π-cálculo: Tan sólo una introducción!

Carlos Martínez Méndez

Workshop en Verificación: (2^{da} *Parte*)
Escuela de Ingeniería Informática
Facultad de Ingeniería
Universidad Diego Portales

Enero 11, 2007

Preliminares

Necesitamos demarcar el contexto de estudio:

Preliminares

Necesitamos demarcar el contexto de estudio:

- *Concurrency Theory:*

Preliminares

Necesitamos demarcar el contexto de estudio:

- *Concurrency Theory:* Es el estudio de programmaciones paralelas, o bien distribuidas en ciencias de la computación,

Preliminares

Necesitamos demarcar el contexto de estudio:

- *Concurrency Theory:* Es el estudio de programmaciones paralelas, o bien distribuidas en ciencias de la computación, Algunos ingredientes en este paradigma, sincronizaciones, comunicación, etc.

Preliminares

Necesitamos demarcar el contexto de estudio:

- *Concurrency Theory:* Es el estudio de programmaciones paralelas, o bien distribuidas en ciencias de la computación, Algunos ingredientes en este paradigma, sincronizaciones, comunicación, etc.
- *Process Álgebra:*

Preliminares

Necesitamos demarcar el contexto de estudio:

- *Concurrency Theory:* Es el estudio de programmaciones paralelas, o bien distribuidas en ciencias de la computación, Algunos ingredientes en este paradigma, sincronizaciones, comunicación, etc.
- *Process Álgebra:* Es un termino acuñado a mediados de los setentas por **Bergstra** y **Klop**.

Preliminares

Necesitamos demarcar el contexto de estudio:

- *Concurrency Theory:* Es el estudio de programmaciones paralelas, o bien distribuidas en ciencias de la computación, Algunos ingredientes en este paradigma, sincronizaciones, comunicación, etc.
- *Process Álgebra:* Es un termino acuñado a mediados de los setentas por **Bergstra** y **Klop**. Se entiende como el estudio de comportamiento *(Behavior)* de un sistema *(Process, Agent)*, donde por ejemplo la execución de un sotfware, o bien una maquina pueden ser visto como tal.

Sucesión de Puntos de Vistas

Progresión de las Álgebras de Procesos:

Sucesión de Puntos de Vistas

Progresión de las Álgebras de Procesos:

- *Calculus of Communicating Systems* (**CCS**), Milner 1973 - 1980.

Sucesión de Puntos de Vistas

Progresión de las Álgebras de Procesos:

- *Calculus of Communicating Systems* (**CCS**), Milner 1973 - 1980.
- *Communicating Sequential Processing* (**CSP**) , Hoare 1975 - 1985.

Sucesión de Puntos de Vistas

Progresión de las Álgebras de Procesos:

- *Calculus of Communicating Systems* (**CCS**), Milner 1973 - 1980.
- *Communicating Sequential Processing* (**CSP**) , Hoare 1975 - 1985.
- *Algebra of Communicating Processes* (**ACP**), Bergstra, Klop 1982 - 1990.

Sucesión de Puntos de Vistas

Progresión de las Álgebras de Procesos:

- *Calculus of Communicating Systems* (**CCS**), Milner 1973 - 1980.
- *Communicating Sequential Processing* (**CSP**) , Hoare 1975 - 1985.
- *Algebra of Communicating Processes* (**ACP**), Bergstra, Klop 1982 - 1990.
- **A Calculus of Mobile Processing** (π - **Calculus**), Milner, Parrow, Walker 1989.

Sucesión de Puntos de Vistas

Progresión de las Álgebras de Procesos:

- *Calculus of Communicating Systems* (**CCS**), Milner 1973 - 1980.
- *Communicating Sequential Processing* (**CSP**) , Hoare 1975 - 1985.
- *Algebra of Communicating Processes* (**ACP**), Bergstra, Klop 1982 - 1990.
- **A Calculus of Mobile Processing** (π - **Calculus**), Milner, Parrow, Walker 1989.
- *Join Calculus*, Fournet, Gonthier 1995.

Sucesión de Puntos de Vistas

Progresión de las Álgebras de Procesos:

- *Calculus of Communicating Systems* (**CCS**), Milner 1973 - 1980.
- *Communicating Sequential Processing* (**CSP**) , Hoare 1975 - 1985.
- *Algebra of Communicating Processes* (**ACP**), Bergstra, Klop 1982 - 1990.
- **A Calculus of Mobile Processing** (π - **Calculus**), Milner, Parrow, Walker 1989.
- *Join Calculus*, Fournet, Gonthier 1995.
- *Ambient Calculus*, Cardelli, Gordon 1998.

Reseña: CCS

Principal contribuidor: **Robin Milner**

Reseña: CCS

Principal contribuidor: **Robin Milner**

- Las primeras publicaciónes se centraron en la formulación semantica de la *composición paralela* en el contexto de semanticas denotacionales, otras de las operaciones que introduce: $*$ composición secuencial, ? composición alternativa, $||$ composición paralela.

Reseña: CCS

Principal contribuidor: **Robin Milner**

- Las primeras publicaciónes se centraron en la formulación semantica de la *composición paralela* en el contexto de semanticas denotacionales, otras de las operaciones que introduce: $*$ composición secuencial, ? composición alternativa, $||$ composición paralela.
- Introduce los *Flow graphs* diagramas que permiten indicar nombre sincronizaciones de puertos e indicar sus co-nombres.

Reseña: CCS

Principal contribuidor: **Robin Milner**

- Las primeras publicaciónes se centraron en la formulación semantica de la *composición paralela* en el contexto de semanticas denotacionales, otras de las operaciones que introduce: $*$ composición secuencial, ? composición alternativa, $||$ composición paralela.
- Introduce los *Flow graphs* diagramas que permiten indicar nombre sincronizaciones de puertos e indicar sus co-nombres.
- Posteriormente introduce nuevas operaciones (prefijos) con sus respectivas reglas. Utiliza *Árboles de Sincronización* como modelos. Obteniendo la presentación finalmente aceptada de *CSS*.

Reseña: CCS

Principal contribuidor: **Robin Milner**

- Las primeras publicaciónes se centraron en la formulación semantica de la *composición paralela* en el contexto de semanticas denotacionales, otras de las operaciones que introduce: $*$ composición secuencial, ? composición alternativa, $||$ composición paralela.
- Introduce los *Flow graphs* diagramas que permiten indicar nombre sincronizaciones de puertos e indicar sus co-nombres.
- Posteriormente introduce nuevas operaciones (prefijos) con sus respectivas reglas. Utiliza *Árboles de Sincronización* como modelos. Obteniendo la presentación finalmente aceptada de *CSS*.
- Junto con M. Hennessy, formulan de manera inductiva las equivalencias observacionales y fuertes en CSS. Establecen una caracterización lógica de la equivalencia de procesos, la así llamada *Logica de Milner-Hennessy*.

Reseña: CSP

Principal contribuidor: **C.A.R Hoare**

Reseña: CSP

Principal contribuidor: **C.A.R Hoare**

- Adopta la noción *message passing paradigm* en comunicaciones

Reseña: CSP

Principal contribuidor: **C.A.R Hoare**

- Adopta la noción *message passing paradigm* en comunicaciones
- Presenta CSP como un lenguage de comandos que describen la sincronización en la comunicación, sin presentar una semantica formal, CSP motiva al Milner a trabajar en CCS.

Reseña: CSP

Principal contribuidor: **C.A.R Hoare**

- Adopta la noción *message passing paradigm* en comunicaciones
- Presenta CSP como un lenguage de comandos que describen la sincronización en la comunicación, sin presentar una semantica formal, CSP motiva al Milner a trabajar en CCS.
- CSP es refinado en TCSP, donde los modelos eran son teoría de trazas *basados en pares de fallas.*. Este tipo de model resulto ser en el *menor* modelo que discrimina el comportamiento de *deadlock*.

Reseña: ASP

Principales contribuidores: **J. Bergstra, J. Klop**

Reseña: ASP

Principales contribuidores: **J. Bergstra, J. Klop**

- Inicialmente estaban preocupados en encontrar soluciones a equaciones recursivas no acotadas, en este contexto definen formalmente *álgebra de procesos* como sigue:

Reseña: ASP

Principales contribuidores: **J. Bergstra, J. Klop**

- Inicialmente estaban preocupados en encontrar soluciones a equaciones recursivas no acotadas, en este contexto definen formalmente *álgebra de procesos* como sigue:

  Definición Un *álgebra de procesos* sobre un conjunto de *acciones atomicas* A es una estructura $\mathcal{A} := \langle \mathbf{A}, +, \cdot, \lfloor, a_i(i \in \mathcal{I}) \rangle$, donde $\mathbf{A}$ es un conjunto que contiene a A, a_i son símbolos constantes correspondientes a cada $a_i \in A$, y se tiene que $+$ (*única*), $\cdot$ (*concadenación*), $\lfloor$ (*combinación izq.*) satisfacen $\forall x, y, z \in \mathbf{A}$ y $a \in A$ los siguientes axiomas:

Reseña: ASP

Principales contribuidores: **J. Bergstra, J. Klop**

- Inicialmente estaban preocupados en encontrar soluciones a equaciones recursivas no acotadas, en este contexto definen formalmente *álgebra de procesos* como sigue:

  Definición Un *álgebra de procesos* sobre un conjunto de *acciones atomicas* A es una estructura $\mathcal{A} := \langle \mathbf{A}, +, \cdot, \lfloor, a_i (i \in \mathcal{I}) \rangle$, donde $\mathbf{A}$ es un conjunto que contiene a A, a_i son símbolos constantes correspondientes a cada $a_i \in A$, y se tiene que $+$ (*únion*), $\cdot$ (*concadenación*), $\lfloor$ (*combinación izq.*) satisfacen $\forall x, y, z \in \mathbf{A}$ y $a \in A$ los siguientes axiomas:

(PA1)	$x + y = y + x$	**(PA5)**	$(x + y)z = xz + yz$
(PA2)	$x + (y + z) = (x + y) + z$	**(PA6)**	$(x + y)\lfloor z = x\lfloor z + y\lfloor z$
(PA3)	$x + x = x$	**(PA7)**	$ax\lfloor y = a(x\lfloor y + y\lfloor x)$
(PA4)	$(xy)z = x(yz)$	**(PA8)**	$a\lfloor y = ay$

Un par de palabras a tener en cuenta

De Robin Milner con amor:

Un par de palabras a tener en cuenta

De Robin Milner con amor:

- (Concurrencia v/s Secuencial) ...Mirando las nociones básicas para un modelo de concurrencia, probablemente es equivoco extrapolar desde el λ-Calculus, con excepción si estamos siguiendo su ejemplo en la busqueda de algo pequeño y poderoso...

Un par de palabras a tener en cuenta

De Robin Milner con amor:

- (Concurrencia v/s Secuencial) ...Mirando las nociones básicas para un modelo de concurrencia, probablemente es equivoco extrapolar desde el λ-Calculus, con excepción si estamos siguiendo su ejemplo en la busqueda de algo pequeño y poderoso...
- (Sobre el π-Calculus) Tiene como objetivo el definir un modelo con un pequeño número de conceptos básicos, en termino de los cuales el comportamiento de iteracciones y **mobilidad** puede ser rigurosamente descrito...

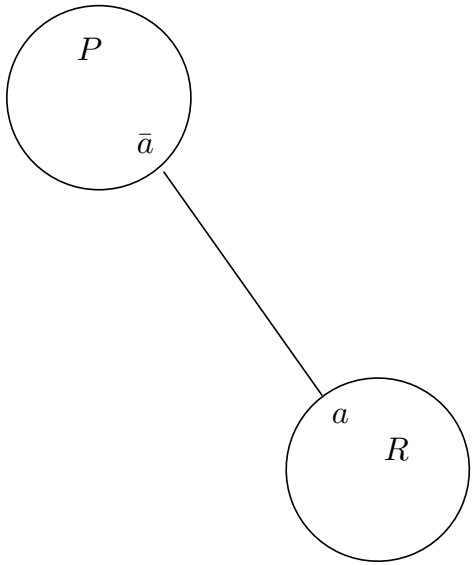
Ejemplo

Trataremos de motivar mediante un ejemplo y notación informal, la propiedad de transmitir canales (*links, referencias*) que es propia de este cálculo, y que no es posible en sus predecesores, especificamente **CCS**.

Ejemplo

Situación: Consideremos el proceso P que desea enviar el valor 5 a otro proceso R a través de un *link* llamado a, tal que R esta disponible a recibir cualquier valor por medio de este link.

Representación Mediante *Flow Graph*



Ejemplo

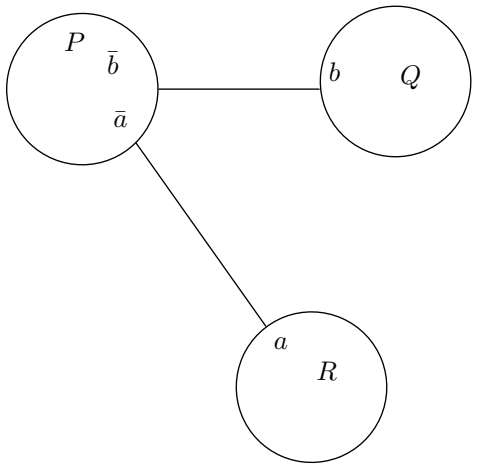
Donde tendremos que cada proceso es respectivamente $P \equiv \bar{a}5.P'$ y $R \equiv a(x).R'$, aquí estamos usando una notación similar a CCS, en que $a(x)$ acota la variable x en R'. Por lo tanto este flow graph estaría representa por le expresión:

$$(\bar{a}5.P'|a(x).R')/a$$

Donde $/a$ representa la restrición de este canal a ambos procesos, una canal privado de P y R

Ejemplo

Consideremos la situación en que P desee delegar esta comunicación a otro proceso Q, la tarea de transmitir 5 a R. Para ellos debemos tener el siguiente diagrama



Como vemos que P y Q esta conectados mediante el link b ($\bar{b}$).

Ejemplo

En esta vez los terminos asociados a nuestra situación son $P \equiv \bar{b}a.\bar{b}5.P'$, que envia a través de b, a y 5 respectivamente.

Ejemplo

En esta vez los terminos asociados a nuestra situación son $P \equiv \bar{b}a.\bar{b}5.P'$, que envia a través de b, a y 5 respectivamente. Supongamos que $Q \equiv b(y).b(z).\bar{y}z.\mathbf{0}$ que recibe un link y un valor a través del canal b.

Ejemplo

En esta vez los terminos asociados a nuestra situación son $P \equiv \bar{b}a.\bar{b}5.P'$, que envia a través de b, a y 5 respectivamente. Supongamos que $Q \equiv b(y).b(z).\bar{y}z.\mathbf{0}$ que recibe un link y un valor a través del canal b. Cabe observar que a no es una expresion en Q, estos quiere decir que no posee un link con R inicialmente. De esto obtenemos el siguiente termino representando la situación anterior.

Ejemplo

En esta vez los terminos asociados a nuestra situación son $P \equiv \bar{b}a.\bar{b}5.P'$, que envia a través de b, a y 5 respectivamente. Supongamos que $Q \equiv b(y).b(z).\bar{y}z.\mathbf{0}$ que recibe un link y un valor a través del canal b. Cabe observar que a no es una expresion en Q, estos quiere decir que no posee un link con R inicialmente. De esto obtenemos el siguiente termino representando la situación anterior.

$$(\bar{b}a.\bar{b}5.P'|b(y).b(z).\bar{y}z.\mathbf{0}|a(x).R')/a/b$$

Ejemplo

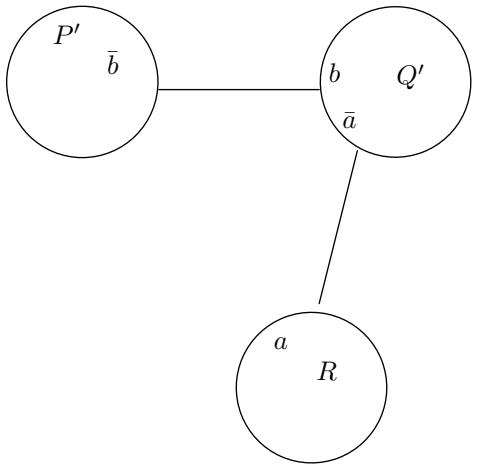
Finalmente después de dos interacciones entre P y Q obtendremos:

Ejemplo

Finalmente después de dos interacciones entre P y Q obtendremos:

$$(P'|\bar{a}5.\mathbf{0}|a(x).R')/a/b$$

Asumiendo que a no ocurre en P' tendremos el siguiente diagrama resultante:

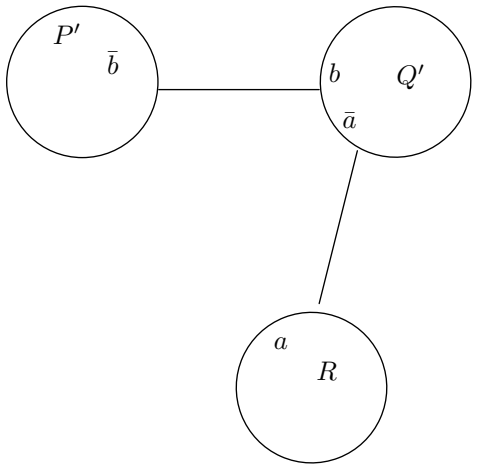


Ejemplo

Finalmente después de dos interacciones entre P y Q obtendremos:

$$(P'|\bar{a}5.\mathbf{0}|a(x).R')/a/b$$

Asumiendo que a no ocurre en P' tendremos el siguiente diagrama resultante:



El hecho importante es que este procedimiento no es posible en **CCS** donde nombres de canales aparecen como argumentos al igual que datos!!

De lo típico

Cada uno con lo suyo:

De lo típico

Cada uno con lo suyo:

Definición *(larga e intuitiva)*

De lo típico

Cada uno con lo suyo:

Definición *(larga e intuitiva)* Consideremos un conjunto infinito de nombres $\mathcal{N}$, sean $u, v, x, y, z, \ldots$ nombres. Sea $\mathcal{P}$ un conjunto de identificadores, cada uno con su ariedad, $A, B, C, \ldots$ variables representado identificadores. Sean $P, Q, R, \ldots$ variables representando expresiones de procesos obtenido recursivamente de la cláusulas siguientes:

De lo típico

Cada uno con lo suyo:

Definición *(larga e intuitiva)* Consideremos un conjunto infinito de nombres $\mathcal{N}$, sean $u, v, x, y, z, \ldots$ nombres. Sea $\mathcal{P}$ un conjunto de identificadores, cada uno con su ariedad, $A, B, C, \ldots$ variables representado identificadores. Sean $P, Q, R, \ldots$ variables representando expresiones de procesos obtenido recursivamente de la cláusulas siguientes:

- *(Suma)* $\sum_{i \in I} P_i$, donde cada conjunto de indices es finito

De lo típico

Cada uno con lo suyo:

Definición *(larga e intuitiva)* Consideremos un conjunto infinito de nombres $\mathcal{N}$, sean $u, v, x, y, z, \ldots$ nombres. Sea $\mathcal{P}$ un conjunto de identificadores, cada uno con su ariedad, $A, B, C, \ldots$ variables representado identificadores. Sean $P, Q, R, \ldots$ variables representando expresiones de procesos obtenido recursivamente de la cláusulas siguientes:

- *(Suma)* $\sum_{i \in I} P_i$, donde cada conjunto de indices es finito
  Este proceso se comporta como uno u otro de los procesos P_i.

De lo típico

Cada uno con lo suyo:

Definición *(larga e intuitiva)* Consideremos un conjunto infinito de nombres $\mathcal{N}$, sean $u, v, x, y, z, \ldots$ nombres. Sea $\mathcal{P}$ un conjunto de identificadores, cada uno con su ariedad, $A, B, C, \ldots$ variables representado identificadores. Sean $P, Q, R, \ldots$ variables representando expresiones de procesos obtenido recursivamente de la cláusulas siguientes:

- *(Suma)* $\sum_{i \in I} P_i$, donde cada conjunto de indices es finito

  Este proceso se comporta como uno u otro de los procesos P_i. Denotaremos por $\mathbf{0}$ la suma vacıa que representará el proceso inactivo.

De lo típico

Cada uno con lo suyo:

Definición *(larga e intuitiva)* Consideremos un conjunto infinito de nombres $\mathcal{N}$, sean $u, v, x, y, z, \ldots$ nombres. Sea $\mathcal{P}$ un conjunto de identificadores, cada uno con su ariedad, $A, B, C, \ldots$ variables representado identificadores. Sean $P, Q, R, \ldots$ variables representando expresiones de procesos obtenido recursivamente de la cláusulas siguientes:

- *(Suma)* $\sum_{i \in I} P_i$, donde cada conjunto de indices es finito

  Este proceso se comporta como uno u otro de los procesos P_i. Denotaremos por $\mathbf{0}$ la suma vacıa que representará el proceso inactivo. Por último es claro como representar las suma binarias, $P_1 + P_2$

Continuación..

- *(Prefijos)* $\bar{y}\langle x\rangle.P$, $y(x).P$, $\tau.P$

Continuación..

- *(Prefijos)* $\bar{y}\langle x\rangle.P$, $y(x).P$, $\tau.P$
  - $\bar{y}\langle x\rangle$ es llamado el *prefijo negativo*, donde $\bar{y}$ puede ser pensado como el *output port* de un proceso que lo contiene. $\bar{y}x.P$ *entrega el nombre de* x *en el puerto* $\bar{y}$ *y luego se comporta como* P

Continuación..

- *(Prefijos)* $\bar{y}\langle x\rangle.P$, $y(x).P$, $\tau.P$
  - $\bar{y}\langle x\rangle$ es llamado el *prefijo negativo*, donde $\bar{y}$ puede ser pensado como el *output port* de un proceso que lo contiene. $\bar{y}x.P$ *entrega el nombre de* x *en el puerto* $\bar{y}$ *y luego se comporta como* P
  - $y(x)$ es llamado el *prefijo positivo*, en este caso podemos pensar en que y es el *input port* de una agente que lo posea. $y(x).P$ *inputs un nombre arbitrario* z *en el puerto* y*, y luego se comporta como* $P\{z/x\}$ *(por definir)*

Continuación..

- *(Prefijos)* $\bar{y}\langle x\rangle.P$, $y(x).P$, $\tau.P$
  - $\bar{y}\langle x\rangle$ es llamado el *prefijo negativo*, donde $\bar{y}$ puede ser pensado como el *output port* de un proceso que lo contiene. $\bar{y}x.P$ *entrega el nombre de* x *en el puerto* $\bar{y}$ *y luego se comporta como* P
  - $y(x)$ es llamado el *prefijo positivo*, en este caso podemos pensar en que y es el *input port* de una agente que lo posea. $y(x).P$ *inputs un nombre arbitrario* z *en el puerto* y*, y luego se comporta como* $P\{z/x\}$ *(por definir)*
  - τ es llamado el *prefijo silencioso*, $\tau.p$ *desempeña una acción silenciosa y luego se comporta como* P

Continuación...

- *(Composición)* $P_1|P_2$

Continuación...

- *(Composición)* $P_1|P_2$
  Esta vez este proceso representará dos procesos ejecutadose en paralelo, cada componete puede actuar de forma independiente, o bien una acción *output* de P_1 en cualquier puerto de salida $\bar{x}$ puede ser sincronizada con una acción *input* de P_2 en x, creando un acción τ de $P_1|P_2$

Continuación...

- *(Composición)* $P_1|P_2$
  Esta vez este proceso representará dos procesos ejecutadose en paralelo, cada componete puede actuar de forma independiente, o bien una acción *output* de P_1 en cualquier puerto de salida $\bar{x}$ puede ser sincronizada con una acción *input* de P_2 en x, creando un acción τ de $P_1|P_2$
- *(Restricción)* $(x).P$

Continuación...

- *(Composición)* $P_1|P_2$
  Esta vez este proceso representará dos procesos ejecutadose en paralelo, cada componete puede actuar de forma independiente, o bien una acción *output* de P_1 en cualquier puerto de salida $\bar{x}$ puede ser sincronizada con una acción *input* de P_2 en x, creando un acción τ de $P_1|P_2$
- *(Restricción)* $(x).P$
  Este proceso se comporta como P con excepción de las acciones del puerto $\bar{x}$ y x que son prohibidas.

Continuación...

- *(Composición)* $P_1|P_2$
  Esta vez este proceso representará dos procesos ejecutadose en paralelo, cada componete puede actuar de forma independiente, o bien una acción *output* de P_1 en cualquier puerto de salida $\bar{x}$ puede ser sincronizada con una acción *input* de P_2 en x, creando un acción τ de $P_1|P_2$
- *(Restricción)* $(x).P$
  Este proceso se comporta como P con excepción de las acciones del puerto $\bar{x}$ y x que son prohibidas. *(Notar eso si que la comunicación entre componentes en P no esta prohibida ya que estas acciones darán paso a acciones silenciosas τ)*

Continuación...

- *(Composición)* $P_1|P_2$
  Esta vez este proceso representará dos procesos ejecutadose en paralelo, cada componete puede actuar de forma independiente, o bien una acción *output* de P_1 en cualquier puerto de salida $\bar{x}$ puede ser sincronizada con una acción *input* de P_2 en x, creando un acción τ de $P_1|P_2$
- *(Restricción)* $(x).P$
  Este proceso se comporta como P con excepción de las acciones del puerto $\bar{x}$ y x que son prohibidas. *(Notar eso si que la comunicación entre componentes en P no esta prohibida ya que estas acciones darán paso a acciones silenciosas τ)*
- *(Matching)* $[x = y]P$

Continuación...

- *(Composición)* $P_1|P_2$
  Esta vez este proceso representará dos procesos ejecutadose en paralelo, cada componete puede actuar de forma independiente, o bien una acción *output* de P_1 en cualquier puerto de salida $\bar{x}$ puede ser sincronizada con una acción *input* de P_2 en x, creando un acción τ de $P_1|P_2$
- *(Restricción)* $(x).P$
  Este proceso se comporta como P con excepción de las acciones del puerto $\bar{x}$ y x que son prohibidas. *(Notar eso si que la comunicación entre componentes en P no esta prohibida ya que estas acciones darán paso a acciones silenciosas τ)*
- *(Matching)* $[x = y]P$
  Este proceso se comporta como P si lo nombres de x e y son idénticos, de otra forma se comporta como $\mathbf{0}$

Observaciones

- *(Identificadores)* $A(y_1, y_2, \ldots, y_n)$

Observaciones

- *(Identificadores)* $A(y_1, y_2, \ldots, y_n)$
  Por cada identificador de procesos A de ariedad n, debe haber una única ecuación definidora $A(x_1, x_2, \ldots, x_n) \equiv^{def} P$, donde los nombres $x_1, x_2, \ldots x_n$ son distintos y son los únicos nombre que ocurren libremente en P.

Observaciones

- *(Identificadores)* $A(y_1, y_2, \ldots, y_n)$
  Por cada identificador de procesos A de ariedad n, debe haber una única ecuación definidora $A(x_1, x_2, \ldots, x_n) \equiv^{def} P$, donde los nombres $x_1, x_2, \ldots x_n$ son distintos y son los únicos nombre que ocurren libremente en P.

Nota: Definiciones a través de identificadores provee recursión, ya que P puede poseer Identificadores, en particular A

En resumen

Definición *(Corta)*

$$
\begin{array}{rcl}
P & := & \mathbf{0} \\
 & | & P_1 + P_2 \\
 & | & \bar{y}\langle x \rangle.P \\
 & | & y(x).P \\
 & | & \tau.P \\
 & | & P_1 | P_2 \\
 & | & (x)P \\
 & | & [x = y]P \\
 & | & A(y_1, y_2, \ldots, y_n)
\end{array}
$$

Que más?

- Al eliminar los *identificadores*, nos quedamos con solo procesos finitarios,

Que más?

- Al eliminar los *identificadores*, nos quedamos con solo procesos finitarios, En cambio *matching* es tan sólo *sintatic sugar*, especial para representar computaciones en estructura de datos.

Que más?

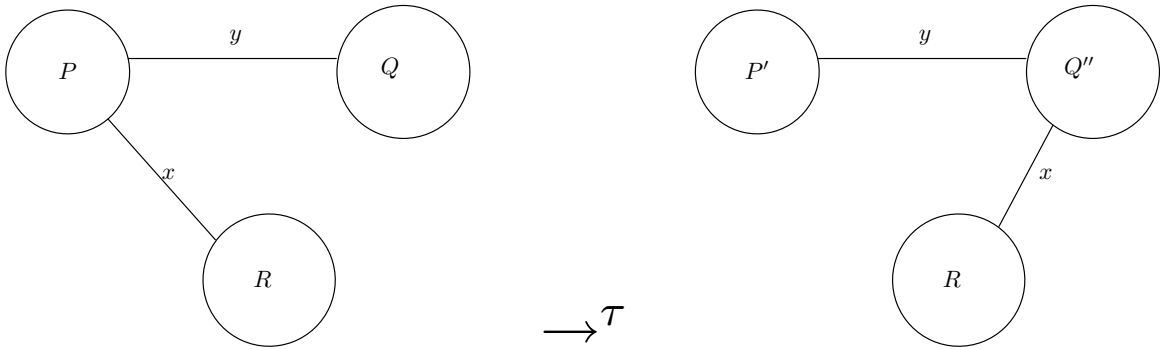
- Al eliminar los *identificadores*, nos quedamos con solo procesos finitarios, En cambio *matching* es tan sólo *sintatic sugar*, especial para representar computaciones en estructura de datos.
- Finalmente, se adscribe a las convenciones y definiciones usuales de *nombre libres, nombres acotados, α-convertibilidad, sustituciones, etc*

El Ejemplo?

El caso es cuando P tiene un link con R, deseando pasar x a través de su link y a Q, donde Q esta disponible a recibirlo. Sean $P \equiv \bar{y}x.P'$ y eventualmente $y(z).Q'$, en este caso tenemos:

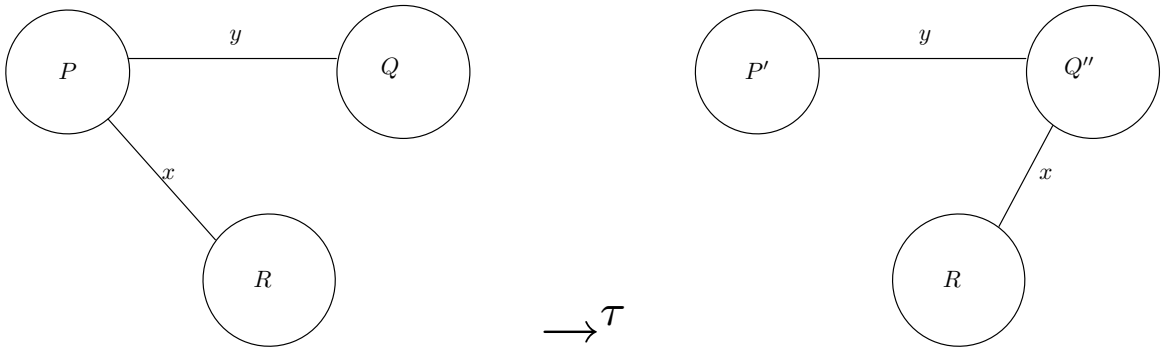
El Ejemplo?

El caso es cuando P tiene un link con R, deseando pasar x a través de su link y a Q, donde Q esta disponible a recibirlo. Sean $P \equiv \bar{y}x.P'$ y eventualmente $y(z).Q'$, en este caso tenemos:



El Ejemplo?

El caso es cuando P tiene un link con R, deseando pasar x a través de su link y a Q, donde Q esta disponible a recibirlo. Sean $P \equiv \bar{y}x.P'$ y eventualmente $y(z).Q'$, en este caso tenemos:



Que sintacticamente la transición es:

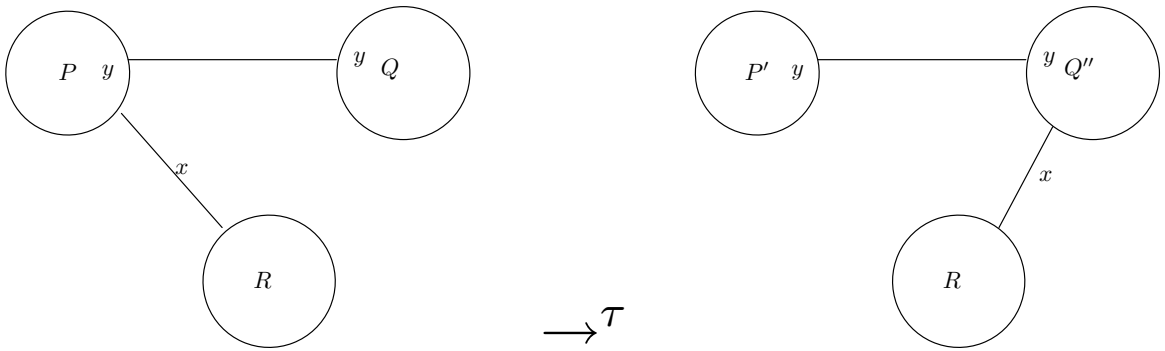
$$\bar{y}x.P'|y(z).Q'|R \rightarrow^{\tau} P'|Q'\{x/z\}|R$$

El Ejemplo?

Que tal si y es un canal privado entre P y Q, en este caso el diagrama apropiado es:

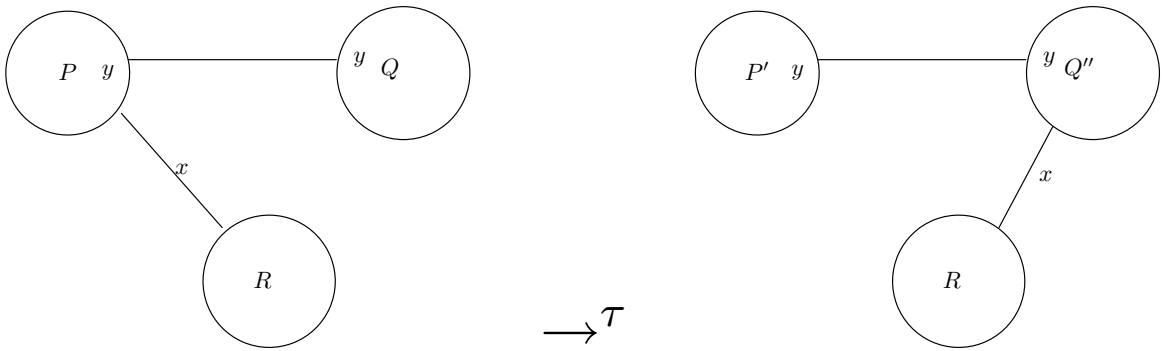
El Ejemplo?

Que tal si y es un canal privado entre P y Q, en este caso el diagrama apropiado es:



El Ejemplo?

Que tal si y es un canal privado entre P y Q, en este caso el diagrama apropiado es:



Que sintacticamente es:

$$(y)(\bar{y}x.P'|y(z).Q')|R \rightarrow^{\tau} (y)(P'|Q'\{x/z\})|R$$

Otro Ejemplo

Pronto podremos establecer ciertas identidades entre procesos, la siguiente es bastante intuitiva,

Otro Ejemplo

Pronto podremos establecer ciertas identidades entre procesos, la siguiente es bastante intuitiva, no?

$$(x)(P_1|P_2) = P_1|(x)(P_2), \text{siendo } x \notin fn(P)$$

Otro Ejemplo

Pronto podremos establecer ciertas identidades entre procesos, la siguiente es bastante intuitiva, no?

$$(x)(P_1|P_2) = P_1|(x)(P_2), \text{siendo } x \notin fn(P)$$

Consideremos ahora el siguiente identificador:

$$Exec(x) =^{def} x(y).\bar{y}$$

Intuitivamente este proceso solo se comunica a través del canal x, llamando el canal y y activandolo. Podemos pensar en que y es el *gatillador* de un proceso que ha sido ejecutado por *Exec*

El otro ejemplo

Consideremos un proceso P queremos obtener el mismo comportamiento en los siguientes casos:

- Ejecutar P directamente.

El otro ejemplo

Consideremos un proceso P queremos obtener el mismo comportamiento en los siguientes casos:

- Ejecutar P directamente.
- Fijando un gatillador z en P de tal manera que pasando z junto a canal x al *Exec* (Asumiendo que $x, z \notin fn(P)$)

El otro ejemplo

Consideremos un proceso P queremos obtener el mismo comportamiento en los siguientes casos:

- Ejecutar P directamente.
- Fijando un gatillador z en P de tal manera que pasando z junto a canal x al *Exec* (Asumiendo que $x, z \notin fn(P)$)

Sea

$$(z)(\bar{x}z|z.P)$$

Que debería comportarse como P bajo interacciones con *Exec*, es decir:

$$(x)((z)(\bar{x}z|z.P)|Exec(x))$$

Desde este ejemplo

Tendremos las reduciones:

$$\frac{\frac{\frac{(x)((z)(\bar{x}z|z.P)|Exec(x))}{(x)((z)(\bar{x}z|z.P)|x(y).\bar{y})}}{\tau.(x)(z)(\mathbf{0}|z.P|\bar{z})}}{\tau.\tau.(x)(z)(\mathbf{0}|P|\mathbf{0})}$$

Finalmente, asumimos que $x, z \notin fn(P)$, es decir

$$\frac{}{\tau.\tau.P}$$

Que es ... a P

Motivación

Los ejemplos anteriores hacen necesario una noción de *congruencia estructural (Alt: Strong Groung Equivalences)* sobre el conjunto de procesos, que no permita identificar los *mismo* procesos:

Motivación

Los ejemplos anteriores hacen necesario una noción de *congruencia estructural (Alt: Strong Groung Equivalences)* sobre el conjunto de procesos, que no permita identificar los *mismo* procesos:

Definición Definiremos la *congruencia estructural*, que denotamos por $\equiv$ a la congruencia más pequeña cerrada bajo las siguientes reglas:

- Si P y Q son α-congruentes entonces $P \equiv Q$

Motivación

Los ejemplos anteriores hacen necesario una noción de *congruencia estructural (Alt: Strong Groung Equivalences)* sobre el conjunto de procesos, que no permita identificar los *mismo* procesos:

Definición Definiremos la *congruencia estructural*, que denotamos por $\equiv$ a la congruencia más pequeña cerrada bajo las siguientes reglas:

- Si P y Q son α-congruentes entonces $P \equiv Q$
- Los operadores $|$ y $+$ forman un monoide abeliano conmutativo con $\mathbf{0}$. Esto es,

$$\begin{array}{lcl} P|Q & \equiv & Q|P \\ P|(Q|R) & \equiv & (P|Q)|R \\ \mathbf{0}|P & \equiv & P \end{array}$$

- *(Unfolding)* $A(\vec{y}) \equiv P\{\vec{y}/\vec{x}\}$, si $A(\vec{x}) =^{def} P$

Continuación

- *(Restricciones)*

Continuación

- *(Restricciones)*

$$
\begin{array}{lcl}
(x)\mathbf{0} & \equiv & \mathbf{0}, \\
(x)(y)P & \equiv & (y)(x)P \\
(x)(P+Q) & \equiv & (x)P + (x)Q \\
(x)\alpha.P & \equiv & \alpha.(x)P \text{ Si } x \neq \alpha \\
(x)\alpha.P & \equiv & \mathbf{0}, \text{ Si } x = \alpha
\end{array}
$$

Continuación

- *(Restricciones)*

$$\begin{array}{lcl} (x)\mathbf{0} & \equiv & \mathbf{0}, \\ (x)(y)P & \equiv & (y)(x)P \\ (x)(P+Q) & \equiv & (x)P + (x)Q \\ (x)\alpha.P & \equiv & \alpha.(x)P \text{ Si } x \neq \alpha \\ (x)\alpha.P & \equiv & \mathbf{0}, \text{ Si } x = \alpha \end{array}$$

- *(Matching)*

Continuación

- *(Restricciones)*

$$\begin{array}{lcl} (x)\mathbf{0} & \equiv & \mathbf{0}, \\ (x)(y)P & \equiv & (y)(x)P \\ (x)(P+Q) & \equiv & (x)P + (x)Q \\ (x)\alpha.P & \equiv & \alpha.(x)P \text{ Si } x \neq \alpha \\ (x)\alpha.P & \equiv & \mathbf{0}, \text{ Si } x = \alpha \end{array}$$

- *(Matching)*

$$\begin{array}{lcl} [x=y]P & \equiv & \mathbf{0}, \text{ Si } x \notin y \\ [x=x]P & \equiv & P \end{array}$$

Ejemplos

- $(x)(P|Q)) \equiv ((x)P|(x)Q)$?

Ejemplos

- $(x)(P|Q)) \equiv ((x)P|(x)Q)$?
  Resp:

Ejemplos

- $(x)(P|Q)) \equiv ((x)P|(x)Q)$?
  Resp: Nop,

Ejemplos

- $(x)(P|Q)) \equiv ((x)P|(x)Q)$?
  Resp: Nop, ya que en el primer procedimiento P y Q pueden interactuar mediante x, en cambio en el segundo no es el caso, sin embargo, en ambos casos se restrigen las mismas ocurrencias de x.

Ejemplos

- $(x)(P|Q)) \equiv ((x)P|(x)Q)$?
  Resp: Nop, ya que en el primer procedimiento P y Q pueden interactuar mediante x, en cambio en el segundo no es el caso, sin embargo, en ambos casos se restrigen las mismas ocurrencias de x.
- $(x)P \equiv P$ siendo $x \notin fn(P)$?

Ejemplos

- $(x)(P|Q)) \equiv ((x)P|(x)Q)$?
  Resp: Nop, ya que en el primer procedimiento P y Q pueden interactuar mediante x, en cambio en el segundo no es el caso, sin embargo, en ambos casos se restrigen las mismas ocurrencias de x.
- $(x)P \equiv P$ siendo $x \notin fn(P)$?
  Resp:

Ejemplos

- $(x)(P|Q)) \equiv ((x)P|(x)Q)$?
  Resp: Nop, ya que en el primer procedimiento P y Q pueden interactuar mediante x, en cambio en el segundo no es el caso, sin embargo, en ambos casos se restrigen las mismas ocurrencias de x.
- $(x)P \equiv P$ siendo $x \notin fn(P)$?
  Resp: Sip,

Ejemplos

- $(x)(P|Q)) \equiv ((x)P|(x)Q)$?
  Resp: Nop, ya que en el primer procedimiento P y Q pueden interactuar mediante x, en cambio en el segundo no es el caso, sin embargo, en ambos casos se restrigen las mismas ocurrencias de x.
- $(x)P \equiv P$ siendo $x \notin fn(P)$?
  Resp: Sip, ya que:

$$\begin{aligned} P &\equiv P|\mathbf{0} \\ &\equiv P|(x).P \\ &\equiv (x)(P|\mathbf{0}) \\ &\equiv (x).P \end{aligned}$$

Motivación

Consideremos

$$P := a(x).([x = y]Q + [x = z]R)$$

En este caso donde el proceso P recibe un nombre a, y ejecuta Q, si el nombre es y, y ejecuta R si el nombre es z.

Motivación

Consideremos

$$P := a(x).([x = y]Q + [x = z]R)$$

En este caso donde el proceso P recibe un nombre a, y ejecuta Q, si el nombre es y, y ejecuta R si el nombre es z. Puede ser implementado mediante los operadores $|$ y $+$ como sigue:

$$P = a(x).(\bar{x}.|(y.Q|z.R))$$

Motivación

Consideremos

$$P := a(x).([x = y]Q + [x = z]R)$$

En este caso donde el proceso P recibe un nombre a, y ejecuta Q, si el nombre es y, y ejecuta R si el nombre es z. Puede ser implementado mediante los operadores $|$ y $+$ como sigue:

$$P = a(x).(\bar{x}.|(y.Q|z.R))$$

Esta vez, al recibir el nombre y y comunicarse con el primer, o segundo proceso ejecutará Q, o bien R respectivamente. Notar que esto simular el matching, solo en el caso en que no existe otro proceso que interfiera con esta comunicación.

Motivación

Hasta ahora:

- *Cálculo Monadico:*

$$\pi := \begin{cases} \bar{x}\langle y \rangle, & \text{envia } \textbf{un} \text{ nombre } y \text{ por medio de } x; \\ x(y), & \text{recibe } \textbf{un} \text{ nombre parametrizado por } y \text{ por medio } x; \\ \tau, & \text{acción silenciosa.} \end{cases}$$

Motivación

Hasta ahora:

- *Cálculo Monadico:*

$$\pi := \begin{cases} \bar{x}\langle y\rangle, & \text{envia } \textbf{un} \text{ nombre } y \text{ por medio de } x; \\ x(y), & \text{recibe } \textbf{un} \text{ nombre parametrizado por } y \text{ por medio } x; \\ \tau, & \text{acción silenciosa.} \end{cases}$$

- *Representación Cálculo Poliadico*

Que tal esta representación:

$$\begin{aligned} x(\overrightarrow{y}).P &:= x(y_1).x(y_2)\ldots.x(y_n).P \\ \bar{x}\langle\overrightarrow{z}\rangle.\mathrm{Q} &:= \bar{x}\langle z_1\rangle.\bar{x}\langle z_2\rangle\ldots.\bar{x}\langle z_n\rangle.Q? \end{aligned}$$

What's wrong in this picture?

Contra Ejemplo:

Considere:

$$x(y_1 y_2).P|\bar{x}\langle z_1, z_2\rangle.Q_1|\bar{x}\langle w_1 w_2\rangle$$

....

Solución

Traducción:

$$\begin{aligned} x(\overrightarrow{y}).P &:= x(w)w(y_1).w(y_2)\ldots.w(y_n).P_0 \\ \bar{x}\langle\overrightarrow{z}\rangle.\mathrm{Q} &:= (w)(\bar{x}\langle w\rangle\bar{w}\langle z_1\rangle.\bar{w}\langle z_2\rangle\ldots.\bar{w}\langle z_n\rangle.Q_0) \end{aligned}$$

Donde P_0 y Q_0 son las respectivas traducciones de P y Q

Solución

Traducción:

$$\begin{array}{lcl} x(\overrightarrow{y}).P & := & x(w)w(y_1).w(y_2)\ldots.w(y_n).P_0 \\ \bar{x}\langle\overrightarrow{z}\rangle.\mathrm{Q} & := & (w)(\bar{x}\langle w\rangle\bar{w}\langle z_1\rangle.\bar{w}\langle z_2\rangle\ldots.\bar{w}\langle z_n\rangle.Q_0) \end{array}$$

Donde P_0 y Q_0 son las respectivas traducciones de P y Q

Que sucede con nuestro ejemplo...

Replicación

Este operador es comunmente introducido en vez de definiciones resursivas,

Replicación

Este operador es comunmente introducido en vez de definiciones resursivas, intuitivamente podemos definirlo como una composición arbitraria de un proceso P, que denotamos por $P!$ con la condición estructural:

$$P! \equiv P|P!$$

Replicación

Este operador es comunmente introducido en vez de definiciones resursivas, intuitivamente podemos definirlo como una composición arbitraria de un proceso P, que denotamos por $P!$ con la condición estructural:

$$P! \equiv P|P!$$

Ejemplo:

$$!x(z).\bar{y}\langle z\rangle.\mathbf{0}$$

...

$$!x(z).!\bar{y}\langle z\rangle.\mathbf{0}$$

....

Definiciones Recursivas?

Considerando replicaciones, podremos simular nuestra definiciones recursivas anteriormente introducidas

$$A(\vec{y}) := Q_A$$

Donde Q_A depende de A y cuyo alcance define un procedimiento P, entonces

Definiciones Recursivas?

Considerando replicaciones, podremos simular nuestra definiciones recursivas anteriormente introducidas

$$A(\vec{y}) := Q_A$$

Donde Q_A depende de A y cuyo alcance define un procedimiento P, entonces

Traducción

- Crear un nombre nuevo, a que represente A

Definiciones Recursivas?

Considerando replicaciones, podremos simular nuestra definiciones recursivas anteriormente introducidas

$$A(\vec{y}) := Q_A$$

Donde Q_A depende de A y cuyo alcance define un procedimiento P, entonces

Traducción

- Crear un nombre nuevo, a que represente A
- Por cada proceso R, se denotara $\widehat{R}$ al resultado de reemplazar cada llamada $A\langle\vec{y}\rangle$ en R por $\bar{a}\langle y\rangle$
- Finalmente reemplace P, y todas las definiciones asociadas de A, por

$$\widehat{\widehat{P}} := (a)(\widehat{P}|!ax.\widehat{Q}_A).$$

Preliminares

Por comodidad adoptemos el siguiente fragmento de nuestro cálculo:

$$\begin{array}{lcl} P & := & \mathbf{0} \\ & | & \bar{y}\langle x\rangle.P \\ & | & y(x).P \\ & | & P_1|P_2 \\ & | & (x)P \end{array}$$

The Chemical Abstract Machine: (Metáfora)

Fue inicialmente presentada por Berry y Boudal como una presentación de una semántica operacional de *CCS*. Esta consiste de dos partes, relaciones sobre procesos:

- *Congruencia Estructural:* Que es la congruencia más pequeña que satisface las reglas I

The Chemical Abstract Machine: (Metáfora)

Fue inicialmente presentada por Berry y Boudal como una presentación de una semántica operacional de *CCS*. Esta consiste de dos partes, relaciones sobre procesos:

- *Congruencia Estructural:* Que es la congruencia más pequeña que satisface las reglas I
- *Reducciones:* Que la congruencia más pequeña que satisface las reglas de redución II.

Regla I

$$P|\mathbf{0} \equiv P \quad P|Q \equiv Q|P \quad P|(Q|R) \equiv (P|Q)|R$$

$$(x).\mathbf{0} \equiv \mathbf{0} \quad (x).(y).P \equiv (y).(x).P$$

$$(x).(P|Q) \equiv P|(x).Q \text{ Si } x \notin fn(P)$$

$$!P \equiv P|!P \quad !\mathbf{0} \equiv \mathbf{0} \quad !!P \equiv !P \quad !(P|Q) \equiv !P|!Q$$

Regla II

$$\bar{y}\langle b\rangle.P|y(a).Q \rightarrow P|Q\{b := a\}$$

$$\frac{P \rightarrow P'}{(x).P \rightarrow (x).P'} \qquad \frac{P \rightarrow P'}{P|Q \rightarrow P'|Q}$$

$$\frac{P \rightarrow P' \quad P \equiv Q \quad P' \equiv Q'}{Q \rightarrow Q'}$$

Sistemas de Transición Etiquetados

Semántica Temprana: Diremos que un *sistema de transición etiquetada* consiste de un conjunto de procesos $\mathcal{P}$, un conjunto de acciones $\mathcal{A}$, y una relación de transición $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{P}$. Donde las etiquetas de la evolución serán vistas como el resulta de una interacciones en un proceso, o bien mediante alguna computación interna.

$$\textbf{Acciones} := x(y)|\bar{x}\langle y\rangle|\bar{x}(y)_\nu|\tau.$$

Respectivamente: input, output libre, output acotado y computación interna.

Regla III

$$y(b).P \rightarrow_{y(a)} P\{b := a\}$$

$$\frac{\ldots}{\ldots}$$

Definiciones

Una relación binaria $\mathcal{S}$ es una *simulación*, si $P\mathcal{S}Q$ entonces:

Si $P \rightarrow^{\alpha} P'$, entonces para algún $Q', Q \rightarrow^{\alpha} Q'$ y $P'\mathcal{S}Q'$

Diremos que una relación binaria $\mathcal{S}$ es una *bisimulación* si y sólo ella y su inversa son simulaciones.

Definiciones

Una relación binaria $\mathcal{S}$ es una *simulación*, si $P\mathcal{S}Q$ entonces:

Si $P \rightarrow^{\alpha} P'$, entonces para algún $Q', Q \rightarrow^{\alpha} Q'$ y $P'\mathcal{S}Q'$

Diremos que una relación binaria $\mathcal{S}$ es una *bisimulación* si y sólo ella y su inversa son simulaciones.

$\equiv$ es la más grande de las bisimulaciones.

Definiciones

Una relación binaria $\mathcal{S}$ es una *simulación*, si $P\mathcal{S}Q$ entonces:

Si $P \rightarrow^{\alpha} P'$, entonces para algún $Q', Q \rightarrow^{\alpha} Q'$ y $P'\mathcal{S}Q'$

Diremos que una relación binaria $\mathcal{S}$ es una *bisimulación* si y sólo ella y su inversa son simulaciones.

$\equiv$ es la más grande de las bisimulaciones.

Se define la relación $\sim^{\cdot}$ *bisimilaridad*, si $P \sim^{\cdot} Q$ si y solo si existe una bisimulación $\mathcal{S}$ tal que $P\mathcal{S}Q$

Temprana (Early)

Diremos que una relación binaria $\mathcal{S}$ es una simulación *Temprana*, si satisface los requerimientos siguientes adicionales:

- Si $P \rightarrow^{\alpha} P'$, y α es una acción libre, entonces para algún Q', $Q \rightarrow^{\alpha} Q'$ y $P' \mathcal{S} Q'$

Temprana (Early)

Diremos que una relación binaria $\mathcal{S}$ es una simulación *Temprana*, si satisface los requerimientos siguientes adicionales:

- Si $P \rightarrow^{\alpha} P'$, y α es una acción libre, entonces para algún Q', $Q \rightarrow^{\alpha} Q'$ y $P' \mathcal{S} Q'$
- **Si** $P \rightarrow^{x(y)} P'$**, donde** $y \notin n(P, Q)$**, entonces para todo** w**, existe** Q'**, tal que** $Q \rightarrow^{x(y)} Q'$ **y** $P'\{w/y\} \mathcal{S} Q'\{w/y\}$

Temprana (Early)

Diremos que una relación binaria $\mathcal{S}$ es una simulación *Temprana*, si satisface los requerimientos siguientes adicionales:

- Si $P \rightarrow^{\alpha} P'$, y α es una acción libre, entonces para algún Q', $Q \rightarrow^{\alpha} Q'$ y $P' \mathcal{S} Q'$
- **Si** $P \rightarrow^{x(y)} P'$**, donde** $y \notin n(P, Q)$**, entonces para todo** w**, existe** Q'**, tal que** $Q \rightarrow^{x(y)} Q'$ **y** $P'\{w/y\} \mathcal{S} Q'\{w/y\}$
- Si $P \rightarrow^{\bar{x}\langle y \rangle} P'$, y $y \notin n(P, Q)$, entonces para algún Q', $Q \rightarrow^{\bar{x}\langle y \rangle} Q'$ y $P' \mathcal{S} Q'$

Temprana (Early)

Diremos que una relación binaria $\mathcal{S}$ es una simulación *Temprana*, si satisface los requerimientos siguientes adicionales:

- Si $P \rightarrow^{\alpha} P'$, y α es una acción libre, entonces para algún Q', $Q \rightarrow^{\alpha} Q'$ y $P' \mathcal{S} Q'$
- **Si** $P \rightarrow^{x(y)} P'$**, donde** $y \notin n(P, Q)$**, entonces para todo** w**, existe** Q'**, tal que** $Q \rightarrow^{x(y)} Q'$ **y** $P'\{w/y\} \mathcal{S} Q'\{w/y\}$
- Si $P \rightarrow^{\bar{x}\langle y \rangle} P'$, y $y \notin n(P, Q)$, entonces para algún Q', $Q \rightarrow^{\bar{x}\langle y \rangle} Q'$ y $P' \mathcal{S} Q'$

Tarde, (Late)

Obtendremos una nueva versión de bisimilaridad, Tardia, al considerar la siguiente segunda clausula alternativa:

Tarde, (Late)

Obtendremos una nueva versión de bisimilaridad, Tardia, al considerar la siguiente segunda clausula alternativa:

- **Si** $P \rightarrow^{x(y)} P'$**, donde** $y \notin n(P, Q)$**, entonces para algún** Q'**,** $Q \rightarrow^{x(y)} Q'$ **y para todo** w**,** $P'\{w/y\} \mathcal{S} Q'\{w/y\}$