Los actores son naturalmente asincronos, la recepcion del mensaje y cuando se consume ocurren en momentos diferentes. Q es quien recibe el mensaje que luego será consumido por el actor en cuestion. Ningún proceso de CSP envia directamente un mensaje a otro actor, todos los mensajes se envian utilizando Q. Cada proceso Q_i cumple la funcion de "mailbox" para un actor especficio, toda comunicacion pasa por el.

$$Q_i(\langle\rangle) = external.i?msg \rightarrow Q_i(\langle msg\rangle)$$
$$Q_i(\langle h\rangle \frown TAIL) = external.i?msg \rightarrow Q_i(\langle msg\rangle \frown \langle h\rangle \frown TAIL)$$
$$\quad \Box\ internal.i?!h \rightarrow Q_i(TAIL)$$
$$Queue = \mathop{\|}_{i=1}^{m} actors?start_q(i) \rightarrow Q_i(\langle\rangle)$$

Este modelo cuenta con dos canales, *internal* el cual está destinado a la comunicación entre la cola y el actor apareado a la misma y *external* que es canal por el cual se reciben los mensajes que envian otros actores. Notar que la sincronización se hace en términos de "i" que es el mailbox con el cual es identificado este proceso.

$$recCustomer_i(n, client) = internal.i?(k) \rightarrow external.client!(k * n) \rightarrow STOP$$
$$recCustomer = \mathop{\|}_{i=1}^{m} actors?start_{rc}(i, n, client) \rightarrow recCustomer_i(n, client)$$

$recCustomer_i$ recibe un mensaje por su canal interno. Su única funcion es enviar un mensaje a *client* con el valor del entero que recibio, k multiplicado por el parametro n el cual oportunamente recibio cuando fue creado. Esta multiplicación claramente engorrosa, intenta describir el espiritu del paralelismo un actor que es una suerte de proceso efectua una computación y envia el resultado, si bien por como se envian los mensajes todo es secuencial es una aproximación a la idea de paralelismo.

$$factorialRec_i = internal.i?(k, client) \rightarrow$$
$$\quad factorialRecCaso0_i(client)$$
$$\quad\quad \lceil k == 0 \rceil$$
$$\quad factorialRecCasoN_i(client, k, createNewMailbox(i))$$

$factorialRec_i$ recibe una tupla con dos valores, un entero k y un mailbox *client*. Si este valor fuera cero, se comporta como $factorialRecCaso0$ sino como $factorialRecCasoN$.

$$factorialRecCaso0_i(client) = external.client!1 \rightarrow factorialRec_i$$

Cuando se comporta como $factorialRecCaso0_i$ envía al mailbox que recibio como parametro el valor 1 y luego se vuelve a comportar como $factorialRec_i$. Sería el fin de la recursión, para el valor 0 enviar al cliente 1.

$$factorialRecCasoN_i(client, k, newMailbox) =$$
$$actors!start_q(newMailbox) \rightarrow actors!start_{rc}(newMailbox, k, client) \rightarrow$$
$$external.i!(k-1, newMailBox) \rightarrow factorialRec_i$$

En el caso $factorialRecCasoN$, este se llama con tres parametros, mailbox del cliente, el valor entero que recibio y un nuevo mailbox creado en el paso anterior. Este paso "instancia" un actor y una cola utilizando los mensajes espaciales $start_q$ y $start_{rc}$. Una vez enviado estos mensajes, se auto-envia un mensaje que contiene $k-1$ y el valor del nuevo mailbox creado.

$$factorialRec = actors!start_q(1) \rightarrow factorialRec_1$$
$$CLIENT = actors!start_q(2) \rightarrow external.1!(2,5) \rightarrow$$
$$internal.2?k \rightarrow HACERALGOCON(k)$$
$$SISTEMA = Queue \parallel recCustomer \parallel FACT \parallel CLIENT$$

En este caso $factorialRec$ es el actor representando el factorial, con el mailbox 1. $CLIENT$ es, quien tiene el mailbox 2, consulta por el mailbox de 5 y se queda esperando por el canal interno la respuesta.