Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Automata for Partial Binding of Services

Tesis de Licenciatura en Ciencias de la Computación

Ezequiel Davidovich Caballero

Director: Carlos G. Lopez Pombo
Codirector: Ignacio Vissani
Buenos Aires, 2020

AUTÓMATAS PARA EL BINDING PARCIAL DE SERVICIOS

Los sistemas distribuidos resultantes de paradigmas emergentes como *service-oriented computing*(SOC) y Cloud/Fog computing están transformando al mundo del software, impulsando lo que se denomina economía de API. La idea subyacente de economía de API es la posibilidad de construir software componiendo servicios externos y registrados previamente en repositorios. Aplicaciones que corren sobre recursos disponibles a nivel global con una infraestructura de comunicación que se reconfiguran en forma dinámica y transparente, a través de un middleware dedicado capaz de descubrir y conectarlas a servicios que puedan cumplir determinados requisitos.
En general los aspectos principales del comportamiento de una API son documentados informalmente limitando la posibilidad de lograr la utopía de SOC: la negociación de servicios en forma automática. Un elemento clave para esto es la existencia de lenguajes formales, junto con técnicas asociadas de análisis, capaces de expresar por completo el contrato de comportamiento de una API. Estos formalismos usualmente están definidos de forma que la correctitud se reduce a la ausencia de ciertas configuraciones consideradas erróneas (i.e., *deadlock*, *receptor no especificado*, y *mensajes huérfanos*), que sólo puede ser comprobada con la presencia de todos los participantes ivolucrado a través de propiedades aún cuand como *Generalized Multiparty Ccompatibility* (GMC).
En este trabajo nos abocaremos al estudio de: 1) una nueva clase de *Communicating Finite State Machines* (CFSMs), llamada Multichannel Communicating Finite State Machines (mCFSMs), que cuenta con una definición explícita de canales de comunicación que permiten, para un participante, la posibilidad de tener más de un canal de comunicación con los otros participantes, 2) una definición de la propiedad de GMC para sistemas de mCFSMs, 3) una clase de *Autómatas Finitos de Comunicación Asincrónica* (AFCAs) con la capacidad de internalizar la comunicación entre participantes como operaciones de lectura/escritura en buffers internos, permitiendo la composición parcial de AFCAs, y 4) una forma de relacionar un AFCA con su correspondiente mCFSM, facilitando un mecanismo para comprobar la propiedad de GMC para la clase de AFCAs.

Palabras claves: Autómatas, CFSM, SOC, GMC formalismos.

AUTOMATA FOR PARTIAL BINDING OF SERVICES

Distributed software resulting from emerging paradigms such as *service-oriented computing* (SOC) and Cloud/Fog computing are transforming the world of software systems, giving impulse to what is called the API's economy. The underlying idea of it is that it is possible to construct software artifacts by composing services provided by third parties and previously registered in repositories. Applications running over globally available computational resources and communication infrastructure dynamically and transparently reconfigured, at run-time, by the intervention of a dedicated middleware capable of discovering and binding a running application with a certain requirement, to a service capable of fulfilling it.
In general the most important aspects of an API's behaviour are documented informally limiting the possibility of achieving SOC's utopia: automatic brokering of services. A key element in this quest is the existence of formal languages, together with associated analysis techniques, capable of fully expressing the API behavioral contract. These formalisms are usually defined in a way that correctness of the communication usually reduced to the absence of certain configurations (*deadlock*, *unspecified reception*, and *orphan message*) can only be asserted in the presence of all the participants involved via properties like *Generalized Multiparty Ccompatibility* (GMC).
In this work we study: 1) a new class of CFSMs, called *Multichannel Communicating Finite State Machines – mCFSMs*, with an explicit definition of the communication channels enabling, for a participant, the possibility of having more than one channel with the other participants 2) a definition of the GMC property for systems of mCFSMs, 3) a class of *Asynchronous Finite Communicating Automatas* (AFCAs) with the capability of internalising the communication as read / write operations on internal buffers, enabling partial composition of communicating automata, and 4) a method for mapping an AFCA to an mCFSM providing a checking mechanism of the GMC property for the class of AFCA.

Keywords: Automata, CFSM, SOC, GMC, formalisms.

AGRADECIMIENTOS

A mis padres Liliana y Carlos por su constante apoyo, por empujarme a seguir hasta el final y la paciencia.
A mi director, el Dr. Carlos Gustavo Lopez Pombo por orientarme, ayudarme y principalmente darme el espacio y presentarme los temas tratados en este trabajo.
A mi codirector, el Dr. Ignacio Vissani por sus aportes y correcciones.
A mi abuelo Ernesto por su constante apoyo en toda mi vida y especialmente en mis años de universidad, hasta sus últimos días. A mi abuela Raquel Amelia que todavía está y sigue preguntando y apoyándome en todo lo que hago.
A mi maestro de Kung Fu Anibal Tanus, sin las artes marciales no sería quien soy.
A mi tía Raquel y a mis primos Fabio y Sergio que más que primos son hermanos.
A mis amigos, Matt, Ana, Lean, Gery, Santi, Mati, Vicka, Ro, Magnus, Indira, Guille y Facu por estar siempre, por acompañarme todos estos años, de cerca y de lejos.
Por último a mi perro Avi que me acompañó hasta principios de este año y fue una alegría constante.

Índice general

1. INTRODUCCIÓN

La tendencia hacia los sistemas distribuidos genera la necesidad de mecanismos de comunicación más complejos. Para manejar esta complejidad se han introducido lenguajes de especificación y métodos formales de análisis que permiten asegurar ciertas propiedades de dichos mecanismos como por ejemplo, las *Communicating Finite State Machines* [BZ83], los *Global Graphs* [CDP12], los *Session types* en su gran diversidad de variantes [HVK98, HYM08] y los *Interface automata* [dH01]. Tomando esto en consideración, los autómatas finitos [HMU01, Def. 2.2.1] resultan útiles como lenguaje primitivo de modelado, subyacente en muchos de los formalismos mencionados anteriormente, que permite representar algunos de estos aspectos de dichos sistemas.

Service oriented computing (SOC) es un paradigma de computación distribuida que cambió el modo en que los sistemas de software son concebidos. El corazón del paradigma son servicios que proveen elementos computacionales autónomos, independientes de la plataforma y que se ejecutan sobre una infraestructura de cómputo y comunicación existente. Estos pueden ser descritos, publicados, descubiertos y programados usando protocolos estándar para construir redes de aplicaciones que colaboran entre sí, incluso distribuidas dentro de distintas fronteras organizacionales, con la misión de, colectivamente, alcanzar un objetivo de negocios. Uno de los elementos centrales de este paradigma es que dichos elementos computacionales son procurados en tiempo de ejecución y bajo demanda; una demanda, resulta local a cada ejecución, lo que implica que no todos los servicios necesarios son al mismo tiempo y algunos, puede que ni siquiera lo sean.

Esta mirada sobre cómo un sistema de software evoluciona reconfigurándose en tiempo de ejecución pone de relieve la necesidad de contar con un lenguaje de descripción que posibilite un enfoque para la composición que sea parcial, y que el resultado de dicha composición resulte una descripción legítima de una componente; una característica que la gran mayoría de los lenguaje utilizados en la descripción de sistemas distribuidos no posee. Volveremos sobre esto en la Sec. 5.2 donde discutiremos otros lenguajes formales relacionados, que hemos mencionado más arriba, al final de la presente.

En este trabajo buscaremos definir un lenguaje formal de modelado de componentes de software con el objeto de satisfacer las necesidades mencionadas anteriormente. Este lenguaje debe satisfacer las siguientes propiedades: 1. debe posibilitar la convivencia de elementos computacionales internos de una componente (transiciones que expresan cambios locales de estado) con su interfaz de comunicación (transiciones que expresan envío o recepción de mensajes), 2. debe poseer un mecanismo claro de composición que no requiera que todos los participantes, y 3. debe tener semántica de comunicación asincrónica, compatible con la semántica de los lenguajes formales conocidos como los mencionados anteriormente).

Para satisfacer estos objetivos definiremos una clase de autómatas finitos, a la que llamaremos *Autómatas Finitos de Comunicación Asincrónica* (AFCA), que cuentan con transiciones internas (denotando cambios locales de estado) y transiciones de comunicación sobre canales de comunicación (que expresan la comunicación con otras componentes), adicionalmente, estos autómatas pueden ser compuestos internalizando la comunicación a través de la creación de buffers dedicados que permiten reemplazar los canales de comunicación. Adaptaremos el lenguaje de las Communicating Finite State Machines, a las llamaremos

multichannel Communicating Finite State Machines (mCFSM), con el objeto de que sean capaces de reflejar la interfaz de comunicación de estos autómatas. Por último, probaremos la equivalencia entre la semántica de la composición de una familia de AFCAs y la del *communicating system* obtenido a partir de la familia de CFSM correspondientes a cada uno de dichos AFCAs (ver Fig. 4.0.1).

Esta tesis se divide en cuatro capítulos. Primero en el capítulo 2 describimos las CFSM como lenguaje y la noción de Generalized Multiparty Compatibility (GMC, [LTY15]) como condición suficiente para que un conjunto de CFSMs formen un communicating system seguro. En el capítulo 3 definimos los AFCA, su composición, y la proyección de su interfaz de comunicación. También definimos mCFSM como extensión de CFSM y proyección de la interfaz de comunicación de un AFCA, y adaptamos la propiedad de GMC para este nuevo modelo. En el capítulo 4 exploramos la equivalencia de la semántica de un conjunto de AFCA y el communicating system resultante de su mCFSM proyectadas, y demostramos la validez de esta propiedad. Por último, en el capítulo 5 cerramos este trabajo con algunas conclusiones que hemos podido derivar del trabajo, presentamos brevemente distintos trabajos relacionados y algunas ideas para trabajo a futuro.

2. PRELIMINARES

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

Definición 1 (Sistema de transición etiquetado, [Kel76]). *Una estructura* $S = \langle Q, \Sigma, \longrightarrow \rangle$ *se dice que es un* sistema de transición etiquetado *si satisface las siguientes propiedades:*

- Q *es un conjunto finito llamado* conjunto de estados,
- Σ *es un conjunto finito llamado* conjunto de etiquetas, *y*
- $\longrightarrow \subseteq Q \times \Sigma \times Q$.

Si $q, q' \in Q$, $a \in \Sigma$, *cuando* $\langle q, a, q' \rangle \in \longrightarrow$ *lo denotaremos como* $q \xrightarrow{a} q'$. *Denotaremos* Σ^* *al conjunto de secuencias (finitas o infinitas)* $\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_n, \ldots$ *tal que para todo* $i \in \mathbb{N}$, $\sigma_i \in \Sigma$. *A su vez, dado* $q \in Q$, *definiremos* $\Sigma^*[q]$, *llamado el lenguaje de* q, *como* $\{\sigma_0, \sigma_1, \ldots, \sigma_n, \ldots \in \Sigma^* \mid$ *existe* $\{q_i\}_{i \in \mathbb{N}}$ *tal que* $q = q_0$ *y para todo* $i \in \mathbb{N}$, $q_i \xrightarrow{\sigma_i} q_{i+1}\}$. *Dados* $q, q' \in Q$ *y* $\sigma \in \Sigma^*$, $q \xrightarrow{\sigma} q'$ *si y solo si* $\sigma = \sigma_0, \sigma_1, \ldots, \sigma_n$ *y existen* $\{q_i\}_{0 \leqslant i \leqslant n} \subseteq Q$ *tal que: 1-* $q_0 = q$, *2-* $q_n = q'$, *y 3- para todo* $0 \leqslant i < n$, $q_i \xrightarrow{\sigma_i} q_{i+1}$.

Definición 2 (Relaciones y clausuras). *Sea* Q *un conjunto y* $R, S \subseteq Q \times Q$*:*

- $R^0 = \{\langle q, q \rangle \in Q \times Q\}$ *(denominada* relación identidad*),*
- $R^{-1} = \{\langle q', q \rangle \in Q \times Q \mid \langle q, q' \in R\}$ *(denominada* relación inversa*),*
- $R \circ S = \{\langle q, q'' \rangle \in Q \times Q \mid$ *existe*$q' \in Q$ *tal que* $\langle q, q' \rangle \in R$ *y* $\langle q', q'' \rangle \in S\}$ *(denominada* relación composición*), y*
- *dado* $i \in \mathbb{N}$, $R^{i+1} = R^i \circ R$ *(denominada* i*-ésima* iteración*).*
- S *es la* clausura reflexiva *de* R *si* $S = R^0 \cup R$,
- S *es la* clausura simétrica *de* R *si* $S = R \cup R^{-1}$, *y*
- S *es la* clausura transitiva *de* R *si* $S = \bigcup_{i \in \mathbb{N} \text{ e } i > 0} R^i$ *(denotada* R^+*).*

S se dice que es la clausura reflexiva y transitiva *si* $S = R^0 \cup R^+$ *(denotada como* R^**) y se dice que es la* clausura reflexiva, simétrica y transitiva *si* $S = R^0 \cup R^{-1} \cup R^+$ *(denotada como* $R^\bullet$*). Cuando una relación es reflexiva, simétrica y transitiva se dice que es una* equivalencia.

Definición 3 (Clases de equivalencia y cociente). *Sea* Q *un conjunto y* $R \subseteq Q \times Q$ *una relación de equivalencia:* $Q|_R = \{s \in \wp(Q) \mid$ *para todo* $q, q' \in s$, $\langle q, q' \rangle \in R$ *y no existe* $q'' \in \overline{s}$ *tal que* $\langle q, q'' \rangle \in R\}$. *Dado* $q \in Q$, *denotaremos* $[q] \in Q|_R$ *al conjunto* $s \in Q|_R$ *tal que* $q \in s$.

En lo subsiguiente introduciremos la noción de equivalencia entre estados que usaremos a lo largo del presente trabajo. Intuitivamente, solo deseamos que dos estados sean distinguibles si tienen la capacidad ser diferenciados por alguna continuación a partir de ellos. A continuación daremos una primera aproximación a esta noción de equivalencia.

Definición 4 (Bisimulación [Mil89], Chap. 4, Def. 1)**.** *Dado un sistema de transición etiquetado* $S = \langle Q, \Sigma, \delta \rangle$*, una* bisimulación *es una relación binaria* $R \subseteq Q \times Q$*, tal que: para cada par de elementos* $p, q \in Q$ *tal que* $\langle p, q \rangle \in R$ *vale:*

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

Dados dos estados $p, q \in Q$*,* p *es bisimilar a* q*, si existe una bisimulación* R *tal que* $\langle p, q \rangle \in R$*.*

Si observamos la definición anterior, dos estados que pertenecen a la relación de bisimulación no tienen posibilidad de ser distinguidos bajo ninguna posible sucesión de transciones comenzando en ellos. Si bien esta definición caracteriza una relación de equivalencia muy útil para razonar sobre sistemas de software, en nuestro caso no será suficiente dado que el tipo de autómatas con los que trabajaremos poseen etiquetas de diferente naturaleza. Para poder razonar sobre ellos se introducen las siguientes definiciones.
En [Mil89, Sec. 5.1, Def. 5] Milner nos da una definición de *bisimulación débil* basada en la eliminación de las transiciones silenciosas, o transiciones ϵ (ver [Mil89, Defs. 1 a 4]). En nuestro caso, adaptaremos dicha definición para hacerla paramétrica en el conjunto de transiciones que deseamos silenciar a los efectos de determinar bisimilaridad débil.

Definición 5. *Sea* Σ *un conjunto de etiquetas,* $\Sigma' \subseteq \Sigma$ *y* $\sigma \in \Sigma^*$*, se define* $\widehat{\sigma}_{\Sigma'}$ *como la secuencia de etiquetas obtenida a partir de eliminar de* σ *toda ocurrencia de etiquetas en* Σ'*.*

Definición 6 (Bisimulación débil)**.** *Dado un sistema de transición etiquetado* $S = \langle Q, \Sigma, \delta \rangle$ *y* $\Sigma' \subseteq \Sigma$ *el conjunto de etiquetas a silenciar, una* bisimulación débil sobre Σ' *es una relación binaria* $R \subseteq Q \times Q$*, tal que: para cada par de elementos* $p, q \in Q$ *tal que* $\langle p, q \rangle \in R$ *vale:*

- $(\forall \sigma \in \Sigma \backslash \Sigma')(p \xrightarrow{\sigma} p' \implies (\exists q' \in Q)(\exists \sigma^* \in \Sigma^*)(q \xrightarrow{\sigma^*} q' \wedge \widehat{\sigma^*}_{\Sigma'} = \sigma \wedge \langle p', q' \rangle \in R))$ *y, simétricamente, vale*
- $(\forall \sigma \in \Sigma \backslash \Sigma')(q \xrightarrow{\sigma} q' \implies (\exists p' \in Q)(\exists \sigma^* \in \Sigma^*)(p \xrightarrow{\sigma^*} p' \wedge \widehat{\sigma^*}_{\Sigma'} = \sigma \wedge \langle p', q' \rangle \in R))$

Dados dos estados $p, q \in Q$*,* p *es débilmente bisimilar sobre* Σ' *a* q*, denotado como* $p \sim_{\Sigma'} q$*, si existe una bisimulación débil* R *tal que* $\langle p, q \rangle \in R$*.*

Comunicación Asincrónica

Como se dijo en la primera sección, buscamos una implementación del paradigma de SOC donde la comunicación entre distintos componentes se realiza en forma asincrónica. Esto no es un requerimiento del paradigma propiamente dicho pero resulta una implementación más eficiente. El intercambio de mensajes en forma asincrónica permite a los componentes maximizar su uso de la cpaacidad de cómputo. El intercambio de mensajes con cada componente es independiente entre sí, salvo que exista alguna necesidad de lo contrario. Para representar este comportamiento utilizamos un tipo de autómata finito denominado *Communicating Finite State Machines* (llamado a partir de ahora CFSM). El concepto de CFSM fue introducido en [BZ83] 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. Las CFSMs son autómatas que modelan únicamente la comunicación externa entre participantes. De este modo cada CFSM representa los distintos componentes del sistema distribuido, pero solo el comportamiento que es relevante a la interacción entre los mismos. Una CFSM se puede ver como la proyección de este comportamiento específico de un autómata más complejo que tenga otro tipo de transiciones adicionales.
La naturaleza dinámica no pre programada de los sistemas SOC hace que se requiera algún mecanismo para verificar que todos los componentes puedan funcionar entre sí en forma correcta. Para esto en esta sección detallamos el concepto de *Generalised multiparty compatibility* (GMC), definido originalmente en [LTY15]. GMC se define a partir de una serie de condiciones que debe cumplir el conjunto de CFSMs participantes del sistema. Esto condiciona al sistema (o a quienes lo diseñen) a tener un conocimiento previo de los posibles sistemas que vayan a interactuar entre sí. Esto no necesariamente rompe con la idea del descubrimiento y binding en tiempo de ejecución dado que se pueden generar métodos para hacer estas comprobaciones a medida que el sistema va conectándose con los distintos participantes.

Definición 7 (Communicating Finite State Machines)**.** *Sea $\mathcal{M}$ un conjunto finito de mensajes y $\mathcal{P}$ 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 *CS* como un conjunto de CFSMs que cumplen determinadas propiedades.

Definición 8 (Communicating System, [LTY15], Def. 2.2)**.** *Dado un conjunto de mensajes $\mathcal{M}$, una CFSM $M_p = \langle Q_p, \mathcal{C}_p, q_{0_p}, \delta_p \rangle$ para cada participante $p \in \mathcal{P}$, la tupla $S = \langle M_p \rangle_{p \in \mathcal{P}}$ es un communicating system (CS).*

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 9 (Estados y configuraciones alcanzables)**.** *Dado un conjunto de mensajes $\mathcal{M}$, una CFSM $M_p = \langle Q_p, \mathcal{C}_p, q_{0_p}, \delta_p \rangle$ para cada participante $p \in \mathcal{P}$ y $S = \langle M_p \rangle_{p \in \mathcal{P}}$ un CS, el sistema de transición etiquetado que determinala semántica de S (denotado como $M_S = \langle Q_S, q_{0S}, \delta_S \rangle$ una configuración de S es un par $s = \langle \overrightarrow{q}; \overrightarrow{\omega} \rangle$ donde $\overrightarrow{q} = (q_p)_{p \in \mathcal{P}}$ con $q_p \in Q_p$ y donde $\overrightarrow{\omega} = (\omega_{pq})_{pq \in C}$ con $\omega_{pq} \in \mathcal{M}^*$. 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 $q_{0S} = \langle \overrightarrow{q_0}; \overrightarrow{\epsilon} \rangle$ con $\overrightarrow{q_0} = (q_{0_p})_{p \in \mathcal{P}}$ y $\overrightarrow{\epsilon} = (\epsilon_p)_{p \in \mathcal{P}}$.*
Dadas $c' = \langle \overrightarrow{q'}, \overrightarrow{\omega'} \rangle$, $c = \langle \overrightarrow{q}, \overrightarrow{\omega} \rangle$ y $l \in \mathcal{C}_s \times \{!, ?\} \times \mathcal{M}$, decimos que c' es alcanzable desde c a través de la etiqueta l, denotado como $\langle c, l, c' \rangle \in \delta_S$, si y sólo si:

1. $l = sr!m$ *y* $\langle q_s, l, q'_s \rangle \in \delta_s$ *y*

   a) $q'_p = q_p$ *para todo* $\mathsf{p} \neq s$*; y*

   b) $\omega'_{sr} = \omega_{sr} \cdot m$ *y* $\omega'_{pq} = \omega_{pq}$ *para todo* $pq \neq sr$*; o bien*

2. $l = sr?m$ *y* $\langle q_r, l, q'_r \rangle \in \delta_r$ *y*

   a) $q'_p = q_p$ *para todo* $p \neq r$*; y*

   b) $\omega_{sr} = m \cdot \omega'_{pq}$ *y* $\omega'_{pq} = \omega_{pq}$ *para todo* $pq' \neq sr$

El conjunto de configuraciones alcanzables de S *es* $RS(S) = \{q \in Q_S | \langle q_{0S}, l_0 \dots l_n, q \rangle \in \delta_S^*\}$ *donde* δ_S^* *es la clausura reflexo-transitiva de* δ_S

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

Definición 10 (Deadlock)**.** *Sea* S *un CS, una transición* t *del mismo y* $s = \langle \overrightarrow{q}; \overrightarrow{\omega} \rangle$ *con* $\overrightarrow{q} = \langle q_1, \dots, q_n \rangle$ *y sea* $\overrightarrow{\omega} = \langle \omega_1, \dots, \omega_n \rangle$ *una de sus configuraciones. Decimos que* s *es una configuración de deadlock si* $\overrightarrow{\omega} = \overrightarrow{\epsilon}$*, existe* $r \in \mathcal{P}$ *tal que* $\langle q_r, sr?a, q'_r \rangle \in \delta_r$*, y para todo* $p \in \mathcal{P}$*,* q_p *es un estado receptor o final. Es decir todos los canales están vacíos, hay al menos una máquina esperando un mensaje y todas las otras máquinas están en un estado final o receptor.*

Definición 11 (Recepción no especificada)**.** *Sea* S *un CS, una transición* t *del mismo y* $s = \langle \overrightarrow{q}; \overrightarrow{\omega} \rangle$ *con* $\overrightarrow{q} = \langle q_1, \dots, q_n \rangle$ *y sea* $\overrightarrow{\omega} = \langle \omega_1, \dots, \omega_n \rangle$ *una de sus configuraciones. Decimos que* s *es una configuración de recepción no especificada si existe* $r \in \mathcal{P}$ *tal que* qr *es un estado receptor y* $\langle q_r, sr?a, q'_r \rangle \in \delta_r$ *implica que* $|\omega_s r| > 0$ *y* $\omega_s r \notin a\mathcal{M}*$*. Una configuración de recepción no especificada corresponde a una configuración en la que la máquina está bloqueada definitivamente por su incapacidad para recibir el mensaje que se encuentra en sus canales.*

Definición 12 (Mensaje Huérfano)**.** *Sea* S *un CS, una transición* t *del mismo y* $s = \langle \overrightarrow{q}; \overrightarrow{\omega} \rangle$ *con* $\overrightarrow{q} = \langle q_1, \dots, q_n \rangle$ *y sea* $\overrightarrow{\omega} = \langle \omega_1, \dots, \omega_n \rangle$ *una de sus configuraciones. Decimos que* s *es una configuración de mensaje huérfano si todos los* $q_p \in \overrightarrow{q}$ *son finales pero* $\overrightarrow{\omega} \neq \overrightarrow{\epsilon}$*. Es decir quedó un mensaje en algún canal pero no hay ninguna transición que lo reciba.*

Definición 13 (CS 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 recepciones no especificadas, y*
3. s *no posee mensajes huérfanos*

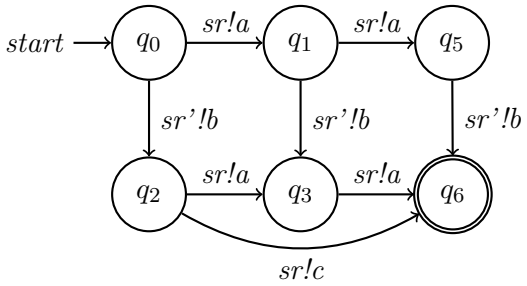
Para poder expresar esta condición de seguridad es necesario identificar conjuntos de acciones que pueden ser llevadas a cabo concurrentemente. Para esto definimos las siguientes relaciones sobre el conjunto de transiciones de una CFSM. Dados $q, q' \in Q$*, se define* $act(q, q') = \{\ell \mid (q, \ell, q') \in \delta\}$ *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* $l \notin act(q_1, q'_1) \wedge act(q_1, q'_1) = act(q_2, q'_2) \wedge act(q_2, q'_2) \neq \varnothing$

- $(q_1, \ell, q_2)\underline{\blacklozenge}(q_1', \ell, q_2')$ *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.*

Ejemplo 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* $((q_0, sr!a, q1)\underline{\Diamond}(q_1, sr'!a, q5))$
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 intercaladas 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 14 (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.*

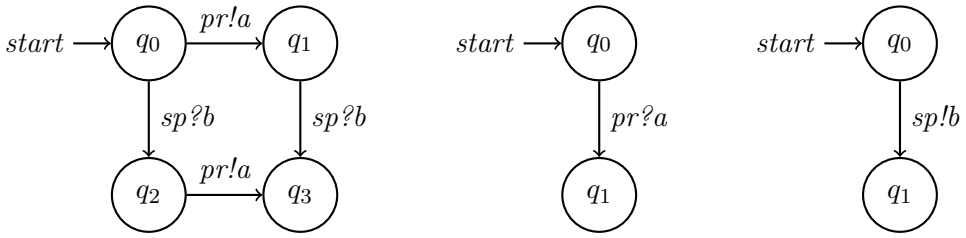
Tomando en consideración la definición anterior, para distinguir el paralelsimo a nivel máquina introducimos una relación de equivalencia sobre eventos que identifica eventos cuyas transiciones son $\blacklozenge$-equivalentes.

Definición 15 (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:*

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

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

Ejemplo 2 (Equivalencia entre eventos)**.** *Considere el siguiente CS:*



En el sistema de arriba podemos ver los siguientes eventos $(q_{0p}, q_{0r}, p \to r : a)$, $(q_{2p}, q_{0r}, p \to r : a)$, $(q_{0s}, q_{0p}, s \to p : b)$ *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 $(q_{0p}, pr!a, q_{1p}) \blacklozenge \underline{\ } (q_{2p}, pr!a, q_{3p})$ *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* $act(q_{0p}, q_{2p}) = act(q_{1p}, q_{3p})$ *y* $act(q_{2p}, q_{2p}) = act(q_{3p}, q_{3p})$. *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* $(q_{0r}, pr?a, q_{1r}) \blacklozenge \underline{\ } (q_{0r}, pr?a, q_{1r})$. *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 16 (Sistema de transición sincrónico)**.** *Dados un* $S = (M_P)_{p \in \mathcal{P}}$ *un CS, sea* $\langle N, \hat{\delta}, E \rangle$, *donde:*

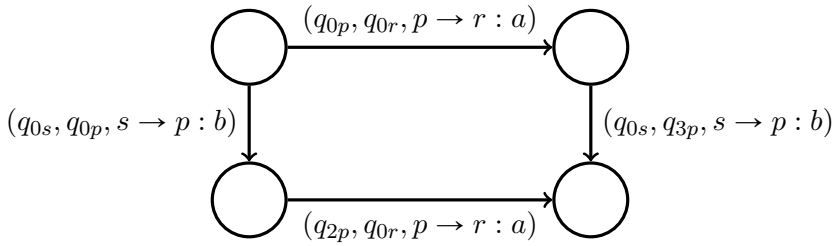
$$N = \{\overrightarrow{q} \mid (\overrightarrow{q}; \overrightarrow{\epsilon}) \in RS_1(S)\},$$

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

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

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. 9 a* $\Rightarrow$ *(ej: si* $\pi = e_1 ... e_k, n_1 \overset{\pi}{\Rightarrow} n_{k+1} si in_1 \overset{e_1}{\Rightarrow} n_2 \overset{e_2}{\Rightarrow} ... \overset{e_k}{\Rightarrow} n_{k+1}$*).*
$TS(S)$ *representa todas las posibles ejecuciones sincrónicas del sistema* S*; y cada transición es etiquetada con un evento* e.

Ejemplo 3 (Sistema de Transición Sincrónico)**.** *Consideremos el CS del 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* $(q_{0s}, q_{0p}, s \to p : b) \bowtie (q_{0s}, q_{3p}, s \to p : b)$*. 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 17 (Proyecciones)**.** *La proyección de un evento e sobre un participante p, denotado por* $e \upharpoonright_p$ *se define de la siguiente manera:*

$$(q_s, q_r, s \to r : a) \upharpoonright_p = \begin{cases} pr!a & \text{if } s = p \\ sp?a & \text{if } r = p \\ \epsilon & \text{en otro caso} \end{cases} \tag{2.0.1}$$

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* $Q = N$*,* $q_0 = n_0$*,* Σ *es el conjunto de etiquetas y* $\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times Q$ *es el conjunto de transiciones, tal que* $(n_1, e \upharpoonright_p, n_2) \in \delta \iff n_1 \overset{e}{\Rightarrow} n_2$

A continuación introducimos el concepto de *Generalized Multiparty Compatibility* (GMC). La noción de GMC originalmente surge en [LTY15] como una condición completa y sólida para construir Global Graphs. En este trabajo la utilizamos como método para comprobar que un conjunto de CFSMs que forman un CS puedan operar correctamente entre sí. 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 traza y elección de cada máquina estén representadas en el sistema de transición.

Definición 18. *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 19 (Representabilidad)**.** *Un sistema S es representable si*

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

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 asegura 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 20 (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*
   $L^i_p = hd(\{e_i \restriction_p \cdot \phi \mid \phi \in LT(S, n'_i, p)\})$ 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. $\exists n' \in N, \pi_1, \pi_2$*:* $n'_1 \stackrel{\pi_1}{\Rightarrow} n' \wedge n'_2 \stackrel{\pi_2}{\Rightarrow} n' \wedge (e_1 \cdot \pi_1) \restriction_p = (e_2 \cdot \pi_2) \restriction_p = \epsilon$
   - *(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 (Def. 19) garantiza que $TS(S)$ contiene suficiente información para decidir propiedades seguras de cualquier ejecución asincrónica de S. La propiedad de ramificación (Def. 20) 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 21 (Generalised Multiparty Compatibilty)**.** *Un CS S es generalised multiparty compatible (GMC) si es representable y posee la propiedad de ramificación.*

Teorema 1 (Solvencia)**.** *Si S es GMC entonces es seguro (no tiene configuraciones de deadlock, recepción no especificada o mensaje huérfano)*

El teorema dice que ninguna ejecución asincrónica de S va a resultar en una configuración de mensaje huérfano, deadlock o recepción no especificada. Basándose en la propiedad de representabilidad (toda transición y ramificación de cada máquina está representada en $TS(S)$), la demostración [LTY15] muestra que todo mensaje enviado es recibido eventualmente y que una máquina en un estado receptor eventualmente recibe el mensaje que espera, según la Def. 19.

3. COMPONENTES ASINCRÓNICAS

Como ya hemos mencionado, en SOC 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 a priori 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, ausencia de mensajes huérfanos y ausencia de situaciones en las que el receptor no se encuentra a la espera de 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.

3.1. Autómatas finitos de comunicación asincrónica

Las CFSMs son un tipo de autómata que modela únicamente la comunicación externa con otros participantes del sistema. Necesitábamos un tipo de autómata que además representara otros comportamientos internos. Para esto primero recurrimos a los Input Output Automata [LT89]. Un I/O automaton modela un componente de un sistema distribuido que puede interactuar con otros sistemas componentes. Es un tipo de máquina de estados en la cual las transiciones están asociadas con determinadas acciones específicas. Existen tres tipos de acciones: input, output, y acciones internas. El autómata usa sus acciones de input y output para comunicarse con el enotrno, mientras que las acciones internas son

sólo visibles para el autómata. A diferencia de las acciones internas y las de output que son seleccionadas y resueltas por el autómata, las de entrada, que llegan del entorno, no están bajo control del autómata. Para componer estos autómatas se requiere que sincronicen las acciones compartidas. Es decir cuando un autómata ejecuta una transición con la etiqueta π, todo autómata que tenga esa acción π la ejecuta en simultáneo. El problema es que los I/O no representan bien la comunicación asincrónica. Las operaciones de comunicación externa sincronizan a todos los participantes del sistema. Es decir que si dos autómatas componentes tienen una misma transición, se ejecutan en simultáneo. Esto va en contra de la idea de asincronsidad que buscamos. Para modelar este comportamiento asincrónico introducimos los Autómatas Finitos de Comunicación Asincrónica (AFCA) (ver Def. 23). Estos autómatas tienen tres tipos de transiciones: 1) internas, que sirven el propósito de representar cómputo realizado por la componente; 2) de buffer, que representan comunicación asincrónica interna entre elementos de la componente, y que permiten representar la comunicación entre dos participantes luego de una composición; y por último 3) de entrada / salida, que modelan acciones de comunicación con otras componentes del sistema.
Como lo que queremos es modelar sistemas distribuidos con comunicación asincrónica, necesitamos una noción de composición. Para esto introducimos una definición de composición para los AFCA (ver Def. 29).

Definición 22 (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 23 (Autómata finito de comunicación asincrónica)**.** *Sea* $\mathcal{P}$ *un conjunto de participantes y* $\mathcal{M}$ *un conjunto de mensajes. Un* autómata finito 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. 22),*
- $\mathcal{C} \subseteq \{pq_n \mid pq \in \mathcal{P}^2, n \in \mathbb{N}, p \neq q\}$ *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* $In(c,m), Out(c,m)$ *dónde* $c \in \mathcal{C}$ *es el canal a través del cual se resuelve la misma y* $m \in \mathcal{M}$ *es el mensaje .*

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 transición por acciones internas de* $A_{\mathcal{P}}$*,*

  2) $\delta_{Ex} \subseteq Q \times \Sigma_{EX} \times Q$ *es la relación de transición de comunicación externa de* $A_{\mathcal{P}}$*,*

  3) $\delta_{Buff} \subseteq Q \times \Sigma_{Buff} \times Q$ *es la relación de transición de comunicación interna de* $A_{\mathcal{P}}$

- $q_0 \in Q$ *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* $P(A) = (p \in \mathcal{P} \mid (\exists \langle p1, p2, c\rangle \in \Sigma_{Ex})\,(p1 = p \vee p2 = p))$

Σ es el conjunto de todas las etiquetas del autómata y lo dividimos en tres subconjuntos disjuntos de la forma 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 entre cada par de participantes hay al menos un canal en cada sentido de la comunicación.
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 que representan el intercambio de mensajes $m \in \mathcal{M}$ entre dos autómatas a través de un canal de comunicación $c \in \mathcal{C}$. Las acciones de entrada o input representan la recepción de un mensaje de algún proceso externo al correspondiente al autómata, denotadas *In(c,m)*. Las acciones de output o salida que representan el envío de un mensaje y se denotan *Out(c,m)*. Estas acciones son la parte externa de la comunicaión asincrónica. Como los canales son unidireccionales decimos que para todo par de participantes existen una cantidad finita de pares de canales.
δ_{Buff} es la relación de transición de comunicación interna mediante buffers.

Formalmente, F ⊆ 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 24 (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 el conjunto 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 25 (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. $\langle q_1, \Omega \rangle \vdash \langle q_2, \Omega \rangle \iff \langle q_1, r, q_2 \rangle \in \delta \wedge r \in \Sigma \backslash \Sigma_{Buff}$
2. $\langle q_1, \{\omega_{b_1}, \ldots, \omega_{b_i}, \ldots, \omega_{b_n}\} \rangle \vdash \langle q_2, \{\omega_{b_1}, \ldots, m : \omega_{b_i}, \ldots, b_n\} \rangle \iff \langle q_1, b_i \ll m, q_2 \rangle \in \delta$
3. $\langle q_1, \{\omega_{b_1}, \ldots, \omega_{b_i} : m, \ldots, \omega_{b_n}\} \rangle \vdash \langle q_2, \{\omega_{b_1}, \ldots, \omega_{b_i}, \ldots, \omega_{b_n}\} \rangle \iff \langle q_1, b_i \gg m, q_2 \rangle \in \delta$

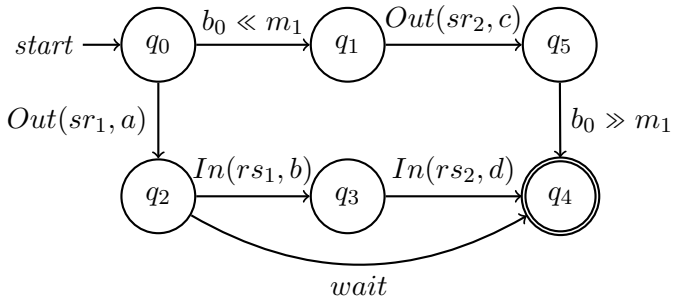
Definición 26 (Traza de un AFCA)**.** *Llamamos traza a una secuencia posible de acciones de un autómata. Se define como una secuencia finita de etiquetas de estado y transición altenradas, que comienza y termina con un estado. Dado un autómata* $A_{\mathcal{P}} = \langle Q, B, \mathcal{C}, \Sigma, \delta, q_0, F \rangle$, *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 27 (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 válida. Una ejecución es una configuración que no entra en deadlock, mensajes huérfanos ni receptor no especificado.*

Ejemplo 4 (Ejemplo de un Autómata Finito de Comunicación Asincrónica)**.** *Considere el 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 de comunicación interna:* $b_0 \ll m_1$ *y* $b_0 \gg m_1$
- *Transiciones de comunicación externa:* $Out(sr_1, a), Out(sr_2, c), In(rs_1, b)$ *y* $In(rs_2, d)$
- *Transiciones internas: wait*

Dado que los AFCA representan procesos, o servicios, que interactúan entre sí como parte de un sistema, necesitamos definir la operación de composición.

Definición 28 (Compatibilidad)**.** *Dados* $\mathcal{P}$ *un conjunto de participantes,* $\mathcal{M}$ *un conjunto de mensajes y* $A_{\mathcal{P}} = \{A_p \mid p \in \mathcal{P}\}$ *un conjunto finito de autómatas finitos de comunicación asincrónica* $A_{p_i} = \langle Q_{p_i}, B_{p_i}, \mathcal{C}_{p_i}.\Sigma_{p_i}, \delta_{p_i}, q_{0p_i}, F_{p_i}\rangle$*, diremos que* $A_{\mathcal{P}}$ *es* compatible *si se satisface que para todos* $p_i, p_j \in \mathcal{P}$*,* $\Sigma_{p_i} \cap \Sigma_{p_j} = \varnothing$ *y* $B_{p_i} \cap B_{p_j} = \varnothing$*.*

Definición 29 (Composición)**.** *Dados* $\mathcal{P}$ *un conjunto de participantes,* $\mathcal{M}$ *un conjunto de mensajes y* $A_{\mathcal{P}} = \{A_p \mid p \in \mathcal{P}\}$ *un conjunto finito de autómatas finitos de comunicación asincrónica* $A_{p_i} = \langle Q_{p_i}, B_{p_i}, \mathcal{C}_{p_i}.\Sigma_{p_i}, \delta_{p_i}, q_{0p_i}, F_{p_i}\rangle$ *compatible. Luego, definimos la composición* $A_{\mathcal{P}} = ||_{1..n} A_{p_i}$ *como sigue:*

- $Q_{\mathcal{P}} = \Pi_{p_i \in \mathcal{P}} Q_{p_i}$ *(i.e. el conjunto de estados de la composición es el producto cartesiano de los estados de los autómatas componentes),*
- $B_{\mathcal{P}} = \bigcup_{p_i \in \mathcal{P}} B_{p_i} \cup \{\omega_c \mid$ *there exists* $p_i, p_j \in \mathcal{P}$ *such that* $c \in \mathcal{C}_{p_i} \cap \mathcal{C}_{p_j}\}$ *(i.e. el conjunto de nombres de buffers del autómata resultante se compone de los buffers de cada autómata participante en la composición, junto con uno nuevo por cada canal compartido entre cada par de autómatas, dichos canales son aquellos mediante los cuales los autómatas a componer intercambian mensajes entre sí),*
- $\mathcal{C}_{\mathcal{P}} = \bigcup_{p_i \in \mathcal{P}} \mathcal{C}_{p_i} \backslash \{c_k \mid$ *for all* $p_i, p_j \in \mathcal{P}, c \in \mathcal{C}_{p_i} \cap \mathcal{C}_{p_j}\}$*,*
- $\Sigma_{\mathcal{P}} = \Sigma_{\mathcal{P}Int} \cup \Sigma_{\mathcal{P}Ex} \cup \Sigma_{\mathcal{P}Buff}$ *tal que: 1)* $\Sigma_{\mathcal{P}Int} = \bigcup_{p_i \in \mathcal{P}} \Sigma_{p_i Int}$*, 2)* $\Sigma_{\mathcal{P}Ex} = \bigcup_{p_i \in \mathcal{P}} \Sigma_{p_i Ex} \backslash \Sigma_{p_i \mapsto p_j}$ *y 3)* $\Sigma_{\mathcal{P}Buff} = \bigcup_{p_i \in \mathcal{P}} \Sigma_{p_i Buff} \cup \Sigma_{p_i \mapsto p_j}$*, donde* $\Sigma_{p_i \mapsto p_j} = \{\langle p_i, p_j, c\rangle \mid p_i, p_j \in \mathcal{P},\ c \in \mathcal{C}_{p_i} \cap \mathcal{C}_{p_j}\}$
- $\delta_{\mathcal{P}} = \delta_{\mathcal{P}Int} \cup \delta_{\mathcal{P}Ex} \cup \delta_{\mathcal{P}Buff}$ *tal que:*

  1. $\delta_{\mathcal{P}Int} \subseteq Q_{\mathcal{P}} \times \Sigma_{\mathcal{P}Int} \times Q_{\mathcal{P}}$ *es la relación de transición interna, tal que que se satisface la siguiente fórmula:*

$$(\forall \langle q, \sigma, q'\rangle \in \delta_{\mathcal{P}Int})(\exists p_i \in \mathcal{P})(\exists q_{p_i}, q'_{p_i} \in Q_{p_i}, \sigma \in \Sigma_{p_i})(\langle q_{p_i}, \sigma, q'_{p_i}\rangle \in \delta_{p_i Int})$$

2. *$\delta_{\mathcal{P}Ex} \subseteq Q_{\mathcal{P}} \times \Sigma_{\mathcal{P}Ex} \times Q_{\mathcal{P}}$ es la relación de transición de comunicación externa, tal que que se satisface la siguiente fórmula:*

$$(\forall \langle q, \sigma, q' \rangle \in \delta_{\mathcal{P}Ex})(\exists p_i \in \mathcal{P})(\exists q_{p_i}, q'_{p_i} \in Q_{p_i}, \sigma \in \Sigma_{p_i})(\langle q_{p_i}, \sigma, q'_{p_i} \rangle \in \delta_{p_i Ex})$$

3. *$\delta_{\mathcal{P}Buff} \subseteq Q_{\mathcal{P}} \times \Sigma_{\mathcal{P}Buff} \times Q_{\mathcal{P}}$ es la relación de transición de comunicación interna tal que se satisface la siguiente fórmula:*

$$\begin{aligned}
&\langle q, \sigma, q' \rangle \in \delta_{\mathcal{P}Buff} \text{ sii } (q = \langle q_k \rangle_{p_k \in \mathcal{P}} \wedge q' = \langle q'_{k'} \rangle_{p_{k'} \in \mathcal{P}} \wedge \\
&\quad ((\exists p_i \in \mathcal{P})((\forall p_k \in \mathcal{P})(p_k \neq p_i \implies q_k = q'_k) \wedge \\
&\qquad (\exists \omega_b \in B_{p_i}, \omega_b\{\gg, \ll\}m \in \Sigma_{p_i Buff})(\langle q_i, \omega_b\{\gg, \ll\}m, q'_i \rangle \in \delta_{p_i Buff}) \wedge \\
&\qquad\quad \sigma = \omega_b\{\gg, \ll\}m) \vee \\
&\qquad (\exists p_i, p_j \in \mathcal{P})(p_i \neq p_j \wedge (\forall p_k \in \mathcal{P})((p_k \neq p_i \wedge p_k \neq p_j) \implies q_k = q'_k) \wedge \\
&\qquad\quad ((\exists Out(c_{p_i,p_{j_n}}, m) \in \Sigma_{p_i Ex})(\langle q_i, Out(c_{p_i,p_{j_n}}, m), q'_i \rangle \in \delta_{p_i Ex} \wedge \\
&\qquad\qquad \sigma = \omega_{c_{p_i p_{j_n}}} \ll m) \vee \\
&\qquad\quad (\exists In(c_{p_j,p_{i_n}}, m) \in \Sigma_{p_i Ex})(\langle q_i, In(c_{p_j,p_{i_n}}, m), q'_i \rangle \in \delta_{p_i Ex} \wedge \\
&\qquad\qquad \sigma = \omega_{c_{p_j,p_{i_n}}} \gg m))))
\end{aligned}$$

Además vale que $\forall q_i, q_j \in Q_{\mathcal{P}},\ m \in \mathcal{M} \mid \langle q_i, [\omega_b]_{b \in B_{\mathcal{P}}} \gg m, q_j \rangle$ $\langle q_i, [\omega_b]_{b \in B_{\mathcal{P}}} \gg m, q_j \rangle \in \delta_{\mathcal{P}Buff} \iff \exists q'_i, q'_j \in Q_{\mathcal{P}}$ tal que $\langle q'_i, [\omega_b]_{b \in B_{\mathcal{P}}} \ll m, q'_j \rangle \in \delta_{\mathcal{P}Buff}$ y se cumple una de las siguientes condiciones:

a) $q'_j = q_i$

b) $\exists \sigma \in \Sigma_{\mathcal{P}}$ *tal que* $\langle q'_j, \sigma, q_i \rangle \in \delta_{\mathcal{P}} \wedge \sigma \neq [\omega_b]_{b \in B_{\mathcal{P}}} \gg m$

c) $\exists s = [q'_j, ..., q_i]$ *donde s es una secuencia finita de estados tales que* $s[0] = q'_j, s[n-1] = q_i$ *y sea* $0 \ll x \ll$ *n-1,*$s[x] \in Q_{\mathcal{P}}$ *y* $\forall s[x], s[x+1], \exists \sigma \in \Sigma_{\mathcal{P}} \wedge \langle s[x], \sigma, s[x+1] \rangle \in \delta_{\mathcal{P}}$

- $q_0 = \langle q_{0_{p_1}}, \ldots, q_{0_{p_n}} \rangle$, *y*
- $F_{\mathcal{P}} = \bigcup_{1..n} F_{p_i}$.

Cada autómata es un componente independiente que cumple una función (o una serie de funciones), y se relaciona con otros a través del envío de mensajes. Para asegurarnos esto pedimos que para todo par de autómatas a componerse tengan conjuntos de etiquetas de acciones que sean disjuntos.
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 por cada par de integrantes, uno para cada sentido de la comunicación, de A_{p_i} a A_{p_j} 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. Hay un caso particular que se da cuando ocurren envíos de mensaje de un autómata componente a otro. En ese caso dado que ambos ahora son parte un mismo autómata, 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 $In(c_{p_i,p_{j_n}}, m)$ y $Out(c_{p_j,p_{i_n}}, m)$ ahora es $\omega_{c_{p_i p_{j_n}}} \ll m$ y $b_{c_{p_j,p_{i_n}}} \gg m$, donde $\omega_{c_{p_i,p_{j_n}}}, \omega_{c_{p_j,p_{i_n}}} \in \mathcal{C}_{\mathcal{P}}$ son los buffers exclusivos del autómata compuesto.

Al componer 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 $\delta_{\mathcal{P}Buff}$ 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).

Ejemplo 5 (Composición de Autómatas Finitos de Comunicación Asincrónica)**.** *En las siguiente figuras vemos un ejemplo de tres comunicación de tres AFCA y como funciona el mecanismo de composición. En la Fig. 3.1.1 tenemos* $P = \langle\{q_0, \ldots, q_4\}, \varnothing, \{PR, SP\}, \{out(PR, a), in(SP, b), in(SP, b), out(PR, a), int_p\}, \delta_P, q_0.\{q_4\}\rangle$*,* $R = \langle\{q_0, \ldots, q_2\}, \varnothing, \{PR\}, \{int_r, in(PR, a)\}, \delta_R, q_0.\{q_2\}\rangle$ *y* $S = \langle\{q_0, q_1\}, \varnothing, \{SP\}, \{out(SP, b)\}, \delta_P, q_0.\{q_1\}\rangle$*. Estos autómatas solo poseen comunicación externa y, en el caso de* P *y* R*, transiciones internas. La comunicación puede comenzar con* P *envíando el mensaje* a *a* R *o con* S *enviando a* P *el mensjae* b*. En ambos casos el autómata destino recibe la comunicación cuando está habilitado. En el caso de* P*, puede recibir* b *desde el inicio de su ejecución o una vez enviado* a*, mientras que* R *primero tiene que pasar por la transición interna* int_r *para poder recibir* a *y terminar. Por último* P *termina su ejecución con la transición interna* int_p*. En la Fig. 3.1.2 podemos ver la composición total de los tres componentes en* $\mathcal{A} = \langle\{q_{000}, \ldots, q_{121}\}, \varnothing, \{PR, SP\}, \Sigma_{\mathcal{A}}, \delta_{\mathcal{A}}, \{q_{000}\}, \{q_{421}\}\rangle$ *donde la comunicación que antes era externa ahora está internalizada. Los canales se convierten en buffers por donde se van mensajes. De esta forma se mantiene el orden original de la comunicación pero representada en su totalidad un único autómata.*

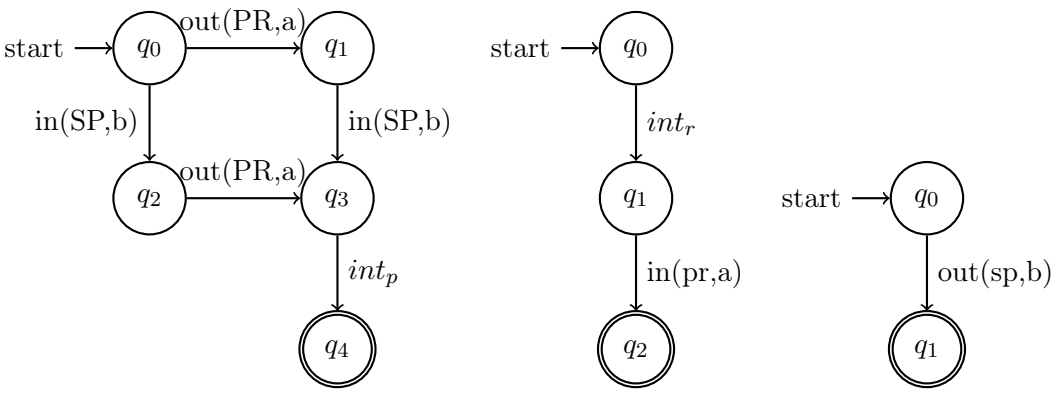


Fig. 3.1.1: Autómatas finitos de comunicación asincrónica A_P, A_R, A_S.

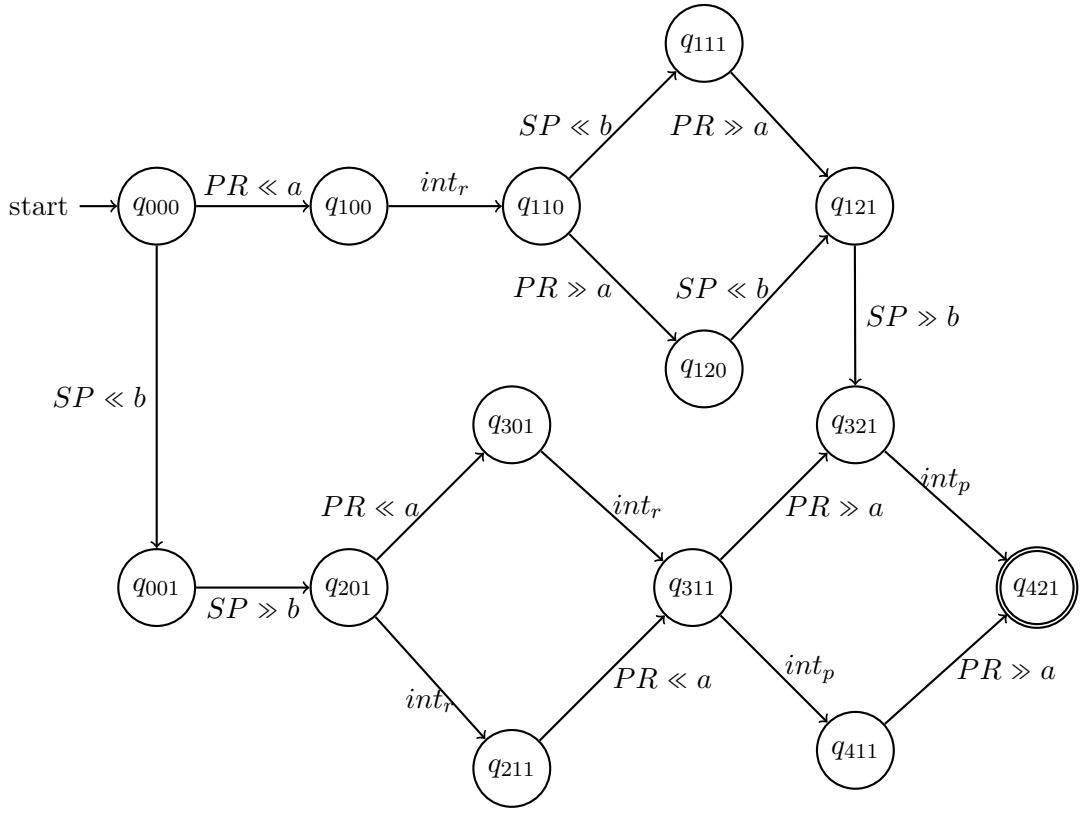


Fig. 3.1.2: Composición de los AFCAs de la Fig. 3.1.1.

3.1.1. AFCAs y composición parcial

El objetivo principal de esta clase de autómatas es posibilitar la composición parcial de componentes de un sistema, como idea de que sirva de reconfiguración dinámica del mismo. Esto debiera visibilizarse a partir de poder, dado un conjunto de AFCAs, tomar un subconjunto y componerlo, y que el resultado siga siendo un conjunto de AFCAs que puede ser compuesto a su vez. En la Fig. 3.1.1 vemos un conjunto de tres AFCA compatibles para la composición. Tomando este mismo conjunto podemos elegir componer solo dos y nos seguiría quedando un conjunto de AFCAs compatibles, como podemos ver en la Fig. 3.1.3 con el autómata compuesto $\mathcal{A}_{PR}$ y el autómata S.

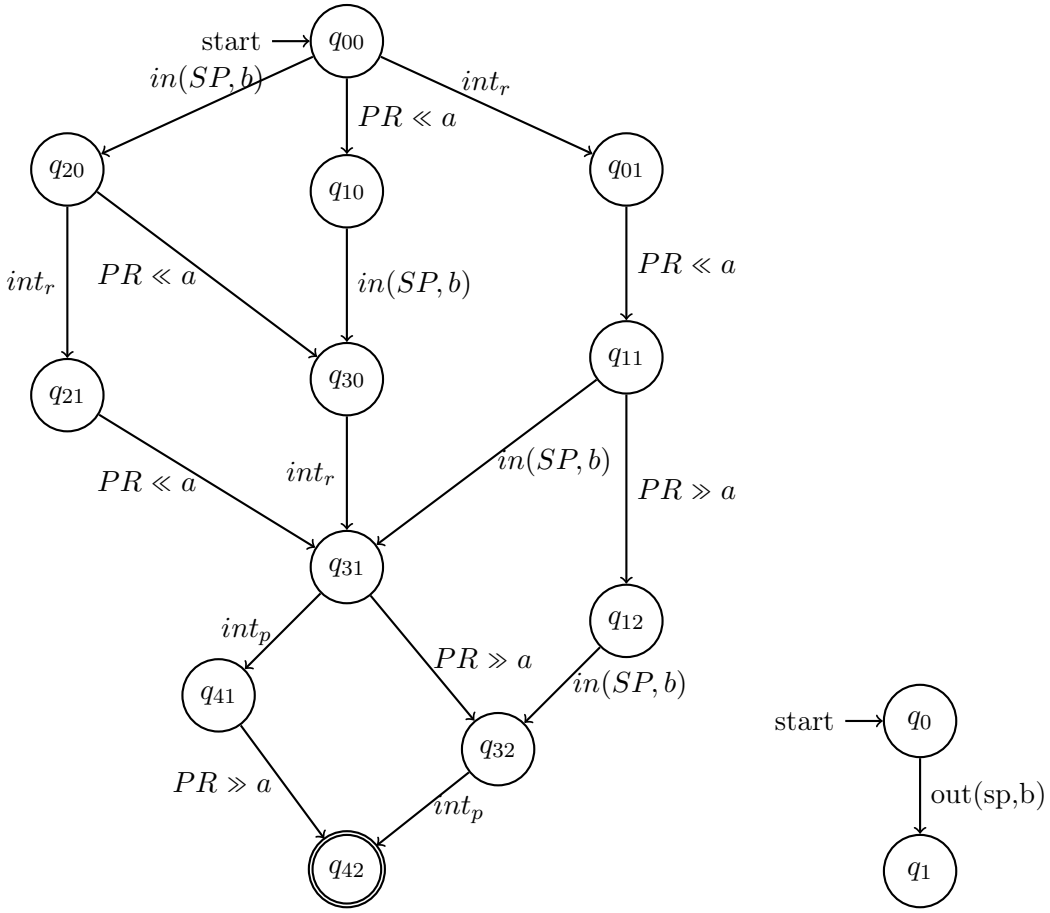


Fig. 3.1.3: Composición parcial de los AFCAs de la Fig. 3.1.1.

3.2. Aspectos comunicacionales de los AFCA

Una de las desventajas principales de utilizar CFSM como lenguaje de especificación de 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 protocolo para poder determinar si el protocolo puede llevarse a cabo sin errores. El propósito de los AFCA es intentar reducir la distancia entre ambas naturalezas. Una de las tareas principales que queremos lograr es obtener, vía una proyección, la inferfaz de comunicación en la forma de una communicating machine. 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 como los AFCA introducen la posibilidad de que haya múltiples canales de comunicación entre dos participantes, este modelo no es suficiente. En esta sección abordamos dicha problemática introduciendo las Multichannel Communicating Finite State Machines [Vis18, Def. 82] (mCFSM). Las mCFSM son una

versión extendida de las CFSM con múltiples canales entre cada par de participantes y la interfaz de comunicación de los AFCA se proyecta como una mCFSM. Se definen, del mismo modo que una CFSM sobre $\mathcal{M}$ de la siguiente manera.

Definición 30 (Multichannel Communicating Finite State Machine - 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 CS 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 CS 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.

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

Definición 31 (Interfaz de comunicación de un AFCA)**.** *Es la mCFSM que resulta de aplicarle el siguiente procedimiento:*

1. *Transformamos toda acción interna y de buffer en una transición ϵ. Esto resulta en un autómata no determinístico con transiciones ϵ al que llamamos A'*
2. *$\forall q \in Q_{A'}$ llamamos $[q]$ a la clase de equivalencia tal que $[q] = \{q' \in Q_{A'} \mid q \xrightarrow{\epsilon^*} q'\}$. Decimos que dos clases $[q]$ y $[q']$ están relacionadas $[q] \xrightarrow{\sigma} [q'']$, con $\sigma \in \Sigma_{Ex}$, sii $\exists q^1 \in [q], q^2 \in [q'']$ tales que $q^1 \xrightarrow{\sigma} q^2 \in \delta$*
3. *$[Q] = \{[q] \mid q, q' \in Q \wedge [q] \xrightarrow{\sigma} [q'] \vee [q'] \xrightarrow{\sigma} [q]\}$ es el conjunto de clases de equivalencia de Q que están relacionadas.*
4. *$[q_0] = q_0$*
5. *$[\delta] = \{[q] \xrightarrow{\sigma} [q'] \mid [q], [q'] \in [Q] \wedge \sigma \in \Sigma_{EX}\}$, dos estados $[q], [q'] \in [Q]$ están relacionados en $[\delta]$ si son clases de equivalencia relaicionadas.*

Ejemplo 6 (Interfaz de comunicación de un AFCA)**.** *Las siguientes mCFSMs son el resultado de proyectar la interfaz de comunicación (Def. 6) de los AFCAs de la Fig. 3.1.1, donde M_1, M_2 y M_3 representan las interfaces de comunicación externa de $\mathcal{A}_1, \mathcal{A}_1$ y $\mathcal{A}_3$, respectivamente.*

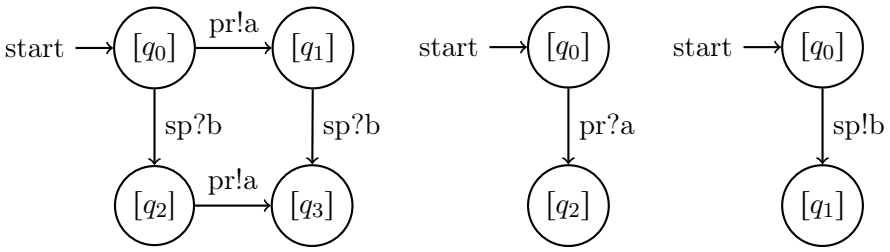


Fig. 3.2.1: mCFSM M_P, M_R y M_S, correspondientes a los AFCAs del Subfig. 3.1.1.

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 CS 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 otros dos 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. El aspecto clave de esta emulación es que preserva el orden de los mensajes.

Definición 32 (Sistema emulado, [Vis18], Def. 84)**.** *Dado un multichannel CS* $(M_p)_{p\in\mathcal{P}}$*, 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 '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 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.

□

4. EQUIVALENCIA DE COMPORTAMIENTO COMUNICACIONAL

Esta sección va a estar enfocada en la definición de composición total de autómatas y la noción de que dado un conjunto $\{\mathcal{A}_i\}_{i\in I}$ de AFCA weak determinate sin transiciones de comunicación interna, la semántica de la composición es equivalente a la del conjunto de mCFSM proyectadas de los elementos de $\{\mathcal{A}_i\}_{i\in I}$.

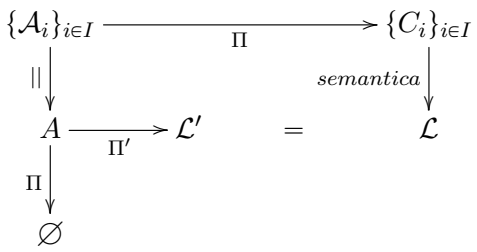
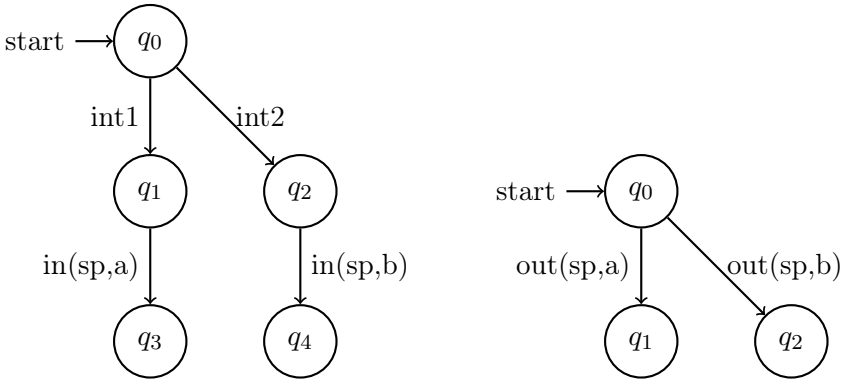
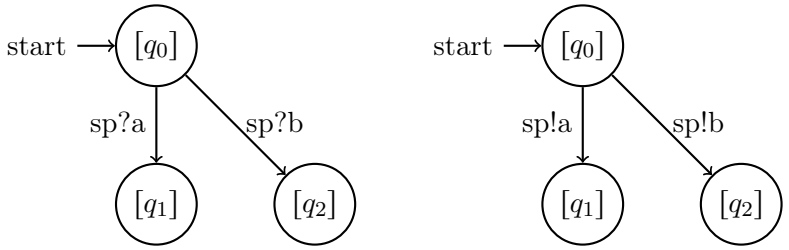


Fig. 4.0.1: La composición de componentes asincrónicas preserva la semántica asincrónica de su interfaz de comunicación.

En general, cuando se trata de formalismos que modelan simultáneamente aspectos del comportamiento interno de una componente y elementos de interfaz observable por otras componentes, la proyección de dicha interfaz no resulta bisimilar con el comportamiento real de la componente. A continuación, en la Fig. 4.0.2, se muestra un ejemplo de este fenómeno recurriendo a la clase de autómatas presentada en la Def. 23, tomando como comportamiento interno las transiciones internas y de buffer, y como comportamiento observable las acciones de comunicación, tanto de entrada como de salida.



(a) Automatas Finitos de Comunicación Asincrónica S y P



(b) Las mCFSM M_S y M_P, correspondientes a los autómatas S y P, respectivamente

Fig. 4.0.2: Sistema distribuido de dos componentes.

En la Fig. 4.0.2 podemos ver un sistema distribuido de dos componentes que a primera vista funciona correctamente pero visto en detalle encontramos que tiene errores en su comportamiento de comunicación y puede entrar en deadlock. Cada componente, expresado en la Fig. 4.0.2a por los autómatas S y P podemos ver que tiene sus respectivas acciones de comunicación externa y transiciones internas. En su estado inicial P puede enviar tanto a como b, indistintamente, pero observando S podemos ver que puede haber deadlock. Esto se debe a que S tiene que ejecutar una transición interna para llegar a un estado donde pueda recibir un mensaje. Dependiendo de qué transición ejecute, es un mensaje u otro. Si ejecuta $int1$, solo puede recibir a y si P envía b el sistema entra en deadlock. Del mismo modo si toma el camino de $int2$ y recibe a, no puede seguir ejecutando porque espera el mensaje b. Pero en la Fig. 4.0.2b se ve como las mCFSM proyectadas funcionan como un sistema, ya que en su estado incial ambas máquinas están en la misma posición. Es decir, M_P en $[q_0]$ puede enviar tanto a como b, y M_S puede ejecutar correctamente al recibir el mensaje.

Para prevenir este tipo de problemas surge la noción de determinismo débil o *weak determinacy* [Mil89, Chap. 11, Def. 3]. La noción de determinismo es importante dado que está relacionada con la predecibilidad del sistema. Si realizamos el mismo experimento dos veces sobre un sistema determinista, comenzando ambas veces del estado inicial, esperamos obtener el mismo resultado o comportamiento cada vez. Además, la predecibilidad es algo que frecuentemente requerimos de sistemas. Por ende si elegimos especificar un sistema S dándole un participante abstracto *Spec* al que S debería ser equivalente, entonces *Spec* va a ser determinista. *Weak Determinacy* es una forma más amplia de determinismo que

nos permite contemplar una equivalencia observacional entre participantes. Esta equivalencia basada en la definición de bisimulación tiene en cuenta que para un participante externo, las transiciones ϵ son invisibles y por lo tanto una serie de estados conectados por transiciones consecutivas de este tipo serían bisimilares. En este caso tomamos las transiciones internas o de comunicación interna como observacionalmente bisimilares para un participante externo.

Definición 33 (Weak Determinacy)**.** *Sea $A = \langle Q, \mathcal{C}, B, \Sigma, \delta, q_0, F\rangle$ un AFCA y $\Sigma' \subseteq \Sigma$ un conjunto de etiquetas a silenciar. Se dice que A es* weak determinate sobre Σ' *si para todo $q, q', q'' \in Q$ y para todo $s, s' \in \Sigma^*$ tal que $\widehat{s}_{\Sigma'} = \widehat{s'}_{\Sigma'}$, $q \xrightarrow{s} q'$ y $q \xrightarrow{s'} q''$, $q' \sim_{\Sigma'} q''$.*

Proposición 2. *Dado un AFCA $A = \langle Q, \mathcal{C}, B, \Sigma, \delta, q_0, F\rangle$ weak determinate tal que $\Sigma_{Int} \subseteq \Sigma$ denota el conjunto de transiciones internas, entonces dados dos estados $q, q' \in Q$ tales que $q \xrightarrow{\sigma} q'$ con $\sigma \in \Sigma^*_{Int}$, $q \sim_{\Sigma_{Int}} q'$.*

Demostración. Supongamos que A es weak determinate sobre Σ_{Int} y existen $q, q' \in Q$ tales que $q \xrightarrow{\sigma} q'$ con $\sigma \in \Sigma^*_{Int}$ y no vale que $q \sim_{\Sigma_{Int}} q'$. Si $q \sim_{\Sigma_{Int}} q'$ no vale, entonces no existen $q'' \in Q$, $s, s' \in \Sigma^*$ tal que $\widehat{s}_{\Sigma'} = \widehat{s'}_{\Sigma'}$, $q \xrightarrow{s} q'$ y $q \xrightarrow{s'} q''$, lo que es absurdo pues se puede tomar $q'' = q$, $s = \epsilon$ y $s' = s\sigma$ satisfaciendo que $\widehat{s}_{\Sigma_{Int}} = \widehat{s'}_{\Sigma_{Int}}$ □

Este lenguaje requiere de determinadas condiciones para asegurar que un sistema distribuido puedan funcionar correctamente. Una vez asegurado eso queremos ver que la composición total de un conjunto de componentes preserva correctamente el comportamiento comunicacional de los mismos. Para esto primero establecemos una propiedad de equivalencia sobre estados de un autómata que permite abstraerse de las acciones internas sin alterar la comunicación. Luego definimos un LTS que resulta de proyectar la semántica de comunicación de un AFCA utilizando la propiedad anterior sobre las transiciones internas. Por último definimos y demostramos.

Definición 34 (Equivalencia de estados por transición interna)**.** *Dado un AFCA $A = \langle Q, \mathcal{C}, B, \Sigma, \delta, q_0, F\rangle$ weak determinate tal que $\Sigma_{Int} \subseteq \Sigma$ denota el conjunto de transiciones internas, definimos $q \Rightarrow q'$ si y solo si existe $\sigma \in \Sigma_{Int}$ tal que $q \xrightarrow{\sigma} q'$. Luego,* $[q]_m$, la clase de equivalencia por comunicación interna *de q, se define como $[q]_m = \{q' \mid q \Rightarrow^\bullet q'\}$.*

Intuitivamente, la clase de equivalencia por comunicación interna de un estado q, $[q]_m$ son aquellos estados q' relacionados en cero o más pasos de transiciones internas con q.
Dado un AFCA $A = \langle Q, \mathcal{C}, B, \Sigma, \delta, q_0, F\rangle$, $\Sigma = \Sigma_{Int} \cup \Sigma_{Buff}$ y $q \in Q$, . Decimos que dos clases de equivalencia están relacionadas $[q]_m \xrightarrow{\sigma} [q']_m$ sii $\exists q_i \in [q]_m, q'_i \in [q']_m$ tales que $q_i \xrightarrow{\sigma} q'_i \in \delta_{\text{Buff}}$.

Definición 35 (Proyección de la comunicación de un AFCA)**.**
Dado un AFCA weak determinate $\mathcal{A} = \langle Q, B, \mathcal{C}, \Sigma, \delta, q_0, F\rangle$, tal que su interfaz de comunicación externa es la mCFSM vacía, es decir $\Pi(\mathcal{A}) = \varnothing$, definimos $\mathcal{T}(\mathcal{A})$ como la proyección de la semántica de comunicación de $\mathcal{A}$. Llamamos $\widehat{\mathcal{A}}$ al sistema de transición etiquetado resultante y decimos que $\mathcal{T}(\langle Q, B, \mathcal{C}, \Sigma, q_0, \delta, F\rangle) = \langle \widehat{Q}, B, \widehat{q_0}, \widehat{\Delta}\rangle$ tal que:

- $\widehat{Q} = \{\langle [q]_m, \overrightarrow{\Omega}\rangle | q \in Q \wedge \overrightarrow{\Omega} = (\Omega_b)_{b \in B}$ *con* $\Omega_b \in \mathcal{M}^*\}$ *Cada estado del sistema de transición resultante es una configuración de $\mathcal{A}$ para una clase de equivalencia por comunicación interna,*

- *B es el conjunto de buffers de $\mathcal{A}$ que se utilizan en las aristas del LTS para denotar de comunicación interna*
- *$\widehat{q_0} = \langle q_0, \Omega \rangle$ es el estado inicial de donde q_0 es el estado inicial de $\mathcal{A}$ y $\Omega = \langle [], \ldots, [] \rangle$ es el conjunto de buffers en su estado inicial sin mensajes, y*
- *$\widehat{\Delta} = \{\langle \langle q_i, \Omega \rangle, \widehat{\sigma}, \langle q_j, \Omega' \rangle \rangle | q_i, q_j \in Q, \widehat{\sigma} \in \{B \times \{\gg, \ll\} \times \mathcal{M}\} \wedge \exists \sigma \in \Sigma$ tal que $[q_i]_m \xrightarrow{\sigma} [q_j]_m\}$. Definimos $\widehat{\Delta}$ como la relación de transición de $\widehat{\mathcal{A}}$. Un estado de $\widehat{q_j}$ es alcanzable desde otro $\widehat{q_i}$ si la clase de equivalencia $[q_j]_m$ es alcanzable desde $[q_i]_{\mathcal{M}}$.*

Partimos de la hipótesis que para poder extraer el LTS de semántica de comunicación interna es necesario que el autómata no tenga transiciones de comunicación externa. Esto es debido a que la comunicación asincrónica externa introduce otra complejidad a tener en cuenta a la hora de abstraernos correctamente de las transiciones internas. Para los propósitos de lo que queremos demostrar estas condiciones son suficientes dado que trabajamos sobre AFCA que resultan de la composición total de un conjunto de autómatas.

A continuación vemos un ejemplo de la Proyección de Comunicación según Def. 35. Dados los AFCA $\mathcal{A}_1, \mathcal{A}_2$ y $\mathcal{A}_3$ (Fig. 3.1.1), y su composición $\mathcal{A}$ (Fig. 3.1.2), obtenemos el LTS resultante $\hat{\mathcal{A}}$ (Fig. 4.0.3). En la figura siguiente (Fig. 4.0.4) vemos otro LTS, en este caso representando al CS correspondiente al conjunto de mCFSMs según Def. 6 (Fig. 3.2.1) que se obtienen de los autómatas anteriores. Comparando ambos sistemas de transición M y $\hat{\mathcal{A}}$, se puede ver que son muy similares. Los nodos de cada LTS muestran distintos estados de la comunicación, incluyendo el contenido de los canales de comunicación, externos en el caso de M e internalizados como buffers para $\hat{\mathcal{A}}$, que en ambos casos es el mismo en cada paso. Las aristas de M y $\hat{\mathcal{A}}$ denotan las mismas acciones comunicación también como externas e internas, respectivamente. Este ejemplo es una intuición de que el mecanismo de composición preserva la semántica de comunicación externa al internalizarla.

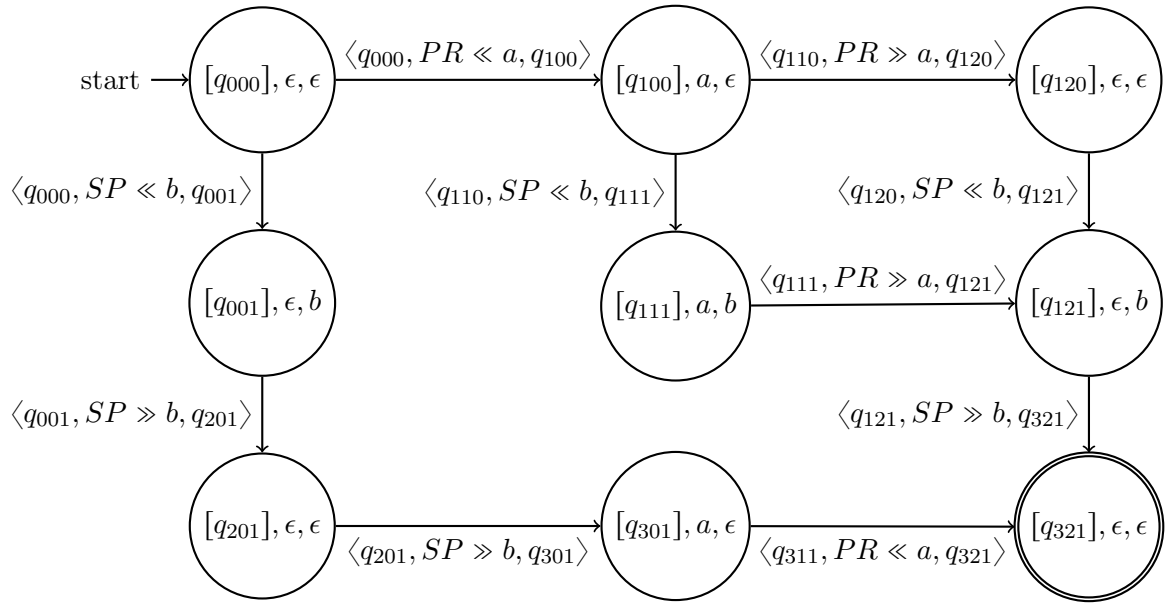


Fig. 4.0.3: Proyección de comunicación interna del AFCA de la Fig. 3.1.2.

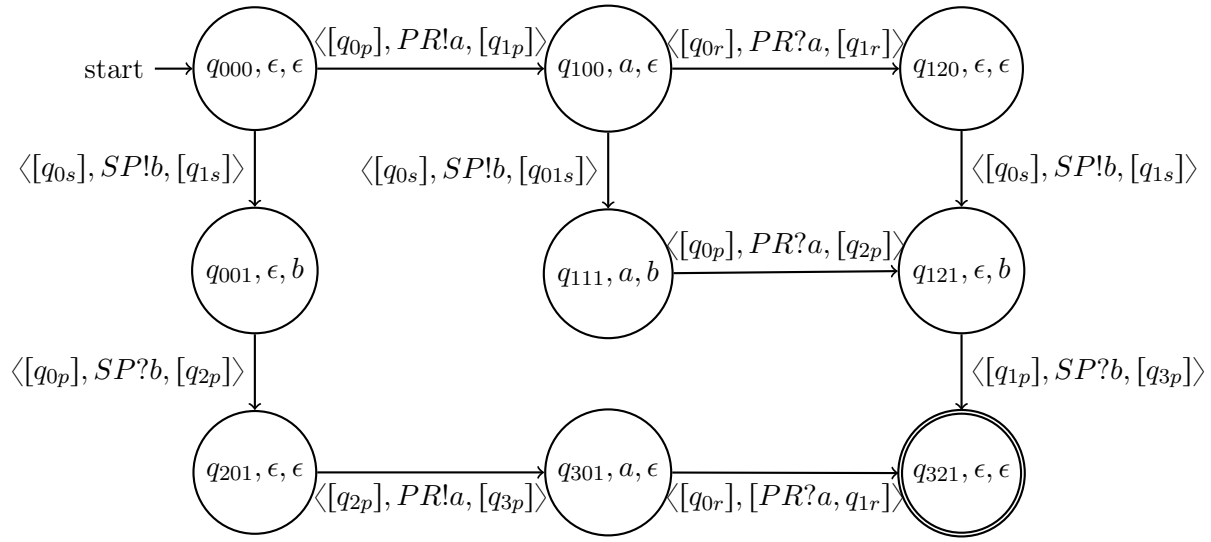


Fig. 4.0.4: LTS del CS formado por las mCFSMs de la Fig. 3.2.1.

Para demostrar la equivalencia semántica de comunicación entre un conjunto de AFCAs compatibles y sus mCFSMs queremos probar que los LTS que representan las respectivas semánticas son bisimilares. Antes de demostrar esto necesitamos demostrar que los estados son equivalentes. Los estados de ambos LTS contienen un estado discreto compuesto y el conjunto de buffers/canales con su contenido en cada estado. En el Lema 1 demostramos que los estados discretos contenidos en los estados de la proyección de comunicación interna son equivalentes a los del LTS que representa el CS, mientras que en el Lema 2 probamos que el comportamiento y el contenido de los buffers es equivalente al de los canales.

Lema 1 (Equivalencia de estados discretos)**.** *Sea $\widehat{\mathcal{A}} = \langle \widehat{Q}, B, \widehat{q_0}, \widehat{\delta} \rangle$, la proyección de comunicación interna del AFCA $\mathcal{A}$ que surge de la composición del conjunto de autómatas $\{\mathcal{A}_i\}_{1 \leqslant i \leqslant n}$ y $M = \langle [Q], \mathcal{C}, [q_0], [\delta] \rangle$, el LTS que representa la semántica del CS correspondiente a $\{M_i\}_{1 \leqslant i \leqslant n}$, el conjunto de mCFSM de $\{\mathcal{A}\}_{1 \leqslant i \leqslant n}$. Para todo $\langle [\overrightarrow{q}]_m, \overrightarrow{\Omega} \rangle \in \widehat{Q}$, existe $\langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}} \rangle \in P$ tal que $q_i \in \overrightarrow{q} \iff q_i \in \overrightarrow{p}$ y viceversa.*

Demostración. $\implies$ Dado $\widehat{q} \in \widehat{Q}$, un estado de la proyección $\widehat{\mathcal{A}}$. Por definición de $\widehat{\mathcal{A}}$, $\widehat{q} = \langle [\overrightarrow{q}]_m, \overrightarrow{\Omega} \rangle$. Por definición de equivalencia por comunicación interna (ver Def. 34) $[\overrightarrow{q}]_m = \left\{ \overrightarrow{q'} \mid \overrightarrow{q} \xrightarrow{\sigma} \overrightarrow{q'} \right\}$ con $\overrightarrow{q}, \overrightarrow{q'} \in Q$ y $\sigma \in \Sigma^*_{Int}$ o sea son los estados alcanzables por transiciones internas desde $\overrightarrow{q}$. Por lo tanto $\langle [\overrightarrow{q}]_m, \overrightarrow{\Omega} \rangle \in \widehat{Q}$ con $\overrightarrow{q} \in Q$. Por definición de composición de AFCA (ver Def. 29) $\overrightarrow{q} = \langle q_1, \ldots, q_i, \ldots, q_n \rangle$ donde $q_i \in Q_i$ es un estado del AFCA $\mathcal{A}_i$. Como el conjunto de AFCA $\{\mathcal{A}_i\}_{1 \leqslant i \leqslant n}$ no posee comunicación interna, la comunicación interna de $\mathcal{A}$ es la comunicación externa de sus componentes internalizada a través de la composición. Por lo tanto, q_i es un estado alcanzable por una sucesión de transiciones internas y externas de $\mathcal{A}_i$. Entonces $\overrightarrow{q} \in Q$ si y solo si $q_i \in Q_i$, para todo $1 \leqslant i \leqslant n$, y por definición de interfaz de comunicación de AFCA (ver Def. 31), existe $[q_i] \in [Q_i]$, el conjunto de estados de la mCFSM M_i. Por lo tanto, por definición de CS (ver Def. 8), existe $\overrightarrow{p} = \langle [q_1], \ldots, [q_i], \ldots, [q_n] \rangle$ tal que, para algún $\overrightarrow{\Omega_{\mathcal{C}}}$, $\langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}} \rangle \in P$.

$\Longleftarrow$ Sea $p \in P$, un estado del LTS M que representa la semántica el CS. Por definición de CS (ver Def. 8), vale $p = \langle\langle [q_1], \ldots, [q_i], \ldots, [q_n]\rangle, \overrightarrow{\Omega}\rangle \in P$ con $[q_i] \in [Q_i]$, para todo $1 \leqslant i \leqslant n$, donde $[Q_i]$ es el conjunto de estados de la mCFSM M_i. Sea $\mathcal{A}_i$ el AFCA cuya interfaz de comunicación es M_i (ver Def. 31) y cuyo conjunto de estados es Q_i, para todo $1 \leqslant i \leqslant n$. Por definición de interfaz de comunicación (ver Def. 31), existen estados $\widehat{q}_i \in Q_i$ alcanzables en cada componente tal que $\widehat{q}_i \in [q_i]$, para todo $1 \leqslant i \leqslant n$. Por definición de composición (ver Def. 29), $\overrightarrow{q} = \langle \widehat{q}_1, \widehat{q}_2, \ldots, \widehat{q}_i, \ldots, \widehat{q}_n\rangle \in Q$. $\overrightarrow{q}$ es alcanzable en $\mathcal{A}$ y, por equivalencia de comunicación interna (ver Def. 34), su clase de comunicación interna $[\overrightarrow{q}]_m$. Entonces, por proyección de comunicación interna (ver Def. 35), existe $\widehat{q} = \langle [\overrightarrow{q}]_m, \overrightarrow{\Omega}\rangle \in \widehat{Q}$, para algún $\overrightarrow{\Omega}$.

□

Lema 2 (Equivalencia entre buffers y canales)**.** *Sean $\widehat{\mathcal{A}} = \langle \widehat{Q}, B, \widehat{q}_0, \widehat{\delta}\rangle$, la proyección de comunicación interna del AFCA $\mathcal{A}$ que surge de la composición del conjunto de autómatas $\{\mathcal{A}_i\}_{1\leqslant i\leqslant n}$ y $M = \langle P, \mathcal{C}, p_0, [\delta]\rangle$, el LTS que representa la semántica del CS correspondiente a $\{M_i\}_{1\leqslant i\leqslant n}$, el conjunto de mCFSM de $\{\mathcal{A}_i\}_{1\leqslant i\leqslant n}$. Dados $\langle [\overrightarrow{q}]_m, \overrightarrow{\Omega}\rangle \in \widehat{Q}$ y $\langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}}\rangle \in P$. Queremos ver que el conjunto de buffers B de $\mathcal{A}$ es el conjunto de canales internalizados de $\mathcal{C}$ de M. Es decir que para todo canal ω vale que $\omega \in \mathcal{C}$ si y solo si $\omega \in B$.*

Demostración. Sea $\langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}}\rangle$ un estado del LTS M. Por definición de CS (ver Def. 8), $\overrightarrow{\Omega_{\mathcal{C}}}$ es el conjunto de buffers asociados a los canales $\mathcal{C}$, con su contenido en el estado discreto $\overrightarrow{p}$. Estos canales corresponden a las mCFSM $\{M_i\}_{1\leqslant i\leqslant n}$ que componen el CS. Entonces vale $\omega \in \overrightarrow{\Omega_{\mathcal{C}}}$ si y solo si $\omega \in \mathcal{C}_i$ con $1 \leqslant i \leqslant n$. Sea $\mathcal{A}_i$ el AFCA tal que M_i es su interfaz de comunicación. Por definición de interfaz de comunicación de AFCA (ver Def. 31), $\omega \in \mathcal{C}_i$ si y solo si $\omega \in \mathcal{C}_{\mathcal{A}_i}$. Por definición de composición de AFCA (ver Def. 29), la comunicación externa se internaliza. Entonces como $\mathcal{A}_i$ no tienen buffers ni comunicación interna, vale $\omega \in \mathcal{C}_{\mathcal{A}_i}$ si y solo si $\omega \in B$ donde B es el conjunto de buffers de comunicación interna de $\mathcal{A}$. Luego, por definición de proyección de comunicación interna (ver Def. 35), el conjunto de buffers de $\mathcal{A}$ es el mismo que en $\widehat{\mathcal{A}}$. Por lo tanto $\omega \in B$ si y solo si $\exists \langle [\overrightarrow{q}]_m, \overrightarrow{\Omega}\rangle \in \widehat{Q}$ tal que $\omega \in \overrightarrow{\Omega}$. □

Teorema 2 (Equivalencia de las semánticas de las mFCSMs y los AFCAs)**.** *Sea $\{\mathcal{A}\}_{1\leqslant i\leqslant n}$ un conjunto de weak determinate AFCA sin transiciones de comunicación interna, tal que su composición $\mathcal{A} = \langle Q, B, \mathcal{C}, \Sigma, \delta, q_0, F\rangle$, también lo es. Llamamos $\widehat{\mathcal{A}} = \langle \widehat{Q}, B, \widehat{q}_0, \widehat{\delta}\rangle$ al sistema de transición etiquetado (LTS) que resulta de la proyección $\mathcal{T}(\mathcal{A})$. Sea el conjunto $\{M_i\}_{1\leqslant i\leqslant n}$ de mCFSM correspondientes a cada uno de los AFCAs en $\{\mathcal{A}\}_{1\leqslant i\leqslant n}$ tal que $M_i = \langle [Q_i], \mathcal{C}, [q_{0_i}], [\delta_i]\rangle$. Llamamos $M = \langle P, \mathcal{C}, p_0, [\delta]\rangle$ al LTS que representa la semántica del CS. Luego $\widehat{\mathcal{A}}$ y M son bisimilares.*

Demostración. Queremos probar que la relación $R \subseteq \widehat{Q} \times P$, definida de la siguiente manera $R = \{\langle\langle [\overrightarrow{q}]_m, \overrightarrow{\Omega}\rangle, \langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}}\rangle\rangle \mid [\overrightarrow{q}]_m R' \overrightarrow{p} \wedge \overrightarrow{\Omega} = \overrightarrow{\Omega_{\mathcal{C}}}\}$ es una bisimulación, con $R' = \{\langle [\overrightarrow{q}]_m, \overrightarrow{p}\rangle \mid q_i \in \overrightarrow{q} \iff q_i \in \overrightarrow{p}\}$.
Es decir que para cada par de elementos $\widehat{q} \in \widehat{Q}$, $p \in P$ para los que vale $\langle \widehat{q}, p\rangle \in R$, queremos probar que:

- $\forall \widehat{\sigma} \in \widehat{\Sigma} | \widehat{q} \xrightarrow{\widehat{\sigma}} \widehat{q}' \implies \exists p' \in P$ tal que $p \xrightarrow{\sigma_M} p' \wedge \langle \widehat{q}', p', \rangle \in R$, y simétricamente
- $\forall \sigma_M \in \Sigma_M | p \xrightarrow{\sigma_M} p \implies \exists \widehat{q}' \in \widehat{Q}$ tal que $\widehat{q} \xrightarrow{\widehat{\sigma}} \widehat{q}' \wedge \langle \widehat{q}', p'\rangle \in R$.

Donde $\widehat{\Sigma} \subseteq B \times \{\gg, \ll\} \times \mathcal{M}$, $\Sigma_M \subseteq \mathcal{A} \times \{!, ?\} \times \mathcal{M}$, el conjunto de buffers B son los canales internalizados del conjunto $\mathcal{C}$ y si $\widehat{\sigma} = (a_k a_j)_l \gg m$ entonces $\sigma_M = (a_k a_j)_l ?m$, si $\widehat{\sigma} = (a_k a_j)_l \ll m$ entonces $\sigma_M = (a_k a_j)_l !m$.
Tomamos inicialmente como etiquetas $\widehat{\sigma} = (a_k a_j)_l \gg m$ y $\sigma_M = (a_k a_j)_l ?m$,
Primero demostraremos la condición $\forall (a_k a_j)_l \gg m \in \widehat{\Sigma} | \widehat{q} \xrightarrow{(a_k a_j)_l \gg m} \widehat{q}' \implies \exists p' \in P$ tal que $p \xrightarrow{(a_k a_j)_l ?m} p' \wedge \langle \widehat{q}', p', \rangle \in R$.

- Dado el par $\langle [\overrightarrow{q}]_m, \overrightarrow{\Omega} \rangle \in \widehat{Q}, \langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}} \rangle \in P$ tal que $\langle \langle [\overrightarrow{q}]_m, \overrightarrow{\Omega} \rangle, \langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}} \rangle \rangle \in R$. Sean $(a_k a_j)_l \gg m \in \widehat{\Sigma}$ y $\langle [\overrightarrow{q}']_m, \overrightarrow{\Omega}' \rangle \in \widehat{Q}$ tales que $\langle [\overrightarrow{q}]_m, \overrightarrow{\Omega} \rangle \xrightarrow{(a_k a_j)_l \gg m} \langle [\overrightarrow{q}']_m, \overrightarrow{\Omega}' \rangle \in \widehat{\Delta}$. Esto vale si y solo si la clase de equivalencia por comunicación interna $[\overrightarrow{q}']_m$ está relacionada con $[\overrightarrow{q}]_m$ vía $(a_k a_j)_l \gg m$ (ver Def. 34), o sea $[\overrightarrow{q}]_m \xrightarrow{(a_k a_j)_l \gg m} [\overrightarrow{q}']_m$. Entonces $\overrightarrow{\Omega} = [(a_1 a_2), \ldots, (a_k a_j)_l \cdot m \ldots (a_n a_m)]$ y $\overrightarrow{\Omega}' = [(a_1 a_2), \ldots, (a_k a_j)_l \ldots (a_n a_m)] \rangle$.

- Sean $\overrightarrow{q_\sigma} \in [\overrightarrow{q}]_m, \overrightarrow{q_\sigma}' \in [\overrightarrow{q}']_m$, por definición de proyección de comunicación interna (ver Def. 35) vale $\overrightarrow{q_\sigma} \xrightarrow{(a_k a_j)_l \gg m} \overrightarrow{q_\sigma}' \in \delta$. Por definición de composición sabemos que vale $\overrightarrow{q_\sigma} = \langle q_0, q_1, \ldots, q_j, \ldots, q_n \rangle$ y $\overrightarrow{q_\sigma}' = \langle q_0, q_1, \ldots, q_j', \ldots, q_n \rangle$ donde q_j, q_j' son estados de $\mathcal{A}_j$, componente de $\mathcal{A}$. Por definición de AFCA (ver Def. 23), $(a_k a_j)_l \gg m$ es una operación de desencolamiento del mensaje m sobre el buffer $(a_k a_j)_l$. Por definición de composición de AFCA (ver Def. 29), como los AFCA del conjunto no tienen acciones de comunicación interna, esta acción proviene de una recepción de un mensaje por parte de $\mathcal{A}_j$ vía el canal $(a_k a_j)_l$. Entonces podemos afirmar que se cumple $\overrightarrow{q_\sigma} \xrightarrow{(a_k a_j)_l \gg m} \overrightarrow{q_\sigma}' \in \delta$ si y solo si $q_j, q_j' \in Q_j \wedge q_j \xrightarrow{(a_k a_j)_l ?m} q_j' \in \delta_j$.

- Por interfaz de comunicación de AFCA (ver Def. 31) podemos obtener M_j, la mCFSM correspondiente a $\mathcal{A}_j$, y sabemos que $q_j \xrightarrow{(a_k a_j)_l ?m} q_j' \in \delta_j$ si y solo si se cumple que $[q_j], [q_j'] \in [Q_j]$, y $[q_j] \xrightarrow{(a_k a_j)_l ?m} [q_j'] \in [\delta_j]$.

- Entonces, por definición del LTS M (ver Def. 8), existen $p' = \langle \overrightarrow{p'}, \overrightarrow{\Omega_{\mathcal{C}}} \rangle \in P, (a_k a_j)_l ?m \in \Sigma_M$ tal que $\overrightarrow{p'} = \langle [q_1], [q_2], \ldots, [q_j'], \ldots [q_n] \rangle, \overrightarrow{\Omega_{\mathcal{C}}'} = [\ldots, (a_k a_j)_l, \ldots]$, y $p \xrightarrow{(a_k a_j)_l ?m} p'$. Donde $\overrightarrow{\Omega_{\mathcal{C}}}$ y $\overrightarrow{\Omega_{\mathcal{C}}}'$ son los canales de comunicación externa para los estados $\overrightarrow{p}$ y $\overrightarrow{p'}$, y vale $\overrightarrow{\Omega_{\mathcal{C}}} = [(a_1 a_2), \ldots, (a_k a_j)_l \cdot m \ldots (a_n a_m)]$ y $\overrightarrow{\Omega_{\mathcal{C}}}' = [(a_1 a_2), \ldots, (a_k a_j)_l \ldots (a_n a_m)] \rangle$. Por lo tanto vale $\langle \widehat{q'}, p' \rangle \in R$.

- Por lo tanto vale que $\forall (a_k a_j)_l \gg m \in \widehat{\Sigma} | \widehat{q} \xrightarrow{(a_k a_j)_l \gg m} \widehat{q}' \implies \exists p' \in P$ tal que $p' \xrightarrow{(a_k a_j)_l ?m} p' \wedge \langle \widehat{q}', p', \rangle \in R$ donde $\widehat{\Sigma} \subseteq B \times \{\gg, \ll\} \times \mathcal{M}$

Demostraremos ahora la condición simétrica $\forall (a_k a_j)_l ?m \in \Sigma_M | p \xrightarrow{(a_k a_j)_l ?m} p' \implies \exists \widehat{q}' \in \widehat{Q}$ tal que $\widehat{q} \xrightarrow{(a_k a_j)_l \gg m} \widehat{q}' \wedge \langle \widehat{q}', p' \rangle \in R$

- Dado el par $\langle q, p \rangle = [\overrightarrow{q}]_m, \overrightarrow{\Omega} \rangle \in \widehat{Q}, \langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}} \rangle \in P$ tal que $\langle \langle [\overrightarrow{q}]_m, \overrightarrow{\Omega} \rangle, \langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}} \rangle \rangle \in R$. Sean $(a_k a_j)_l ?m$ y $p' = \langle \overrightarrow{p}', \overrightarrow{\Omega_{\mathcal{C}}}' \rangle \in P$ tales que $\langle \overrightarrow{p}, \overrightarrow{\Omega_{\mathcal{C}}} \rangle \xrightarrow{(a_k a_j)_l ?m} \langle \overrightarrow{p}', \overrightarrow{\Omega_{\mathcal{C}}}' \rangle \in \delta_P$. Por definición del LTS M (ver Def. 8), $(a_k a_j)_l ?m$ es una operación sobre el canal $(a_k a_j)_t$ de la mCFSM M_j, componente del CS. Dado que el LTS M representa la semántica del CS compuesto por el conjunto de mCFSMs, vale que $\overrightarrow{p} =$

$\langle [q_1], [q_2], \ldots [q_j], \ldots [q_n] \rangle$, $\overrightarrow{p}' = \langle [q_1], [q_2], \ldots, [q'_j], \ldots [q_n] \rangle$, donde $[q_j]$ y $[q'_j]$ son estados de la mCFSM M_j, y vale $[q_j] \xrightarrow{(a_k a_j)_l ? m} [q'_j] \in [\delta]_j$

- Por definición de intefaz de comunicación de AFCA (ver Def.31) sabemos que M_J es la mCFSM correspondiente a $\mathcal{A}_j$, componente de $\mathcal{A}$. Por el Lema 1, como $[q_j] \xrightarrow{(a_k a_j)_l ? m} [q'_j] \in [\delta_j]$ entonces $q_j \xrightarrow{(a_k a_j)_l ? m} q'_j \in \delta_j$.
- Por definición de composición de AFCA (ver Def. 29) q_j y q'_j son componentes de dos estados compuestos que llamamos $\overrightarrow{q}_\sigma$ y $\overrightarrow{q}'_\sigma$, respectivamente. De modo tal que $\overrightarrow{q}_\sigma \xrightarrow{(a_k a_j)_l \gg m} \overrightarrow{q}'_\sigma \in \delta$. Donde $(a_k a_j)_l$ es el canal internalizado en forma de buffer.
- Por definición de equivalencia por comunicación interna (ver Def. 34) como vale $\overrightarrow{q}_\sigma \xrightarrow{(a_k a_j)_l \gg m} \overrightarrow{q}'_\sigma \in \delta$, entonces existen las clases de equivalencia $[\overrightarrow{q}]_m$, $[\overrightarrow{q}']_m$, tales que $\overrightarrow{q}_\sigma \in [\overrightarrow{q}]_m$, $\overrightarrow{q}'_\sigma \in [\overrightarrow{q}']_m$, y $[\overrightarrow{q}]_m \xrightarrow{(a_k a_j)_l \gg m} [\overrightarrow{q}']_m$. Por proyección de comunicación interna (ver Def. 35), existen $\widehat{q}' = \langle [\overrightarrow{q}']_m, \overrightarrow{\Omega}' \rangle \in \widehat{Q}, (a_k a_j)_l \gg m \in \widehat{\Sigma}$, tales que $\langle [\overrightarrow{q}]_m, \overrightarrow{\Omega} \rangle \xrightarrow{(a_k a_j)_l \gg m} \langle [\overrightarrow{q}']_m, \overrightarrow{\Omega}' \rangle$. Donde, $\overrightarrow{\Omega}$, y $\overrightarrow{\Omega}'$ son los buffers de $\mathcal{A}$ para los estados $\overrightarrow{q}_\sigma$ y $\overrightarrow{q}'_\sigma$, respectivamente, y se cumple que $\overrightarrow{\Omega} = [(a_1 a_2), \ldots, (a_k a_j)_l \cdot m \ldots (a_n a_m)]$ y $\overrightarrow{\Omega}' = [(a_1 a_2), \ldots, (a_k a_j)_l \ldots (a_n a_m)] \rangle$. Por lo tanto $\langle \widehat{q}', p' \rangle \in R$
- Por lo tanto vale que $\forall (a_k a_j)_l ? m \in \Sigma_M | p \xrightarrow{(a_k a_j)_l ? m} p \implies \exists \widehat{q}' \in \widehat{Q}$ tal que $\widehat{q} \xrightarrow{(a_k a_j)_l \gg m} \widehat{q}' \wedge \langle \widehat{q}', p' \rangle \in R$ donde $\Sigma_M \subseteq \mathcal{C} \times \{!, ?\} \times \mathcal{M}$

□

5. CONCLUSIONES, TRABAJO RELACIONADO Y TRABAJO FUTURO

En esta sección resumimos las contribuciones de esta tesis junto con conclusiones, trabajos relacionados que nos parecen de relevancia y posibles futuras líneas de investigación.

5.1. Conclusiones

El objetivo de esta tesis era el estudio y posible extensión de los modelos formales para servicios, dado que consideramos que son necesarios para la implementación de SOC en toda su capacidad. Partiendo del ideal del paradigma SOC, es decir, un contexto que permita la reconfiguración dinámica de un artefacto de software a partir de posibilitar el binding en tiempo de ejecución, el objetivo principal fue crear un formalismo que soporte binding parcial. Con este objetivo en mente desarrollamos los AFCA que permiten internalizar el envío y la recepción de mensajes como comportamiento interno de una componente resultante de la composición. Dicho mecanismo de composición, además, preserva la comunicación externa de las componentes involucradas.
Para expresar la interfaz externa de comunicación de ese tipo de autómatas propusimos las mCFSMs, una extensión de las CFSMs con múltiples canales entre cada par de participantes. esto facilita la unificación de las interacciones de dos componente a ser compuestas, con una tercera. Como método para garantizar que los distintos participantes puedan interoperar en forma segura extendimos la noción de GMC para mCFSMs. Por último pudimos demostrar la equivalencia entre el comportamiento de la composición de un conjunto de AFCA y el de sus respectivas mCFSMs.
Todos estos elementos proporcionan los fundamentos de un lenguaje formal que provee un mecanismo de composición parcial, en el sentido de lo explicado en la Sec. 3.1.1, y los vincula con resultados conocidos del campo de los sistemas distribuidos que permiten garantizar una interoperación libre de errores.

5.2. Trabajo relacionado

Como se ha dicho anteriormente, la propuesta de la presente tesis es la presentación de un lenguaje que permita describir componentes de un sistema distribuido, con las siguientes particularidades:

- debe posibilitar la convivencia de acciones internas, vinculadas al cómputo que realiza localmente la componente, junto con primitivas de comunicación utilizadas para interactuar con las restantes componentes del sistema, y

- se espera que posea un mecanismo de composición parcial, en relación al conjunto esperado de participantes.

Existe una gran variedad de lenguajes que permiten la formalización de la comunicación entre componentes de un sistema distribuido. Algunos ejemplos de interés en este campo son:

Session types [HVK98, HYM08]: Los *session types* constituyen un cálculo que introduce una nueva noción de tipos para procesos móviles en la que las interacciones, que incluyen múltiples participantes, se abstraen directamente como un escenario global. Un global type cumple el rol de un acuerdo compartido entre pares que se comunican y es la base de *type checking* eficiente a través de su proyección sobre participantes individuales. Algunas propiedades fundamentales de un conjunto bien tipado de participantes son *communication safety*, *progress* y *session fidelity*. Esta noción de acuerdos entre participantes, formalizada a través de los global types, permite modelar las interacciones entre participantes de una comunicación y demostrar las propiedades mencionadas anteriormente.

Global Graphs [CDP12]: Los *Global Graphs* son una superclase de los *Generalized Global Types* de [DY12] que a su vez son una versión de los *multi-party session global types*. Esta versión de *global types* presenta un lenguaje simplificado equipado con semántica basada en trazas de un autómata finito. Los estados de dicho autómata representan estados del sistema distribuido conformado por los participantes de la comunicación y las aristas indican eventos de comunicación (en realidad sólo prescriben el envío del mensaje puesto que la comunicación es asincrónica) de la forma "el participante `a` envía el mensaje `m` al participante `b`".

En el caso de todo ellos, el objetivo central es el de caracterizar formalmente la comunicación entre un conjunto de componentes de forma que se posible garantizar que esta se lleva a cabo en forma correcta. Usualmente se entiendo como correcto la satisfacción de tres condiciones esenciales: 1) **ausencia de deadlock**: si un participante espera un mensaje determinado de otro, este será recibido en algún momento, 2) **ausencia de recepción no especificada**: cuando un participante espera un mensaje determinado no puede ocurrir que este reciba un mensaje diferente, y 3) **ausencia de mensajes huérfanos**: todo mensaje que es enviado, eventualmente es recibido por su destinatario.

Por otro lado, la propia definición de *communicating systems* en cualquiera de estos lenguajes impone el conocimiento de todos los participantes que formarán parte de la comunicación, sea porque son parte de la descripción global de esta, o porque es necesaria la vista local de todos ellos. A continuación resumimos algunos lenguajes que comparten la motivación con esta tesis, la provisión de un lenguaje que permite la descripción de sistemas abiertos en los que no todos los participantes son conocidos proveyendo, entre otras cosas, un mecanismo de composición parcial de componentes.

Interface automata [dH01]: Los sistemas de tipos convencionales especifican interfaces en términos de valores y dominios. Interface Automata es un lenguaje basado en autómatas que captura el comportamiento asumido respecto al orden en el que los métodos de una componente son invocados, y el orden en el que se invocan métodos externos. Similar a las CFSMs esto permite modelar la interacción entre distintas componentes de un sistema. Al igual que los AFCAs estos autómatas modelan no solo comportamiento de la comunicación (a través de acciones de input y output) sino también internas, y pueden componerse dadas ciertas condiciones de compatibilidad. La desventaja respecto a los AFCAs es que las interacciones entre estos autómatas tienen semántica sincrónica, lo que se hace explícito en la composición de dos autómatas donde una acción de salida de un autómata que coincide con la de entrada de otro se transforman en una única acción.

Global Types for Open Systems [BdH18]: Los formalismos basados en *global types* permiten describir el comportamiento general de un sistema distribuido y al mismo tiempo formalizar ciertas propiedades de seguridad para la comunicación entre las componentes del sistema. La visión centralizada de los *global types* es adecuada para describir sistemas cerrados, es decir, sistemas en los que todas las componentes que forman parte de él son conocidas. Esto impide que un sistema descrito en base a *global types* se pueda ver como un módulo independiente que puede conectarse a otros sistemas. De la necesidad de resolver esta problemática surge una noción de *global types abiertos*. En este enfoque un *global type abierto*, denominado *global type with interface roles* (GTIR), denota un número de sistemas abiertos de CFSMs donde ciertos participantes (roles en este contexto) son identificados como interfaces en vez de como participantes propiamente dichos. Para lo cual se introduce una sintaxis paramétrica que, dado un formalismo basado en *global types*, extiende su sintaxis, permitiéndole identificar algunos roles como *interface roles* y definir una composición de *global types abiertos*, interpretados semánticamente como sistemas de CFSMs. Esta sintaxis no depende de un formalismo particular, sino en general mientras las componentes individuales o *end points* puedan ser interpretados como CFSMs. Esta nueva sintaxis además permite asegurar la preservación de las condiciones de seguridad mencionadas anteriormente bajo la composición, siempre que se satisfagan ciertas condiciones. Similar a otros de los modelos presentados, los GTIRs presentan una visión global de la comunicación donde lo que se conectan son CSs. A diferencia de los Interface Automata, sólo se enfoca en la comunicación y, dado el nivel de abstracción, no modela comportamiento interno de los componentes. Por último los GTIRs son un modelo basado en Global Types y, por lo tanto, difieren de nuestro enfoque en que nosotros perseguimos una vista local del problema de la composición que permita tomar cuenta del cómputo que se lleva a cabo en cada una de las componentes que participan del sistema.

Choreography automata [BLT20]: De la tendencia de modelar microservicios y sistemas distribuidos como coreografías, y los autómatas como modelo de especificación ya establecido y sustentado por múltiples resultados, surgen los *Choreography Automata*. Estos autómatas finitos, que pueden ser sincrónicos o asincrónicos, modelan coreografías de communicating systems. La proyección de un choreography automata produce un CS del que se pueden extraer sus componentes en forma de CFSMs. A diferencia de los AFCAs, que modelan componentes desde una visión local, con la posibilidad de escalar a través de la composición, estos autómatas presentan un enfoque global. Es un lenguaje de más alto nivel que especifica la comunicación desde un punto más abstracto pero que a través de proyecciones se llega a más específico como CSs, de los que a su vez pueden obtener CFSMs. La otra discrepancia con nuestro trabajo es que los choreography automata se abstraen de modelar canales o buffers explícitos y solo ven el comportamiento comunicaconal, no especifican cómputo interno de los componentes.

5.3. Trabajo futuro

Si bien el lenguaje soporta binding parcial, queda por estudiar en mayor profundidad si el lenguaje cumple con las condiciones esperadas. Para esto es necesario definir formalmente la composición parcial de AFCAs y ver la relación entre un AFCA obtenido por una serie de composiciones parciales y uno de una composición total. Esto permite vincular los crterios de corrección de la comunicación establecidos por el lenguaje formal de las CFSM con el

comportamiento del resultado de la composición dinámica del sistema.
Una condición que se pide en este modelo es que el conjunto de autómatas y el autómata compuesto sean weak determinate. Esta condición es muy fuerte y restrictiva dado que la propiedad de weak determinacy no se preserva en la composición. De esta condición surge la pregunta de si es posible lingüísticamente generar autómatas weak determinate que al componerse den uno que también lo sea. Una forma posible de solucionar esto sería un modelo donde el comportamiento interno no esté representado en las aristas del autómata, sino de alguna manera encapsularlo dentro de los estados. De este modo potencialmente se evitaría el problema de que la composición de autómatas weak determinate pueda genera uno que no lo es.

Bibliografía

[BdH18] Franco Barbanera, Ugo de'Liguoro, and Rolf Hennicker. Global types for open systems. In Massimo Bartoletti and Sophia Knight, editors, *Proceedings of the 11th Interaction and Concurrency Experience, ICE 2018*, volume 279 of *Electronic Proceedings in Theoretical Computer Science*, pages 4–20, June 2018.

[BLT20] Franco Barbanera, Ivan Lanese, and Emilio Tuosto. Choreography automata. In Simon Bliudze and Laura Bocchi, editors, *Proceedings of Coordination Models and Languages - 22nd IFIP WG 6.1 International Conference, COORDINATION 2020, Held as Part of the 15th International Federated Conference on Distributed Computing Techniques, DisCoTec 2020*, volume 12134 of *Lecture Notes in Computer Science*, pages 86–106. Springer-Verlag, June 2020.

[BZ83] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.

[CDP12] Giuseppe Castagna, Mariangiola Dezani-Ciancaglini, and Luca Padovani. On global types and multi-party session. *Logical Methods in Computer Science*, 8(1), 2012.

[dH01] Luca de Alfaro and Thomas A. Henzinger. Interface automata. In A. Min Tjoa and Volker Gruhn, editors, *Proceedings of the 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 109–120, Vienna, Austria, September 2001. ACM Press.

[DY12] Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty session types meet communicating automata. In Helmut Seidl, editor, *Proceedings of 21st European Symposium on Programming, ESOP 2012, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2012*, volume 7211 of *Lecture Notes in Computer Science*, pages 194–213, Berlin, Heidelberg, 2012. Springer-Verlag.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison–Wesley Longman Publishing Co., Inc., New York, NY, USA, 2001.

[HVK98] Kohei Honda, Vasco Thudichum Vasconcelos, and Makoto Kubo. Language primitives and type discipline for structured communication-based programming. In Chris Hankin, editor, *Proceedings of 7th European Symposium on Programming: Programming Languages and Systems, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98*, Lecture Notes in Computer Science, pages 122–138, Berlin, Germany, 1998. Springer-Verlag.

[HYM08] Kohei Honda, Nobuko Yoshida, and Carbone Marco. Multiparty asynchronous session types. In George Necula, editor, *Proceedings of 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 273–284, New York, NY, USA, 2008. ACM Press.

[Kel76] Robert M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, 1976.

[LT89] Nancy A. Lynch and Mark R. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 2(3):219–246, 1989. Also available as MIT Technical Memo MIT/LCS/TM-373.

[LTY15] Julien Lange, Emilio Tuosto, and Nobuko Yoshida. From communicating machines to graphical choreographies. In Sriram K. Rajamani and David 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 Press.

[Mil89] Robin Milner. *Communication and concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[Vis18] Ignacio Vissani. *Aspectos formales de un modelo de ejecución orienteada a servicios*. PhD thesis, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, 2018. Advisor: Carlos G. Lopez Pombo.