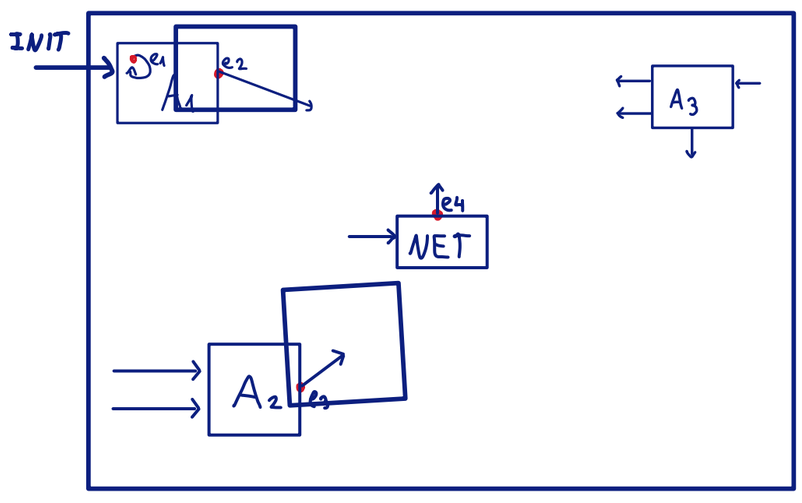
# Introducción

**Características**

- Cada componente del sistema es modelado mediante un autómata independiente ⇒ Adecuado para sistemas asíncronos
- Las operaciones de entrada siempre están habilitadas
- Es indeterminista
- Permite la composición de autómatas
- Modo de operación:
  - Autómatas ⇒ generan *ejecuciones* (secuencia alterna de estados y acciones)
  - Problemas ⇒ son formalizados mediante un conjunto de secuencias de acciones (externas)
    - ▷ Un autómata *resuelve* un problema si los *comportamientos equitativos* del autómata están incluidos en los del problema
- Permite la descripción detallada tanto de problemas como de algoritmos (concurrentes)
- Permite la descripción de algoritmos y sistemas a distintos niveles de abstracción
- Puede ser utilizado como base formal para realizar demostraciones de corrección
- Puede ser utilizado para realizar análisis de complejidades, prueba de límites, etc.



## Modelo de Autómata E/S

DEFINICIÓN **1** *Definimos un autómata $A$ de la siguiente manera:*

1. *Una signatura de un conjunto de acciones, $acc(A)$: $\{ent(A), sal(A), int(A)\}$*
   - $ext(A) = ent(A) \cup sal(A)$
   - $local(A) = sal(A) \cup int(A)$
2. *Un conjunto de estados, $estados(A)$*
3. *Un conjunto no vacío de estados iniciales, $inicio(A) \subseteq estados(A)$*
4. *Una relación de transición, $pasos(A) \subseteq estados(A) \times acc(A) \times estados(A)$, cumpliéndose que $\forall\ s \in estados(A)$ y $\forall\ \pi \in ent(A) \Rightarrow$*
   $\exists\ (s, \pi, s') \in pasos(A)$
5. *Una relación de equivalencia $part(A)$, dividiendo el conjunto $local(A)$ en, al menos, un número finito de clases de equivalencia*

▷ Cada elemento de la relación de transición representa un posible paso en la computación del sistema que modela el autómata (*paso*).

Si $(s, \pi, s')$ es un paso de $A$, diremos que $\pi$ está habilitado en $s'$.

▷▷ El modelo está *habilitado en entrada*

▷ Cuando un autómata se "ejecuta", éste genera una cadena que representa la ejecución del sistema que el autómata modela.

Un *fragmento de ejecución* de $A$ es una secuencia (finita o infinita) alterna de estados y acciones, $s_0, \pi_1, s_1, \pi_2, s_2, \pi_3, ...$, tal que $(s_i, \pi_{i+1}, s_{i+1})$ es un paso de $A$ para cada $i$.

Una *ejecución* es un fragmento de ejecución, empezando con un estado inicial ($ejecs(A)$).

Un estado es *alcanzable* si es el estado final de una ejecución finita.

Un *paso extendido* de un autómata $A$ consiste en un triplete $(s, \alpha, s')$, donde $s$ y $s'$ son estados de $A$, $\alpha$ es una secuencia de acciones de $A$ (que puede ser vacía) y existe un fragmento de ejecución de $A$ con $s$ como estado inicial, $s'$ como estado final y $\alpha$ como historia.

▷ En ciertas ocasiones sólo estamos interesados en las acciones ejecutadas y no en los estados.

La *historia* de un fragmento de ejecución $\alpha$ es la subsecuencia formada por las acciones que aparecen en $\alpha$ ($hist(\alpha)$).

- $\beta$ es una historia de un autómata $A$ si es una historia de una ejecución de $A$.
- $hists(A)$ = conjunto de historias de $A$
- $finhists(A)$ = conjunto de historias finitas de $A$

Los estados no se detallan completamente, solo la parte relevante

▷ El *comportamiento* de una ejecución o ~~historia~~ $\alpha$ es la subsecuencia formada por las acciones externas que aparecen en $\alpha$ ($comp(\alpha)$). Intuitivamente, $comp(\alpha)$ es la porción de $\alpha$ externamente observable.

Comportamiento: propiedades requeridas a una ejecución.

- $\beta$ es un comportamiento de un autómata $A$ si es un comportamiento de una ejecución de $A$.
- $comp(A)$ = conjunto de comportamientos de $A$
- $fincomp(A)$ = conjunto de comportamientos finitos de $A$

Notación: *evento* = instancia de una acción

**Ejemplo**: Modelado de máquinas de autoservicio

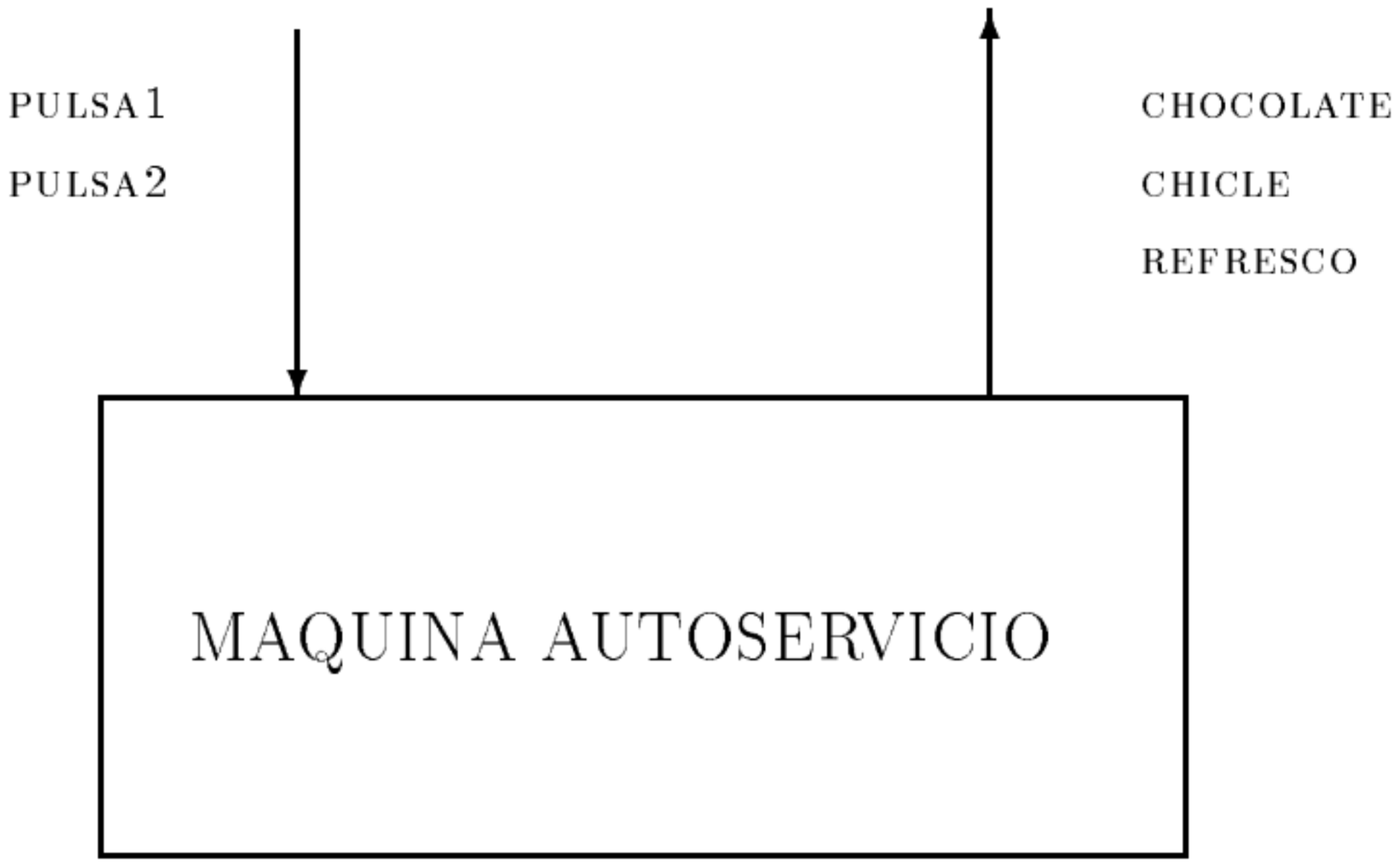
Signatura de acciones:

- Acciones de entrada: PULSA1, PULSA2
- Acciones de salida: CHOCOLATE, CHICLE, REFRESCO
- Acciones internas: ninguna

Estados: → Variables

- Botón_Pulsado: toma como valores 0, 1 y 2 (siendo el valor inicial 0)

Partición: {CHOCOLATE, CHICLE, REFRESCO}



**MA-1**: Cuando se pulsa el botón 1 proporciona chocolate y cuando se pulsa el botón 2 o chicle o refresco (de manera indeterminista)

Las acciones se guardan en variables

Especificación

Las variables son luego precondiciones de algún estado

- PULSA1:

  Efecto: Botón_Pulsado $\leftarrow 1$

- PULSA2:

  Efecto: Botón_Pulsado $\leftarrow 2$

- CHOCOLATE:

  Precondición: Botón_Pulsado $= 1$

  Efecto: Botón_Pulsado $\leftarrow 0$

- CHICLE:

  Precondición: Botón_Pulsado $= 2$

  Efecto: Botón_Pulsado $\leftarrow 0$

- REFRESCO:

  Precondición: Botón_Pulsado $= 2$

  Efecto: Botón_Pulsado $\leftarrow 0$

Estado quiescente: modo pasivo, esperando entrada

$\alpha(MA\text{-}1) = \boxed{P1}\ CHIO \cdot P2 \cdot REF \cdot P2 \cdot REF$
$\boxed{P2}\ \ P1 \qquad\qquad\qquad\qquad\qquad\quad CHI$
$\qquad\quad P2 \qquad\qquad\qquad\qquad\qquad\quad P1$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad P2$

Hay dos enfoques:

- encolar acciones
- acción-ejecución

Pulsa_1
Efecto:
$q \leftarrow q \cdot P_1$

Pulsa_2
Efecto:
$q \leftarrow q \cdot P_2$

CHIO
Prec:
$q = P_1 \cdot q'$
Efecto:
$q \leftarrow q'$

Una ejecución justa/equitativa prohibe que haya una acción habilitada de manera permanente (infinito) y no se ejecute nunca.

$\alpha = P1 \cdot CH0 \cdot P2 \cdot CHI \cdot P2 \cdot CHI \cdot P2 \cdot CHI$

$\{CHI, CH0, REF\}$ Justa * Hilo por conjunto

$\{\{CH0\}\ \{CHI, REF\}\}$ Justa

$\{\{CH0, CHI\}, \{REF\}\}$ NO : La condición de REF la escucha el primer hilo

Un hilo por precondición o todas las precondiciones en un hilo

$\alpha = P1 \cdot CH0\ P1 \cdot CH0 \ldots$ / $\{P1, P2\}$ ✓ / $\{\{P1\}, \{P2\}\}$ ✗

$\alpha = \{\}$ No

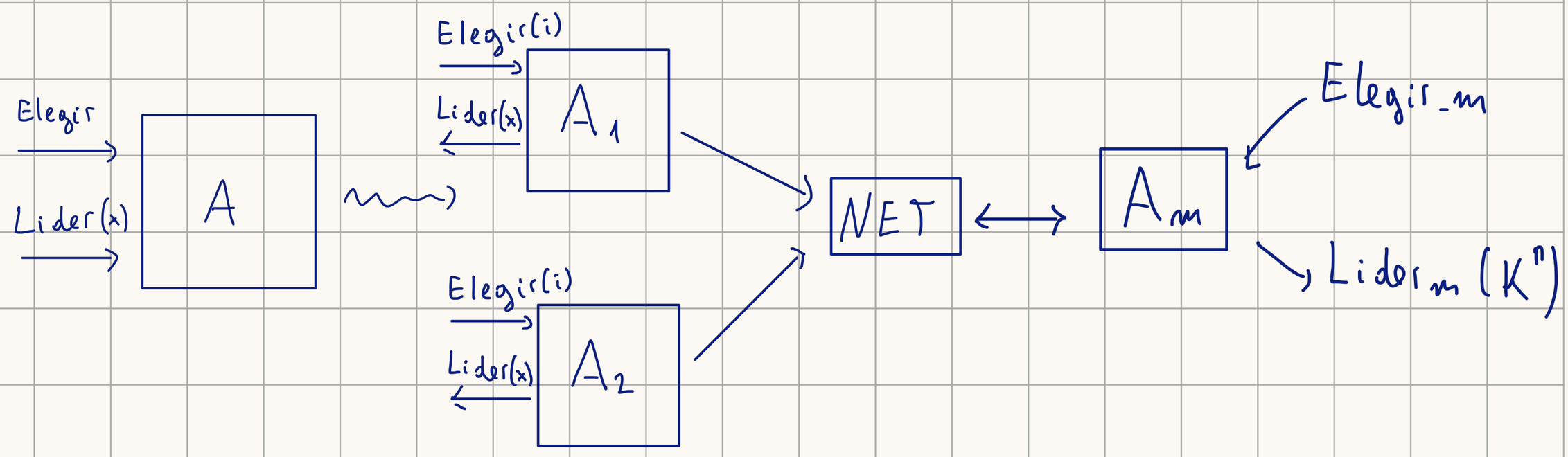
Justicia en un conjunto Finito

$\alpha = P1$. No. CH0 está habilitado pero no se ejecuta

$\alpha = P1 \cdot CH0$. Sí $\alpha = \ldots$ CH0 CHI REF Sí

$\alpha = \{\}$ ej_eqs(CLI-3): $S_i$

# ELECCIÓN DE LÍDER




Propiedades

(1) Seguridad

(2) viveza

(2) $\alpha = \alpha_1 \cdot ELEGIR_i \cdot \alpha_2$

$\Longrightarrow \alpha_2 = \alpha_{2_1} \cdot LIDER(K) \cdot \alpha_{22}$

(1) $\forall \alpha \in fin\_ejecs(A)$

$\alpha = \alpha_1 \cdot Lider_i(K) \cdot \alpha_2 \cdot Lider_j(K) \cdot \alpha 3 \cdot Lider_i(K)$

$\Longrightarrow K = K''$

1.2 $\alpha \in fin\_ejecs(A) : \alpha = \alpha_i \cdot Lider_i(K) \cdot \alpha_2$

**MA-2**: Cuando se pulsa el botón 1 proporciona chocolate y cuando se pulsa el botón 2 chicle.

Especificación

- PULSA1:

  Efecto: Botón_Pulsado $\leftarrow 1$

- PULSA2:

  Efecto: Botón_Pulsado $\leftarrow 2$

- CHOCOLATE:

  Precondición: Botón_Pulsado $= 1$

  Efecto: Botón_Pulsado $\leftarrow 0$

- CHICLE:

  Precondición: Botón_Pulsado $= 2$

  Efecto: Botón_Pulsado $\leftarrow 0$

- REFRESCO:

  Precondición: Falso

  Efecto: Botón_Pulsado $\leftarrow 0$

**MA-3**: Nunca dispensa nada.

Especificación

- PULSA1:

  Efecto: Botón_Pulsado $\leftarrow 1$

- PULSA2:

  Efecto: Botón_Pulsado $\leftarrow 2$

- CHOCOLATE:

  Precondición: Falso

  Efecto: Botón_Pulsado $\leftarrow 0$

- CHICLE:

  Precondición: Falso

  Efecto: Botón_Pulsado $\leftarrow 0$

- REFRESCO:

  Precondición: Falso

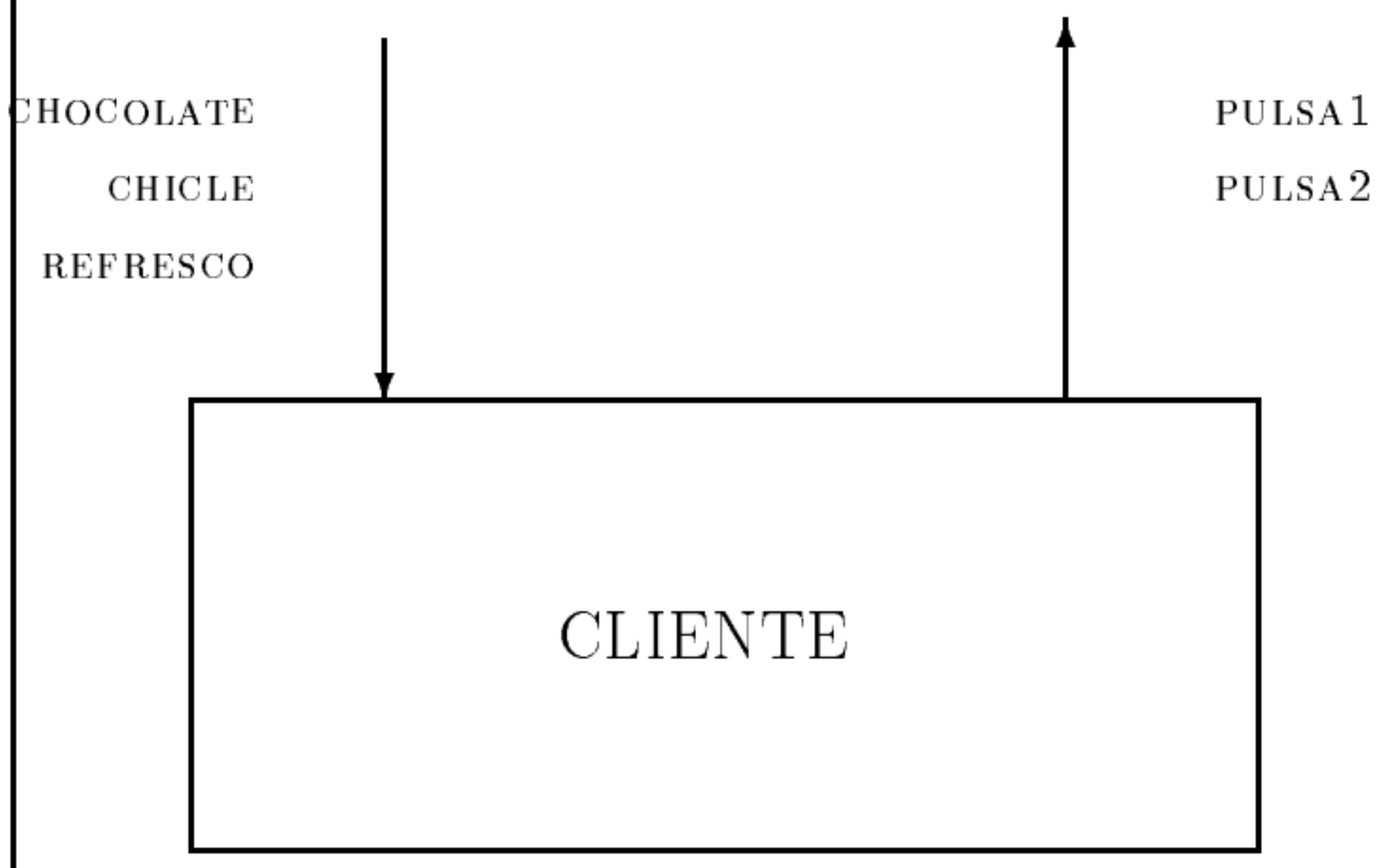
  Efecto: Botón_Pulsado $\leftarrow 0$

**Ejemplo**: Modelado del cliente

Signatura de acciones:

- Acciones de entrada: CHOCOLATE, CHICLE, REFRESCO
- Acciones de salida: PULSA1, PULSA2
- Acciones internas: ninguna

Partición: {PULSA1, PULSA2}



**CLI-1**: Cuando el cliente pulsa un botón, espera el producto.

Estados:

- Espera: toma como valores *si* y *no* (siendo el valor inicial *no*)

Especificación

- CHOCOLATE:

  Efecto: Espera ← no

- CHICLE:

  Efecto: Espera ← no

- REFRESCO:

  Efecto: Espera ← no

- PULSA1:

  Precondición: Espera = no

  Efecto: Espera ← si

- PULSA2:

  Precondición: Espera = no

  Efecto: Espera ← si

**CLI-2**: Pulsa repetidamente el botón 2 hasta recibir un refresco, y después sólo el botón 1.

Estados:

- Espera: toma como valores *si* y *no* (siendo el valor inicial *no*)
- Refresco_Recibido: toma como valores *si* y *no* (siendo el valor inicial *no*)

Especificación

- CHOCOLATE:

  Efecto: Espera ← no

- CHICLE:

  Efecto: Espera ← no

- REFRESCO:

  Efecto: Espera ← no; Refresco_Recibido ← si

- PULSA1:

  Precondición: Espera = no; Refresco_Recibido = si

  Efecto: Espera ← si

- PULSA2:

  Precondición: Espera = no; Refresco_Recibido = no

  Efecto: Espera ← si

**CLI-3**: Actúa de manera similar a CLI-1, excepto que puede saciarse (y ya no pide nada más).

Signatura de acciones:

- Acciones de entrada: CHOCOLATE, CHICLE, REFRESCO
- Acciones de salida: PULSA1, PULSA2
- Acciones internas: SACIARSE

Estados:

- Espera: toma como valores *si* y *no* (siendo el valor inicial *no*)
- Saciado: toma como valores *si* y *no* (siendo el valor inicial *no*)

Partición: {PULSA1, PULSA2, SACIARSE}

Especificación

- CHOCOLATE:

  Efecto: Espera $\leftarrow$ no

- CHICLE:

  Efecto: Espera $\leftarrow$ no

- REFRESCO:

  Efecto: Espera $\leftarrow$ no

- PULSA1:

  Precondición: Espera = no; Saciado = no

  Efecto: Espera $\leftarrow$ si

- PULSA2:

  Precondición: Espera = no; Saciado = no

  Efecto: Espera $\leftarrow$ si

- SACIARSE:

  Precondición: Espera = no; Saciado = no

  Efecto: Saciado $\leftarrow$ si

## Equidad de Autómatas

Supongamos que el cliente que utiliza MA-1 no espera el producto antes de volver a pulsar un botón. Entonces, un comportamiento de MA-1 sería la secuencia infinita PULSA1 PULSA1 PULSA1 PULSA1 PULSA1 $\cdots$, en la cual se pulsa repetidamente el botón 1 sin dar la oportunidad a la máquina a dispensar el producto.

Observación: La máquina sólo puede realizar su trabajo correctamente cuando es tratada de forma equitativa, es decir, cuando tiene la oportunidad de responder a las entradas.

Por ello estaremos interesados en las ejecuciones en las que los componentes son tratados equitativamente.

DEFINICIÓN **2** *Una ejecución equitativa de un autómata $A$ es definida como una ejecución $\alpha$ de $A$ tal que para cada clase $C \in part(A)$ se cumple:*

1. *Si $\alpha$ es finita, entonces ninguna acción en $C$ está habilitada en el estado final de $\alpha$*
2. *Si $\alpha$ es infinita, entonces se cumple que, o $\alpha$ contiene un número infinito de eventos de $C$, o $\alpha$ contiene un número infinito de estados en los que ninguna acción de $C$ está habilitada*

▷ La anterior definición dice que una ejecución equitativa proporciona turnos equitativos a cada clase $C \in part(A)$, y por lo tanto a cada componente del sistema que está siendo modelado.

Denotaremos al conjunto de ejecuciones equitativas de $A$ como $equejecs(A)$.

- $\beta$ es una historia equitativa de $A$ si es la historia de una ejecución equitativa de $A$ ($equhists(A)$).
- $\beta$ es un comportamiento equitativo de $A$ si es un comportamiento de una ejecución equitativa de $A$ ($equcomps(A)$).

Ejemplo: Las acciones localmente controladas de MA-1 son $\{$CHOCOLATE, CHICLE, REFRESCO$\}$.

▷ El comportamiento PULSA1 PULSA1 PULSA1 PULSA1 $\cdots$, de MA-1 no es equitativo, mientras que PULSA1 PULSA1 CHOCOLATE PULSA1 PULSA1 CHOCOLATE $\cdots$, si que lo es ya que infinitamente a menudo una acción de salida de MA-1 se ejecuta.

▷ Sin embargo PULSA2 CHICLE PULSA2 CHICLE $\cdots$ nunca da refresco. Si queremos que también lo de, podemos variar la partición de manera que $part(MA-1) = \{\{$CHOCOLATE, CHICLE$\}, \{$REFRESCO$\}$.

▷ Pero si tenemos PULSA1 CHOCOLATE PULSA2 CHICLE PULSA2 CHICLE $\cdots$, tampoco da nunca refresco. Tenemos pues que variar la partición $part(MA-1) = \{\{$CHOCOLATE$\}, \{$CHICLE$\}, \{$REFRESCO$\}$.

Ejemplo: En CLI-3, cualquier ejecución finita terminando con la acción SACIARSE es una ejecución equitativa, ya que en el estado que sigue a dicha acción ninguna acción está habilitada.

## Especificación de Problemas

▷ Para especificar un problema es necesario definir las "soluciones" que permite, así como la interface entre la solución y el entorno.

▷ Un autómata "resuelve" la especificación de un problema si cada una de sus "soluciones" está incluida en el conjunto de las "soluciones" permitidas por la especificación del problema.

DEFINICIÓN **3** *Definimos un módulo de historias H, mediante:*

1. *Una signatura de acciones,* $sig(H)$
2. *Un conjunto de historias,* $hists(H)$

$hists(H) \rightarrow finhists(H) \rightarrow equhists(H)$

$comps(H) \rightarrow fincomps(H) \rightarrow equcomps(H)$

▷ Un *problema* es simplemente un módulo de historias $P$.

DEFINICIÓN **4** *Un autómata A resuelve un problema P si P y A tienen la misma signatura externa y* $equcomps(A) \subseteq equcomps(P)$.

DEFINICIÓN **5** *Un autómata A implanta un problema P si P y A tienen la misma signatura externa y* $fincomps(A) \subseteq fincomps(P)$.

▷ Utilizaremos el término *módulo* para referirnos, tanto a un autómata como a un módulo de historias.

DEFINICIÓN **6** *Sean $M$ y $M'$ dos módulos (bien autómatas o módulos de historias). $M$ resuelve $M'$ si $M$ y $M'$ tienen la misma signatura externa y $equcomps(M) \subseteq equcomps(M')$. $M$ implanta $M'$ si $M$ y $M'$ tienen la misma signatura externa y $fincomps(M) \subseteq fincomps(M')$.*

Esto nos permite realizar pruebas jerarquicas de forma modular:
Demostraremos que un autómata resuelve un problema demostrando que el autómata resuelve otro autómata, el cual resuelve otro autómata, el cual resuelve otro autómata, etc., hasta que el autómata final resuelve el problema original.

En general no se cumple que si un módulo $M$ resuelve otro módulo $M'$ entonces también lo resuelve. Lo mismo se aplica para la implantación.

Ejemplo: En CLI-1 todas las acciones controladas localmente (PULSA1 y PULSA2) están en la misma partición, al igual que en CLI-3 (PULSA1, PULSA2 y SACIARSE). Además ambos autómatas tienen la misma signatura externa.

Aunque CLI-3 implanta CLI-1, no lo resuelve: la secuencia vacía obtenida a partir de la historia SACIARSE es un comportamiento equitativo de CLI-3, aunque no de CLI-1 (ya que PULSA1 y PULSA2 están habilitados).

## Especificación de Propiedades

**Propiedad**: La modelaremos mediante un conjunto de comportamientos.

**Propiedades de seguridad**: De manera informal, podemos decir que están caracterizadas por el hecho de que especifican un propiedad que debe mantenerse en cualquier estado de la computación. Así pues, debe satisfacerse en cada prefijo finito de la computación $\Rightarrow$ la noción de comportamiento más adecuado es el de comportamientos finitos.

Ejemplo de propiedad de seguridad en las máquinas MAs: Cada CHOCOLATE es inmediatamente precedido de PULSA1 y cada CHICLE o REFRESCO precedido de PULSA2.

Queremos demostrar que las máquinas de autoservicio que hemos definido cumplen está propiedad.

Para ello definimos un módulo de historias MA-SEGURO con la misma signatura que las MAs y con las historias satisfaciendo la propiedad de seguridad anterior.

Objetivo: Demostrar que las MAs *implantan* MA-SEGURO.

Demostración: (Por inducción).

**Propiedades de viveza**: Están caracterizadas por el hecho de que especifican un propiedad que debe cumplirse en algún estado futuro de la computación. Así pues, no es una propiedad necesariamente de comportamientos finitos, sino más bien de comportamientos equitativos.

Ejemplo de propiedad de viveza de las máquinas MAs: Cada PULSA1 es inmediatamente seguido de CHOCOLATE y cada PULSA2 inmediatamente seguido de CHICLE o REFRESCO.

Problema: Sabemos que los autómatas están habilitados en entrada. Por lo tanto no pueden bloquear las acciones de entrada.

Solución: Si queremos controlarlas hemos de introducir, explícitamente, condiciones de *buena formación* entre el autómata y el entorno. Por lo tanto, si queremos restringir nuestra atención a las interacciones en las que las acciones de pulsado y dispensación se alternen estrictamente, respecto a las acciones de entrada debemos imponer una condición de buena formación: En el caso de MAs, que exista una alternancia de acciones de entrada y salida, empezando con una de entrada, es decir, que hasta que no se obtenga un producto no se vuelva a pulsar un botón (*MA-bien formadas*).

Así pues, definimos un módulo de historias MA-VIVO con la misma signatura que las MAs y con las historias satisfaciendo el que si están MA-bien formadas, entonces cumplen la propiedad de viveza de MA.

Objetivo: MA-1 y MA-2 *resuelven* MA-VIVO. MA-3 no lo *resuelve*.

Demostración: (Reducción al absurdo).

Objetivo: MA-1 y MA-2 no *resuelven* MA-VIVO.

Demostración: Trivial, existe una ejecución finita con el estado final en el que hay acciones activadas.

## Técnicas de Demostración: Descomposición Jeráquica

Metodología: Para demostrar que un autómata resuelve un problema se demuestra que un autómata dado resuelve un segundo autómata, éste un tercero y así de forma sucesiva hasta demostar que el autómata final resuelve el problema.

Una manera de probar que un autómata $A$ resuelve otro autómata $B$ consiste en establecer una relación entre los estados de $A$ y los de $B$ y usar dicha relación para demostrar que los comportamientos equitativos de $A$ son comportamientos equitativos de $B$.

Supongamos que $A$ y $B$ tienen la misma signatura externa, y supongamos que $f$ es un mapeo de$estados(A)$ en $\mathcal{P}(estados(B))$ (es decir, si $s$ es un estado de $A$ entonces $f(s)$ es un conjunto de posibles estados de $B$). $f$ es un *mapa de posibilidades* de $A$ en $B$ si se cumple:

1. Para todo estado inicial $s_0$ de $A$, existe un estado inicial $t_0$ de $B$ tal que $t_0 \in f(s_0)$.
2. Si $s$ es un estado alcanzable de $A$, $t \in f(s)$ es un estado alcanzable de $B$ y $(s, \pi, s')$ es un paso de $A$, entonces existe un paso extendido $(t, \gamma, t')$ de $B$ tal que:
   (a) $\gamma \mid ext(B) = \pi \mid ext(A)$
   (b) $t' \in f(s')$

Ejemplo: Existe un mapa de posibilidades de CLI-2 en CLI-1 que mapea cada estado $s$ de CLI-2 en el conjunto conteniendo el estado CLI-1 que sólo contiene la variable "espera" de $s$.

En el caso en que estemos interesados en comportamientos finitos y no comportamientos equitativos se cumple:

PROPOSICIÓN **1** *Supongamos que $A$ y $B$ son autómatas con la misma signatura externa. Si existe un mapa de posibilidades de $A$ en $B$, entonces $A$ implanta $B$.*

Ejemplo: La existencia de un mapa de posibilidades de CLI-2 en CLI-1 implica que CLI-2 implanta CLI-1.

## Composición de Autómatas

▷ El formalismo permite la construcción de autómatas que modelan sistemas complejos mediante la composición de los autómatas que modelan los componentes de dicho sistema.

Para ello se identifican las salidas de un autómata con las entradas de otro distinto: cuando un autómata ejecuta una acción de salida $\pi$, todos los autómatas con esa acción como entrada la ejecutan de manera simultánea (los restantes no hacen nada).

Ejemplo: Composición de los autómatas MAs con los CLIs.

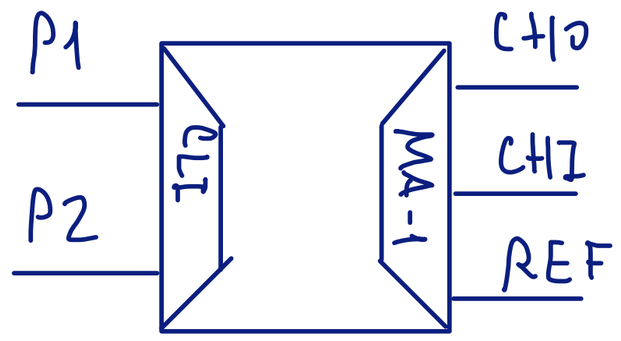
Restricciones en la composición:

1. Dado que las acciones internas de un autómata $A$ no son observables por otro autómata $B$, el conjunto de las acciones internas de $A$ debe ser disjunto con el conjunto de las acciones internas de $B$ (en caso contrario una acción interna de $A$ obligaría a $B$ a tomar un paso).

2. Las acciones de salida de $A$ y $B$ deben formar conjuntos disjuntos (un sólo autómata debe controlar la ejecución de cada acción de salida)

3. Cada acción de la composición debe ser una acción de sólo un número finito de componentes de la composición (en caso contrario, si la composición está formada por infinitos autómatas, una sóla acción podría hacer un "trabajo" infinito)

$A = CLI\text{-}1 \circ MA1 \rightarrow$



DEFINICIÓN **7** *Una colección contable* $\{S_i\}_{i \in I}$ *de signaturas de acciones es fuertemente compatible si* $\forall\, i, j \in I : i \neq j$ *se cumple:*

1. $sal(S_i) \cap sal(S_j) = \emptyset$
2. $int(S_i) \cap sig(S_j) = \emptyset$
3. *No existe ninguna acción incluida en todas las signaturas* $sig(S_i)$*, siendo* $i$ *infinito*

Ejemplo: MA-1 y CLI-1 son fuertemente compatibles

Cuando componemos una colección de autómatas:

1. Las acciones internas de los componentes se convierten en acciones internas de la composición
2. Las acciones externas de los componentes se convierten en acciones externas de la composición
3. Las restantes se convierten en acciones de entrada

Ejemplo: En la composición MA-1 y CLI-1, todas las acciones se transforman en acciones de salida.

DEFINICIÓN **8** *La composición* $S = \Pi_{i \in I} S_i$ *de una colección contable de signaturas de acciones fuertemente compatibles* $\{S_i\}$, *es definida como una signatura de acciones con:*

- $ent(S) = \cup_{i \in I} ent(S_i) - \cup_{i \in I} sal(S_i)$
- $sal(S) = \cup_{i \in I} sal(S_i)$
- $int(S) = \cup_{i \in I} int(S_i)$

DEFINICIÓN **9** *La composición* $A = \Pi_{i \in I} A_i$ *de una colección contable de autómatas fuertemente compatibles* $\{A_i\}_{i \in I}$, *es definida como un autómata cumpliendo:*

- $sig(A) = \Pi_{i \in I} sig(A_i)$
- $estados(A) = \Pi_{i \in I} estados(A_i)$ *(producto cartesiano)*
- $inicio(A) = \Pi_{i \in I} inicio(A_i)$
- $pasos(A) = (\vec{s}, \pi, \vec{s}') : \forall\, i \in I,\ si\ \pi \in acc(A_i)\ entonces$
  $(\vec{s}[i], \pi, \vec{s}'[i]) \in pasos(A_i)\ y\ si\ \pi \neq acc(A_i)\ entonces\ \vec{s}[i] = \vec{s}'[i]$
- $part(A) = \cup_{i \in I} part(A_i)$

PROPOSICIÓN **2** *Sea $\{A_i\}_{i\in I}$ una colección de autómatas fuertemente compatibles y sea $A = \Pi_{i\in I} A_i$. Si $\alpha \in ejecs(A)$ entonces $\alpha \mid A_i \in ejecs(A_i), \forall\, i \in I$.*

El mismo resultado se mantiene si $ejecs()$ es sustituido por $finejecs()$, $equejects()$, $hists()$, $finhists()$, $equhists()$, $comp()$, $fincomp()$ o $equcomp()$.

PROPOSICIÓN **3** *Sea $\{A_i\}_{i\in I}$ una colección de autómatas fuertemente compatibles y sea $A = \Pi_{i\in I} A_i$. Supongamos que $\alpha_i \in ejecs(A_i)$ y supongamos que $\beta$ es una secuencia de acciones de $acc(A)$ tal que $\beta \mid A_i = hist(\alpha_i)$. Entonces $\exists\, \alpha \in ejecs(A) : \beta = hist(\alpha)$ y $\alpha_i = \alpha \mid A_i$.*

El mismo resultado se mantiene si $acc()$, $ejecs()$ e $hist()$ son sustituidos por $ext()$, $equejecs()$ y $comp()$ respectivamente.

PROPOSICIÓN **4** *Sea $\{A_i\}_{i\in I}$ una colección de autómatas fuertemente compatibles y sea $A = \Pi_{i\in I} A_i$. Sea $\beta$ una secuencia de acciones de $acc(A)$. Si $\beta \mid A_i = hist(\alpha_i)$ entonces $\beta \in hists(A)$.*

El mismo resultado se mantiene si $acc()$ es sustituido por $ext()$ e $hist()$ es sustituido por $equhists()$, $comp()$ o $equcomp()$.

PROPOSICIÓN **5** *Sea $\{A_i\}_{i\in I}$ una colección de autómatas fuertemente compatibles y sea $A = \Pi_{i\in I} A_i$. Entonces $hists(A) = \Pi_{i\in I} hists(A_i)$, $equhists(A) = \Pi_{i\in I} equhists(A_i)$, $comps(A) = \Pi_{i\in I} comps(A_i)$, $equcomps(A) = \Pi_{i\in I} equcomps(A_i)$.*

Cuestión: Ya que representan una comunicación entre los autómatas componentes, no deberían de ser internas ?

Respuesta: Sea un autómata $A$ que tiene una acción de salida $\pi$, la cual es una acción de entrada de los autómatas $B$ y $C$ (básicamente un "broadcast"). Entonces no sería lo mismo componer $(A \cdot B) \cdot C$ que componer $A \cdot B \cdot C$. En el primer caso la acción se ocultaría previamente a la composición de $C$.

▷ Esto dificultaría el razonar de una manera modular.

▷▷ Solución: Continuar con la definición e introducir una operación de "ocultación" que transforme ciertas operaciones en internas.

Si $S$ es una signatura de acciones y $\Sigma \subseteq acc(S)$, entonces $oculta_\Sigma(S) = S'$, donde $ent(S') = ent(S) - \Sigma$, $sal(S') = sal(S) - \Sigma$ e $int(S') = int(S) \cup \Sigma$.

Si $A$ es un autómata y $\Sigma \subseteq acc(A)$, entonces $oculta_\Sigma(A)$ es el autómata $A'$ obtenido a partir de $A$ sustituyendo $sig(A)$ por $oculta_\Sigma(sig(A))$

Ejemplo: Sea CLI-4 una versión de CLI-1 en la cual las precondiciones son siempre verdaderas, es decir, el cliente no espera el producto antes de volver a pulsar un botón.

En el ejemplo anterior, las acciones localmente controladas de MA-1 y CLI-4 son $\{\text{CHOCOLATE}, \text{CHICLE}, \text{REFRESCO}\}$ y $\{\text{PULSA1}, \text{PULSA2}\}$.

Así pues, la partición de las acciones localmente controladas de la composición MA-1 $\cdot$ CLI-4 tiene dos clases de equivalencia ($\{\text{PULSA1}, \text{PULSA2}\}$ y $\{\text{CHOCOLATE}, \text{CHICLE}, \text{REFRESCO}\}$).

La definición de la composición de autómatas garantiza que una clase de equivalencia de un autómata componente sea una clase de equivalencia del autómata composición.

De ese modo, la composición mantiene la "estructura" de los componentes.

Ejemplo: El comportamiento PULSA1 PULSA1 PULSA1 PULSA1 $\cdots$, de la composición MA-1 $\cdot$ CLI-4 no es equitativo, mientras que PULSA1 PULSA1 CHOCOLATE PULSA1 PULSA1 CHOCOLATE $\cdots$, si que lo es ya que infinitamente a menudo una acción de salida de MA-1 se ejecuta, e infinitamente a menudo una acción de salida de CLI-4 también se ejecuta.

## Técnicas de Demostración: Descomposición Modular

Metodología: Se razona acerca del funcionamiento de la composición a través de razonar acerca del funcionamiento de los autómatas componentes.

A menudo, un autómata se comporta correctamente en el contexto de ciertas restricciones sobre las entradas.

Modalidades:

- En el contexto de la composición
- Restricciones definidas por el problema, que describe bajo que condiciones una solución de comporta correctamente (e.g., condición de buena formación).

DEFINICIÓN **10** *Un conjunto de secuencias $\mathcal{P}$ es cerrado bajo prefijo si $\forall\ \beta \in \mathcal{P}$ se cumple que $\beta$ es un prefijo de $\alpha$ y $\alpha \in \mathcal{P}$.*

Ejemplo: El conjunto de secuencias MA-bien formadas es cerrado bajo prefijo.

DEFINICIÓN **11** *Un módulo $M$ es cerrado bajo prefijo si $fincomps(M)$ es cerrado bajo prefijo.*

Ejemplo: El módulo de historias MA-SEGURO es cerrado bajo prefijo.

Un módulo *preserva* una propiedad si se cumple que, si el entorno no viola la propiedad, tampoco la viola el módulo.

DEFINICIÓN **12** *Sea $M$ un módulo cerrado bajo prefijo y sea $\mathcal{P}$ un conjunto no vacío cerrado bajo prefijo de secuencias de acciones $\Phi$ tal que $\Phi \cap int(M) = \emptyset$. Diremos que $M$ preserva $\mathcal{P}$ si $\beta\pi \mid \Phi \in \mathcal{P}$ cuando $\beta \mid \Phi \in \mathcal{P}$, $\pi \in sal(M)$ y $\beta\pi \mid M \in fincomps(M)$.*

Ejemplo: MA-1 preserva MA-buena formación: aunque se puede pulsar el botón varias veces sin esperar producto, la máquina sólo dispensa producto si un botón ha sido pulsado desde que el último producto fue dispensado.

Un autómata es *cerrado* si no tiene acciones de entrada.

PROPOSICIÓN **6** *Sea $A$ un autómata cerrado y sea $\mathcal{P}$ un conjunto no vacío cerrado bajo prefijo de secuencias de acciones $\Phi$ tal que $\Phi \cap int(A) = \emptyset$. Si $A$ preserva $\mathcal{P}$ entonces $fincomps(A) \mid \Phi \subseteq \mathcal{P}$.*

PROPOSICIÓN **7** *Sea $\{A_i\}_{i \in I}$ una colección de autómatas fuertemente compatibles cumpliendo que $A = \Pi_{i \in I} A_i$ es un autómata cerrado. Sea $P$ un problema con la signatura externa de $A$. Si $\forall\, i \in I, A_i$ preserva $fincomps(P)$ entonces $A$ implanta $P$.*

Así pues, si conseguimos demostrar que cada componente $A_i$ preserva el comportamiento externo requerido por el problema $P$, entonces habremos demostrado que la composición $A$ preserva el comportamiento externo, y dado que $A$ no tiene acciones de entrada que puedan ser responsables de violar el comportamiento requerido por $P$, entonces todos los comportamientos finitos de $A$ lo serán también de $P$.

PROPOSICIÓN **8** *Sea* $\{A_i\}_{i \in I}$ *una colección de autómatas fuertemente compatibles y sea* $\{P_i\}_{i \in I}$ *una colección de problemas. Si* $\forall\, i \in I, A_i$ *resuelve* $P_i$*, entonces* $\Pi_{i \in I} A_i$ *resuelve* $\Pi_{i \in I} P_i$*.*

Por lo tanto podemos demostrar que la composición de los autómatas $\{A_i\}_{i \in I}$ resuelve un problema, demostrando que cada componente $A_i$ resuelve un problema $P_i$ y después probando que la composición de dichos problemas resuelve el problema original.

Ejemplo: