

## 1. INTRODUCCIÓN

## 2. PROPIEDADES DE LA COMUNICACIÓN ASINCRÓNICA

### 2.1. INTRODUCCIÓN

### 2.2. DEFINICIONES PRELIMINARES

A continuación presentaremos algunas definiciones y resultados preliminares que serán de utilidad en las restantes secciones de esta tesis.

- Def. Clausura reflexo-transitiva. Sea  $\rightarrow$  una relación sobre un conjunto  $S$ . La clausura reflexo-transitiva de  $\rightarrow$ , denotada  $\rightarrow^*$ , se define como la relación reflexiva y transitiva más pequeña sobre  $S$  que contiene a  $\rightarrow$ .
- Def. Relación de equivalencia es una relación binaria reflexiva, simétrica y transitiva. La relación .es igual a.es el ejemplo canónico donde para todo trío de objetos  $a$ ,  $b$  y  $c$ , vale:
  1.  $a = a$  (propiedad reflexiva)
  2. Si  $a = b$  entonces  $b = a$  (propiedad de simetría)
  3. Si  $a = b$  y  $b = c$  entonces  $a = c$  (propiedad transitiva)
- Prop. La clausura reflexo-transitiva de una relación binaria es una relación de equivalencia

### 2.3. COMMUNICATING FINITE STATE MACHINES

El concepto de *Communicating Finite State Machines* – CFSM fue introducido en [1] con el objetivo modelar y estudiar el comportamiento de sistemas distribuidos constituidos por un conjunto de procesos secuenciales que ejecutan concurrentemente y se comunican a partir de intercambiar mensajes a través de canales de comunicación previamente declarados. Fijamos .

**Definición 1** (Communicating Finite State Machines). Sea  $\mathcal{M}$  un conjunto finito de mensajes y un conjunto finito de mensajes y un conjunto finito de participantes  $\mathcal{P}$  un conjunto finito de participantes, definimos una *communicating finite state machine* – CFSM sobre  $\mathcal{M}$  como un sistema de transición finito  $(Q, C, q_0, \mathcal{M}, \delta)$  donde

- $Q$  es un conjunto finito de estado;
- $C = \{pq \in \mathcal{P}^2 \mid p \neq q\}$  es un conjunto de canales
- $q_0 \in Q$  es el estado inicial;
- $\delta \subseteq Q \times (C \times \{!, ?\} \times \mathcal{M}) \times Q$  es un conjunto finito de transiciones.

A partir de la introducción de las CFSMs, la siguiente definición introduce el concepto de *Communicating System* [2, Def.2.2] como un conjunto de CFSMs que cumplen determinadas propiedades.

**Definición 2** (Communicating System). *Dada una CFSM  $M_p = (Q_p, q_{0_p}, \mathcal{M}, \delta_p)$  para cada participante  $p \in \mathcal{P}$ , la tupla  $S = (M_p)_{p \in \mathcal{P}}$  es un communicating system CS. Una configuración de  $S$  es un par  $s = (\vec{q}; \vec{w})$  donde  $\vec{q} = (q_p)_{p \in \mathcal{P}}$  con  $q_p \in Q_p$  y donde  $\vec{w} = (w_{pq})_{pq \in C}$  con  $w_{pq} \in \mathcal{M}^*$ . La componente  $\vec{q}$  es el estado de control y  $q_p \in Q_p$  es el estado local de la máquina  $M_p$ . La configuración inicial de  $S$  es  $s_0 = (\vec{q}_0; \vec{\epsilon})$  con  $\vec{q}_0 = (q_{0_p})_{p \in \mathcal{P}}$*

La semántica de los CSs está dada por un sistema de transición etiquetado [?] cuyos estados y transiciones determinan las posibles ejecuciones del conjunto de procesos sobre el que está definido el sistema.

**Definición 3** (Estados y configuraciones alcanzables). *Una configuración  $s' = (\vec{q}', \vec{w}')$  es alcanzable desde otra configuración  $s = (\vec{q}, \vec{w})$  a través de la ejecución de la transición  $t$  (escrito  $s \xrightarrow{t} s'$ ) si existe un  $m \in \mathcal{M}$  tal que ocurre una de las siguientes alternativas:*

1.  $t = (q_p, pq!m, q'_p) \in \delta_p$  y
  - a)  $q'_{p'} = q_{p'}$  for all  $p' \neq p$ ; y
  - b)  $w'_{pq} = w_{pq} \cdot m$  and  $w'_{p'q'} = w_{p'q'}$  for all  $p'q' \neq pq$ ; O bien
2.  $t = (q_q, pq?q, q'_q) \in \delta_q$  y
  - a)  $q'_{p'} = q_{p'}$  para todo  $p' \neq q$ ; y
  - b)  $m \cdot w'_{pq} = w_{pq}$  y  $w'_{p'q'} = w_{p'q'}$  para todo  $p'q' \neq pq$

Escribimos  $s_1 \xrightarrow{t_1 \dots t_m} s_{m+1}$  cuando para algún  $s_2, \dots, s_m$ ,  $s_1 \xrightarrow{t} s_2 \dots s_m \xrightarrow{t_m} s_{m+1}$ . El conjunto de configuraciones alcanzables de  $S$  es  $RS(S) = \{s | s_0 \rightarrow^* s\}$

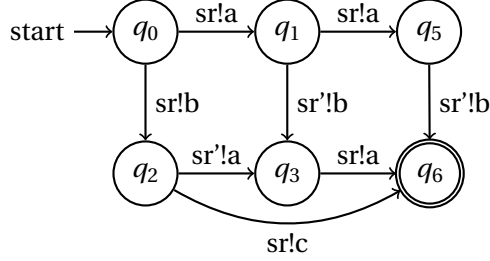
En adelante presentaremos definiciones y resultados que permiten expresar propiedades de los CSs introducidos en la definición anterior.

**Definición 4** (Communicating System seguro). *Sea  $S$  un CS; se dice que  $S$  es seguro si para cada  $s \in RS(S)$ : 1.  $s$  no es una configuración de deadlock, 2.  $s$  no posee mensajes huérfanos, y 3.  $s$  no posee recepciones no especificadas. Para poder expresar esta condición de seguridad es necesario identificar conjuntos de acciones que pueden ser llevadas a cabo concurrentemente. Para esto definimos las siguientes relaciones sobre el conjunto de transiciones de una CFSM. Dados  $q, q' \in Q$ , se define  $act(q, q') = \{\ell \mid (q, \ell, q') \in \delta\}$  y  $\diamond, \blacklozenge \subseteq \delta \times \delta$  como las relaciones de equivalencia más pequeñas que contienen  $\underline{\diamond}$  y  $\underline{\blacklozenge}$  donde:*

- $(q_1, \ell, q_2) \underline{\diamond} (q'_1, \ell, q'_2)$  sii  $\ell \notin act(q_1, q'_1) \wedge act(q_1, q'_1) = act(q_2, q'_2) \wedge act(q_2, q'_2) \neq \emptyset$
- $(q_1, \ell, q_2) \underline{\blacklozenge} (q'_1, \ell, q'_2)$  sii  $(q_1, \ell, q_2) \underline{\diamond} (q'_1, \ell, q'_2)$  y para todo  $(q, \ell, q') \in [(q_1, \ell, q_2)]^{\underline{\diamond}}$ ,  $act(q_1, q) = act(q_2, q') \wedge act(q'_1, q) = act(q'_2, q')$

donde  $[(q_1, \ell, q_2)]^\diamond$  es la clase de equivalencia de  $(q, \ell, q')$  respecto de la relación  $\diamond$  (resp.  $\blacklozenge$ ). Intuitivamente dos transiciones están  $\blacklozenge$ -relacionadas si se refieren a la misma acción aún teniendo en cuenta el interleaving.

**Example 1** (Ejemplo).



- 1)  $(q_0, sr!a, q_1) \underline{\diamond} (q_2, sr!a, q_3)$
- 2)  $(q_0, sr!a, q_1) \blacklozenge (q_2, sr!a, q_3)$
- 3)  $\neg((q_0, sr!a, q_1) \underline{\diamond} (q_1, sr!b, q_5))$
- 4)  $(q_0, sr!b, q_2) \underline{\diamond} (q_1, sr!b, q_3)$
- 5)  $\neg((q_0, sr!b, q_2) \blacklozenge (q_1, sr!b, q_3))$

Las relaciones en (1-2) se sostienen dado que ambas transiciones están interleaved (traducir) con  $sr!b$ . La relación en (3) no se sostiene debido a que la transición entre el origen de una ( $q_0$ ) y el del otro ( $q_1$ ) pasa por  $sr!a$ . Ambas transiciones en (3) son secuenciales, no concurrentes. La relación en (4) se sostiene, pero en (5) no porque  $(q_5, sr!b, q_6)$  está en la clase de *Diamond*-equivalencia de  $(q_0, sr!b, q_2)$  para la cual la condición no se sostiene (debido a la transición con la etiqueta  $sr!c$ ).

**Definición 5** (Eventos). *Dados un conjunto de participantes  $\mathcal{P}$  y un conjunto de mensajes  $\mathcal{M}$  definimos un evento  $e$  como:*

- $e = (q_s, q_r, s, r, a) = (q_s, q_r, s \rightarrow r : a)$  tal que  $s, r \in \mathcal{P}$  son CFSMs

Un evento  $e$  es una tupla  $(q_s, q_r, s, r, a)$ , también escrita como  $(q_s, q_r, s \rightarrow r : a)$  indica que  $s$  y  $r$  se pueden intercambiar el mensaje  $a$ , cuando están en el estado  $q_s$  y  $q_r$ , respectivamente. Para distinguir el paralelismo a nivel máquina introducimos una relación de equivalencia sobre eventos que identifica eventos cuyas transiciones son  $\blacklozenge$ -equivalentes.

**Definición 6** (Equivalencia entre eventos). *Definimos la equivalencia entre eventos como  $\triangleright \triangleleft_s = \triangleright \triangleleft_s \cap \triangleright \triangleleft_r \subseteq \mathcal{E} \times \mathcal{E}$  donde  $\mathcal{E}$  es el conjunto de eventos del sistema y se cumplen las siguientes condiciones*

- $(q_1, q_2, s \rightarrow r : a) \triangleright \triangleleft_s (q'_1, q'_2, s \rightarrow r : a) \iff \forall (q_1, SR!a, q_3), (q'_1, SR!a, q'_3) \in \delta_s : (q_1, SR!a, q_3) \blacklozenge (q'_1, SR!a, q'_3)$
- $(q_1, q_2, s \rightarrow r : a) \triangleright \triangleleft_r (q'_1, q'_2, s \rightarrow r : a) \iff \forall (q_2, SR?a, q_3), (q'_2, SR?a, q'_4) \in \delta_r : (q_2, SR?a, q_3) \blacklozenge (q'_2, SR?a, q'_4)$

**Definición 7** (Sistema de transición sincrónico). *Dados un sistema  $S = (M_P)_{P \in \mathcal{P}}$ , sean*

$N = \{ \vec{q} \mid (\vec{q}; \vec{e}) \in RS_1(S) \}$ ,  $\hat{\delta} = \{ (n, e, n') \mid (n; \vec{e})_{s_1} \xrightarrow{sr!asr?a} (n'; \vec{e}) \wedge e = n[s], n[r], s \rightarrow r : a \}$  y  $E = \{ \exists n, n' \in N : (n, e, n') \in \hat{\delta} \} \subseteq \mathcal{E}$  el sistema de transición sincrónico de  $S$  es  $TS(S) = (N, n_0, E / \triangleright \triangleleft, \rightarrow)$  donde  $n_0 = \vec{q}_0$  es el estado inicial,  $n \stackrel{[e]}{\longmapsto} n' \iff (n, e, n') \in \hat{\delta}$ . Fijamos un conjunto  $\hat{E}$  de elementos representativos de cada clase de equivalencia  $\triangleright \triangleleft$  ( $e \downarrow : \hat{E} \subseteq E$  y  $\forall e \in E, \exists! e' \in \hat{E} : e' \in [e]$ )

y escribimos  $n \xrightarrow{e'} n'$  para  $n \xrightarrow{[e]} n'$  cuando  $e' \in [e] \cap \hat{E}$ . Las secuencias de eventos se notan con un símbolo  $\pi$  y extendemos la notación de  $\rightarrow$  en la [?, Def. XXX] a  $\rightarrow$  (ej: si  $\pi = e_1 \dots e_k, n_1 \xrightarrow{\pi} n_{k+1}$  si  $n_1 \xrightarrow{e_1} n_2 \xrightarrow{e_2} \dots \xrightarrow{e_k} n_{k+1}$ ).

$TS(S)$  representa todas las posibles ejecuciones sincrónicas del sistema  $S$ ; y cada transición es etiquetada con un evento  $e$ .

**Definición 8** (Proyecciones). La proyección de un evento  $e$  sobre un participante  $p$ , denotado por  $e \downarrow_p$  se define de la siguiente manera:

$$(q_s, q_r, s \rightarrow r : a) \downarrow_p = \begin{cases} pr!a & \text{if } s = p \\ sp?a & \text{if } r = p \\ \epsilon & \text{en otro caso} \end{cases} \quad (2.1)$$

La proyección se define sobre secuencias de eventos en el modo obvio. La proyección  $TS(S) = (N, n_0, \hat{E}, \rightarrow)$  sobre el participante  $p$ , notada  $TS(S) \downarrow_p$ , es el autómata  $(Q, q_0, Act, \delta)$  donde  $Q = N$ ,  $q_0 = n_0$ ,  $Act$  es el conjunto de etiquetas y  $\delta \subseteq Q \times Act \cup \{\epsilon\} \times Q$  es s.t.  $(n_1, e \downarrow_p, n_2) \in \delta \iff n_1 \xrightarrow{e} n_2$

Introducimos el concepto de generalized multiparty compatibility (GMC) como una condición completa y sólida para construir CFSMs. A partir de este punto, fijamos un sistema  $S = (M_p)_{p \in \mathcal{P}}$  con  $TS(S) = (N, n_0, \hat{E}, \rightarrow)$ . GMC depende de dos condiciones, representabilidad y propiedad de ramificación. La primera determina que para cada máquina, cada traza y cada elección estén representadas. La segunda, que cada vez que hay una decisión en un sistema  $TS(S)$ , una única máquina toma esa decisión y cada uno de los otros participantes es notificado de la rama que se tomó o no participa en esa elección. Representabilidad garantiza que  $TS(S)$  contiene suficiente información para decidir propiedades seguras de cualquier ejecución asincrónica de  $S$ . La propiedad de ramificación asegura que si una rama en  $TS(S)$  representa una elección esta está "bien formada".

**Definición 9.** Para un lenguaje  $\mathcal{L}$ ,  $hd(\mathcal{L})$  devuelve las primeras acciones de  $\mathcal{L}$  si las tiene:

$$hd(\mathcal{L}) = \{\ell \mid \exists q \in Act^* : \ell \cdot q \in \mathcal{L}\} \quad hd(\{\epsilon\}) = \{\epsilon\}$$

Dado  $n \in N$ , sea  $TS(S)\langle n \rangle$  el sistema de transición  $TS(S)$  donde se reemplaza al estado inicial  $n_0$  por  $n$ . Escribimos  $LT(S, n, p)$  para  $\mathcal{L}(TS(S)\langle n \rangle) \downarrow_p$ , es decir  $LT(S, n, p)$  es el lenguaje que se obtiene estableciendo a  $n$  como nodo inicial de  $TS(S)$  y proyectando el nuevo sistema de transición sobre  $p$ .

**Definición 10** (Representabilidad). Un sistema  $S$  es representable si

1.  $\mathcal{L}(M_p) = LT(S, n_0, p)$  y
2.  $\forall q \in Q_p \exists n \in N : n[p] = q \wedge \cup_{(q, \ell, q') \in \delta_p} \{\ell\} \subseteq hd(LT(S, n, p))$

Nota de Eze: es esto lo que pedís en la nota abajo?  
(Charlie)

Agregar intuición sobre qué dicen las definiciones y para que serán usadas. Ayuda al lector a seguir el trabajo.  
(Charlie)

para todo  $p \in \mathcal{P}$

La primera condición es necesaria para asegurar que cada traza de cada máquina esté en  $TS(S)$ . La segunda condición es necesaria para asegurar que cada elección en cada máquina esté representada en  $TS(S)$ .

**Definición 11** (Propiedad de ramificación). *Un sistema  $S$  posee la propiedad de ramificación si para todo  $n \in N$  y para todo  $e_1 \neq e_2 \in \hat{E}$  tq  $n \xrightarrow{e_1} n_1$  y  $n_1 \xrightarrow{e_2} n_2$  luego tenemos*

1. o bien existe  $n' \in N$  tal que  $n_1 \xrightarrow{e_2} n'$  y  $n_2 \xrightarrow{e_1} n'$ , o
2. para cada  $(n'_1, n'_2) \in \text{In}(n, e_1, e_2)$  quedando  
 $L_p^i = \text{hd}(\{e_i \downarrow_p \cdot \phi \mid \phi \in \text{LT}(S, n'_i, p)\})$  con  $i \in \{1, 2\}$  y  $p \in P$ , y se cumplen las condiciones 2a, 2b y 2c definidas abajo
  - (a) *Choice awareness*:  $\forall p \in \mathcal{P}$  valen
    - I.  $L_p^1 \cap L_p^2 \subseteq \{\epsilon\}$  y  $\epsilon \in L_p^1 \iff \epsilon \in L_p^2$ , o
    - II.  $\exists n' \in N, \pi_1, \pi_2: n'_1 \xrightarrow{\pi_1} n' \wedge n'_2 \xrightarrow{\pi_2} n' \wedge (e_1 \cdot \pi_1) \downarrow_p = (e_2 \cdot \pi_2) \downarrow_p = \epsilon$
  - (b) *selector único*:  $\exists! s \in \mathcal{P}: L_s^1 \cap L_s^2 = \emptyset \wedge \exists sr!a \in L_s^1 \cup L_s^2$
  - (c) *no race*:  $\forall r \in \mathcal{P}: L_r^1 \cap L_r^2 = \emptyset \Rightarrow \forall s_1 r?a_1 \in L_r^1, \forall s_2 r?a_1 \in L_r^2: \forall i \neq j \in \{1, 2\}: n'_i \xrightarrow{\pi_i} \Rightarrow \text{dep}(s_i \rightarrow r: a_i, e_i \cdot \pi_i, s_j \rightarrow r: a_j)$

Con esto definimos que toda ramificación es o bien (1) la ejecución concurrente de dos eventos; o, para cada participante  $p$  (2(a)I) si  $p$  no termina antes de  $n$  entonces las primeras dos acciones de  $p$  en dos ramas distintas son disjuntas; o (2(a)II)  $p$  no está involucrado en la elección, o sea la ramas se juntan antes de que  $p$  realice ninguna acción; (2(b)) hay un único participante  $s$  tomando la decisión; y (2(c)) para cada participante  $r$  involucrado en la elección, no puede haber race condition entre los mensajes que puede recibir  $r$ . La no race condition asegura que en ninguna ejecución (asincrónica) de  $S$ , si una máquina tiene más de un buffer no vacío, entonces puede leerlos en cualquier orden (interleaving es posible). Notar que si una máquina  $r$  recibe todos sus mensajes de un mismo emisor, entonces hay una  $\triangleleft$ -relación entre todas sus acciones.

**Definición 12** (General Multiparty Compatibility). *Un sistema  $S$  es generalised multiparty compatible (GMC) si es representable y posee la propiedad de ramificación.*

### 3. COMPOSICIÓN PARCIAL DE COMPONENTES ASINCRÓNICAS

#### 3.1. MOTIVACIÓN

En SOC (service oriented computing) el objetivo es que los servicios se obtengan por demanda. Es decir un sistema de este tipo utiliza distintos servicios según la necesidad. Esto nos da la idea binding incremental, dado que a medida que se vayan necesitando distintas funciones, el sistema va agregando servicios con los cuales se conecta. Por otro lado existe un

procedimiento para determinar si un conjunto de CFSMs puede interactuar entre sí sin errores. El problema es que este procedimiento necesita tener a mano las CFSMs para cada uno de los participantes de la comunicación. Esto va en contra de la idea de conectarse a servicios en tiempo de ejecución. Para modelar esto necesitamos un tipo de autómatas que soporten la composición parcial. Al componer dos CFSMs puede ocurrir que cada máquina se comunique a través de un canal con un tercer participante. Estos canales son independientes en el sentido de que son diferentes buffers FIFO. Por lo tanto para preservar la semántica es necesario que el autómata resultante de la composición tenga dos canales con este tercer participante. Para modelar este comportamiento introducimos los Autómatas Finitos de Comunicación Asíncrona (AFCA). Estos autómatas tienen tres tipos de transiciones: internas que representan procesamiento propio; de buffer que representan comunicación asíncrona interna y permiten representar la comunicación entre dos participantes luego de una composición; y por último transiciones que modelan acciones de comunicación externa como en CFSMs.

### 3.2. AUTÓMATAS FINITOS DE COMUNICACIÓN ASÍNCRÓNICA

**Definición 13** (Cola). *Una cola es un tipo de estructura de datos caracterizada por ser una secuencia de elementos first in-first out (FIFO) debido a que el primer elemento en entrar es el primer elemento en salir. Para manejarla se definen dos operaciones, una para agregar y otra para sacar elementos de la una cola. Para los usos de este trabajo vamos a definir colas en las cuales se depositan y de las cuales se retiran mensajes, con las operaciones:*

- Cola vacía: denotado  $[]$
- Encolar mensaje: denotado como  $b \ll m$ , si  $b$  es una cola y  $m$  es un mensaje.
- Desencolar mensaje: denotado como  $b \gg m$ , si  $b$  es una cola y  $m$  es un mensaje.
- Sea una cola vacía  $b = []$  al aplicarle la operación  $b \ll m$  queda  $b = [m]$
- Sea una cola no vacía  $b = [m_1, \dots, m_n]$  al aplicarle la operación  $b \ll m$  queda  $b = [m, m_1, \dots, m_n]$  del mismo modo al aplicarle  $b \gg m_n$  queda  $b = [m, m_1, \dots]$

**Definición 14** (Autómata finitos de comunicación asíncrona). *Sea  $\mathcal{P}$  un conjunto de participantes y  $\mathcal{M}$  un conjunto de mensajes. Un autómata finitos de comunicación asíncrona es una estructura  $A_{\mathcal{P}} = \langle Q, B, \mathcal{C}, \Sigma, \delta, q_0, F \rangle$  tal que:*

- $Q$  es un conjunto finito de estados,
- $B \subseteq \{pq_n \mid pq \in \mathcal{P}^2, n \in \mathbb{N}, p \neq q\}$  es un conjunto finito de buffers (i.e. colas, ver Def. 13),
- $\mathcal{C} \subseteq \{pq_n \mid pq \in \mathcal{P}^2, n \in \mathbb{N}, p \neq q\}$  es un conjunto de canales tales que  $B \cap \mathcal{C} = \emptyset$
- $\Sigma = \{\Sigma_{Int} \cup \Sigma_{Ex} \cup \Sigma_{Buff}\}$ ,  $\Sigma \cap \mathcal{M} = \emptyset$  es el conjunto de etiquetas del autómata, siendo  
1)  $\Sigma_{Int}$  las acciones internas del autómata

2)  $\Sigma_{Ex}$  un conjunto de etiquetas de la forma  $\langle p_1, p_2, c \rangle$  donde  $p_1, p_2 \in \mathcal{P}$  son, respectivamente, el emisor y el receptor de la comunicación y  $c \in \mathcal{C}$  es el canal a través del cual se resuelve la misma.

3)  $\Sigma_{Buff}$  es el conjunto de etiquetas de las acciones sobre los buffers de la forma  $b \ll m$  o  $b \gg m$ , donde  $b \in B$  y  $m \in \mathcal{M}$ .

■  $\delta = (\delta_{Int} \cup \delta_{Ex} \cup \delta_{Buff})$  siendo:

1)  $\delta_{Int} \subseteq Q \times \Sigma_{Int} \times Q$  es la relación de transición por acciones internas de  $A_{\mathcal{P}}$ ,

2)  $\delta_{Ex} \subseteq Q \times \{In(c, m), Out(c, m) | c \in \Sigma_{Ex} \wedge m \in \mathcal{M}\} \times Q$  es la relación de transición de comunicación externa de  $A_{\mathcal{P}}$ ,

3)  $\delta_{Buff} \subseteq Q \times \Sigma_{Buff} \times Q$

■  $q_0 \in Q$  es el estado inicial, y

■  $F \subseteq Q$  es el conjunto de estados finales.

Se denota  $P(A)$ , al conjunto de participantes que integran el autómata  $A$ , y se define como  $P(A) = \{p \in \mathcal{P} | (\exists \langle p_1, p_2, c \rangle \in \Sigma_{Ex})(p_1 = p \vee p_2 = p)\}$

Explicación de la intuición de la definición.

$\Sigma$  es el conjunto de todas las etiquetas del autómata y lo dividimos en tres subconjuntos disjuntos que la forman  $\Sigma_{Int}$ ,  $\Sigma_{Ex}$  y  $\Sigma_{Buff}$  que corresponden a las acciones de procesamiento interno, las de comunicación externa con otros participantes y las de comunicación interna vía buffers.

$\Sigma_{Ex}$  es el conjunto de etiquetas correspondientes a la comunicación externa del autómata. Como la comunicación es dirigida punto a punto, cada etiqueta incluye a dos participantes  $p_1, p_2 \in \mathcal{P}$  (emisor y receptor) y un canal  $c \in \mathcal{C}$ . Necesariamente en todas las etiquetas vale que  $p_1$  y  $p_2$  son distintos y .

$\Sigma_{Buff}$  es el conjunto de etiquetas de acciones de comunicación interna vía buffers. Se especifican las acciones de encolar y desencolar como  $b \ll m$ ,  $b \gg m$ , respectivamente, donde  $b \in B$  y  $m$  es el mensaje que se inserta/extrae del mismo.

$\delta$  es el conjunto de transiciones que se compone de  $\delta_{Int}$ , las acciones internas,  $\delta_{Ex}$  las transiciones de comunicación con otros agentes y  $\delta_{Buff}$  las acciones de buffer.

$\delta_{Ex}$  se compone de dos tipos de acciones denotadas  $In(c, m)$  y  $Out(c, m)$ , con  $c \in \Sigma_{Ex}$  y  $m \in \mathcal{M}$ . Las acciones de entrada o input que representan la recepción de un mensaje de algún proceso externo al correspondiente al autómata, y las acciones de output o salida que representan el envío de un mensaje y se denotan. Estas acciones son la parte externa de la comunicación asincrónica. Como los canales son unidireccionales decimos que para todo par de participantes existen dos canales, uno por cada sentido.

$\delta_{Buff}$  es la relación de transición de comunicación interna mediante buffers.

Formalmente,  $F \subseteq Q$ . Es el conjunto de estados finales, los posibles estados a los que una ejecución llega y es aceptada como válida. Para los autómatas asincrónicos pedimos también que al llegar a este estado final los buffers se encuentren vacíos. Para esto definimos lo siguiente:

**Definición 15** (Configuración instantánea). *Dados  $\mathcal{P}$  un conjunto de participantes y  $\mathcal{M}$  un conjunto de mensajes. A un autómata con comunicación asincrónica  $A_{\mathcal{P}} = \langle Q, B, \mathcal{C}, \Sigma, \delta, q_0, F \rangle$ , definimos:*

*Una configuración instantánea del autómata como  $\langle q, \Omega \rangle \in Q \times \{[m_b]_{b \in B} \mid m_b \in \mathcal{M}^*\}$  donde  $\mathcal{M}^* = \{[s_0, \dots, s_n] \mid \forall i \in [0, \dots, n], s_i \in \mathcal{M}\}$  y  $[m_b]_{b \in B}$*

- *$q$  es el estado actual*
- *$\sigma$  es la secuencia de acciones por realizar, y*
- *$\Omega$  es la secuencia de buffers (con sus respectivos contenidos hasta el momento). Decimos que  $\Omega$  es de la forma  $[\omega_{b_1}, \omega_{b_2}, \dots, \omega_{b_n}]$  con  $b_i \in B$ .*
- *Decimos que una configuración es inicial si  $q = q_0$  y  $\Omega = [\square, \dots, \square]$*
- *Decimos que una configuración es final si  $q \in F$  y  $\Omega = [\square, \dots, \square]$*

**Definición 16** (Relación de transición entre configuraciones). *Sean  $A_{\mathcal{P}} = \langle Q, B, \mathcal{C}, \Sigma, \delta, q_0, F \rangle$ ,  $q_1, q_2 \in Q$ ,  $m \in \mathcal{M}$  definimos  $\vdash$ , relación de transición entre configuraciones, como:*

1.  $\langle q_1, \Omega \rangle \vdash \langle q_2, \Omega \rangle \iff \langle q_1, r, q_2 \rangle \in \delta \wedge r \in \Sigma \setminus \Sigma_{\text{Buff}}$
2.  $\langle q_1, [\omega_{b_1}, \dots, \omega_{b_i}, \dots, \omega_{b_n}] \rangle \vdash \langle q_2, [\omega_{b_1}, \dots, m : \omega_{b_i}, \dots, \omega_{b_n}] \rangle \iff \langle q_1, b_i \ll m, q_2 \rangle \in \delta$
3.  $\langle q_1, [\omega_{b_1}, \dots, \omega_{b_i} : m, \dots, \omega_{b_n}] \rangle \vdash \langle q_2, [\omega_{b_1}, \dots, \omega_{b_i}, \dots, \omega_{b_n}] \rangle \iff \langle q_1, b_i \gg m, q_2 \rangle \in \delta$

**Definición 17** (Traza de un ACFSA). *Llamamos traza a una secuencia posible de acciones de un autómata. Se define como una secuencia finita de etiquetas de estado y transición alternadas, que comienza y termina con un estado. Dado un autómata  $A_{\mathcal{P}} = \langle Q, B, \mathcal{C}, \Sigma, \delta, q_0, F \rangle$ , una traza tiene las siguientes características*

- *Tiene la forma  $[q_0, \sigma_1, q_1, \dots, q_{n-1}, \sigma_n, q_n]$  donde*
- *$q_0$  es el estado inicial del autómata*
- *$q_i \in Q$ ,*
- *$\sigma_i \in \Sigma$  y*
- *$\langle q_{i-1}, \sigma_i, q_i \rangle \in \delta$*

*El comportamiento de un ACFSA es el conjunto de todas las trazas posibles.*

**Definición 18** (Ejecución de un ACFSA). *. Una ejecución es un  $\tau = \tau_0, \dots, \tau_n$  donde  $\tau_0$  es la configuración inicial,  $\langle \tau_i, \tau_{i+1} \rangle \in \vdash$ ,  $\tau_n$  es una configuración final y  $\forall i \in [1, \dots, n]$ ,  $\tau_i$  es una configuración. Una ejecución es una configuración que no entra en deadlock, mensajes huérfanos ni receptor no especificado.*

**Definición 19** (Deadlock). *Sean  $A_{\mathcal{P}} = \langle Q, B, \Sigma, \delta, q_0, F \rangle$ ,  $q_i \in Q$ ,  $\sigma \in \Sigma^*$ ,  $\Omega = [\omega_{b_1}, \dots, \omega_{b_i}, \dots, \omega_{b_n}]$ , decimos que una configuración está en deadlock cuando ocurre:*



- En la configuración  $\langle q_i, \Omega \rangle$ ,
- $\forall \langle q_i, s, q_j \rangle \in \delta, s \in \Sigma_{\text{Buff}} \wedge (s = b \gg m \Rightarrow \omega_b = [])$

Decimos que ocurre deadlock cuando partiendo de un estado la única transición posible hacia un estado siguiente es consumiendo un mensaje de algún buffer y esos buffers se encuentran vacíos.

**Definición 20** (Mensajes huérfanos). Sean  $A_{\mathcal{D}} = \langle Q, B, \Sigma, \delta, q_0, F \rangle$ ,  $q_i \in Q$ ,  $\sigma \in \Sigma^*$ ,  $\Omega = [\omega_{b_1}, \dots, \omega_{b_i}, \dots, \omega_{b_n}]$ , decimos que una configuración tiene mensajes huérfanos cuando ocurre:

- En la configuración  $\langle q_i, \Omega \rangle$ ,
- $\exists \omega_b \in \Omega \mid \omega_b \neq []$
- $\delta(q_i) = \emptyset$

Decimos que la configuración tiene mensajes huérfanos si, en un estado  $q_i$  quedan mensajes sin consumir en algún buffer y no hay ninguna transición saliente.

**Definición 21** (Receptor no especificado). Sean  $A_{\mathcal{D}} = \langle Q, B, \Sigma, \delta, q_0, F \rangle$ ,  $q_i \in Q$ ,  $\sigma \in \Sigma^*$ ,  $\Omega = [\omega_{b_1}, \dots, \omega_{b_i}, \dots, \omega_{b_n}]$  decimos que una configuración es de receptor no especificado cuando ocurre:

- En la configuración  $\langle q_i, \Omega \rangle$ ,
- $\exists \omega_b \in \Omega \mid \omega_b \neq []$
- $\forall \langle q_i, s, q_j \rangle \in \delta, s \in \Sigma_{\text{Buff}} \wedge (s = b \gg m \Rightarrow \omega_b = [] \vee \omega_b = [\dots, m'], m' \neq m)$

**Definición 22** (Estados alcanzables).

La Figura 3.2 muestra un ejemplo de estos autómatas.

Dibujo

Explicación en castellano del ejemplo

Figura 3.1: Ejemplo de autómata asíncrono de comunicación

Como la idea es que los autómatas con comunicación asincrónica representen procesos o servicios que pueden formar parte de un sistema más grande, necesitamos definir la operación de composición. Para que un par de autómatas E y R sean compatibles para la composición debemos pedirles las siguientes condiciones

1.  $\Sigma_E \cap \Sigma_R = \emptyset$  el conjunto de etiquetas debe ser disjunto, tanto internas, como de entrada/salida y de buffer
2.  $B_E \cap B_R = \emptyset$  el conjunto de Buffers de ambos autómatas debe ser disjunto

**Definición 23** (Composición). *Dados  $\mathcal{P}$  un conjunto de participantes y  $\mathcal{M}$  un conjunto de mensajes. Dos autómatas  $E_{\mathcal{P}} = \langle Q_E, B_E, \mathcal{C}_E, \Sigma_E, \delta_E, q_{0E}, F_E \rangle$  y  $R_{\mathcal{P}} = \langle Q_R, B_R, \mathcal{C}_R, \Sigma_R, \delta_R, q_{0R}, F_R \rangle$ , que cumplan  $\Sigma_E \cap \Sigma_R = \emptyset$  y  $B_E \cap B_R = \emptyset$ , definimos la composición  $E || R_{\mathcal{P}}$  componente a componente.*

- $Q_{ER} = Q_E \times Q_R$  El conjunto de estados de la composición es el producto cartesiano de los estados de los autómatas componentes.
- $B_{ER} = B_E \cup B_R \cup \{\mathcal{C}_E \cap \mathcal{C}_R\}$  es el conjunto de nombres de buffers del autómata resultante. Se compone de los buffers internos de ambos autómatas junto con uno nuevo por cada canal compartido entre ambos autómatas. Los canales compartidos son aquellos mediante los cuales los autómatas a componer intercambian mensajes entre sí.
- $\mathcal{C}_{ER} = \{\mathcal{C}_E \cup \mathcal{C}_R\} \setminus \{\mathcal{C}_E \cap \mathcal{C}_R\}$
- $\Sigma_{ER} = (\Sigma_{ERInt} \cup \Sigma_{EREx} \cup \Sigma_{ERBuff})$  tal que:
  - 1)  $\Sigma_{ERInt} = \Sigma_{EInt} \cup \Sigma_{RInt}$ ,
  - 2)  $\Sigma_{EREx} = \Sigma_{EEx} \cup \Sigma_{REx} \setminus \Sigma_{EfromtoR}$  y
  - 3)  $\Sigma_{ERBuff} = \Sigma_{EBuff} \cup \Sigma_{RBuff} \cup \Sigma_{EfromtoR}$  donde
  - 4)  $\Sigma_{EfromtoR} = \{\langle p_1, p_2, c \rangle \mid p_1, p_2 \in \mathcal{P}, c \in \mathcal{C} \text{ y } ((p_1 = E \wedge p_2 = R) \vee (p_1 = R \wedge p_2 = E))\}$
- $\delta_{ER} = \{\delta_{ERInt} \cup \delta_{EREx} \cup \delta_{ERBuff}\}$
- $\delta_{ERInt} = Q_{ER} \times \Sigma_{ERInt} \times Q_{ER}$
- $\delta_{EREx} : Q_{ER} \times \{In(e, m), Out(e, m) \mid e \in \Sigma_{EREx} \wedge m \in \mathcal{M}\} \times Q_{ER}$  es la relación de transición de comunicación externa de  $E || R_{\mathcal{P}}$ ,
- $\delta_{ERBuff} : Q_{ER} \times \Sigma_{ERBuff} \times Q$  y decimos que  $\forall q \in Q_{ER}, m \in \mathcal{M} \mid \langle q, \omega_{ERi} \gg m, q' \rangle \in \delta_{ERBuff} \iff \exists$  una configuración  $\langle q, [\omega_{ER1}, \dots, \omega_{ERi} : m, \dots, \omega_{ERn}] \rangle$
- $q_0 = \langle q_{0E}, q_{0R} \rangle$
- $F_{ER} = F_E \times F_R$

Explicación de la intuición de definición

Cada autómata es un sistema independiente que cumple una función (o una serie de funciones), y se relaciona con otros a través del envío de mensajes. Decimos que si dos autómatas tienen una acción con la misma etiqueta, al componerlos, ambas transiciones se ejecutarían a la par. Como estos autómatas son de comunicación asíncrona, queremos evitar que dos transiciones se sincronizen de ese modo. Para asegurarnos esto pedimos que todo par de autómatas a componerse tengan conjuntos de etiquetas de acciones que sean disjuntos. Formalmente que dados E y R,  $\Sigma_E \cap \Sigma_R = \emptyset$ .

Del mismo modo, cada autómata tiene su propio conjunto de buffers que pedimos sean disjuntos, para distinguir la comunicación interna de cada autómata componente de la que ocurra entre componentes o con participantes externos. Para modelar la comunicación interna entre componentes, agregamos dos buffers, uno para cada sentido de la comunicación, de E a R y viceversa.

Como los conjuntos de acciones son disjuntos podemos decir que las acciones internas, de comunicación externa y de buffer, de cada componente se preservan, siempre y cuando tenga sentido con la composición de estados. Existe un caso particular que ocurre cuando existían envíos de mensaje de un autómata componente a otro. En ese caso dado que ambos ahora son parte un mismo autóamta, la comunicación pasa a ser envío de mensajes interno. Para representar este tipo de comunicación es que utilizamos buffers. De este modo el intercambio que antes era  $\text{In}((E,R,c),m)$  y  $\text{Out}((E,R,c),m)$  ahora es  $b_{ER} \ll m$  y  $b_{ER} \gg m$ , donde  $b_{ER} \in \{\mathcal{C}_E \cap \mathcal{C}_R\}$  son los buffers exclusivos del autómata compuesto.

Al componer dos autómatas la comunicación que antes era externa y ahora es de buffer puede generar problemas. Puntualmente pueden aparecer transiciones de consumo de un buffer (que antes eran envío de mensajes) donde antes no había. De este modo pueden aparecer secuencias de estados y transiciones donde se consume un mensaje antes de que este sea depositado en el buffer correspondiente. Para esto pedimos que  $\delta_{ER\text{Buff}}$  cumpla con una condición especial. Solo pueden haber transiciones de consumo saliendo de un estado si en alguna secuencia de acciones que termina en ese estado, hay transiciones de producción (es decir se encola un mensaje en el buffer).

La composición es conmutativa y se puede generalizar a una cantidad arbitraria (finita) de autómatas.

La Figura 3.2 muxestra un ejemplo de una composición de dos autómatas.

Dibujo

Figura 3.2: Ejemplo de composición autómata asíncrono de comunicación

**Definición 24** (Determinismo). *Decimos que un autómata es determinístico cuando cumple que no hay dos transiciones con la misma etiqueta que partan de un mismo estado y vayan a estados distintos. Es decir*

*Sea un autómata  $\Lambda = \langle Q, \Sigma, \delta, q_0, F \rangle$  se cumple*

$$\forall q_i, q_j, q_k \in Q_{j \neq k}, \nexists \delta_i, \delta_j \in \delta, t \in \Sigma \parallel \delta_1 = \langle q_i, t, q_j \rangle, \delta_2 = \langle q_i, t, q_k \rangle$$

*Decimos que la composición de estos autómatas preserva el determinismo. Esto es un resultado directo de que ambos autómatas no comparten acciones y de la definición de la composición de  $\delta$ .*

### 3.3. MULTICHANNEL CFSM

Al principio de esta sección introdujimos los AFCA para modelar el comportamiento deseado en SOC de binding incremental. Ahora nos queremos enfocar en los aspectos puramente

comunicacionales, es decir la interacción entre distintos participantes. Para hacer énfasis en estos aspectos existen las CFSMs, pero los AFCA introducen con la composición la posibilidad de que haya múltiples canales de comunicación entre dos participantes. En esta sección atendemos dicha problemática introduciendo las Multichannel Communicating Finite State Machines. Se definen, del mismo modo que una CFSM sobre  $\mathcal{M}$  de la siguiente manera.

**Definición 25** (mCFSM). *Una multichannel communicating finite state machine (mCFSM) sobre  $\mathcal{M}$  es un sistema finito de transición  $(Q, \mathcal{C}, q_0, \mathcal{M}, \delta)$  donde*

- $Q$  es un conjunto finito de estados
- $\mathcal{C} = \{pq_n \mid pq \in \mathcal{P}^2, n \in \mathbb{N}, p \neq q\}$  es un conjunto de canales
- $q_0 \in Q$  es el estado inicial;
- $\delta \subseteq Q \times (\mathcal{C} \times \{!, ?\} \times \mathcal{M}) \times Q$  es un conjunto finito de transiciones.

Un communicating system es un mapa  $S$  que asigna un mCFSM  $S(p)$  a cada  $p \in \mathcal{P}$ . Escribimos  $q \in S(p)$  cuando  $q$  es un estado de la máquina  $S(p)$  y  $\tau$  es una transición de  $S(p)$ .

Al igual que antes la semántica de un communicating system se obtiene considerando configuraciones. Estas configuraciones son iguales a las de las CFSM puras con la salvedad que ahora los canales no están restringidos a un único par entre cada par de participantes.

**Definición 26** (Semántica de una mCSFSM). *La configuración de un multichannel communicating system se define en términos de transiciones entre configuraciones como se ve a continuación:*

La configuración de un communicating system  $S$  es un par  $s = (\vec{q}, \vec{w})$  donde  $\vec{q} = (q_p)_{p \in \mathcal{P}}$  donde  $q_p \in S(p)$  para cada  $p \in \mathcal{P}$  y  $\vec{w} = (w_{pq})_{pq \in \mathcal{C}}$  con  $w_{pq} \in \mathcal{M}$ .

Una configuración  $s' = (\vec{q}', \vec{w}')$  es alcanzable desde otra configuración  $s = (\vec{q}, \vec{w})$  a través de la ejecución de la transición  $\tau$  (escrito  $s \xrightarrow{\tau} s'$ ) si existe un  $m \in \mathcal{M}$  tal que ocurre una de las siguientes alternativas:

1.  $t = (q_p, pq_n!m, q'_p) \in \delta_p$  y
  - a)  $q'_{p'} = q_{p'}$  for all  $p' \neq p$ ; y
  - b)  $w'_{pq_n} = w_{pq_n} \cdot m$  and  $w'_{p'q'_m} = w_{p'q'_m}$  for all  $p'q'_m \neq pq_n$ ;

O bien

2.  $t = (q_q, pq?m, q'_q) \in \delta_q$  y
  - a)  $q'_{p'} = q_{p'}$  for all  $p' \neq q$ ; y
  - b)  $m \cdot w'_{pq_n} = w_{pq_n}$  and  $w'_{p'q'_m} = w_{p'q'_m}$  for all  $p'q'_m \neq pq_n$

Multichannel CFSMs con un único canal para cada par ordenado de participantes son equivalentes a CFSMs puras.

### 3.4. ASPECTOS COMUNICACIONALES DE LOS AFCA

Una de las desventajas principales de utilizar CFSM como lenguaje de especificación de contratos en Service Oriented Computing es que la naturaleza del binding en SOC choca con el método de establecer si un conjunto de participantes pueden interactuar libres de errores de comunicación. En SOC el descubrimiento y binding de servicios se realiza por demanda, eso significa que a pesar de que se necesiten muchos servicios para resolver una tarea, se obtienen uno por uno según se van necesitando. Por el otro lado las CFSM requieren el conjunto total de participantes de un protocolo para poder determinar si el protocolo puede llevarse a cabo sin errores. La idea de los ACFSAs es intentar reducir la distancia entre ambas naturalezas. Por lo tanto una de las tareas principales que queremos lograr con ACFSAs es obtener, vía una proyección, la interfaz de comunicación en la forma de una mCFSM.

**Definición 27** (Interfaz de comunicación de un ACFSa). *Es la mCFSM que resulta de aplicarle el siguiente procedimiento a un aCFSa.*

1. *Transformamos toda acción interna y de buffer en una transición  $\epsilon$ . Esto resulta en una autómatas no determinístico con transiciones  $\epsilon$*
2. *Transformamos el autómatas no determinístico rsultante del paso anterior en uno determinístico.*

### 3.5. GMC PARA MCFSMS

En la sección 2.3 explicamos como funciona la condición de Generalized Multiparty Compatibility para CFSMs. Necesitamos ver que la misma condición es aplicable a las Multichannel CFSMs, para esto la solución más práctica que se encontró es ver que podemos emular una Multichannel CFSM con una CFSM pura. Mostrando esa equivalencia vemos que si podemos aplicar GMC al emulador, la condición aplica al emulado. Dado que un multichannel communicating system es libre de errores de comunicación si y solo si el sistema emulado también lo es. El procedimiento consiste en generar un nuevo participante para cada canal entre dos otros participantes. Este nuevo participante es un simple repetidor de mensajes de uno de los participantes a otro. De este modo una comunicación multichannel entre un par de participantes es reemplazada por múltiples comunicaciones de un canal con un repetidor en el medio. Una aplicación de este procedimiento se ve en la Figura X. El aspecto clave de esta emulación es que preserva el orden de los mensajes.

**Definición 28** (Sistema emulado). *Dado un multichannel communicating system  $(M_p)_{p \in \mathcal{P}}$ , agrandamos el conjunto  $\mathcal{P}$  agregando un participante adicional por cada canal en el sistema original  $\mathcal{P}' = \mathcal{P} \cup \bigcup_{p \in \mathcal{P}} \{p^{pq_n} \mid pq_n \in \mathcal{C}_p\} \cup \bigcup_{p \in \mathcal{P}} \{p^{qp_n} \mid qp_n \in \mathcal{C}_p\}$ . Cada participante nuevo  $p^{sr_n} \in \bigcup_{p \in \mathcal{P}} \{p^{pq_n} \mid pq_n \in \mathcal{C}_p\} \cup \bigcup_{p \in \mathcal{P}} \{p^{qp_n} \mid qp_n \in \mathcal{C}_p\}$  se define con la siguiente mCFSM:*

- $Q_{p^{sr_n}} = \{q_0\} \cup \bigcup_{m \in \mathcal{M}} \{q_m\}$
- $\mathcal{C}_{p^{sr_n}} = \{sp_n^{sr_n}, p^{sr_n}r_n, p^{sr_n}s_n, rp_n^{sr_n}\}$
- $q_{0_{p^{sr_n}}} = q_0$

$$\delta_{p^{sr_n}} = \bigcup_{m \in \mathcal{M}} \{(q_0, sp_n^{sr_n}?m, q_m), (q_m, p^{sr_n}r_n!m, q_0)\}$$

Cada viejo participante  $q \in \mathcal{P}$  se reemplaza por un nuevo participante  $q'$  donde:

$$\begin{aligned} \mathcal{C}_{q'} &= \{qp_n^{qr_n} \mid qr_n \in \mathcal{C}_q\} \cup \{p^{sq_n}q_n \mid sq_n \in \mathcal{C}_q\} \\ \delta_{q'} &= \bigcup_{m \in \mathcal{M}} \{(q, qp_n^{qr_n}!m, q') \mid (q, qr_n!m, q') \in \delta_q\} \cup \{(q, p^{sq_n}q_n?m, q') \mid (q, sq_n?m, q') \in \delta_q\} \end{aligned}$$

Queda claro que si bien transformamos cada canal (buffer) en dos canales nuevos, el orden de los mensajes está garantizado como resultado del modo en que  $\delta$  está definido para los repetidores. Nótese que los repetidores respetan ese orden porque al consumir un mensaje desde el canal de "entrada" (el canal que sirve para recibir mensajes del emisor original) la única acción posible del repetidor es mandar el mensaje a través del canal de salida (el canal que sirve para enviar mensajes al receptor original). Como todo canal es FIFO y los repetidores se comportan del mismo modo, el orden entre dentro de cada canal queda garantizado. Por otro lado, no hay garantía respecto al orden de la comunicación entre canales del modelo original. Por lo tanto podemos decir que los repetidores no introducen más concurrencia que la que estaba en el modelo original y debido a eso los dos sistemas, a pesar de no ser bisimilares y no tener las mismas trazas, son equivalentes respecto a la ausencia de deadlock, recepciones no especificadas y mensajes huérfanos. Esta observación es importante porque provee una forma de chequear multichannel communication systems recurriendo nuevamente al chequeo de GMC del sistema emulado.

**Proposición 1.** *El sistema emulado preserva los errores comunicacionales*

*Demostración.* Primero notemos que los repetidores no pueden generar errores de comunicación dado que en el estado inicial pueden recibir el rango completo de mensajes y luego de consumir un mensaje su única acción posible es reenviarlo. Recordemos también que el orden de los mensajes se preserva. Esto significa que si en el sistema original los mensajes  $m$  y  $m'$  fueron enviados en ese orden sobre el canal  $sr$  en el sistema emulado se envían en el mismo orden sobre el canal  $sp^{sr}$  y por lo tanto son enviados en ese mismo orden a través del canal  $p^{sr}r$ . Entonces se puede mostrar que para cada error de configuración alcanzable en el sistema original, hay una configuración que presenta el mismo error que es alcanzable en el sistema emulado y viceversa.

- $\Rightarrow$  Consideremos cualquier configuración de error  $e$  alcanzable en el sistema original y consideremos cualquier camino que alcance  $e$ . Luego reemplacemos toda acción  $sr!m$  con la secuencia de acciones  $sp^{sr}!m, sp^{sr}?m, p^{sr}r!m$  y cada acción  $sr?m$  con  $p^{sr}?m$ , este nuevo camino está presente en el sistema emulado y alcanza el estado donde las configuraciones de buffer y transiciones permitidas son las mismas para el conjunto de máquinas compartidas (es decir, todas las máquinas de sistema emulado menos los repetidores) y los repetidores están en su estado inicial con buffers vacíos.
- $\Leftarrow$  Consideremos cualquier error de configuración  $e$  alcanzable en el sistema emulado y consideremos cualquier camino que alcance  $e$ . Ya establecimos que el error no puede ser culpa de los repetidores dado que siempre pueden progresar. Por lo tanto sin importar el estado de los buffers en  $e$  hay una configuración  $e'$  que presenta el mismo

error de comunicación y donde el estado de los buffers de los repetidores está vacío y están en su estado inicial. Luego consideremos que cualquier camino que alcance  $e'$  está formado por secuencias de la forma  $sp^{sr!}m, \dots, sp^{sr?}m, \dots, p^{sr}r!m$  y  $p^{sr}r?m$  (donde los puntos suspensivos denota la posibilidad de interleaving de otras acciones). Entonces alcanza con reemplazar cada secuencia  $sp^{sr!}m, \dots, sp^{sr?}m, \dots, p^{sr}r!m$  con la secuencia  $\dots_0, \dots_1, sr!m$  y cada  $p^{sr}r?m$  con  $sr?m$  para obtener un camino que existe en el sistema original y alcanza una configuración que tiene sus buffers en el mismo estado con las mismas transiciones permitidas.

□

## 4. TRABAJO FUTURO

**Definición 29.** Sea  $\mathcal{P}$  un conjunto de participantes,  $\mathcal{C}$  un conjunto de canales de comunicación unidireccionales y  $\mathcal{M}$  un conjunto de mensajes. Un autómata es una estructura  $A_{\mathcal{P}} = \langle Q, B, \Sigma, \delta, q_0, F \rangle$  tal que:

- $Q$  es un conjunto finito de estados,
- $B$  es el conjunto de nombres de buffers,
- $\Sigma = \{\Sigma_{Int} \cup \Sigma_{Ex} \cup \Sigma_{Buff}\}$  es el conjunto de etiquetas del autómata, siendo:
  - 1)  $\Sigma_{Int}$  las acciones internas del autómata
  - 2)  $\Sigma_{Ex}$  un conjunto de etiquetas de la forma  $\langle p_1, p_2, c \rangle$  donde  $p_1, p_2 \in \mathcal{P}$  son, respectivamente, el emisor y el receptor de la comunicación y  $c \in \mathcal{C}$  es el canal a través del cual se resuelve la misma.
  - 3)  $\Sigma_{Buff}$  es el conjunto de etiquetas de las acciones sobre los buffers de la forma  $b \ll m$  o  $b \gg m$ , donde  $b \in B$  y  $m \in \mathcal{M}$ .
- $\delta = (\delta_{Int} \cup \delta_{Ex} \cup \delta_{Buff})$  siendo:
  - 1)  $\delta_{Int} \subseteq Q \times \Sigma_{Int} \times Q$  es la relación de transición por acciones internas de  $A_{\mathcal{P}}$ ,
  - 2)  $\delta_{Ex} \subseteq Q \times \{In(c, m), Out(c, m) | c \in \Sigma_{Ex} \wedge m \in \mathcal{M}\} \times Q$  es la relación de transición de comunicación externa de  $A_{\mathcal{P}}$ ,
  - 3)  $\delta_{Buff} \subseteq Q \times \Sigma_{Buff} \times Q$
- $q_0 \in Q$  es el estado inicial, y
- $F \subseteq Partes(Q)$  es el conjunto de conjuntos de estados finales.

Se denota  $P(A)$ , al conjunto de participantes que integran el autómata  $A$ , y se define como  $P(A) = \{p \in \mathcal{P} | (\exists \langle p_1, p_2, c \rangle \in \Sigma_{Ex})(p_1 = p \vee p_2 = p)\}$

Los autómatas de asincrónicos reactivos se comportan como los autómatas asincrónicos de comunicación en casi todo sentido. La diferencia principal es que con estos queremos representar el comportamiento de sistemas que no tienen una ejecución finita. Para esto tomamos comportamiento de los autómatas de Muller que nos genera una condición de aceptación doble. En este caso el conjunto de estados finales es un conjunto de conjuntos y decimos que el sistema terminó su ejecución cuando pasa al menos una cantidad infinita de veces por alguno de los conjuntos que componen a  $F$ . Para asegurarnos de que la ejecución sea correcta, pedimos también que en ese ciclo infinito los buffers estén vacíos.

## 5. CONCLUSIONES

## 6. BIBLIOGRAFÍA

### REFERENCIAS

- [1] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323-342, 1983.
- [2] Julien Lange, Emilio Tuosto, Nobuko Yoshida. From Communicating Machines to Graphical Choreographies. In S. K. Rajamani and D. Walker, editors, *Proceedings of 42rd. Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '15*, pages 221–232, New York, NY, USA, 2015. ACM.