質。

對於$P/\underline{\leftrightarrow}$內的有限程序(即對應 closed term 的程序)，如果 BPA□ s = t 若且唯若 s$\underline{\leftrightarrow}$t，如果 s = t 當然有 s$\underline{\leftrightarrow}$t 結果，反過來說如果 s$\underline{\leftrightarrow}$t 成立，因$\underline{\leftrightarrow}$保持分支的特性，所以 s = t。這說明了 BPA 的公設對於$P/\underline{\leftrightarrow}$內有限程序的集合為 complete 公設。

當δ加入形成P_δ，因δ並不影響程序的 action relation，因此同樣用$\underline{\leftrightarrow}$來將相同的程序歸成一個等價類，這樣形成的模型為$P_\delta/\underline{\leftrightarrow}$，$P_\delta/\underline{\leftrightarrow}$與$P/\underline{\leftrightarrow}$一樣具有上面相同的討論結果。

不論是ε加入 P 形成P_ε或者加入P_δ形成$P_{\delta\varepsilon}$，從第三章對BPA_ε的討論得知，ε的加入會改變 action relation，因此我們需要有不同的 bisimulation，這個 bisimulation 僅需將前述之 strong bisimulation 的 iii 改為下列敘述即可：

iii. 如果 R(p,q) p,q∈P_ε(或$P_{\delta\varepsilon}$) 則 p↓ ⇔ q↓(p↓表示 p 可成功結束)。

我們同樣也是以$\underline{\leftrightarrow}$符號表示它。$P_\varepsilon/\underline{\leftrightarrow}$與$P_{\delta\varepsilon}/\underline{\leftrightarrow}$這兩個模型都和$P/\underline{\leftrightarrow}$一樣，滿足 RDP、RAIP 及 RSP 但不滿足 AIP。

從第三章我們知道 PAε所構成的項可以經由公設化成$BPA_{\delta\varepsilon}$所構成的項，因此由

$BPA_{\delta\varepsilon}$所架構的模型$P_{\delta\varepsilon}/\underline{\leftrightarrow}$也就是$PA\varepsilon$的模型。同樣的$P_{\delta\varepsilon}/\underline{\leftrightarrow}$也是$ACP\varepsilon$的模型。

第五節　ACP_{ε}^{τ}的模型

第三章第十一節我們討論過ACP_{ε}^{τ}，如果涵蓋變數時，ACP_{ε}^{τ}所產生的 term 模型與$BPA_{\delta\varepsilon}^{\tau}$所產生的 term 模型是一樣的，而$BPA_{\delta\varepsilon}^{\tau}$與$BPA_{\delta\varepsilon}$的 action relation 除了允許$\tau$為一個 action 外它們都是相同的，現在我們要架構一個ACP_{ε}^{τ}的模型，就必須定義一個含有τ的 bisimulation。因為$BPA_{\delta\varepsilon}$所有程序的集合為$P_{\delta\varepsilon}$，也可以說在$P_{\delta\varepsilon}$上定義一個含有τ的 bisimulation 就可以轉成ACP_{ε}^{τ}的模型。

為了定義一個含有τ的 bisimulation，需用到 generalized τ-step，generalized τ-step 定義如下：

一連串 n 個τ ($n\geq 0$)，使得$s\xrightarrow{\tau}\cdots\xrightarrow{\tau}t$，我們把它寫成$s\Rightarrow t$。

現在我們開始定義 branching bisimulation，在定義裡面我們以$P_{\delta\varepsilon}$代表ACP_{ε}^{τ}所建構出來所有程序的集合(與前面的$P_{\delta\varepsilon}$比較，現在的$P_{\delta\varepsilon}$允許τ在程序中出現)、$p,q\in P_{\delta\varepsilon}$、 $a\in A\cup\{\tau\}$。

一個 branching bisimulation 是一種在$P_{\delta\varepsilon}$上的二元關係 R，滿足下列條件：

i. 如果$p\xrightarrow{a}p'$且 R(p,q) 則既是$a=\tau$且 R(p' ,q)或是存在 v，q' 滿足

$q\Rightarrow v\xrightarrow{a}q'$，而且 R(p,v) 及 R(p',q')均成立。

ii. 同 i.交換 p，q。

iii. 如果 p↓ 且 R(p,q) 則存在一個 q' 滿足 $q\Rightarrow q'$ 且 q'↓且 R(p，q')。

iv. 同 iii.交換 p，q。

若 p 和 q 有 branching bisimulation 關係，我們寫成$p\underline{\leftrightarrow}_b q$。

Branching bisimulation 對 ACP_{ε}^{τ}的+及 ||_ 沒有取代的關係，例如雖然$\tau\cdot$

$a \underline{\leftrightarrow}_b a$，但是 $\tau a + b \underline{\leftrightarrow}_b a + b$ 卻不成立，$\tau a \between b \underline{\leftrightarrow}_b a \between b$ 也不成立。為了讓 $\underline{\leftrightarrow}_b$ 具有取代的關係，就必需排除在+及 $\between$ 運算中子程序一開始就有 τ，排除的方式就是另外增加兩個條件如下：

i. 如果 $p \xrightarrow{a} p'$ 然後存在一些 q' 滿足 $q \xrightarrow{a} q'$ 且 $p' \underline{\leftrightarrow}_b q'$ $a \in A \cup \{\tau\}$

ii. 同 i.交換 p，q。

這兩個條件的意義為第一步任何一個程序執行 a，另一個程序也必需配合執行 a，往後只要具有 $\underline{\leftrightarrow}_b$ 關係即可。因為這兩個條件針對第一步作限制，所以稱為根條件，我們把具有 $\underline{\leftrightarrow}_b$ 關係又滿足根條件寫成 $\underline{\leftrightarrow}_{rb}$。

現在我們可以使用 $\underline{\leftrightarrow}_{rb}$ 關係將 $P_{\delta\varepsilon}$ 分割成等價類，每一個等價類中的程序都是相等的而且對 ACP_ε^τ 的運算子都是可以互相取代的，所以 $P_{\delta\varepsilon} / \underline{\leftrightarrow}_{rb}$ 就是一個 ACP_ε^τ 的模型。

我們在討論 guarded 遞迴等式定義時，不允許 τ 作為 guarded 的原子 action，原因是會產生許多解。例如 $X = \tau X$，任何程序左邊加一個以上的 τ 即為它的解，如 p 為一程序則 τp 就是一個解，因此 τa、τab 等等都是 $X = \tau X$ 的解(使用 $x\tau = x$ 公設)。我們希望遞迴等式規格只有一個解，使得我們能利用它來定義程序時更為明確，所以我們不允許 τ 作為 guarded 的原子 action，但 ACP_ε^τ 引進 τ_I 算子，τ_I 算子會產生 τ，因此我們也不允許 τ_I 算子出現在 guarded 遞迴等式定義中，這樣一來第二節中所定義的**可定義的(definable)程序**在 ACP_ε^τ 中就需要更改，因此**可定義的程序**在沒有 τ 的程序代數中移到有 τ 的程序代數中被細分成兩類，一類為**可架構的(constructible)程序**、另一類為**可規格化的(specifiable)程序**，**可架構的程序**指使用所屬之代數規格裡面的原子 action、運算子以及 guarded 遞迴等式規格所構成的程序(可以將 τ_I 用在 guarded 遞迴等式上，會產生 τ^ω 程序)，**可規格化的程序**是指 guarded 遞迴等式規格所定義的程序(程序 τ^ω 不是可規格化的程序，因 guarded 遞迴等式不允許使用 τ 及 τ_I)。

因為$\pi_n(\tau x) = \tau\ \pi_n(x)$，因此$\pi_n(\tau^{\omega}) = \tau^{\omega}$，$\tau^{\omega}$為一無限程序，所以可架構的程序之 projection 不一定會相等於 closed term。至於可規格化的程序，因為不允許使用τ及τ_I所以它相等於 ACP 中的可定義的程序，因此可化成 HNF 以及具有 BND 性質。所以$P_{\delta\varepsilon}/\underline{\leftrightarrow}_{rb}$模型滿足 RAIP、RDP 以及 RSP[2]。