Capítulo II:

Communicating Sequential Process

II.1. LAS ALGEBRAS DE PROCESOS

Cuando se analiza un programa concurrente se ha de dar respuesta a una nueva clase de requerimientos que son desconocidos en los sistemas secuenciales y que implica el replanteamiento de las propiedades de correctitud de dichos sistemas.

El diseño de un sistema correcto se convierte entonces en una etapa crítica importante que va antes de la implementación de un sistema concurrente. Este diseño se basa en una buena especificación del *comportamiento de los procesos que intervienen en él.*

El *modelo de las álgebras de procesos* se basa en un conjunto de procesos elementales, un conjunto de operaciones que pueden llevarse a cabo y un conjunto de leyes algebraicas que se utilizan para obtener términos de procesos equivalentes.

Ejemplos de Álgebras de Procesos concretas son CSP o CCS que pertenecen a la categoría de *modelos abstractos de cálculo paralelo* y son útiles para la descripción de un conjunto complejo de procesos concurrentes, es decir, de un sistema, aplicación o programa concurrente.

Al utilizar alguno de estos modelos de cálculos abstractos se pueden llegar a cubrir los objetivos siguientes:

a) Derivar términos de procesos complejos a partir de un conjunto simple de procesos, utilizando un número pequeño de operadores con una semántica bien definida.
b) Ser capaz de razonar acerca de las propiedades de los términos de procesos, utilizando su interpretación semántica y un conjunto de leyes algebraicas.
c) Tener una base segura para derivar un programa concreto a partir de una especificación formal previa preservando la equivalencia semántica.

El álgebra de procesos que se estudiará en este capítulo será el Álgebra de procesos llamada ***Communicating Sequential Process*** comúnmente conocida como ***CSP***.

II.2. ADOPCION DE LA HIPOTESIS DEL MODELO

La hipótesis adoptada para el estudio del modelo CSP contempla los siguientes puntos:

- Supondremos que existe un paralelismo de sucesos (llamados *eventos*) que se pueden modelizar a través del *entrelazamiento* de acciones atómicas de los procesos (*comunicaciones abstractas*). De esta forma un proceso secuencial es indistinguible respecto de una red de procesos paralelos si en ambos se produce una secuencia de entrelazamiento idéntica.
- Se define un *entorno externo* a los procesos, es decir:
    - Un sistema ofrece un conjunto de comunicaciones posibles
    - El entorno decide en que comunicación se va a implicar o va a participar
- Supondremos que el sistema toma decisiones autónomas respecto del entorno, es decir:

  - El sistema puede llevar a cabo acciones internas que ocasionan un comportamiento del sistema de manera no determinística
  - Por tanto, el trazado del sistema no presentará siempre la misma secuencia de entrelazamiento de las comunicaciones entre los procesos.
- Supondremos la existencia de una notación (semántica) bien definida para los términos de procesos.

II.3. EL CSP

El *Álgebra de Procesos CSP (Communicating Sequential Processes)* propuesto por Hoare en 1978 es un lenguaje algebraico con el que se puede describir el comportamiento de la comunicación entre procesos y puede ser verificado a través de eventos y operadores.

II.3.1. EL ALGEBRA DE PROCESOS DEL CSP

En CSP un proceso describe el patrón de comportamiento de un objeto en términos de eventos, operadores y otros procesos. Para incluir eventos en una descripción de procesos, se utiliza el operador más básico del CSP que es el operador *prefijo*.

Operador Prefijo
Si *a* es un evento y *P* es un proceso, entonces:

(a → P)

es un proceso que ejecuta el evento *a* antes de comportarse como el proceso *P*. Un evento puede ser un evento de comunicación, el cual es representado por el par:

c.v

Donde *c* es el nombre del canal en donde el evento toma lugar, y *v* es el valor del mensaje que es pasado por dicho canal. La entrada y la salida son respectivamente la acción de lectura ***c?v*** y la acción de escritura ***c!v*** las cuales se llevan a cabo cuando ambos procesos se encuentran listos y la comunicación sobre el canal *c* se realiza. Definimos un proceso de entrada como:

(c?v → P(v))

Un proceso de entrada es un proceso que se comporta como un proceso *P(v)* después de que el valor *v* es leido por el canal *c*. El valor *v* es de tipo *T* denotado como:

(c?(v:T) → P(v)).

Definimos ahora un proceso de salida como:

(c!v → P)

Un proceso de salida es un proceso que se comporta como un proceso *P* después de que el valor *v* es enviado por el canal *c*. Esto es definido por el protocolo de comunicación del CSP llamado *rendezvous o cita*. El conjunto de mensajes comunicables en un canal *c* es definido como:

$Type(c)=\{v \mid c.v \in P\}$

El alfabeto de P es el conjunto de eventos en un proceso P. El tipo del valor *v* es especificado por el proceso de salida $(c!v \rightarrow P) = (c.v \rightarrow P)$ y debe ser aceptado por el proceso de entrada $(c?x \rightarrow P(x)) = \square_{v:type(c)}(c.v \rightarrow P(v))$, donde el símbolo $\square$ denota opciones. El proceso de salida define un tipo y el proceso de entrada acepta uno o más tipos.

Dados dos procesos P y Q, éstos pueden ser observados en varias composiciones:

$(P;Q)$ **Composición Secuencial**; Es un proceso que se comporta como *P* hasta que éste componente es totalmente terminado y entonces se comporta como Q.

$(P \mid\mid Q)$ **Composición Paralela;** es un proceso o composición en la cual P es capaz de ejecutarse en cualquier evento del conjunto αP, y Q es capaz de ejecutarse en cualquier evento del conjunto αP. Los dos procesos pueden cooperar para llevar a cabo cualquier evento el cual es común a ambos alfabetos. Esta composición paralela puede ser descrita por $P \mid [A] \mid Q$ con $A=\alpha P \mid \alpha Q$ como el conjunto compartido de eventos.

$(P \mid\mid\mid Q)$ **Composición Paralela (con interpolación);** donde los dos procesos P y Q se desarrollan independientemente sin cooperación entre ellos en cada ocurrencia de cualquier evento de P y Q. Esta composición paralela es igual a $P \mid [A] \mid Q$ con el conjunto *A* como el conjunto vacío. Si un proceso no puede ejecutarse en la acción, entonces debe pasar a ser el otro proceso; pero si ambos procesos pueden ejecutarse en la misma acción, la elección entre ellos es no determinística.

$(P \square Q)$ **Composición Alternativa;** Se comporta como P si la primera acción de P puede ejecutarse, sino se comporta como Q si la primera acción de Q puede ejecutarse. Si ambas acciones pueden llevarse a cabo entonces la elección entre ellos se hace de manera no determinística (el ambiente bajo el cual se ejecutan los procesos puede controlar qué proceso podría ser seleccionado P o Q.

$(P \sqcap Q)$ **Composición Alternativa (no determinística);** La elección entre P y Q se basa en una selección arbitraria, sin el conocimiento o control del ambiente externo.

STOP Es proceso que nunca se ejecuta en ningún evento. Describe el comportamiento de un objeto roto o de un objeto bloqueado.

SKIP Es un proceso que no hace nada, pero termina completamente.

Una regla importante en CSP muy utilizada es que:

Dado		P	=	$\underset{x:X}{}$ **$x \rightarrow P_x$**
		Q	=	$\underset{y:Y}{}$ **$y \rightarrow P_y$**
Entonces	$P \mid [A \mid B] \mid Q$		**=**	$\underset{z:Z}{}$ **$z \rightarrow (P_z' \mid [A \mid B] \mid Q_z')$**
	Donde	P_z'	=	**P_z Si $z \in X$; P en otro caso**
	y	Q_z'	=	**Q_z Si $z \in Y$; Q en otro caso**
	y	Z	=	**$(X \cap Y) \cup (X - B) \cup (Y - A)$**
	Asumiendo	$X \subseteq A$ y $Y \subseteq B$		

Regla 1. Regla General del Operador Selección (choice)

Regla General del Operador Selección (choice)
Esta regla proporciona un algoritmo para evaluar el comportamiento de dos procesos en paralelo (P | | Q). La notación $P = \underset{x:X}{} x \rightarrow P_x$ es una serie de opciones o selecciones que dependen en los eventos *x* del conjunto *X*. Una instancia $P= \underset{x:\{a,b,c\}}{} x \rightarrow P_x$ es igual a $P=(a \rightarrow P_a) \quad (b \rightarrow P_b) \quad (c \rightarrow P_c)$. El conjunto Z es definido como la colección de elementos en *X* y *Y* y sin elementos dobles. La notación **(X - B)** es el conjunto **{ x | x ∈ X, x ∉ B }**.

II.3.2. COMUNICANDO PROCESOS

Los procesos *P* y *Q* se comunican a través de canales compartidos. Las acciones de lectura y escritura en un canal se consideran eventos compartidos que son llevados a cabo cuando ambos procesos están listos para comunicarse. Un canal proporciona un camino de comunicación entre dos procesos. Para tener dos caminos de comunicación se requiere del uso de dos canales. Un canal que es utilizado por un proceso sólo para la salida es llamado *canal de salida* y aquel canal que es utilizado por un proceso sólo para la entrada de datos es llamado *canal de entrada*.

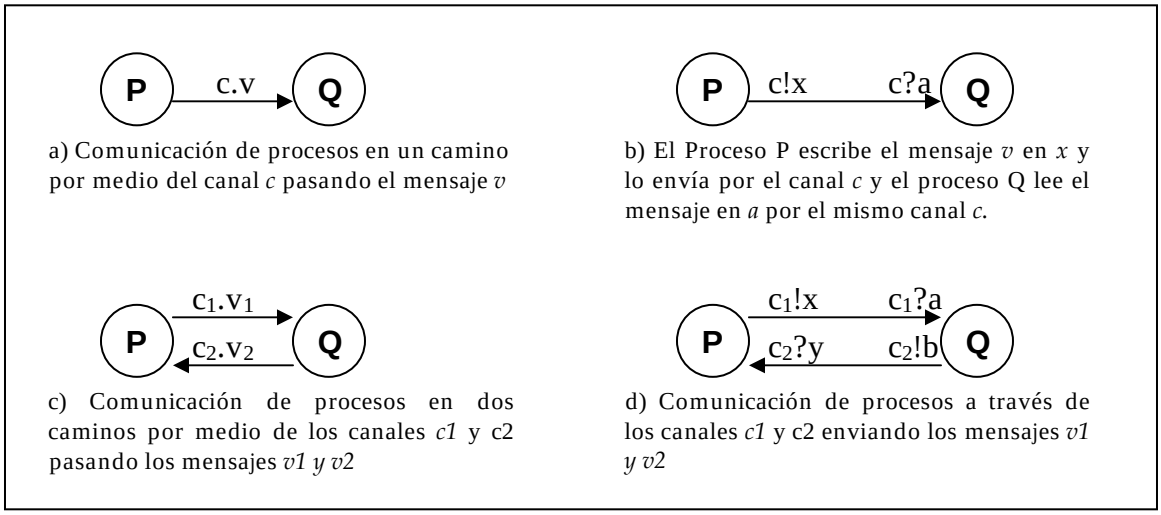


Fig 1. Comunicando Procesos

La figura 1 muestra la comunicación entre procesos con uno y dos caminos. La figura 1.a) muestra una composición paralela donde *P* y *Q* comparten el mismo canal *c*. La comunicación ocurre en el canal *c* en cada ocasión que *P* envía un mensaje y Q recibe el mensaje, simultáneamente. Un proceso que este enviando un dato especifica un único valor para el mensaje, por lo que el proceso que esta recibiendo el dato esta preparado para aceptar cualquier valor comunicable. Así, el evento que puede ocurrir es la comunicación *c.v*, donde *v* es el valor del mensaje especificado por el proceso de salida. El proceso *P* ejecuta la acción ***c!a*** y el proceso *Q* ejecuta la acción ***c?a*** tal como lo ilustra la figura 1.b), en donde las variables *x* y *a* constituyen el mensaje *v*. Las figuras 1.c) y 1.d) muestran la comunicación vía dos canales entre los procesos *P* y *Q*.

En el caso de la comunicación vía dos caminos o bien en presencia de un ciclo, deberemos evitar el ínter bloqueo en cualquier instante de tiempo. La siguiente sección discute la aparición del ínter bloqueo (deadlock) causado por un típico orden secuencial de comunicación entre procesos.

II.3.2.1. La Composición Secuencial es un ínter bloqueo sensitivo

Un hilo (Thread) es un flujo de control secuencial. Los procesos *P* y *Q* son considerados hilos de control que invocan acciones de lectura y escritura en ambos canales en un orden secuencial. Si este orden no es correctamente seleccionado por el programador o por el flujo de control entonces se puede dar el ínter bloqueo. Ambos procesos pueden entonces esperar infinitamente a comunicarse (figura 2).

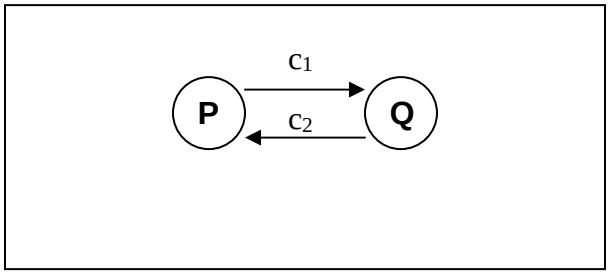


Figura 2. Comunicación cíclica de Procesos

El sistema de procesos de la figura 2, puede describirse como sigue:

```
P: do sequential          Q: do sequential
     {                         {
      c1 ! x                    c2 ! b
      c2 ? y                    c1 ? a
     }                         }
```

O en el lenguaje CSP:

SYSTEM = (P||Q) = (P ||[αP|αQ]|| Q) con αP={c1,c2} y αQ={c1,c2}

En donde los procesos *P* y *Q* son descritos como:

$$P = (c1 \rightarrow c2 \rightarrow P')$$

$$P = (c2 \rightarrow c1 \rightarrow Q')$$

Los procesos *P'* y *Q'* son procesos sufijos de *P* y *Q* respectivamente, después de la composición. Simplemente decimos entonces que $c1 \rightarrow c2$ *y* $c2 \rightarrow c1$ sólo pueden ser llevados a cabo cuando *c1 = c2*. Esto nunca es verdadero y por tanto los procesos *P y Q* son mutuamente exclusivos o bien se produce un ínter bloqueo. Se concluye entonces que la comunicación secuencial en un orden erróneo puede provocar un ínter bloqueo.

Teorema: La Comunicación Secuencial en un orden erróneo provoca ínter bloqueo.

Demostración: Demostrar que el SISTEMA es igual al proceso STOP.

Dados: $P = (c_1 \rightarrow c_2 \rightarrow P') = (c_1 \rightarrow P_{c1})$ *con* $P_{c1} = (c_2 \rightarrow P')$

$Q = (c_2 \rightarrow c_1 \rightarrow Q') = (c_2 \rightarrow Q_{c2})$ *con* $Q_{c2} = (c_1 \rightarrow Q')$

Entonces

$SYSTEM = ((c_1 \rightarrow c_2 \rightarrow P') \,|[\, \{c1, c2\} \mid \{c1, c2\} \,]|\, (c_2 \rightarrow c_1 \rightarrow Q'))$

Siguiendo la regla 1:

$X = \{c1\},\ Y = \{c2\}$ *y* $Z = \{\ \}$

Con

$P = \underset{x:\{c1\}}{} x \rightarrow P_x = c1 \rightarrow P_{c1}$

$Q = \underset{y:\{c2\}}{} y \rightarrow Q_y = c2 \rightarrow Q_{c2}$

Y

$P \,|[\alpha P \mid \alpha Q]|\, Q = \underset{z:\{\ \}}{} z \rightarrow (P_z' \,|[A \mid B]|\, Q_z') = STOP$

O bien

$SYSTEM = STOP$

Q.E.D.

El Proceso SYSTEM nunca se ejecuta en ningún evento de *P* y *Q* e inmediatamente se produce el ínter bloqueo. La demostración puede extenderse al uso de más de dos procesos que se estén comunicando.

II.3.2.2. La Composición Paralela elimina el ínter bloqueo

Las acciones de lectura y escritura del proceso *P* y las acciones de lectura y escritura del proceso *Q* son mutuamente independientes. La acción de lectura y escritura puede ser ejecutada en un orden arbitrario ya que el orden actual es indefinido. Como se mencionó anteriormente en una composición secuencial, esto es, en un proceso secuencial, el orden de las acciones de lectura y escritura deben ser cuidadosamente elegidas. Un enfoque mucho más seguro es utilizar una composición paralela para sí eliminar el interbloqueo, como por ejemplo:

```
P: do sequential          Q: do parallel
     {                         {
      c1 ! x                    c2 ! b
      c2 ? y                    c1 ? a
     }                         }
```

En la siguiente demostración se muestra que el proceso SYSTEM siempre lleva a cabo todas las acciones. Nuevamente, esta demostración puede extenderse a la comunicación de más de dos procesos.

Teorema: Una composición paralela previene un orden erróneo de comunicación y elimina el ínter bloqueo potencial.

Demostración: Demostrar que una composición paralela siempre satisface el comportamiento del sistema.

Dados: $P = (c_1 \rightarrow c_2 \rightarrow P') = (c_1 \rightarrow P_{c1})$ *con* $P_{c1} = (c_2 \rightarrow P')$

$Q = (c_2 \rightarrow c_1 \rightarrow Q') \quad (c_1 \rightarrow c_2 \rightarrow Q') = (c_2 \rightarrow Q_{c2}) \quad (c_1 \rightarrow Q_{c1})$

con $Q_{c1} = (c_1 \rightarrow Q')$ *y* $Q_{c2} = (c_2 \rightarrow Q')$

Entonces

$SYSTEM = (c_1 \rightarrow c_2 \rightarrow P') \,|[\, \{c1, c2\} \mid \{c1, c2\}]| \, ((c_2 \rightarrow c_1 \rightarrow Q') \quad (c_1 \rightarrow c_2 \rightarrow Q'))$

Siguiendo la regla 1:

$X = \{c1\}, \quad Y = \{c1, c2\}$, *y* $Z = \{c1\}$

Con

$P = {}_{x:\{c1\}}\ x \rightarrow P_x = c1 \rightarrow P_{c1}$

$Q = {}_{y:\{c1,c2\}}\ y \rightarrow Q_y = (c_1 \rightarrow Qc_1) \quad (c_2 \rightarrow Qc_2)$

Y

$P \,|[\alpha P \mid \alpha Q]|\, Q = {}_{z:\{c1\}}\ z \rightarrow (P_z' \,|[A \mid B]|\, Q_z') = c1 \rightarrow (P_{c1} \,|[A \mid B]|\, Q_{c1})$

El siguiente comportamiento es:

$X = \{c2\}, \quad Y = \{c1, c2\}$, *y* $Z = \{c2\}$

Con

$P_z' = {}_{x:\{c2\}}\ x \rightarrow P_x = c2 \rightarrow P_{c2}$

$Q_z' = {}_{y:\{c1,c2\}}\ y \rightarrow Q_y = (c_1 \rightarrow Q') \quad (c_2 \rightarrow Q')$

Y

$P \,|[\alpha P_z \mid \alpha Q_z]|\, Q = {}_{z:\{c2\}}\ z \rightarrow (P_z'' \,|[A \mid B]|\, Q_z'') = c2 \rightarrow (P' \,|[A \mid B]|\, Q')$

O bien

$SYSTEM = c1 \rightarrow c2 \rightarrow (P' \,|[A \mid B]|\, Q')$

Q.E.D.

El Proceso SYSTEM se puede ejecutar en ambos eventos *c1* y *c2* de *P* y *Q* y sin embargo este proceso satisface el comportamiento del proceso SYSTEM.

El Sistema siempre realiza todas las acciones. Y en el caso en donde las acciones de lectura y/o escritura son mutuamente independientes, estas acciones suelen ser encerradas en una composición paralela. El programador no suele preocuparse por el orden de ejecución. El anterior ejemplo se podría entonces rescribir de la siguiente manera:

```
P: do parallel          Q: do parallel
     {                       {
      c1 ! x                  c2 ! b
      c2 ? y                  c1 ? a
     }                       }
```

II.3.3. Composiciones de Comunicación de procesos

En esta sección se discuten las tres situaciones más comunes en que los procesos pueden comunicarse (ver figura 3). Se puede ver que estas tres situaciones pueden ser descritas por la misma descripción, valga la redundancia, para las tres diferentes composiciones que se pueden llevar a cabo sobre un proceso *P*. Las flechas *a, b, c,* son canales sobre los cuales los procesos *P, Q, R* y *S* pasan información. La información puede ser pasada acorde al principio de *cita*. La dirección de los canales es en un sentido denotada por las flechas.

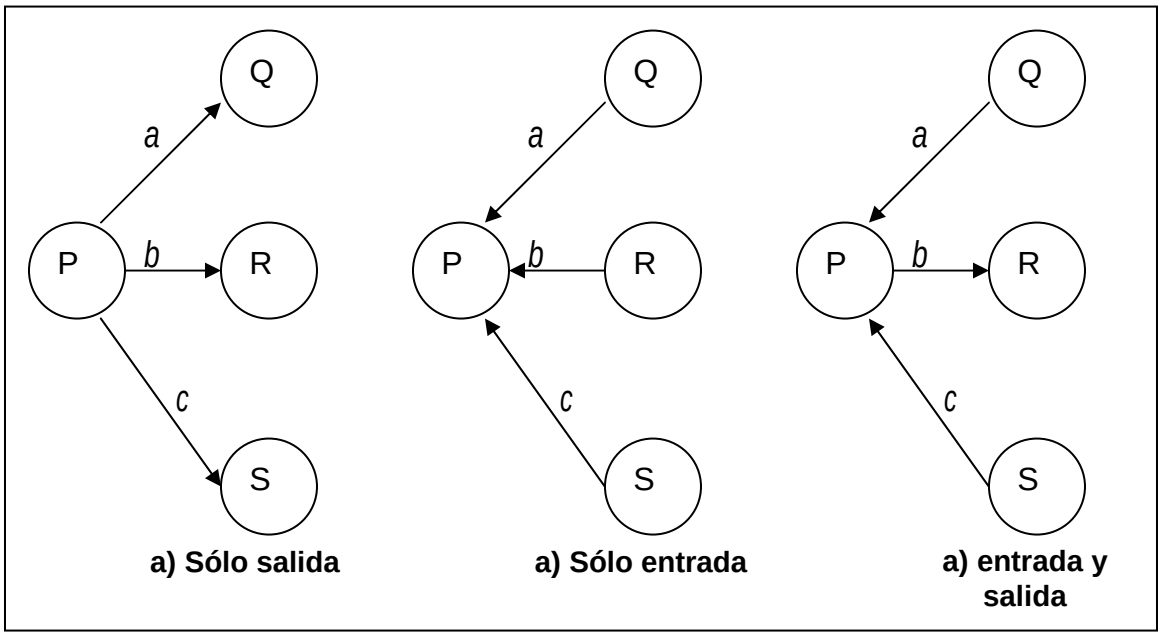


Figura 3. Tres situaciones sobre el proceso P

En la representación de la figura 3, se esta considerando al proceso *P*, como el proceso primario y a los otros procesos como procesos secundarios. Estas situaciones muestran la red de comunicación de un programa y el grado del paralelismo, es decir, la estructura y la geometría del paralelismo. Ahora bien, el orden de comunicación realizado por el proceso *P* se discute a través de tres diferentes composiciones. No se discuten los aspectos del tiempo.

Los procesos secundarios sólo se comunican con el proceso *P* y no hay comunicación mutua entre los procesos secundarios. El sistema puede ser descrito entonces como:

$$SYSTEM = P \;|[\alpha P \,|\, \alpha Q \,|\, \alpha R \,|\, \alpha S]|\; (Q \,|||\, R \,|||\, S) \qquad (*)$$

$$Con\;\; \alpha P = \{a, b, c\},\; \alpha Q = \{a\},\; \alpha R = \{b\},\; \alpha S = \{c\}$$

Cada proceso se comporta acorde con su definición, pero con la restricción de aquellos eventos los cuales están en el alfabeto de ambos procesos: *P y (Q|||R|||S)*, esto es, los eventos *a, b, c* requieren participación simultanea. Sin embargo ellos pueden procesar se independientemente en aquellos eventos que contemplen sus alfabetos, es decir, *Q, R y S* se procesan independientemente. Podemos entonces escribir el conjunto $\{\alpha P \,|\, \alpha Q \,|\, \alpha R \,|\, \alpha S\}$ *como { {a, b, c}|{a}|{b}|{c} } o bien como {a, b, c}* simplificando las ecuaciones. El proceso primario *P* puede comunicarse con los procesos secundarios *Q, R y S* en orden secuencial, en paralelo o mediante una elección. Por consiguiente, podemos considerar de forma respectiva las comunicaciones de; (i)la composición secuencial, (ii) la composición paralela, y (iii) la composición alternativa. Este tipo de composiciones (ilustradas en la figura 3, son discutidas a continuación.

II.3.3.1. La Composición Secuencial

Una composición secuencial realiza una comunicación en un cierto orden. En el caso del proceso *P*, éste es un hilo de control, esto es, un flujo de control secuencial, este proceso es secuencial por naturaleza. El programador puede especificar un cierto orden que puede ser definido como:

$$P = (a \rightarrow b \rightarrow c \rightarrow P')$$
$$Q = (a \rightarrow Q')$$
$$R = (b \rightarrow R')$$
$$S = (c \rightarrow S')$$

Donde los procesos *P′, Q′, R′ y S′* son procesos sufijos después de la composición. Después de sustituir estos comportamientos en el sistema etiquetado con (*), la composición secuencial se puede escribir como:

$$SEQ = (a \rightarrow b \rightarrow c \rightarrow P') \;|[\{a,b,c\}]|\; ((a \rightarrow Q') \,|||\, (b \rightarrow R') \,|||\, (c \rightarrow S'))$$

Esta definición es la misma para todas las tres situaciones de la figura 3. La secuencia es completamente determinada por el proceso *P*. La dirección en la que se hace la comunicación es por tanto no relevante.

II.3.3.2. La Composición Paralela

Una composición paralela realiza una comunicación en paralelo. Si el orden de comunicación por el proceso *P* es indefinido entonces la composición paralela es la

composición que debería utilizarse si se quiere evitar el ínter bloqueo. El comportamiento del proceso P en donde se quiere llevar a cabo la comunicación de forma simultánea es definido como:

$$P = (a \rightarrow b \rightarrow c \rightarrow P') \quad (a \rightarrow c \rightarrow b \rightarrow P') \quad (b \rightarrow a \rightarrow c \rightarrow P') \quad (b \rightarrow c \rightarrow a \rightarrow P') \quad (c \rightarrow a \rightarrow b \rightarrow P') \quad (c \rightarrow b \rightarrow a \rightarrow P')$$

$$Q = (a \rightarrow Q')$$
$$R = (b \rightarrow R')$$
$$S = (c \rightarrow S')$$

La ecuación entera es:

$$PAR = ((a \rightarrow b \rightarrow c \rightarrow P') \quad (a \rightarrow c \rightarrow b \rightarrow P') \quad (b \rightarrow a \rightarrow c \rightarrow P') \quad (b \rightarrow c \rightarrow a \rightarrow P') \quad (c \rightarrow a \rightarrow b \rightarrow P') \quad (c \rightarrow b \rightarrow a \rightarrow P')) \; |[\{a, b, c\}]| \; ((a \rightarrow Q') \; ||| \; (b \rightarrow R') \; ||| \; (c \rightarrow S'))$$

II.3.3.3. La Composición Alternativa

La composición alternative simple se puede basar en un evento que puede ser descrito por el proceso:

$$ALT = (a \rightarrow P) \quad (\neg a \rightarrow Q)$$

El proceso ALT puede comportarse como *P* si este puede ejecutarse en *a*, sino puede entonces comportarse como *Q*. Este comportamiento continúa de forma inmediata y es análogo a una sentencia *if-then-else*. El proceso para la sentencia *if-then* sin la sentencia *else*, se describe como:

$$ALT = (a \rightarrow P) \quad (\neg a \rightarrow SKIP)$$

La siguiente expresión describe una composición alternativa referente a la figura 3. El proceso *P* hace elecciones basadas en los eventos *a, b y c* y se describe como:

$$P = (a \rightarrow P_a') \quad (b \rightarrow P_b') \quad (c \rightarrow P_c')$$
$$Q = (a \rightarrow Q')$$
$$R = (b \rightarrow R')$$
$$S = (c \rightarrow S')$$

La ecuación completa es:

$$ALT = ((a \rightarrow P_a') \quad (b \rightarrow P_b') \quad (c \rightarrow P_c')) \; |[\{a, b, c\} | \{a\} | \{b\} | \{c\}]| \; ((a \rightarrow Q') \; ||| (b \rightarrow R') \; ||| (c \rightarrow S'))$$

Si más de un evento es verdadero, entonces la elección puede ser no determinística. La elección puede llevarse a cabo por un mecanismo justo o un mecanismo de prioridad. Los eventos *a, b, y c* son llamados *guardas* de la composición alternativa. Un guarda puede ser de entrada o salida.

II.4. Concepto de Término de Procesos

- **Término de Procesos**

$P ::= op(P_1, P_2, \ldots, P_n) \mid X \mid \mu X.P$

Donde: P_i *es un proceso* $\subseteq$ *Comunicación*
X *es un identificador*
$\mu X.P$ *denota recursividad*

- **El conjunto de términos de Procesos recursivos (*rec*)**

P, Q $\in$ *Rec P ::= stop (interbloqueo)*
| *div (divergencia)*
| *a.P (Prefijo)*
| *P+Q (selección)*
| *P | Q (paralelismo)*
| *P[φ] (Morfismo)*
| *X (identificador)*
| $\mu X.P$ *(recursión)*

- **Morfismos**

Renombrado: *Para diferentes eventos a1, a2,. . ., an con n ≥1*
$P[b1, \ldots, bn / a1, \ldots, an] = P[\varphi]$

$$\varphi(u) = \begin{cases} \iota_i & \text{si } u = a_i \text{ para } i \in 1, 2, \ldots, n \\ u & \text{si no} \end{cases}$$

donde $u \in A$

Ocultamiento: $P \setminus B = P[\varphi]$

$$\varphi(u) = \begin{cases} \tau & \text{si } u \in B \\ u & \text{si no} \end{cases}$$

Para cada término P asignamos un alfabeto de comunicación $\alpha(P)$ definido como sigue:

$\alpha(STOP:A) = \alpha(div: A) = A$
$\alpha(a.P) = \{a\} \cup \alpha(P)$
$\alpha(P + Q) = \alpha(P \mid Q) = \alpha(P) \cup \alpha(Q)$
$\alpha(P[\varphi]) = \varphi(\alpha(P)) - \{\tau\}$
$\alpha(X) = A \text{ si } X \in Id(A)$
$\alpha(\mu X.P) = \alpha(X) \cup \alpha(P)$

II.5. Ejemplos

II.5.1 Prefijo

x . P

Este término representa un proceso que después de implicarse en la comunicación x se comporta como el proceso P.

$\alpha(x . P) = \alpha P, \forall x \in \alpha P$

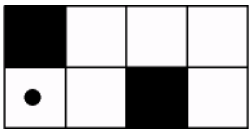
1. Una máquina de ventas estropeada:

(moneda . STOP)

2. Una máquina de ventas que sólo sirve a 2 usuarios

(moneda . (choc . (moneda . (choc . STOP))))

3. Un contador que comienza en la esquina inferior izquierda de un tablero y que puede moverse arriba y a la derecha:



$\alpha TABLERO_1 = \{ arriba, derecha \}$
$TABLERO_1 = (derecha . arriba . derecha . derecha . STOP)$

II.5.2 Recursión

$\mu X : A.F(X)$

- Dado el término de procesos ***F(X)*** sin variables libres y $A = \alpha X$.
- La variable ***X*** está ligada por el operador μ.
- ***F(X)*** tiene que ser un término guardado para poseer solución única.
- La expresión $\mu X : A.F(X)$ representa la solución a la ecuación $X = F(X)$.

Ejemplo 4. El reloj perpetuo

$$CLOCK = \mu X . (tick . X)$$

Ejemplo 5. Una máquina de ventas simple que sirve continuamente chocolate a sus usuarios.

$$MVS = (moneda . (choc . MVS))$$

Ejemplo 6. Una máquina de cambio que da cambio de 5p (5 pesos) continuamente:

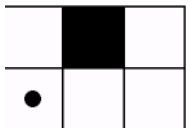
$$\alpha C5p = \{ in5p, out2p, out1p \}$$
$$C5p = (in5p . out2p . outlp . out2p . C5p)$$

II.5.3 Selección

$(x . P + y . Q)$

- Si el primer suceso es *x*, el término de procesos se convierte en *P*
- Si el primer suceso es *y*, se convierte en *Q*
- Los sucesos *x* e *y* han de ser diferentes:
- Los alfabetos son constantes: $\alpha(x . P + y . Q) = \alpha P,$
  $\{ x , y \} \subseteq \alpha P \wedge \alpha P = \alpha Q$

Ejemplo 7. Un nuevo tablero, en este caso son posibles los movimientos alternativos



$$(arriba . STOP + derecha . derecha . arriba . STOP)$$

Ejemplo 8. Una máquina de cambio que ofrece la posibilidad de 2 combinaciones de cambio diferentes:

$$CHAN5C = in5p.\ (\ outlp\ .\ outlp\ .\ outlp\ .\ out2p\ .\ CHAN5C + out2p\ .\ outlp\ .\ out2p\ .\ CHAN5C\ )$$

Ejemplo 9. Una máquina de ventas que sirve chocolate o café en cada transacción

$$MVCHC = \mu X\ .\ moneda\ .\ (\ choc\ .\ X + coffee\ .\ X\ )$$

II.5.4 Operador de Composición Paralela

$P \mid Q$

- Si P puede realizar la comunicación $a \notin \alpha Q$ y se convierte en P' después, entonces $P \mid Q \equiv a.(P' \mid Q)$.
- Si Q puede realizar la comunicación $b \notin \alpha P$ y se convierte en Q' después, entonces $P \mid Q \equiv b.(P \mid Q')$.
- Si P y Q pueden realizar una comunicación común c y después se convierten en P' y Q' respectivamente, entonces $P \mid Q \equiv c.(P' \mid Q')$.

Ejemplo 10. Un usuario tacaño quiere obtener cosas de la máquina sin pagar:

$$\begin{aligned} \alpha MVCHC &= \alpha GRD\_USR = \{moneda,\ choc,\ cafe\} \\ MVCHC &= \mu X.moneda.(choc.X + cafe.X) \\ \\ GRD\_USR &= (cafe.GRD\_USR \\ &\quad + choc.GRD\_USR \\ &\quad + moneda.choc.GRD\_USR) \end{aligned}$$

Si el usuario tacaño intenta obtener cosas de la máquina MVCHC, ésta nunca le dará café:

$$(GRD\_USR | MVCHC) = \mu X.(moneda.choc.X)$$

II.5.5 Ocultamiento

$P \backslash a$

- Resulta útil abstraer (ignorar) parte de la actividad de comunicación de un proceso.
- $P \backslash a$ es un proceso en el cuál la participación de la comunicación a no es visible

Ejemplo 11. Una máquina de ventas ruidosa,

$$NOISYMV = moneda.clink.choc.clock.NOISYMV$$

resulta aislada si ocultamos las acciones siguientes:

$$NOISYMV \backslash \{clink, clonk, cafe\} = MVS$$

II.5.6 Renombrado

$f : \alpha P.A$

- El término de procesos $f(P)$ se habría implicado en el suceso $f(c)$ siempre que el término de procesos P lo hubiera hecho en el suceso c.

- Es cierto que $\alpha f(P) = f(\alpha P)$.

- El renombrado de símbolos que sucede después de aplicar una función inyectiva no modifica el comportamiento de comunicación del proceso.

- El renombrado es útil cuando se ha de construir un conjunto de procesos con el mismo código, pero dichos procesos no han de comunicarse entre ellos.

Ejemplo 12. Hay tres personas sentadas alrededor de una mesa. Cada una tiene una pila de papel impreso. El comportamiento de cada persona consiste en colocar un papel en la mesa y esperar a que una cuarta persona tome un papel de cada pila y los engrape.

$$Persona = papel.signal.wait.Persona$$

$$Persona_i = f_i(Persona) = [papel_i \mid papel] \; [signal_i \mid signal] \; [wait_i \mid wait]$$

$$Encuadernador = signal_1.tomar_1.wait_1.... \; signal_3.tomar_3.wait_3.Encuadernador$$

$$Encuadernar = (|_{i\in\{1,...,n\}} Persona_i) | Encuadernador$$

II.6. Programando con Communicating Sequential Processes

Para describir la comunicación mediante mensajes en CSP es necesario extender el álgebra de procesos con una clase especial de sucesos denominada *comunicación* y denotada por el siguiente par $c.v$:
c: nombre de canal que sirve para comunicar a 2 procesos.
v: mensaje enviado por el canal.

Considerar ahora el alfabeto de comunicación del proceso:

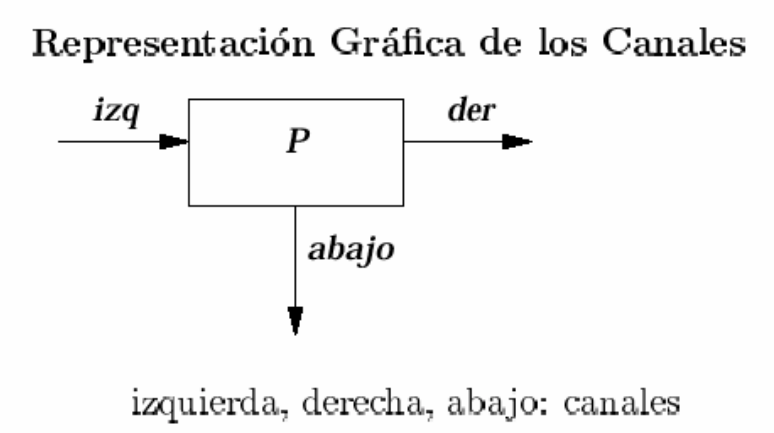
$$\alpha c(P) = \{ v / \; c.v \in \alpha P \}$$

II.6.1. Ordenes de Comunicación: *Operaciones de Entrada/Salida*

Dado el mensaje $v \in \alpha c(P)$:

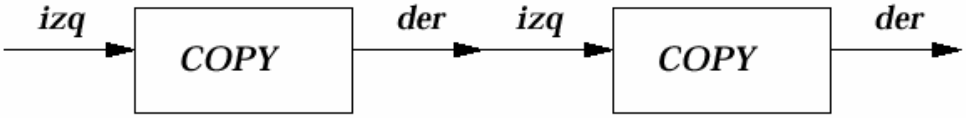
Mensaje de Salida: ***(c ! v → P)***
Mensaje de Entrada: ***(c ? x → P(x))***

Todo canal es utilizado para comunicar exactamente 2 procesos, uno que realiza la salida y el otro la entrada en dicho canal.

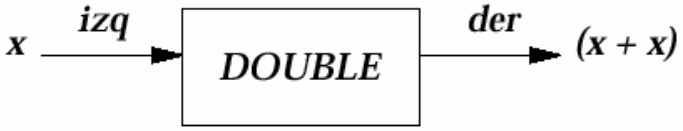


Ejemplo 13. dos buffers que almacenan sólo un elemento

$$\text{COPY} = \mu X.(izq?x \rightarrow der!x \rightarrow X)$$



$$\text{DOUBLE} = \mu X.(izq?x \rightarrow der!(x + x) \rightarrow X)$$



Ejemplo 14. Reemplazar cada par de asteriscos consecutivos "**" por "↑":

$$\alpha_{izq}(\text{SQUASH}) = \alpha_{der}\ (\text{SQUASH})\ \text{-}\{\uparrow\}$$

$$\begin{aligned}\text{SQUASH} = \mu\text{X. izq ? x} \rightarrow\ & \\ \text{IF x} \neq\ ^{*}\ \text{THEN}\ & \ (\text{der ! x} \rightarrow \text{X}) \\ \text{ELSE izq ? y} \rightarrow\ & \ (\text{IF y} = {}^{*}\ \text{THEN (der ! } \uparrow \rightarrow \text{X))} \\ & \ \text{ELSE (der ! } {}^{*} \rightarrow \text{der ! y} \rightarrow \text{X)}\end{aligned}$$

II.6.2. Orden Paralela

Dados *P y Q*, estos dos procesos satisfacen:

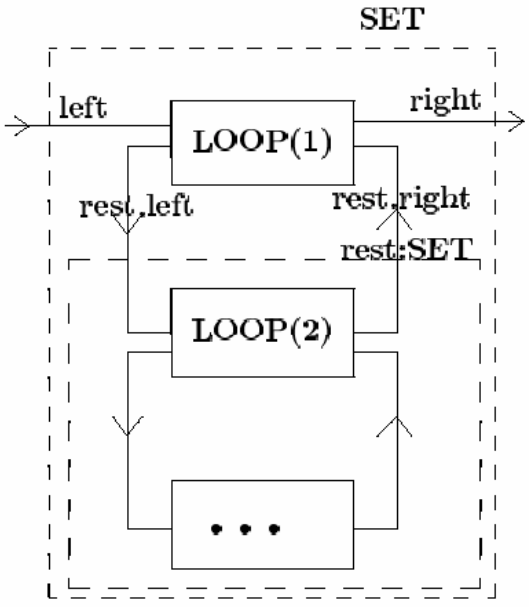
$$input.channel(P) = c = output.channel(Q)\ y\ \ \alpha c(P) = \alpha c(Q)$$

La construcción paralela $(P \mid Q)$ significa que existe *handshaking* entre la salida de *Q* y la entrada de *P* a través del canal *c*.

Ejemplo 15:

$$SET = left?x \rightarrow right!NO \rightarrow (rest : SET \mid LOOP(x))$$

$$\begin{aligned}\text{LOOP (x): } \mu\text{ X. left ? y} \rightarrow\ & \text{(IF y = x THEN right ! YES} \rightarrow \text{X} \\ & \text{ELSE (rest.left ! y} \rightarrow \text{rest.right ? z} \\ & \rightarrow \text{right ! z} \rightarrow \text{X)}\end{aligned}$$



traza: s= <left.1, right.NO, left.2, right.NO>

$$SET/ < left.1 >= right!NO \rightarrow (rest :| LOOP(1))$$

$$\Rightarrow SET/ < left.1, right.NO >= (rest : SET \parallel LOOP(1))$$

$$\Rightarrow SET/ < left.1, right.NO, left.2 >= (rest : SET \mid rest.left!2 \rightarrow rest.right?z \rightarrow right!z \rightarrow LOOP(1))$$

$$\Rightarrow (rest : (right!NO \rightarrow (rest : SET \parallel LOOP(2))) \parallel rest.right?z \rightarrow right!z \rightarrow LOOP(1))$$

$$\Rightarrow SET/ < left.1, right.NO, left.2, right.NO > = (rest : SET \parallel LOOP(2)) \mid LOOP(1))$$

II.6.3. Orden Alternativa

Considerar ***c*** y ***d***, que son 2 canales diferentes, y después considerar el siguiente término de procesos:

$$(\ c\ ?\ x \rightarrow P(x)\ \mid\ d\ ?\ y \rightarrow Q(y)\ )$$

Denota un proceso que o bien recibe ***x*** inicialmente en el canal ***c*** y después se convierte en el término ***P(x)***, o recibe ***y*** en el canal ***d*** y después se convierte en el término ***Q(y)***.

La selección real dependerá de la primera orden de salida que se complete con éxito por cualquier otro proceso en cualquiera de los canales.

Si varios procesos han realizado simultáneamente órdenes de salida en varios canales, entonces se selecciona uno de ellos no-determinísticamente (si no, podría existir peligro de ínter bloqueo).

Ejemplo 16. Inter Bloqueo (si el canal *d* es siempre determinísticamente seleccionado en P_2):

$$P_1 : (c!2 \rightarrow d!4 \rightarrow P) \parallel P_2 : (c?x \rightarrow P(x) \mid d?y \rightarrow Q(y))$$

Ejemplo 17. Un proceso no-determinístico

$$\begin{aligned} \alpha_{izq1} &= \alpha_{izq2} = \alpha_{der} \\ MERGE &= (izq1?x \to der!x \to MERGE \mid izq2?x \to der!x \\ &\quad \to MERGE) \end{aligned}$$

Ejemplo 18. Un proceso buffer

$$\begin{aligned} BUFFER &= P<> \\ P<> &= izq?x \to P<x> \\ P<x>\cdot s &= (izq?y \to P<x>\cdot s\cdot <y> \mid der!x \to Ps) \end{aligned}$$

Ejemplo 19. Un proceso Pila

$$\begin{aligned} STACK &= P<> \\ P<> &= (empty \to P<> \mid izq?x \to P<x>) \\ P<x>\cdot s &= (izq?y \to P<y>\cdot <x>\cdot s \mid der!x \to Ps) \end{aligned}$$