Verificación Formal de Protocolos de Comunicación con Álgebra de Procesos

Miguel Valero
Centrum voor Wiskunde en Informatica
SEN 2 - Amsterdam - The Netherlands

Verificación Formal:

- Probar formalmente que un *programa* es correcto.

Verificación Formal:

- Probar formalmente que un *programa* es correcto.
- Criterios de Corrección:
  - Actúa conforme a una especificación.
  - Satisface una serie de requisitos o propiedades.

Verificación Formal:

- Probar formalmente que un *programa* es correcto.
- Criterios de Corrección:
  - Actúa conforme a una especificación.
  - Satisface una serie de requisitos o propiedades.
- Especificación formal:
  - Descripción de un sistema cuya interpretación es única (no contiene ambigüedades).
  - Lenguajes formales: Lógica, Autómata de Estados, Álgebra de procesos, Redes de Petri, ...

Verificación Formal:

- Probar formalmente que un *programa* es correcto.
- Criterios de Corrección:
  - Actúa conforme a una especificación.
  - Satisface una serie de requisitos o propiedades.
- Especificación formal:
  - Descripción de un sistema cuya interpretación es única (no contiene ambigüedades).
  - Lenguajes formales: Lógica, Autómata de Estados, Álgebra de procesos, Redes de Petri, ...
- Requisitos:
  - Descritos por formulas en alguna lógica (proposicional, temporal, modal, ...)
  - Descritos por otra especificación abstracta.

Protocolos de Comunicación

Protocolos de Comunicación

- Diferentes agentes (servidores, clientes, ...) intercambian información a través de un determinado medio.

Protocolos de Comunicación

- Diferentes agentes (servidores, clientes, ...) intercambian información a través de un determinado medio.
- Los componentes del sistema se ejecutan en paralelo, en un entorno distribuido.

Protocolos de Comunicación

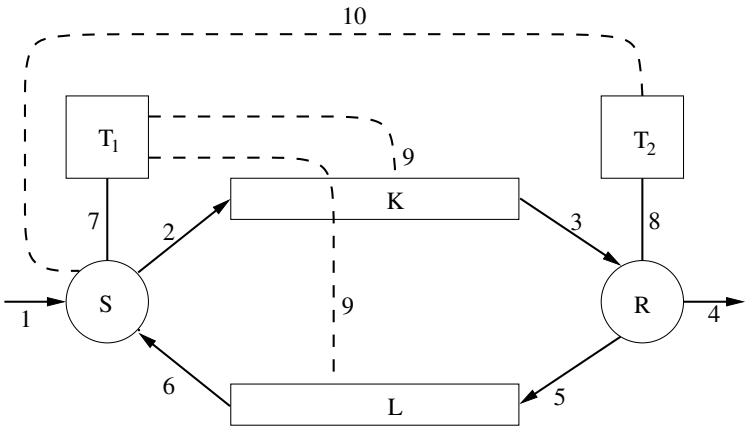
- Diferentes agentes (servidores, clientes, ...) intercambian información a través de un determinado medio.
- Los componentes del sistema se ejecutan en paralelo, en un entorno distribuido.
- El comportamiento global del sistema depende de los procesos y de los datos.

Protocolos de Comunicación

- Diferentes agentes (servidores, clientes, ...) intercambian información a través de un determinado medio.
- Los componentes del sistema se ejecutan en paralelo, en un entorno distribuido.
- El comportamiento global del sistema depende de los procesos y de los datos.
- El sistema es no-determinista.

Protocolos de Comunicación

- Diferentes agentes (servidores, clientes, ...) intercambian información a través de un determinado medio.
- Los componentes del sistema se ejecutan en paralelo, en un entorno distribuido.
- El comportamiento global del sistema depende de los procesos y de los datos.
- El sistema es no-determinista.
- Ejemplos: Alternating bit protocol, Bounded Retransmission Protocol, Sliding Window Protocol...



Técnicas de Automáticas de Verificación

Testing:

Model Checking:

Técnicas de Automáticas de Verificación

Testing:

- Simulación de algunos escenarios.

Model Checking:

Técnicas de Automáticas de Verificación

Testing:

- Simulación de algunos escenarios.
- En general no es completo → no detecta todos los errores.

Model Checking:

Técnicas de Automáticas de Verificación

Testing:

- Simulación de algunos escenarios.
- En general no es completo → no detecta todos los errores.
- Es difícil testear sistemas distribuidos por que contienen no-determinismo.

Model Checking:

Técnicas de Automáticas de Verificación

Testing:

- Simulación de algunos escenarios.
- En general no es completo → no detecta todos los errores.
- Es difícil testear sistemas distribuidos por que contienen no-determinismo.

Model Checking:

- Comprobación de que todos los escenarios son correctos.

Técnicas de Automáticas de Verificación

Testing:

- Simulación de algunos escenarios.
- En general no es completo → no detecta todos los errores.
- Es difícil testear sistemas distribuidos por que contienen no-determinismo.

Model Checking:

- Comprobación de que todos los escenarios son correctos.
- Completo y Automático.

Técnicas de Automáticas de Verificación

Testing:

- Simulación de algunos escenarios.
- En general no es completo → no detecta todos los errores.
- Es difícil testear sistemas distribuidos por que contienen no-determinismo.

Model Checking:

- Comprobación de que todos los escenarios son correctos.
- Completo y Automático.
- Necesita la representación explícita de todos los comportamientos del sistema

Técnicas de Automáticas de Verificación

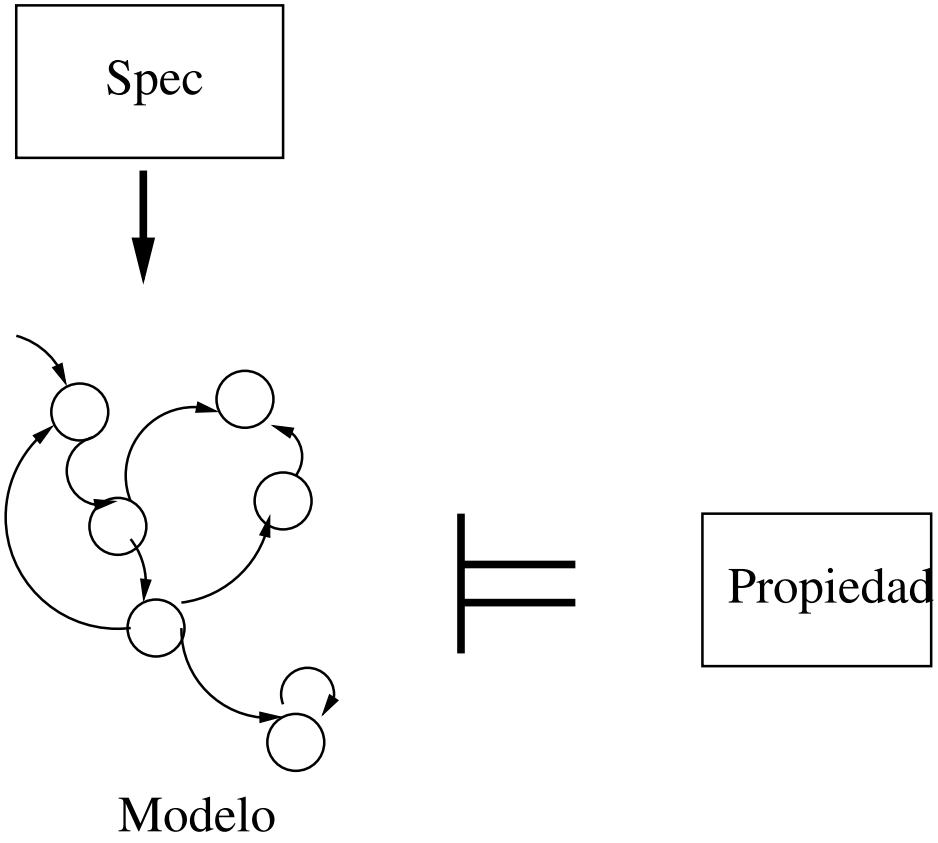
Testing:

- Simulación de algunos escenarios.
- En general no es completo → no detecta todos los errores.
- Es difícil testear sistemas distribuidos por que contienen no-determinismo.

Model Checking:

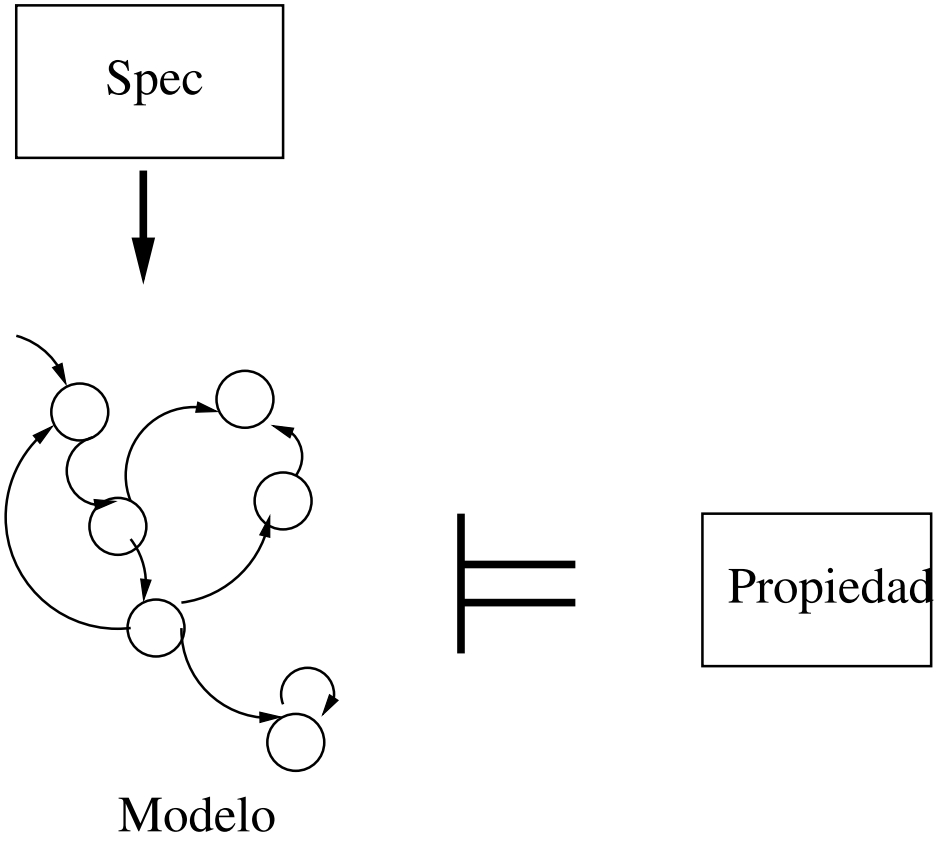
- Comprobación de que todos los escenarios son correctos.
- Completo y Automático.
- Necesita la representación explícita de todos los comportamientos del sistema
- Problema: crecimiento exponencial de estados.→ Sólo útil para sistemas finitos (y pequeños).

Model Checking:



Propiedades:

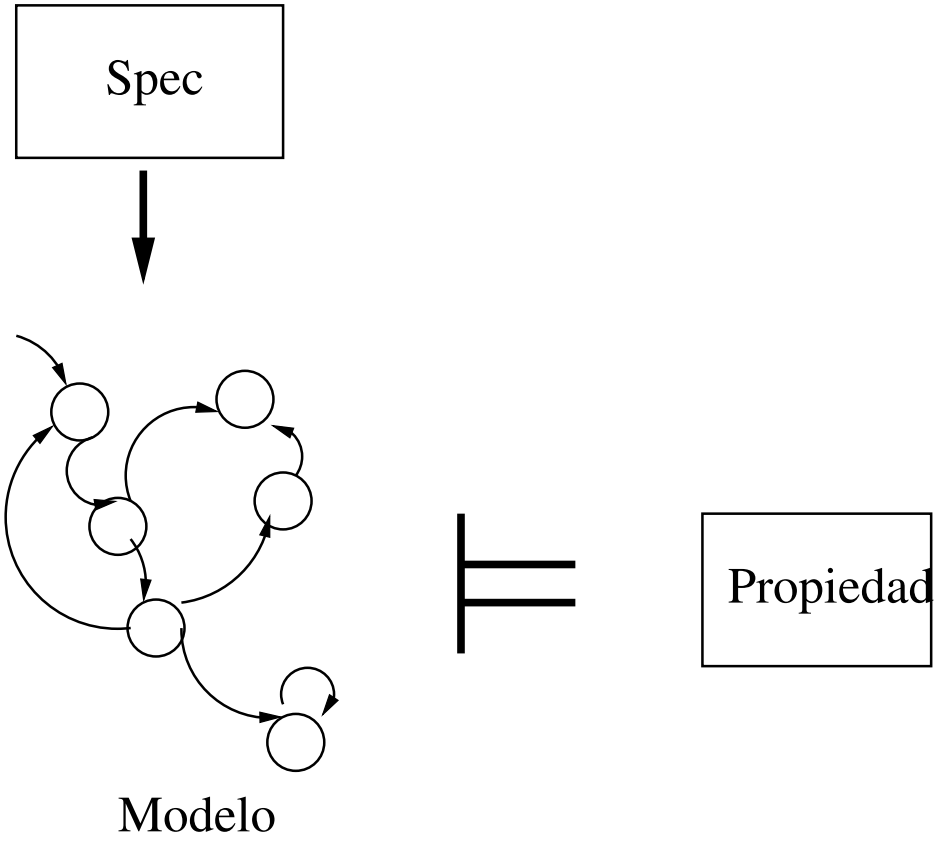
Model Checking:



Propiedades:

- "Si S envía un dato d entonces R recibirá eventualmente d"

Model Checking:



Propiedades:

- "Si S envía un dato d entonces R recibirá eventualmente d"
- Ausencia de bloqueo (Deadlock).

Comprobación de Equivalencia:

Comprobación de Equivalencia:

- Probar que un sistema $\mathcal{I}$ es "equivalente" a otro $\mathcal{E}$ ($\mathcal{I} \simeq \mathcal{E}$)

Comprobación de Equivalencia:

- Probar que un sistema $\mathcal{I}$ es "equivalente" a otro $\mathcal{E}$ ($\mathcal{I} \simeq \mathcal{E}$)
- $\mathcal{E}$(specificación). Es una descripción abstracta, representa lo que queremos que el sistema haga.

Comprobación de Equivalencia:

- Probar que un sistema $\mathcal{I}$ es "equivalente" a otro $\mathcal{E}$ ($\mathcal{I} \simeq \mathcal{E}$)
- $\mathcal{E}$(specificación). Es una descripción abstracta, representa lo que queremos que el sistema haga.
- $\mathcal{I}$(mplementación). Es una descripción detallada, representa lo que el sistema hace.

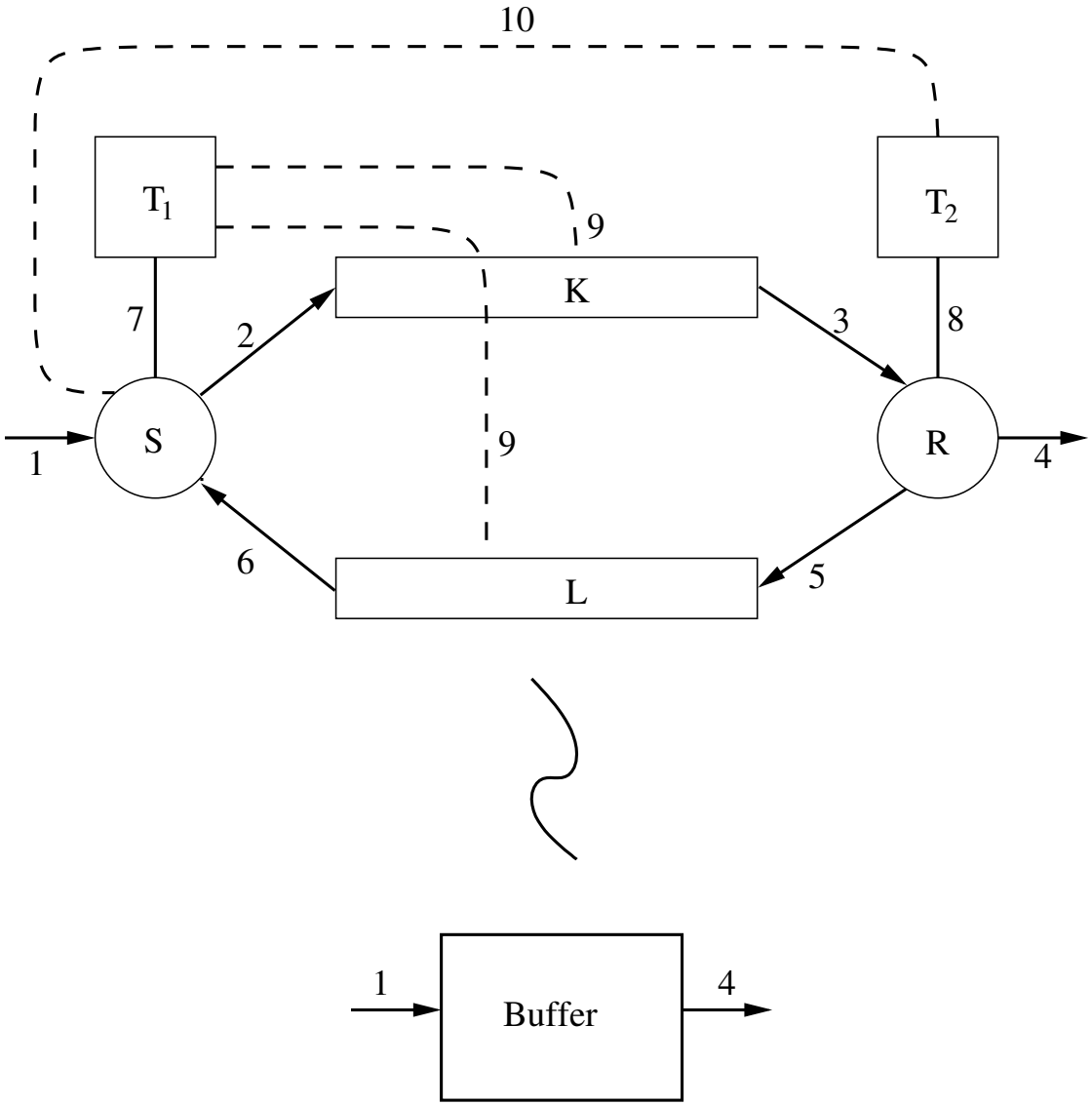
Comprobación de Equivalencia:

- Probar que un sistema $\mathcal{I}$ es "equivalente" a otro $\mathcal{E}$ ($\mathcal{I} \simeq \mathcal{E}$)
- $\mathcal{E}$(specificación). Es una descripción abstracta, representa lo que queremos que el sistema haga.
- $\mathcal{I}$(mplementación). Es una descripción detallada, representa lo que el sistema hace.
- Hay que definir una noción de equivalencia satisfactoria. ($\simeq$)

Comprobación de Equivalencia:

- Probar que un sistema $\mathcal{I}$ es "equivalente" a otro $\mathcal{E}$ ($\mathcal{I} \simeq \mathcal{E}$)
- $\mathcal{E}$(specificación). Es una descripción abstracta, representa lo que queremos que el sistema haga.
- $\mathcal{I}$(mplementación). Es una descripción detallada, representa lo que el sistema hace.
- Hay que definir una noción de equivalencia satisfactoria. ($\simeq$)
- Si ambos sistemas son finitos (y pequeños) se puede probar automáticamente.
- De lo contrario, hay que hacerlo manualmente (asistidos por un Theorem Prover).

Ejemplo:



Resumen:

Hasta ahora:

- Introducción a la Verificación Formal.
- Técnicas de verificación.
- Características de los protocolos de comunicación.

Resto de charla:

- Especificación Algebraica.
- Álgebra de Procesos.
- Nociones de Equivalencia de Procesos.
- Nociones de Model Checking.

Especificación Algebraica:

Especificación Algebraica:

- Especificación de los naturales.
  - 0 es un natural
  - El sucesor S de un natural es un natural

Especificación Algebraica:

- Especificación de los naturales.
  - 0 es un natural
  - El sucesor S de un natural es un natural

- Axiomas de la suma y la multiplicación:

$$
\begin{array}{lrcl}
A1. & plus(x, 0) & = & x \\
A2. & plus(x, S(y)) & = & S(plus(x, y)) \\
A3. & mul(x, 0) & = & 0 \\
A4. & mul(x, S(y)) & = & plus(mul(x, y), x)
\end{array}
$$

Especificación Algebraica:

Especificación Algebraica:

- Teorema: ($3 + 1 = 2 \times 2$)

$$plus(S(S(S(0))), S(0)) = mul(S(S(0)), S(S(0))$$

Especificación Algebraica:

- Teorema: ($3 + 1 = 2 \times 2$)

$$plus(S(S(S(0))), S(0)) = mul(S(S(0)), S(S(0))$$

- Prueba:

$$\begin{aligned} plus(S(S(S(0))), S(0)) &= (A2) \quad S(plus(S(S(S(0))), 0) \\ &= (A1) \quad S(S(S(S(0)))) \end{aligned}$$

$$\begin{aligned} mul(S(S(0)), S(S(0)) &= (A4) \quad plus(mul(S(S(0)), S(0)), S(S(0))) \\ &= (A4) \quad plus(plus(mul(S(S(0)), 0), S(S(0))), S(S(0))) \\ &= (A3) \quad plus(plus(0, S(S(0))), S(S(0))) \\ &= \quad \ldots \\ &= \quad S(S(S(S(0)))) \end{aligned}$$

Álgebra de Procesos

Álgebra de Procesos

- Hay muchas álgebras de procesos: CCS, CSP, ACP, calculo-π, ...
- En BPA (Basic Process Algebra), es un subconjunto de ACP, en que los procesos básicos están compuestos:

Álgebra de Procesos

- Hay muchas álgebras de procesos: CCS, CSP, ACP, calculo-π, ...
- En BPA (Basic Process Algebra), es un subconjunto de ACP, en que los procesos básicos están compuestos:

- Acciones atómicas. Una acción a representa un acto indivisible.
- Composición alternativa $+$: El proceso $t_1 + t_2$ ejecuta t_1 o t_2.
- Composición secuencial $\cdot$: El proceso $t_1 \cdot t_2$ primero ejecuta t_1, y después t_2.

Álgebra de Procesos

- Hay muchas álgebras de procesos: CCS, CSP, ACP, calculo-π, ...
- En BPA (Basic Process Algebra), es un subconjunto de ACP, en que los procesos básicos están compuestos:

- Acciones atómicas. Una acción a representa un acto indivisible.
- Composición alternativa $+$: El proceso $t_1 + t_2$ ejecuta t_1 o t_2.
- Composición secuencial $\cdot$: El proceso $t_1 \cdot t_2$ primero ejecuta t_1, y después t_2.

- Ejemplo: $a \cdot (b + a) \cdot (a + a \cdot b)$

Semántica operacional

- A cada proceso de BPA le corresponde un grafo etiquetado (Labelled Transition System o LTS).
- El grafo representa el comportamiento completo (la semántica) del sistema.

Semántica operacional

- A cada proceso de BPA le corresponde un grafo etiquetado (Labelled Transition System o LTS).
- El grafo representa el comportamiento completo (la semántica) del sistema.
- Reglas de Transición de BPA (SOS):

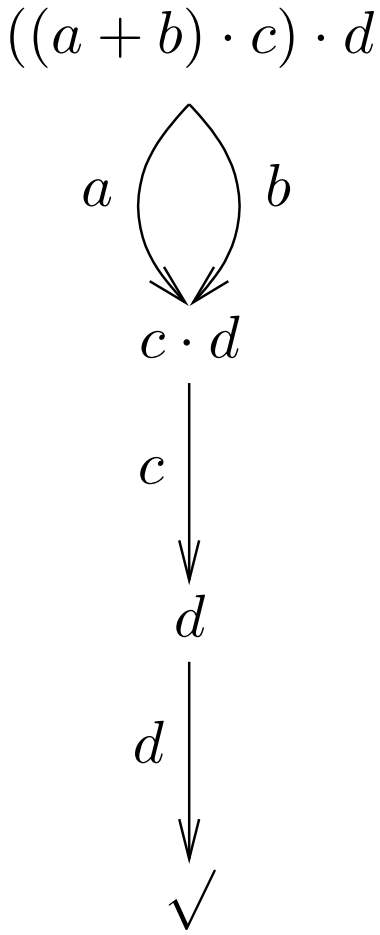
$$\frac{}{a \xrightarrow{a} \surd}$$

$$\frac{x \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd} \qquad \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} \surd}{x \cdot y \xrightarrow{a} y}$$

$$\frac{y \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd} \qquad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'} \qquad \frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y}$$

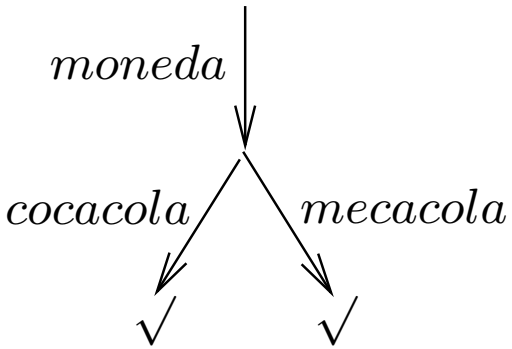
Ejemplo:

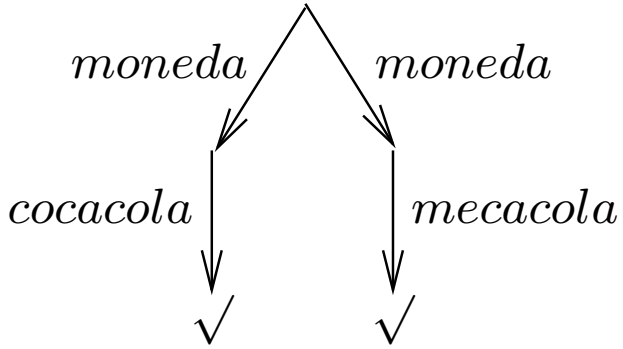
El siguiente término $((a + b) \cdot c) \cdot d$ representa el siguiente grafo:



Equivalencia entre procesos:

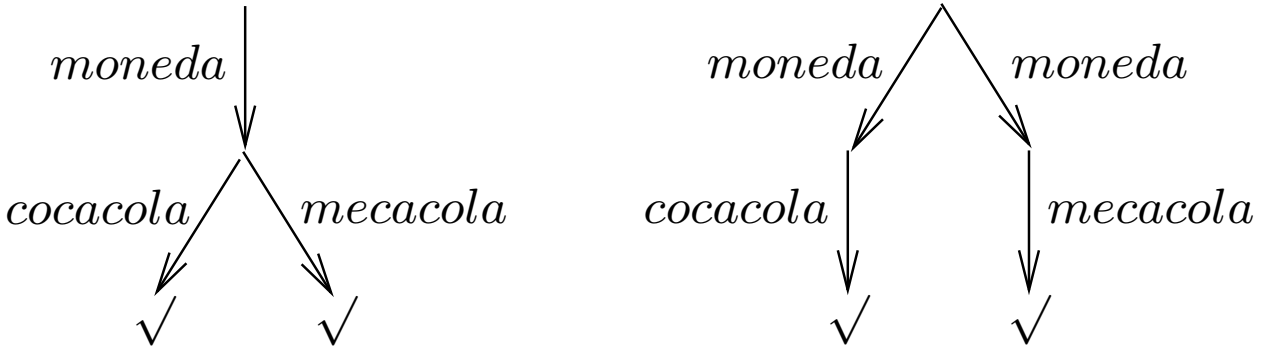
- Especificación de una máquina de bebidas:





Equivalencia entre procesos:

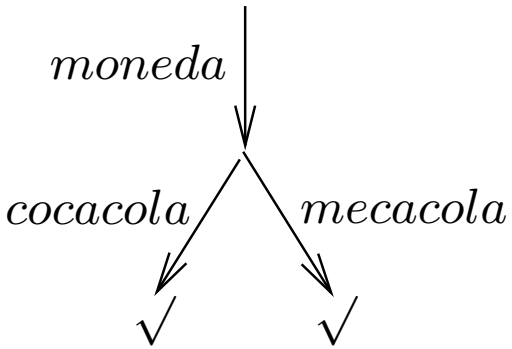
- Especificación de una máquina de bebidas:

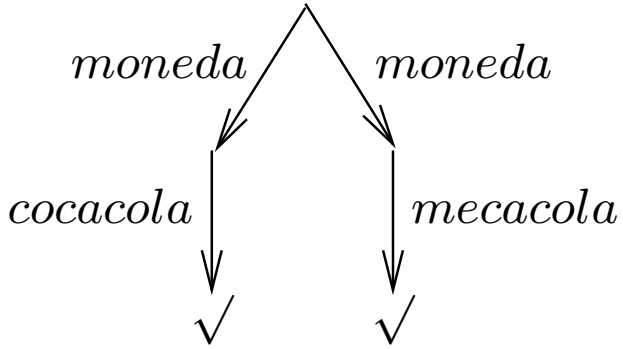


- Los dos procesos tienen las mismas trazas (aunque no tienen el mismo comportamiento).

Equivalencia entre procesos:

- Especificación de una máquina de bebidas:





- Los dos procesos tienen las mismas trazas (aunque no tienen el mismo comportamiento).
- Queremos una noción de equivalencia que equipare los procesos que puedan evolucionar de la misma manera desde cualquier estado.

Bisimulación:

- Hay muchas nociones de equivalencia: simulation, bisimulation, trace equivalence, ready simulation, failure traces...
- La bisimulación es de las que más discrimina.

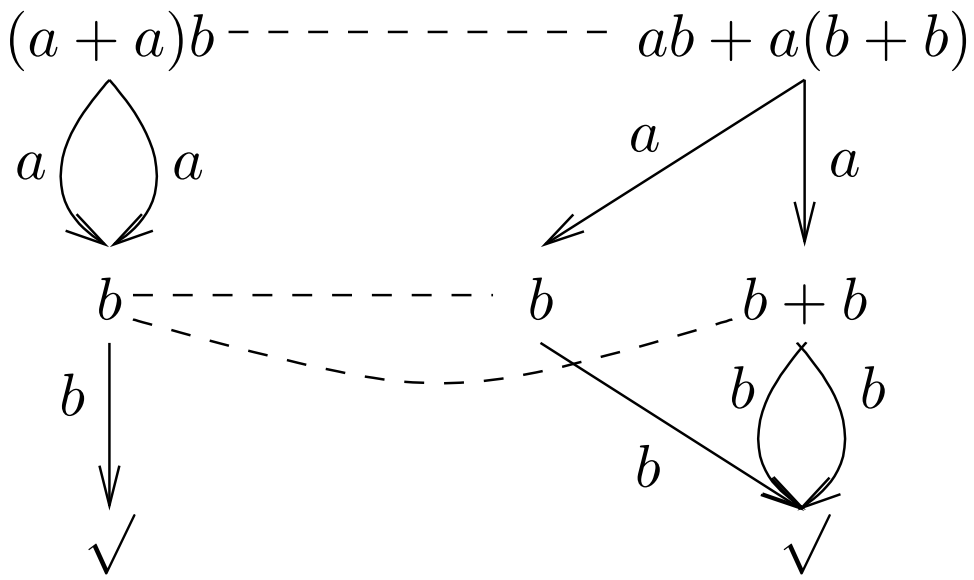
Bisimulación:

- Hay muchas nociones de equivalencia: simulation, bisimulation, trace equivalence, ready simulation, failure traces...
- La bisimulación es de las que más discrimina.
- Una bisimulación es una relación $\mathcal{B}$ de equivalencia entre procesos, tal que:
  1. Si $p \,\mathcal{B}\, q$ y $p \xrightarrow{a} p'$, entonces $q \xrightarrow{a} q'$ con $p' \,\mathcal{B}\, q'$
  2. Si $p \,\mathcal{B}\, q$ y $q \xrightarrow{a} q'$, entonces $p \xrightarrow{a} p'$ con $p' \,\mathcal{B}\, q'$
  3. Si $p \,\mathcal{B}\, q$ y $p \xrightarrow{a} \surd$, entonces $q \xrightarrow{a} \surd$
  4. Si $p \,\mathcal{B}\, q$ y $q \xrightarrow{a} \surd$, entonces $p \xrightarrow{a} \surd$

  Dos procesos p y q son bisimilares ($p \underline{\leftrightarrow} q$), si existe una relación de bisimilaridad $\mathcal{B}$, tal que $p \,\mathcal{B}\, q$.

Ejemplo:

$(a+a)b \underline{\leftrightarrow} ab + a(b+b)$



La relación de bisimulación $\mathcal{B}$ es:

$(a+a)b \;\mathcal{B}\; ab + a(b+b),\;\; b \;\mathcal{B}\; b$ y $\;\; b \;\mathcal{B}\; b + b.$

Axiomas de BPA (modulo bisimulación):

$$
\begin{array}{lrcl}
\text{A1} & x + y & = & y + x \\
\text{A2} & (x + y) + z & = & x + (y + z) \\
\text{A3} & x + x & = & x \\
\text{A4} & (x + y) \cdot z & = & x \cdot z + y \cdot z \\
\text{A5} & (x \cdot y) \cdot z & = & x \cdot (y \cdot z)
\end{array}
$$

Axiomatización:

- Teorema: $(a + a)b = ab + a(b + b)$
- Prueba:

$$\begin{aligned} (a+a)b &= (A3) \quad ab \\ \\ ab + a(b+b) &= (A3) \quad ab + ab \\ &= (A3) \quad ab \end{aligned}$$

Axiomatización:

- Teorema: $(a + a)b = ab + a(b + b)$
- Prueba:

$$\begin{aligned} (a + a)b & = (A3) \quad ab \\ \\ ab + a(b + b) & = (A3) \quad ab + ab \\ & = (A3) \quad ab \end{aligned}$$

- IMPORTANTE: La axiomatización de BPA, es consistente y completa módulo bisimulación (sound and complete):
  - (Consistente) $p = q$ implica $p \underline{\leftrightarrow} q$
  - (Completa) $p \underline{\leftrightarrow} q$ implica $p = q$

Ejemplo:

$(a+a)(cd)+(bc)(d+d) = ((b+a)(c+c))d$

$$
\begin{aligned}
& (\underline{a+a})(cd)+(bc)(d+d) \\
\overset{\text{A3}}{\rightarrow} \quad & a(cd)+(bc)(\underline{d+d}) \\
\overset{\text{A3}}{\rightarrow} \quad & a(cd)+\underline{(bc)d} \\
\overset{\text{A5}}{\rightarrow} \quad & a(cd)+b(cd)
\end{aligned}
$$

$$
\begin{aligned}
& ((b+a)(\underline{c+c}))d \\
\overset{\text{A3}}{\rightarrow} \quad & \underline{((b+a)c)d} \\
\overset{\text{A5}}{\rightarrow} \quad & \underline{(b+a)(cd)} \\
\overset{\text{A4}}{\rightarrow} \quad & b(cd)+a(cd)
\end{aligned}
$$

Axiomatización Unsound:

Podemos Probar cosas que no son ciertas.

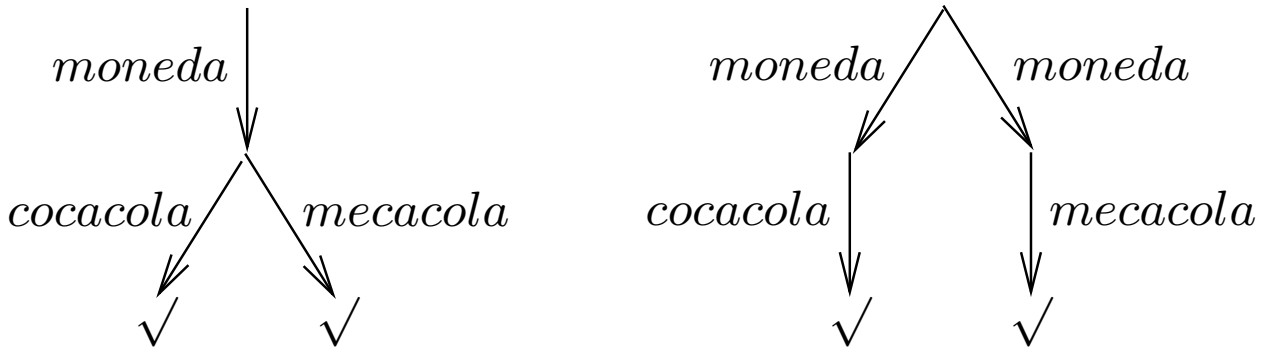
$$
\begin{array}{lrcl}
\text{A1} & x + y & = & y + x \\
\text{A2} & (x + y) + z & = & x + (y + z) \\
\text{A3} & x + x & = & x \\
\text{A4} & (x + y) \cdot z & = & x \cdot z + y \cdot z \\
\text{A5} & (x \cdot y) \cdot z & = & x \cdot (y \cdot z) \\
\text{A6} & z \cdot (y + z) & = & x \cdot y + x \cdot z
\end{array}
$$

Axiomatización Unsound:

Podemos Probar cosas que no son ciertas.

$$
\begin{array}{lrcl}
\text{A1} & x + y & = & y + x \\
\text{A2} & (x + y) + z & = & x + (y + z) \\
\text{A3} & x + x & = & x \\
\text{A4} & (x + y) \cdot z & = & x \cdot z + y \cdot z \\
\text{A5} & (x \cdot y) \cdot z & = & x \cdot (y \cdot z) \\
\text{A6} & z \cdot (y + z) & = & x \cdot y + x \cdot z
\end{array}
$$

- Serían equivalentes según los axiomas (aunque no son bisimilares)



Axiomatización Uncomplete:

No Podemos Probar cosas que son ciertas.

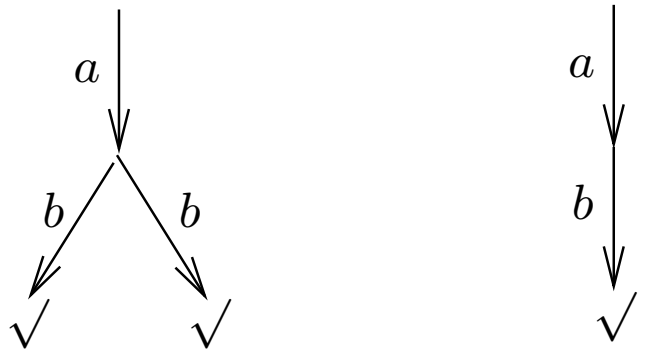
$$
\begin{array}{lrcl}
\text{A1} & x + y & = & y + x \\
\text{A2} & (x + y) + z & = & x + (y + z) \\
\text{A4} & (x + y) \cdot z & = & x \cdot z + y \cdot z \\
\text{A5} & (x \cdot y) \cdot z & = & x \cdot (y \cdot z)
\end{array}
$$

Axiomatización Uncomplete:

No Podemos Probar cosas que son ciertas.

$$
\begin{array}{lrcl}
\text{A1} & x + y & = & y + x \\
\text{A2} & (x + y) + z & = & x + (y + z) \\
\text{A4} & (x + y) \cdot z & = & x \cdot z + y \cdot z \\
\text{A5} & (x \cdot y) \cdot z & = & x \cdot (y \cdot z)
\end{array}
$$

- Son bisimilares pero no podemos probarlo con los axiomas



Nuevos Operadores:

ACP (Algebra for Communicating Processes).

Nuevos Operadores:

ACP (Algebra for Communicating Processes).

- Comunicación $|$. Permite a dos acciones sincronizar ($a \mid b = c$).

Nuevos Operadores:

ACP (Algebra for Communicating Processes).

- Comunicación $|$. Permite a dos acciones sincronizar ($a \mid b = c$).
- Composición Paralela ($\|$). Permite la ejecución de varios procesos en paralelo.

Nuevos Operadores:

ACP (Algebra for Communicating Processes).

- Comunicación $|$. Permite a dos acciones sincronizar ($a \mid b = c$).
- Composición Paralela ($\|$). Permite la ejecución de varios procesos en paralelo.
- Deadlock (δ).. Representa el bloqueo.

Nuevos Operadores:

ACP (Algebra for Communicating Processes).

- Comunicación $|$. Permite a dos acciones sincronizar ($a \mid b = c$).
- Composición Paralela ($\|$). Permite la ejecución de varios procesos en paralelo.
- Deadlock (δ).. Representa el bloqueo.
- Encapsulación (∂_H). Fuerza a procesos a sincronizar.

Nuevos Operadores:

ACP (Algebra for Communicating Processes).

- Comunicación $|$. Permite a dos acciones sincronizar ($a \mid b = c$).
- Composición Paralela ($\|$). Permite la ejecución de varios procesos en paralelo.
- Deadlock (δ).. Representa el bloqueo.
- Encapsulación (∂_H). Fuerza a procesos a sincronizar.
- Recursión ($X = a.X$). Permite representar procesos con comportamiento infinito.

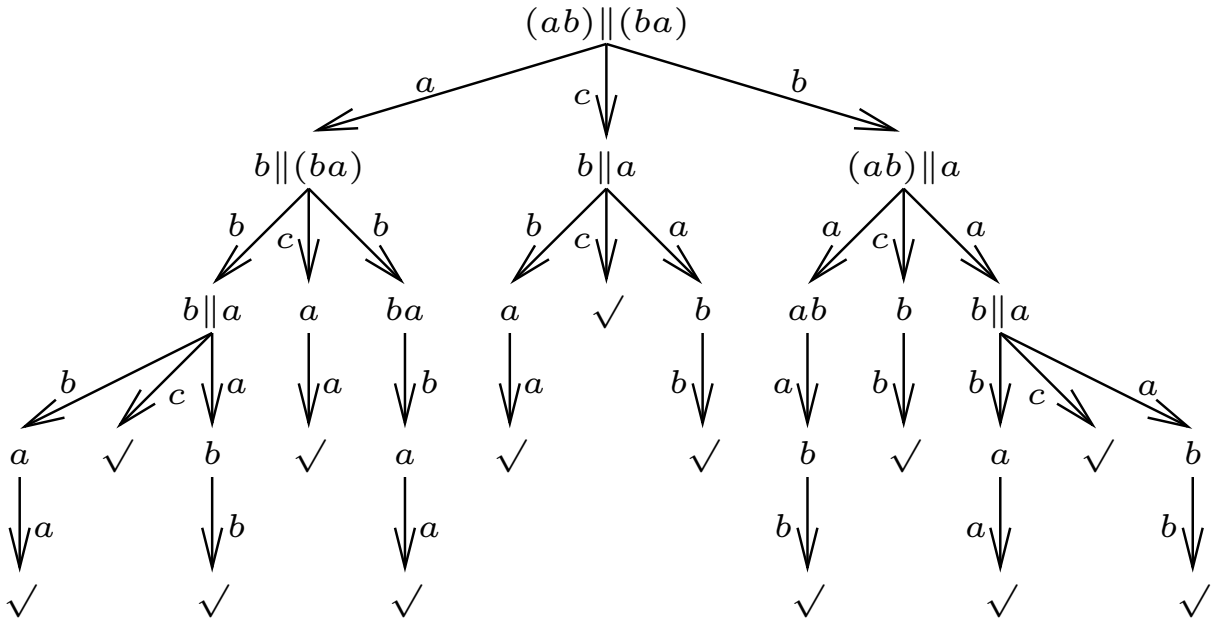
Nuevos Operadores:

ACP (Algebra for Communicating Processes).

- Comunicación $|$. Permite a dos acciones sincronizar ($a \mid b = c$).
- Composición Paralela ($\|$). Permite la ejecución de varios procesos en paralelo.
- Deadlock (δ).. Representa el bloqueo.
- Encapsulación (∂_H). Fuerza a procesos a sincronizar.
- Recursión ($X = a.X$). Permite representar procesos con comportamiento infinito.
- Abstracción (τ_I). Oculta el comportamiento de algunas acciones.

Composición Paralela y Comunicación:

El grafo correspondiente a $(ab)\|(ba)$, asumiendo que a y b pueden comunicar resultando en c ($a \mid b = c$), es:

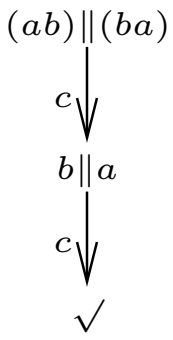


Deadlock y Encapsulación:

- Deadlock (δ) no tiene ningún comportamiento. No hay ninguna transición asociada con él.

Deadlock y Encapsulación:

- Deadlock (δ) no tiene ningún comportamiento. No hay ninguna transición asociada con él.
- ∂ fuerza a los procesos a comunicar.
- El grafo correspondiente a $\partial_{\{a,b\}}(ab)\|(ba)$, es:



Abstracción:

Abstracción:

- La abstracción se utiliza para ocultar partes de la ejecución.

Abstracción:

- La abstracción se utiliza para ocultar partes de la ejecución.
- Una acción oculta se denota con τ.

Abstracción:

- La abstracción se utiliza para ocultar partes de la ejecución.
- Una acción oculta se denota con τ.
- Las acciones ocultas no son observables desde el exterior.

Abstracción:

- La abstracción se utiliza para ocultar partes de la ejecución.
- Una acción oculta se denota con τ.
- Las acciones ocultas no son observables desde el exterior.
- Es útil para abstraer el comportamiento interno del sistema.

Abstracción:

- La abstracción se utiliza para ocultar partes de la ejecución.
- Una acción oculta se denota con τ.
- Las acciones ocultas no son observables desde el exterior.
- Es útil para abstraer el comportamiento interno del sistema.
- Nueva noción de equivalencia que tiene en cuenta la abstracción (*Branching Bisimulation*).

Abstracción:

- La abstracción se utiliza para ocultar partes de la ejecución.
- Una acción oculta se denota con τ.
- Las acciones ocultas no son observables desde el exterior.
- Es útil para abstraer el comportamiento interno del sistema.
- Nueva noción de equivalencia que tiene en cuenta la abstracción (*Branching Bisimulation*).
- Por ejemplo: $a.\tau.b = a.b$

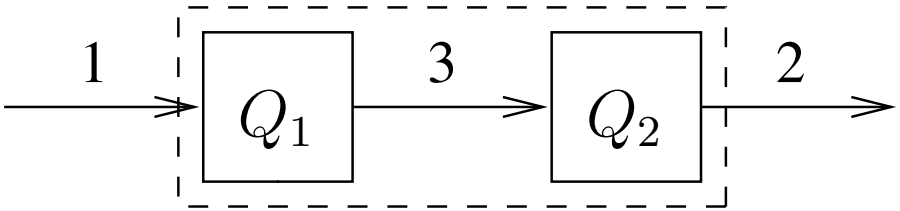
Axiomatización:

Existe un axiomatización sound and complete módulo bisimulación de ACP

Existe un axiomatización sound and complete módulo branching bisimulación de ACP_τ

Ejemplo:

$$
\begin{aligned}
Q_1 &= r_1\, s_3\, Q_1 \\
Q_2 &= r_3\, s_2\, Q_2
\end{aligned}
$$



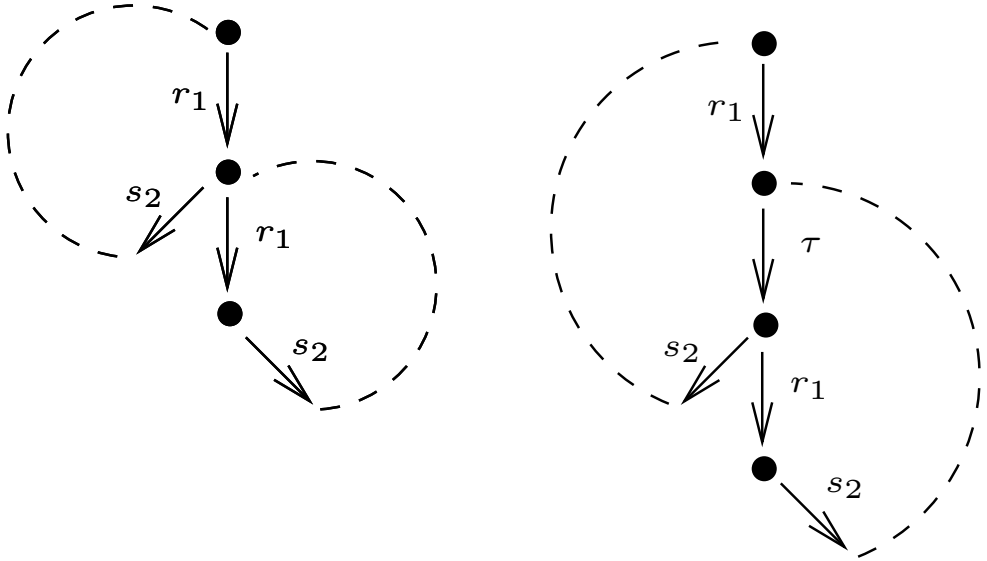
Dos buffers Q_2 y Q_1 de capacidad uno

$$\tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1))$$

son equivalentes a un buffer de capacidad dos:

$$
\begin{aligned}
X &= r_1\, Y \\
Y &= r_1\, Z + s_2\, X \\
Z &= s_2\, Y
\end{aligned}
$$

Ejemplo:

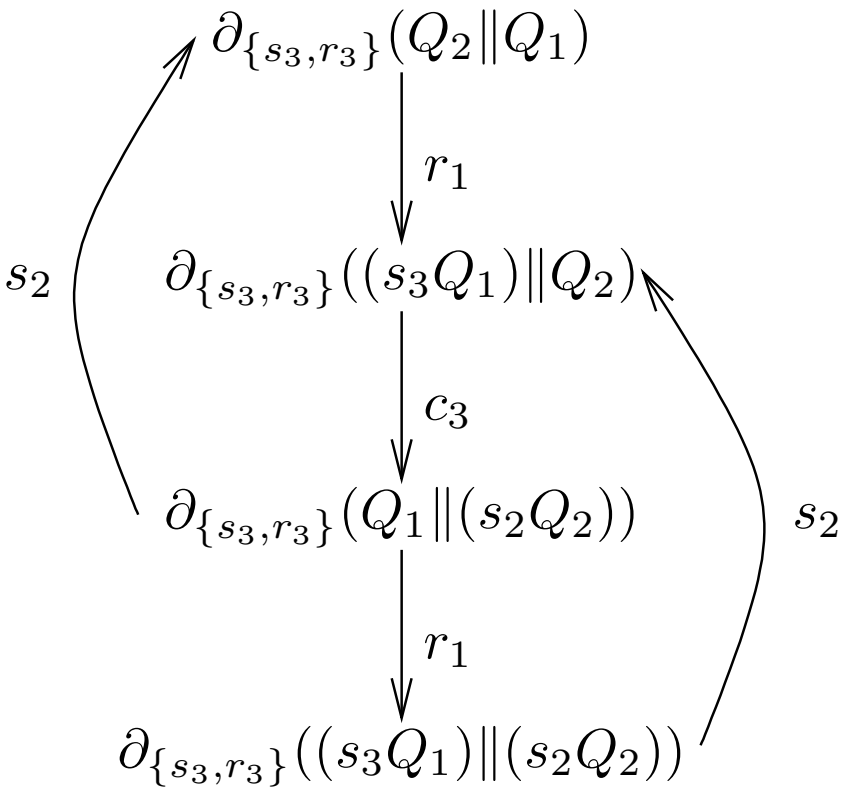


$$
\begin{aligned}
& Q_2 \| Q_1 \\
= \quad & Q_2 \between Q_1 + Q_1 \between Q_2 + Q_2 | Q_1 \\
= \quad & (r_3 s_2 Q_2) \between Q_1 + (r_1 s_3 Q_1) \between Q_2 \\
& + (r_3 s_2 Q_2) | (r_1 s_3 Q_1) \\
= \quad & r_3 \cdot ((s_2 Q_2) \| Q_1) + r_1 \cdot ((s_3 Q_1) \| Q_2) \\
& + \delta \cdot ((s_2 Q_2) \| (s_3 Q_1)) \\
= \quad & r_3 \cdot ((s_2 Q_2) \| Q_1) + r_1 \cdot ((s_3 Q_1) \| Q_2)
\end{aligned}
$$

$$
\begin{aligned}
& \partial_{\{s_3, r_3\}} (Q_2 \| Q_1) \\
= \quad & \partial_{\{s_3, r_3\}} (r_3 \cdot ((s_2 Q_2) \| Q_1) + r_1 \cdot ((s_3 Q_1) \| Q_2)) \\
= \quad & \partial_{\{s_3, r_3\}} (r_3 \cdot ((s_2 Q_2) \| Q_1)) \\
& + \partial_{\{s_3, r_3\}} (r_1 \cdot ((s_3 Q_1) \| Q_2)) \\
= \quad & \partial_{\{s_3, r_3\}} (r_3) \cdot \partial_{\{s_3, r_3\}} ((s_2 Q_2) \| Q_1) \\
& + \partial_{\{s_3, r_3\}} (r_1) \cdot \partial_{\{s_3, r_3\}} ((s_3 Q_1) \| Q_2) \\
= \quad & \delta \cdot \partial_{\{s_3, r_3\}} ((s_2 Q_2) \| Q_1) \\
& + r_1 \cdot \partial_{\{s_3, r_3\}} ((s_3 Q_1) \| Q_2) \\
= \quad & r_1 \cdot \partial_{\{s_3, r_3\}} ((s_3 Q_1) \| Q_2)
\end{aligned}
$$

Podemos derivar:

$$\partial_{\{s_3,r_3\}}((s_3Q_1)\|Q_2) = c_3 \cdot \partial_{\{s_3,r_3\}}(Q_1\|(s_2Q_2))$$
$$\partial_{\{s_3,r_3\}}(Q_1\|(s_2Q_2)) = r_1 \cdot \partial_{\{s_3,r_3\}}((s_3Q_1)\|(s_2Q_2)) + s_2 \cdot \partial_{\{s_3,r_3\}}(Q_2\|Q_1)$$
$$\partial_{\{s_3,r_3\}}((s_3Q_1)\|(s_2Q_2)) = s_2 \cdot \partial_{\{s_3,r_3\}}((s_3Q_1)\|Q_2)$$



$$
\begin{aligned}
& \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1)) \\
= \quad & \tau_{\{c_3\}}(r_1 \cdot \partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2)) \\
= \quad & r_1 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2)) \\
= \quad & r_1 \cdot \tau_{\{c_3\}}(c_3 \cdot \partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \\
= \quad & r_1 \cdot \tau \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \\
= \quad & r_1 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2)))
\end{aligned}
$$

Podemos derivar:

$$
\begin{aligned}
\tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \; = \\
& r_1 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2))) \\
& + s_2 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1)) \\
\tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2))) \; = \\
& s_2 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2)))
\end{aligned}
$$

Por tanto:

$$
\begin{aligned}
X \quad &:= \quad \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1)) \\
Y \quad &:= \quad \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \\
Z \quad &:= \quad \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2)))
\end{aligned}
$$

Álgebra de Procesos con Datos:

Especificación algebraica de los datos + integración

Álgebra de Procesos con Datos:

Especificación algebraica de los datos + integración

- Acciones con Parámetros $n : Nat,\ send(n)$

Álgebra de Procesos con Datos:

Especificación algebraica de los datos + integración

- Acciones con Parámetros $n : Nat,\ send(n)$
- Procesos con Parámetros $P(n : Nat, t : Time, M : multiset[Nat])$

Álgebra de Procesos con Datos:

Especificación algebraica de los datos + integración

- Acciones con Parámetros $n : Nat,\ send(n)$
- Procesos con Parámetros $P(n : Nat, t : Time, M : multiset[Nat])$
- Condicional:

$$\begin{aligned} P(n : Nat) &= n > 0 \rightarrow a.P(n-1) \\ &+ n = 0 \rightarrow \tau.P(N) \end{aligned}$$

Álgebra de Procesos con Datos:

Especificación algebraica de los datos + integración

- Acciones con Parámetros $n : Nat,\ send(n)$
- Procesos con Parámetros $P(n : Nat, t : Time, M : multiset[Nat])$
- Condicional:

$$\begin{aligned} P(n : Nat) &= n > 0 \rightarrow a.P(n-1) \\ &+ n = 0 \rightarrow \tau.P(N) \end{aligned}$$

- Cuantificación sobre datos ($\sum$):

$$\begin{aligned} P &= send(0).P + send(1).P + send(2).P + ... \\ &\sim \\ P &= \sum_{n:Nat} send(n).P \end{aligned}$$

Bounded Buffer:

Bounded Buffer:

- Datos:

$$\begin{array}{llll} empty: & \rightarrow List & cons: & Nat \times List \rightarrow List \\ head: & List \rightarrow Nat & tail: & List \rightarrow List \\ len: & List \rightarrow Nat & & \end{array}$$

$$\begin{array}{llll} head(cons(n, L)) = & n & tail(cons(n, L)) = & L \\ len(empty) = & 0 & len(cons(n, L)) = & S(len(L)) \end{array}$$

Bounded Buffer:

- Datos:

$$\begin{array}{llll} empty: & \rightarrow List & cons: & Nat \times List \rightarrow List \\ head: & List \rightarrow Nat & tail: & List \rightarrow List \\ len: & List \rightarrow Nat & & \end{array}$$

$$\begin{array}{llll} head(cons(n,L)) = & n & tail(cons(n,L)) = & L \\ len(empty) = & 0 & len(cons(n,L)) = & S(len(L)) \end{array}$$

- Proceso:

$$\begin{array}{lll} Buffer(l:List) & = & \sum_{n:Nat} len(l) < Max \rightarrow put(n).Buffer(cons(n,l)) \\ & + & len(l) > 0 \rightarrow get(head(l)).Buffer(tail(l)) \end{array}$$

Model Checking:

- Algoritmo automático sobre sistemas de transiciones.

Model Checking:

- Algoritmo automático sobre sistemas de transiciones.
- Sólo se puede usar en sistemas finitos (aunque hay especificaciones algebraicas infinitas)

Model Checking:

- Algoritmo automático sobre sistemas de transiciones.
- Sólo se puede usar en sistemas finitos (aunque hay especificaciones algebraicas infinitas)
- Las propiedades normalmente se expresan en una lógica temporal.

$$\varphi ::= T \mid F \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle a \rangle \varphi \mid [a] \varphi$$

Model Checking:

- Algoritmo automático sobre sistemas de transiciones.
- Sólo se puede usar en sistemas finitos (aunque hay especificaciones algebraicas infinitas)
- Las propiedades normalmente se expresan en una lógica temporal.

$$\varphi ::= T \mid F \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle a \rangle \varphi \mid [a] \varphi$$

  - $\langle a \rangle \varphi$. Existe una transición a que lleva a un estado que satisface φ.

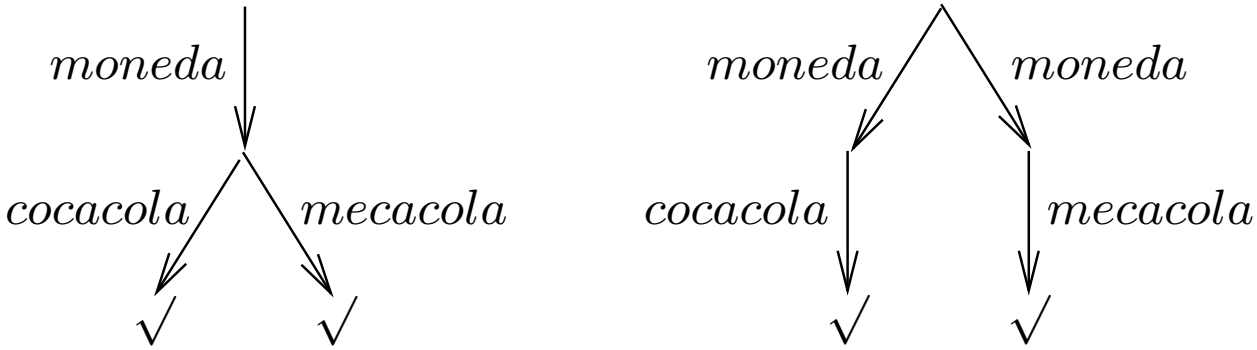
Model Checking:

- Algoritmo automático sobre sistemas de transiciones.
- Sólo se puede usar en sistemas finitos (aunque hay especificaciones algebraicas infinitas)
- Las propiedades normalmente se expresan en una lógica temporal.

$$\varphi ::= T \mid F \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle a \rangle \varphi \mid [a] \varphi$$

  - $\langle a \rangle \varphi$. Existe una transición a que lleva a un estado que satisface φ.
  - $[a] \varphi$. Todas las transiciones a llevan a un estado que satisface φ.

Ejemplo:



$$[moneda]\langle mecacola\rangle T$$

El de la derecha satisface la formula, el de la izquierda no.

Conclusión:

- El álgebra de procesos es potente formalismo para especificar sistemas concurrentes.

Conclusión:

- El álgebra de procesos es potente formalismo para especificar sistemas concurrentes.
- Es completamente formal → basada en teorías algebraicas.

Conclusión:

- El álgebra de procesos es potente formalismo para especificar sistemas concurrentes.
- Es completamente formal → basada en teorías algebraicas.
- Integra diversos métodos de verificación.

Conclusión:

- El álgebra de procesos es potente formalismo para especificar sistemas concurrentes.
- Es completamente formal → basada en teorías algebraicas.
- Integra diversos métodos de verificación.
- Existen extensiones para modelar sistemas más complejos: con tiempo, con probabilidades, ...

Conclusión:

- El álgebra de procesos es potente formalismo para especificar sistemas concurrentes.
- Es completamente formal → basada en teorías algebraicas.
- Integra diversos métodos de verificación.
- Existen extensiones para modelar sistemas más complejos: con tiempo, con probabilidades, ...
- Puede parecer críptica o no intuitiva.

Conclusión:

- El álgebra de procesos es potente formalismo para especificar sistemas concurrentes.
- Es completamente formal → basada en teorías algebraicas.
- Integra diversos métodos de verificación.
- Existen extensiones para modelar sistemas más complejos: con tiempo, con probabilidades, ...
- Puede parecer críptica o no intuitiva.
- Está destinada a la especificación y no a la implementación.

Conclusión:

- El álgebra de procesos es potente formalismo para especificar sistemas concurrentes.
- Es completamente formal → basada en teorías algebraicas.
- Integra diversos métodos de verificación.
- Existen extensiones para modelar sistemas más complejos: con tiempo, con probabilidades, ...
- Puede parecer críptica o no intuitiva.
- Está destinada a la especificación y no a la implementación.
- Verificar requiere mucho esfuerzo y conocimiento.