1. Definiciones preliminares

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

Definición 1 (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 $\rightarrow$.*

Definición 2 (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)*

Proposición 1. *La clausura reflexo-transitiva de una relación binaria es una relación de equivalencia*

1.1. Comunicación Asincrónica

Introducción a la comunicación asincrónica. CFSM
El concepto de *Communicating Finite State Machines* (llamado a partir de ahora CFSM) fue introducido en [1] con el objetivo modelar y estudiar el comportamiento de sistemas distribuidos constituídos 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.

Definición 3 (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 CFSM sobre $\mathcal{M}$ como un sistema de transición finito $\langle Q, C, q_0, \mathcal{M}, \delta \rangle$ donde*

- *Q es un conjunto finito de estados;*
- *$C = \{pq \in \mathcal{P}^2 \mid p \neq q\}$ 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 4 (Communicating System)**.** *Dada una CFSM* $M_p = \langle Q_p, q_{0_p}, \mathcal{M}, \delta_p \rangle$ *para cada participante* $p \in \mathcal{P}$*, la tupla* $S = \langle M_p \rangle_{p \in \mathcal{P}}$ *es un communicating system CS. Una configuración de* S *es un par* $s = \langle \overrightarrow{q}; \overrightarrow{w} \rangle$ *donde* $\overrightarrow{q} = (q_p)_{p \in \mathcal{P}}$ *con* $q_p \in Q_p$ *y donde* $\overrightarrow{w} = (w_{pq})_{pq \in C}$ *con* $w_{pq} \in \mathcal{M}^*$*. La componente* $\overrightarrow{q}$ *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* $s_0 = \langle \overrightarrow{q_0}; \overrightarrow{\epsilon} \rangle$ *con* $\overrightarrow{q_0} = (q_{0_p})_{p \in \mathcal{P}}$ *y* $\overrightarrow{\epsilon} = (\epsilon_p)_{p \in \mathcal{P}}$

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 5 (Estados y configuraciones alcanzables)**.** *Una configuración* $c' = \langle \overrightarrow{q'}; \overrightarrow{w'} \rangle$ *es* alcanzable *desde otra configuración* $c = \langle \overrightarrow{q}; \overrightarrow{w} \rangle$ *a través de la* ejecución de la transición l *(escrito* $s \xrightarrow{l} s'$*) si existe un* $m \in \mathcal{M}$ *tal que ocurre una de las siguientes alternativas:*

1. $l = sr!m, \langle q_s, l, q'_s \rangle \in \delta_s$ *y*
   a) $q'_p = q_p$ *para todo* $\mathsf{p} \neq s$*; y*
   b) $w'_{sr} = w_{sr} \cdot m$ *y* $w'_{pq} = w_{pq}$ *para todo* $pq \neq sr$*; o bien*
2. $l = sr?m, \langle q_r, l, q'_r \rangle \in \delta_r$ *y*
   a) $q'_p = q_p$ *para todo* $p \neq r$*; y*
   b) $w_{sr} = m \cdot w'_{pq}$ *y* $w'_{pq} = w_{pq}$ *para todo* $pq' \neq sr$

Escribimos $s_1 \xrightarrow{t1 \ldots tm} s_{m+1}$ *cuando para algún* $s_2, ..., s_m, s_1 \xrightarrow{t} s_2 ... s_m \xrightarrow{t_m} s_{m+1}$ *. 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 6 (Communicating System seguro)**.** *Sea* S *un CS; se dice que* S *es* seguro *si para cada* $s \in RS(S)$*:*

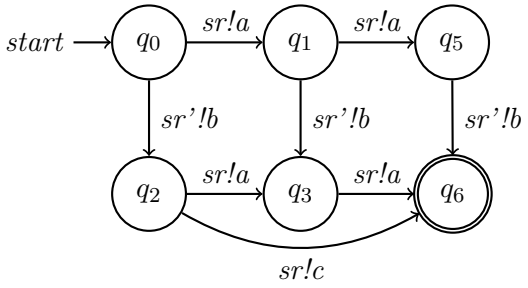
1. *s no es una configuración de deadlock (cuando la única transición de salida necesita consumir un mensaje que no está en el buffer)*
2. *s no posee mensajes huérfanos (cuando no hay transiciones salientes y quedaron mensajes en el buffer) , y*
3. *s no posee recepciones no especificadas (cuanndo la única transición saliente consume un mensaje disinto al primero de la cola).*

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* $\diamondsuit, \blacklozenge \subseteq \delta \times \delta$ *como las relaciones de equivalencia más pequeñas que contienen* $\underline{\diamondsuit}$ *y* $\underline{\blacklozenge}$ *donde:*

- $(q_1, \ell, q_2) \underline{\Diamond} (q_1', \ell, q_2')$ *sii* $l \notin act(q_1, q_1') \wedge act(q_1, q_1') = act(q_2, q_2') \wedge act(q_2, q_2') \neq \varnothing$
- $(q_1, \ell, q_2) \underline{\blacklozenge} (q_1', \ell, q_2')$ *sii* $(q_1, \ell, q_2) \underline{\Diamond} (q_1', \ell, q_2')$ *y* $\forall (q, \ell, q') \in [(q_1, \ell, q_2)]^{\Diamond}$, $act(q_1, q) = act(q_2, q') \wedge act(q_1', q) = act(q_2', q')$

donde $[(q_1, \ell, q_2)]^{\Diamond}$ *es la clase de equivalencia de* (q, ℓ, 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 (Relaciones $\underline{\Diamond}$ y $\underline{\blacklozenge}$)**.** *Consideremos la siguiente CFSM:*



1. $(q_0, sr!a, q1) \underline{\Diamond} (q_2, sr!a, q3)$
2. $(q_0, sr!a, q1) \underline{\blacklozenge} (q_2, sr!a, q3)$
3. *No vale* $((q_0, sr!a, q1) \underline{\Diamond} (q_1, sr'!b, q5))$
4. $(q_0, sr'!b, q2) \underline{\Diamond} (q_1, sr'!b, q3)$
5. *No vale* $((q_0, sr'!b, q2) \underline{\blacklozenge} (q_1, sr'!b, q_3))$

Las relaciones en Ej. 1.1– 1.2 se sostienen dado que ambas transiciones están entrelazadas con $sr'!b$*. La relacion en Ej. 1.3 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 Ej. 1.3 son secuenciales, no concurrentes. La relación en Ej. 1.4 se sostiene, pero en Ej. 1.5 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 7 (Eventos)**.** *Dados un conjunto de participantes* $\mathcal{P}$ *y un conjunto de mensajes* $\mathcal{M}$ *definimos un evento e como es una tupla* $\langle q_s, q_r, s, r, a \rangle$ *(también escrita como* $\langle q_s, q_r, s \to r : a \rangle$*), tal que* $s, r \in \mathcal{P}$*, indicando que* s *y* r *pueden intercambiar el mensaje* a*, cuando están en el estado* q_s *y* q_r*, respectivamente.*

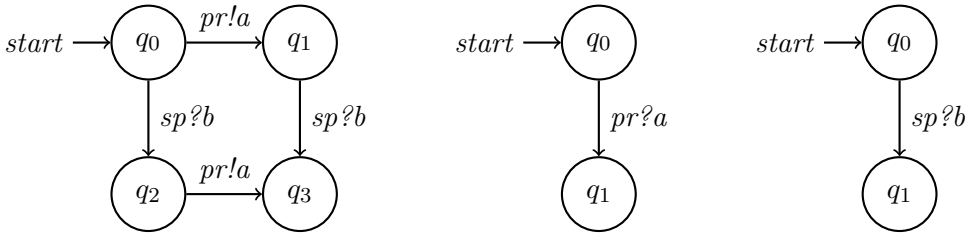
REvisar

Tomando en consideración la definición anterior, para distinguir el paralelsimo a nivel máquina introducimos una relación de equivalencia sobre eventos que identifica eventos cuyas transiciones son $\blacklozenge$-equivalentes.

Definición 8 (Equivalencia entre eventos)**.** *Definimos la equivalencia entre eventos como* $\bowtie = \bowtie_s \cap \bowtie_r \subseteq \mathcal{E} \times \mathcal{E}$ *donde* $\mathcal{E}$ *es el conjunto de eventos del sistema y se cumplen las siguientes condiciones:*

- $(q_1, q_2, s \to r : a) \bowtie_s (q_1', q_2', s \to r : a) \iff$
  $\forall (q_1, sr!a, q_3), (q_1', sr!a, q_3') \in \delta_s : (q_1, sr!a, q_3) \underline{\blacklozenge} (q_1', SR!a, q_3')$
- $(q_1, q_2, s \to r : a) \bowtie_r (q_1', q_2', s \to r : a) \iff$
  $\forall (q_2, sr?a, q_3), (q_2', sr?a, q_4') \in \delta_r : (q_2, sr?a, q_4) \underline{\blacklozenge} (q_2', sr?a, q_4')$

Example 2 (Equivalencia entre eventos)**.** *Considere el siguiente Communicating System*



En el sistema de arriba podemos ver los siguientes eventos $(q_{0p}, q_{0r}, p \to r : a)$, $(q_{2p}, q_{0r}, p \to r : a)$, $(q_{0s}, q_{0p}, s \to p :)$ *y* $(q_{0s}, q_{3p}, s \to p : b)$. *Queremos ver si se cumple que* $(q_{0p}, q_{0r}, p \to r : a) \bowtie (q_{2p}, q_{0r}, p \to r : a)$, *es decir que son eqivalentes tanto bajo* $\bowtie_p$ *como en* $\bowtie_r$.
Para el primero queremos ver que $(q_{0p}, pr!a, q_{1p}) \underline{\blacklozenge} (q_{2p}, pr!a, q_{3p})$ *Para esto necesitamos que valga* $(q_{0p}, pr!a, q_{1p}) \underline{\Diamond} (q_{2p}, pr!a, q_{3p})$. *Esto se cumple dado que* $pr!a \notin act(q_{0p}, q_{2p})$ *y* $act(q_{0p}, q_{2p}) = act(q_{1p}, q_{3p}) \neq \varnothing$. *Ahora tenemos que ver la clase de equivalencia* $[(q_{0p}, pr!a, q_{1p})]^{\Diamond}$. *La misma es el conjunto unitario* $\{(q_{2p}, pr!a, q_{3p})\}$. *Por último queremos ver que* $act(q_{0p}, q_{2p}) = act(q_{1p}, q_{3p}) \wedge act(q_{2p}, q_{2p}) = act(q_{3p}, q_{3p})$. *La segunda es trivial, la primera se cumple siendo el conjunto unitario* $\{sp?b\}$. *Con esto demostramos que* $(q_{0p}, q_{0r}, p \to r : a) \bowtie_p (q_{2p}, q_{0r}, p \to r : a)$
Nos queda probar que vale $(q_{0p}, q_{0r}, p \to r : a) \bowtie_r (q_{2p}, q_{0r}, p \to r : a)$. *Esta equivalencia es más sencilla de probar, dado que tenemos una única transición. Entonces vemos que es trivial que* $(q_{0r}, pr?a, q_{1r}) \underline{\blacklozenge} (q_{0r}, pr?a, q_{1r})$. *Con esto vemos que* $(q_{0p}, q_{0r}, p \to r : a)$ *y* $(q_{2p}, q_{0r}, p \to r : a)$ *son equivalentes en* $\bowtie_p$ *y* $\bowtie_r$, *por lo tanto están en* $\bowtie = \bowtie_p \cap \bowtie_r$.

A continuación definimos la noción de Sistema de Transición Sincrónico, para reflejar el comportamiento de un sistema cuando envíos y recepciones son emparejados para mostrar que ocurren al mismo tiempo.

Definición 9 (Sistema de transición sincrónico)**.** *Dados un* $S = (M_P)_{p\in\mathcal{P}}$ *un CS, sea* $\langle N, \hat{\delta}, E\rangle$*, donde:*

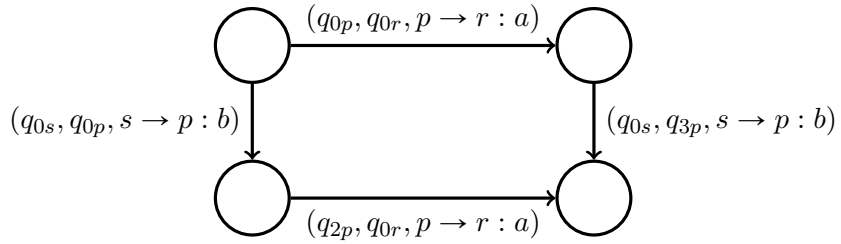
$$N = \{\overrightarrow{q} \mid (\overrightarrow{q}; \overrightarrow{\epsilon}) \in RS_1(S)\},$$

$$\hat{\delta} = \{(n, e, n') \mid (n; \overrightarrow{\epsilon})s_1 \xrightarrow{sr!a\ sr?a} (n'; \overrightarrow{\epsilon}) \wedge e = n[s], n[r], s \to r : a\},\ y$$

$$E = \{\exists n, n' \in N : (n, e, n') \in \hat{\delta}\} \subseteq \mathcal{E},$$

el Sistema de Transición Sincrónico *de* S *es* $TS(S) = \langle N, n_0, E/\bowtie, \Rightarrow\rangle$ *donde* $n_0 = \overrightarrow{q_0}$ *es el estado inicial,* $n \overset{[e]}{\Rightarrow} n' \iff (n, e, n') \in \hat{\delta}$*. Fijamos un conjunto* $\hat{E}$ *de elementos representativos de cada clase de equivalencia* $\bowtie$ *(ej:* $\hat{E} \subseteq E$ *y* $(\forall e \in E)\left(\exists! e' \in \hat{E}\right)(e' \in [e])$*) y escribimos* $n \overset{e'}{\Rightarrow} n'$ *para* $n \overset{[e]}{\Rightarrow} n'$ *cuando* $e' \in [e] \cap \hat{E}$*. Las secuencias de eventos se notan con un símbolo* π *y extendemos la notación de* $\to$ *en la Def. 24 a* $\Rightarrow$ *(ej:* $si\pi = e_1...e_k, n_1 \overset{\pi}{\Rightarrow} n_{k+1} siin_1 \overset{e_1}{\Rightarrow} n_2 \overset{e_2}{\Rightarrow} ... \overset{e_k}{\Rightarrow} n_{k+1}$*).*
$TS(S)$ *representa todas las posibles ejecuciones sincrónicas del sistema* S*; y cada transición es etiquetada con un evento* e*.*

Example 3 (Sistema de Transición Sincrónico)**.** *Consideremos el Ej. 2, su Sistema de Transición Sincrónico es el siguiente*



Tenemos $(q_{0p}, q_{0r}, p \to r : a) \bowtie (q_{2p}, q_{0r}, p \to r : a)$ *y* $(q_{0s}, q_{0p}, s \to p :) \bowtie (q_{0s}, q_{3p}, s \to p : b)$*. Podemos considerar equivalentes los eventos de las transiciones verticales por un lado y las de las transiciones horizontales por el otro. Esto nos permite identificar un par de interacciones concurrentes, pero seguir diferenciándolas de otras instancias de comunicación* $p \to r : a$ *y* $s \to p : b$

Definición 10 (Proyecciones)**.** *La proyección de un evento* e *sobre un participante* p*, denotado por* $e \upharpoonright_p$ *se define de la siguiente manera:*

$$(q_s, q_r, s \to r : a) \upharpoonright_p = \begin{cases} pr!a & if\ s = p \\ sp?a & if\ r = p \\ \epsilon & en\ otro\ caso \end{cases} \tag{1.1}$$

La proyección se define sobre secuencias de eventos en el modo evidente. La proyección $TS(S) = (N, n_0, \hat{E}, \Rightarrow)$ *sobre el participante* p*, notada* $TS(S) \upharpoonright_p$*, es el autómata* (Q, q_0, Σ, δ) *donde* $Q = N,\ q_0 = n_0,\ \Sigma$ *es el conjunto de etiquetas y* $\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times Q$ *es el conjunto de transiciones, tal que* $(n_1, e \upharpoonright_p, n_2) \in \delta \iff n_1 \overset{e}{\Rightarrow} n_2$

Example 4. *Proyecciones*

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 propiedad de representabilidad determina que cada máquina, traza y elección estén representadas en el sistema de transición.

Definición 11. *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 $\mathcal{L}(TS(S)\langle n \rangle) \upharpoonright_p$, 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 12 (Representabilidad)**.** *Un sistema S es representable si*

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

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

La primera condición asegura que cada traza de cada máquina esté en $TS(S)$, a su vez, la segunda condición es necesaria para asegurar que cada elección en cada máquina esté representada en $TS(S)$.

La propiedad de ramificación estimula 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.

Definición 13 (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 $n \stackrel{e_1}{\Rightarrow} n_1$ y $n_1 \stackrel{e_2}{\Rightarrow} n_2$ luego tenemos*

1. *o bien existe $n' \in N$ tal que $n_1 \stackrel{e_2}{\Rightarrow} n'$ y $n_2 \stackrel{e_1}{\Rightarrow} n'$, o*
2. *para cada $(n'_1, n'_2) \in ln(n, e_1, e_2)$ quedando*
   $L^i_p = hd(\{e_i \upharpoonright_p \cdot \phi \mid \phi \in LT(S, n'_i, p)\})$ 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. $L^1_p \cap L^2_p \subseteq \{\epsilon\}$ *y* $\epsilon \in L^1_p \iff \epsilon \in L^2_p$, *o*

II. $\exists n' \in N, \pi_1, \pi_2$: $n'_1 \stackrel{\pi_1}{\Rightarrow} n' \wedge n'_2 \stackrel{\pi_2}{\Rightarrow} n' \wedge (e_1 \cdot \pi_1) \downarrow_p = (e_2 \cdot \pi_2) \downarrow_p = \epsilon$

(b) selector único: $\exists! s \in \mathcal{P} : L^1_s \cap L^2_s = \varnothing \wedge \exists s r! a \in L^1_s \cup L^2_s$

(c) no race: $\forall r \in \mathcal{P} : L^1_r \cap L^2_r = \varnothing \Rightarrow \forall s_1 r?a_1 \in L^1_r, \forall s_2 r?a_1 \in L^2_r : \forall i \neq j \in \{1,2\} : n'_i \stackrel{\pi_i}{\Rightarrow}$
$\Rightarrow dep(s_i \to r : a_i, e_i \cdot \pi_i, s_j \to r : a_j)$

Representabilidad garantiza que TS(S) contiene suficiente información para decdir 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".

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 $\lhd$-relación entre todas sus acciones.

Definición 14 (General Multiparty Compatibilty)**.** *Un sistmea S es generalised multiparty compatible (GMC) si es representable y posee la propiedad de ramifiación.*

2. Compponentes Asincrónicas

2.1. Autómatas finitos de comunicación asincrónica

Definición 15 (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* $b = [m_1, .., m_n]$ *al aplicarle la operación* $b \ll m$ *queda* $b = [m, m_1, .., m_n]$ *del mismo modo al aplicarle* $b \gg m_n$ *queda* $b = [m, m_1, ..]$

Definición 16 (Autómata finitos de comunicación asincrónica)**.** *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. 15),*
- *$\mathcal{C} \subseteq \{pq_n \mid pq \in \mathcal{P}^2, n \in \mathbb{N}, p \neq q\}$ es un conjunto de canales tales que $B \cap \mathcal{C} = \varnothing$*
- *$\Sigma = \{\Sigma_{Int} \cup \Sigma_{Ex} \cup \Sigma_{Buff}\}$, $\Sigma \cap \mathcal{M} = \varnothing$ es el conjunto de etiquetas del autómata, siendo*

  1) Σ_{Int} las acciones internas del autómata

  2) Σ_{Ex} un conjunto de etiquetas de la forma $\langle p_1, p_2, c \rangle$ dónde $p_1, p_2 \in \mathcal{P}$ 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) Σ_{Buff} es el conjunto de etiquetas de las acciones sobre los buffers de la forma $b \ll m$ o $b \gg m$, dónde $b \in B$ 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 transicion 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 $P(A) = (p \in \mathcal{P} \mid (\exists \langle p1, p2, c \rangle \in \Sigma_{Ex})\, (p1 = p \vee p2 = p))$

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

δ es el conjunto de transciciones que se compone de δ_{Int}, las acciones internas, δ_{Ex} las transiciones de comunicación con otros agentes y δ_{Buf} las acciones de buffer.
δ_{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 comunicaión asincrónica. Como los canales son unidireccionales decimos que para todo par de participantes existen dos canales, uno por cada sentido.
δ_{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 17 (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 $\langle q, \Omega \rangle \in Q \times \{[m_b]_{b \in B} | m_b \in \mathcal{M}^*\}$ donde $\mathcal{M}^* = \{[s_o, \ldots, s_n] \mid \forall i \in [0, \ldots, n], s_i \in \mathcal{M}\}$ y $[m_b]_{b \in B}$

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

Definición 18 (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. $(q_1, \Omega) \vdash (q_2, \Omega) \iff \langle q_1, r, q_2 \rangle \in \delta \wedge r \in \Sigma \backslash \Sigma_{Buff}$
2. $(q_1, [\omega_{b_1}, \ldots, \omega_{b_i}, \ldots, \omega_{b_n}]) \vdash (q_2, [\omega_{b_1}, \ldots, m : \omega_{b_i}, \ldots, b_n]) \iff \langle q_1, b_i \ll m, q_2 \rangle \in \delta$
3. $(q_1, [\omega_{b_1}, \ldots, \omega_{b_i} : m, \ldots, \omega_{b_n}]) \vdash (q_2, [\omega_{b_1}, \ldots, \omega_{b_i}, \ldots, \omega_{b_n}]) \iff \langle q_1, b_i \gg m, q_2 \rangle \in \delta$

Definición 19 (Traza de un AFCA)**.** *Llamamos traza a una secuencia posible de acciones de un autómata. Se define como una secuencia finita de etiquetas de estado y transición altenradas, 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 caracterísitcas*

- *Tiene la forma $[q_0, \sigma_1, q_1, ..., q_{n-1}, \sigma_n, q_n]$ 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 AFCA es el conjunto de todas las trazas posibles.

Definición 20 (Deadlock)**.** *Sean* $A_{\mathcal{P}} = \langle Q, B, \Sigma, \delta, q_0, F \rangle$, $q_i \in Q$, $\sigma \in \Sigma^*$, $\Omega = [\omega_{b_1}, .., \omega_{b_i}, .., \omega_{b_n}]$, *decimos que una configuración está en deadlock cuando ocurre:*

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

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

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

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.

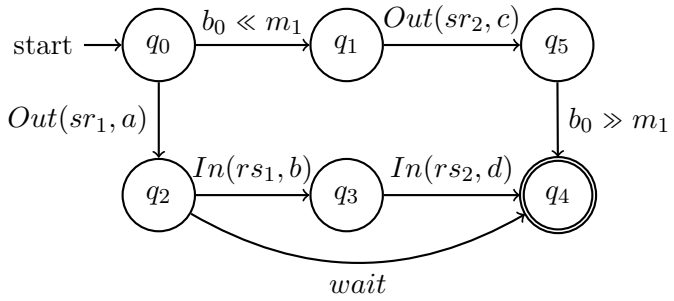
Definición 22 (Receptor no especificado)**.** *Sean* $A_{\mathcal{P}} = \langle Q, B, \Sigma, \delta, q_0, F \rangle$, $q_i \in Q$, $\sigma \in \Sigma^*$, $\Omega = [\omega_{b_1}, .., \omega_{b_i}, .., \omega_{b_n}]$ *decimos que una configuración es de receptor no especificado cuando ocurre:*

- *En la configuración* $\langle q_i, \Omega \rangle$,
- $\exists \omega_b \in \Omega \mid \omega_b \neq [\ ]$
- $(\forall \langle q_i, b \gg m, q_j \rangle \in \delta)\, (\omega_b = [\ldots, m'])$, *con* $m' \neq m$

Decimos que una configuración está en modo Receptor no Especificado si la única transición saliente de un estado q_i consume un mensaje m y el primer mensaje de la cola es un mensaje m' distinto de m.

Definición 23 (Ejecución de un AFCA)**.** . *Una ejecución es un* $\tau = \tau_0, \ldots, \tau_n$ *donde* τ_0 *es la configuración inicial,* $\langle \tau_i, \tau_{i+1} \rangle \in \vdash$, τ_n *es una configuración final y* $\forall i \in [1, \ldots, n], \tau_i$ *es una configuración. Una ejecución es una configuración que no entra en deadlock, mensajes huérfanos ni receptor no especificado.*

La Figura. 3.1 muestra un ejemplo de estos autómatas.



La figura corresponde al AFCA $S = \langle\{q_0, .., q_5\}, [b_0], \{sr_1, sr_2, rs_1\}, \Sigma_A, \delta_A, q_0, \{q_4\}\rangle$. Donde $\Sigma = \{wait, b_0 \ll m_1, b_0 \gg m_1, Out(sr_1, a), Out(sr_2, c), In(rs_1, b), In(rs_2, d)\}$ En el ejemplo podemos ver los tres tipos de transiciones que los AFCA pueden realizar. Transiciones internas de buffer interno como son $b_0 \ll m_1$ y $b_0 \gg m_1$,

Figura 2.1: Ejemplo de autómata asíncrono de comunicación

2.2. 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 interacturar 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 protocol,o para poder determinar si el protocolo puede llevarse a cabo sin errores. La idea de los AFCA es intentar reducir la distancia entre ambas naturalezas. Por lo tanto una de las tareas principales que queremos lograr con AFCAs es obtener, vía una proyección, la inferfaz de comunicación en la forma de una mCFSM.

Definición 24 (mCFSM)**.** *Una multichannel communicating finite state machine (mCFSM) sobre $\mathcal{M}$ es una 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 τ 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 25 (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 $s = (\vec{\mathbf{q}}, \vec{\mathbf{w}})$ donde $\vec{\mathbf{q}} = (q_{\mathsf{p}})_{\mathsf{p}\in\mathcal{P}}$ donde $q_{\mathsf{p}} \in S(p)$ para cada $p \in \mathcal{P}$ y $\vec{\mathbf{w}} = (w_{pq})_{pq\in\mathsf{C}}$ con $w_{pq} \in \mathcal{M}$.
Una configuración $s' = (\vec{\mathbf{q}'}, \vec{\mathbf{w}'})$ es alcanzable *desde otra configuración $s = (\vec{\mathbf{q}}, \vec{\mathbf{w}})$ a través de la* ejecución de la transición *τ (escrito $s \xrightarrow{t} s'$) si existe un $m \in \mathcal{M}$ tal que ocurre una de las siguientes alternativas:*

1. $t = (q_p, pq_n!m, q'_p) \in \delta_p$ *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.

Definición 26 (Interfaz de comunicación de un AFCA)**.** *Es la mCFSM que resulta de aplicarle el siguiente procedimiento a un AFCA.*

1. *Transformamos toda acción interna y de buffer en una transición ϵ. Esto resulta en una autómata no determinístico con transiciones ϵ*

2. *Transformamos el autómata no determinístico rsultante del paso anterior en uno determinístico.*

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 applicación de este procedimiento se ve en la Figura X. El aspecto clave de esta emulación es que preserva el orden de los mensajes.

Definición 27 (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:*

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

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 δ *está definido para los repetidores. Nótese que los repetidores respetan ese orden porque al consumir un mensaje desde el canal de .*^{en}*trada"(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 garantiado. Por oro 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 multichannesl communication systems recurriendo nuevamente al chequeo de GMC del sistema emulado.*

Proposición 2. *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 ocnsumir 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.

⇒ 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.

- ⇐ Consideremos cualquier error de configuración e alcanzable en el sistema emulado y ocnsideremos 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 $sp^{sr}!m, \ldots, sp^{sr}?m, \ldots, p^{sr}r!m$ y $p^{sr}r?m$ (donde los puntos suspensivos denota la posibilidad de interleaving de otras acciones). Entonces alcanza con reemplazar cada secuencia $sp^{sr}!m, \ldots_0, sp^{sr}?m, \ldots_1, p^{sr}r!m$ con la secuencia $\ldots_0, \ldots_1, sr!m$ y cada $p^{sr}r?m$ con $sr?m$ para obtener un camino que existe en el sistma original y aclanza una configuración que tiene sus buffers en el mismo estado con las mismas transiciones permitidas.

□

3. Composición de Componentes Asincrónicas

3.1. Motivación

Como ya hemos mencionado, en SOC (Service-Oriented Computing) los sistemas son concebidos como objetos dinámicos construidos en *run-time* en la medida que su ejecución llega a un estado en el que la intervención de servicios externos se hace necesaria. Es decir un sistema de este tipo utilizará distintos servicios según las necesidades que se manifiesten a lo largo de una ejecución particular.
Una aplicación que se encuentra ejecutando se conecta con los servicios que le son necesarios a través de canales de comunicación por los cuales se envían o reciben mensajes. Estos canales pueden establecer una comunicación entre un número fijo (para cada canal particular) pero no acotado de servicios. En la sección anterior hemos detallado un conjunto de propiedades que garantizan una comunicación sin errores a través de estos canales (i.e. ausencia de deadlock, aunsencia de mensajes huérfanos y ausencia de situaciones en las que el receptor no espera un mensaje que le fue enviado) y un procedimiento para garantizarlas. Estas condiciones y su procedimiento de análisis parten de la hipótesis de que las CFSMs correspondientes a todos y cada uno de los participantes de la comunicación sobre dicho canal se encuentran disponibles.
Ahora bien, que la aplicación arribe a un estado en el que un servicio se hace necesario sobre un canal particular, no implica que todos los participantes también lo sean en ese

mismo instante y por ello, con el objeto de profundizar esta concepción incremental, a demanda, que se tiene sobre los sistemas de software surge, más o menos naturalmente, la idea de poder dotar al *middleware* de la capacidad de realizar un *binding* parcial sobre los canales. A esta práctica la llamaremos *binding incremental*.
Esta percepción parcial del *binding* sobre un canal requiere la utilización de un un lenguaje de descripción que soporten dichos mecanismos de composición. Por ejemplo, al componer dos CFSMs puede ocurrir que cada máquina se comunique a través de un canal con un tercer participante, estos dos canales son independientes y, además, una vez que se ha realizado la composición, deben ser percibidos por este tercer participante como canales de comunicación separados. Por lo tanto, para preservar la semántica de la comunicación, es necesario que el CFSM resultante de la composición tenga dos canales con este tercer participante. Este fenómeno será el eje rector de las modificaciones que introduciremos en esta sección. Esta característica no solo será necesario a nivel de CFSM (interfaz de comunicación de un servicio) sino también de los autómatas que caracterizan el cómputo, cuya interfaz de comunicación es expresada a través de una CFSM.
Para modelar este comportamiento introducimos los Autómatas Finitos de Comunicación Asincrónica (AFCA). Estos autómatas tienen tres tipos de transiciones: internas que representan procesamiento propio; de buffer que representan comunicación asincrónica 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.
Como la idea es que los autómatas con comunicación asincrónica reperesenten 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 pedirlas siguientes condiciones

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

Definición 28 (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 $\Sigma_E \cap \Sigma_R = \varnothing$ y $B_E \cap B_R = \varnothing$, 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.*
- $\mathcal{C}_{ER} = \{\mathcal{C}_E \cup \mathcal{C}_R\} \backslash \{\mathcal{C}_E \cap \mathcal{C}_R\}$

- $\Sigma_{ER} = (\Sigma_{ERInt} \cup \Sigma_{EREx} \cup \Sigma_{ERBuff})$ *tal que:*

  1) $\Sigma_{ERInt} = \Sigma_{EInt} \cup \Sigma_{RInt}$,

  2) $\Sigma_{EREx} = \Sigma_{EEx} \cup \Sigma_{REx} \backslash \Sigma_{EfromtoR}$ *y*

  3) $\Sigma_{ERBuff} = \Sigma_{EBuff} \cup \Sigma_{RBuff} \cup \Sigma_{EfromtoR}$ *donde*

  4) $\Sigma_{EfromtoR} = \{\langle p_1, p_2, c\rangle \mid p_1, p_2 \in \mathcal{P},\ c \in \mathcal{C}$ *y* $((p_1 = E \wedge p_2 = R) \vee (p_1 = R \wedge p_2 = E))\}$

- $\delta_{ER} = \{\delta_{ERInt} \cup \delta_{EREx} \cup \delta_{ERBuf}\}$
- $\delta_{ERInt} = Q_{ER} \times \Sigma_{ERInt} \times Q_{ER}$
- $\delta_{EREx} : Q_{ER} \times \{In(e,m), Out(e,m) | e \in \Sigma_{EREx} \wedge m \in \mathcal{M}\} \times Q_{ER}$ *es la relación de transición de comunicación externa de* $E||R_{\mathcal{P}}$,
- $\delta_{ERBuf} : Q_{ER} \times \Sigma_{ERBuf} \times Q$ *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* $\langle q, [\omega_{ER1}, ..., \omega_{ERi} : m, ..., \omega_{ERn}]\rangle$
- $q_0 = \langle q0_E, q0_R\rangle$
- $F_{ER} = F_E x F_R$

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 transsiciones se ejecutarían a la par. Como estos autómatas son de comunicación asincrónica, queremos evitar que dos transiciones se sicronizen 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, $\Sigma_E \cap \Sigma_R = \varnothing$.
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 In((E,R,c),m) y 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 δ_{ERBuff} 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 4.1 muxestra un ejemplo de una composición de dos autómatas.

Dibujo

Figura 3.1: Ejemplo de composición autómata asíncrono de comunicación

Definición 29 (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*

$$\textit{Sea un autómata } \Lambda = \langle Q, \Sigma, \delta, q_0, F\rangle \textit{ se cumple}$$
$$\forall\ q_i, q_j, q_k \in Q_{j\neq k},\ \nexists\ \delta_i, \delta_j \in \delta, t \in \Sigma \parallel \delta_1 = \langle q_i, t, q_j\rangle,\ \delta_2 = \langle q_i, t, q_k\rangle$$

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 δ.

3.2. Composición parcial vs composición total

En esta sección definimos los Autómatas Finitos de Comunicación Asincrónica para modelar la composición parcial de CFSMs. Ahora necesitamos demostrar que esta composición parcial es equivalente a una composición total.
Como sabemos que todas las CFSMs a componer se encuentran en $\mathcal{P}$ podemos decir que conocemos a priori todos los componentes de la composición final. Dado un conjunto finito $\mathcal{P}$ de CFSMs denominamos $p_1, p_2, p_3, \ldots, p_n$.
Si componemos p_1 y p_2 nos quedaría el conjunto $\mathcal{P}_1 = \{p_{12}\}, p_3, \ldots, p_n\}$, podemos hacer un paso siguiente componiendo p_{n-1} y p_n. De esto obtenemos $\mathcal{P}_{\in 2} = \{p_{12}, p_3, \ldots, p_n - 1\}$.
Podemos continuar este proceso hasta llegar a tener un único autómata.
Decimos que en cada paso de la composición parcial tengo una función suryectiva, pero no inyectiva que garantiza que cada punto del codominio es la composición de al menos dos puntos de la preimagen.
Llamemos TM1 y TM2 a los autómatas resultantes de la composición de todos los elementos de $\mathcal{P}$ por sucesión de composiciones parciales y por composición total respectivamente. Queremos ver que $q \in Q_{TM1} \iff q \in Q_{TM2}$ y en ambos casos está entre los estados alcanzables.
Queremos demostrar que

1. Las configuraciones alcanzables entre tm1 y tm2 son las mismas

2. tm1 es bisimilar a tm2, es decir que para cada configuración las acciones realizables son las mismas

Definición 30 (Bisimulación)**.** *Dado un sistema de transición con etiquetas $S = \langle Q, \Sigma, \delta \rangle$, una bisimulación es una relación binaria $R \subseteq Q \times Q$, tal que tanto R como su transpuesta R^T son simulaciones. Equivalentemente R es una bisimulación si para cada par de elementos $p, q \in Q$ vale $\langle p, q \rangle \in R$ y para todo $\sigma \in \Sigma$ vale:*

- $(\forall p' \in \Sigma \mid p \xrightarrow{\sigma} p' \implies \exists q' \in Q$ *talque* $q \xrightarrow{\sigma} q' \wedge \langle p', q' \rangle \in R)$ *y, simétricamente vale*
- $(\forall q' \in \Sigma \mid q \xrightarrow{\sigma} q' \implies \exists p' \in Q$ *talque* $p \xrightarrow{\sigma} p' \wedge \langle p', q' \rangle \in R)$

Dados dos estados $p, q \in Q$, p es bisimilar a q, se denota $p \sim q$, si existe una bisimulación R tal que $\langle p, q \rangle \in R$. La relación de bisimilaridad $\sim$ es una relación de equivalencia. Además es la relación de bisimulación más grande sobre un sistema dado.

4. Trabajo futuro

Definición 31. *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 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) Σ_{Int} las acciones internas del autómata

  2) Σ_{Ex} un conjunto de etiquetas de la forma $\langle p_1, p_2, c \rangle$ dónde $p_1, p_2 \in \mathcal{P}$ 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) Σ_{Buff} es el conjunto de etiquetas de las acciones sobre los buffers de la forma $b \ll m$ o $b \gg m$, dónde $b \in B$ 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 transicion 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 $P(A) = \{p \in \mathcal{P} \mid (\exists \langle p1, p2, c \rangle \in \Sigma_{Ex})(p1 = p \vee p2 = p)\}$

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 ChoreographiesIn 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.