Trabajo de grado

Prioridades en un modelo de verdadero paralelismo.

Leticia Ramos

Resumen

El lenguaje $\mathcal{L}$
$t ::= a; \mid a^1 \mid t.t' \mid t + t' \mid t \mid_s t' \mid [k]t \mid \varepsilon \mid \delta \mid \theta(t)$

El lenguaje $\mathcal{K}^1$
$t ::= a; \mid a^1 \mid t.t' \mid t + t' \mid t \mid_s t' \mid [k]t \mid \varepsilon \mid \delta \mid t +_> t' \mid t$

Prioridad en los STE
$G \quad (S, i, \mathbf{A}, \dashrightarrow) \ \theta : G \to G$

$$\mathrm{PRI}_{\mathrm{ste}} \frac{p \overset{a}{\dashrightarrow} q \ \ \forall b > a.p \overset{b}{\not\dashrightarrow}}{\theta(p) \overset{a}{\dashrightarrow} \theta(q)} \qquad \mathrm{PRI}'_{\mathrm{ste}} \frac{p \overset{\surd}{\dashrightarrow} \delta}{\theta(p) \overset{\surd}{\dashrightarrow} \delta}$$

Prioridad en los STT's
$G \quad (S, i, \mathbf{A}, \longrightarrow) \ \theta : G \to G$

$$\mathrm{PRI\ I} \frac{p \overset{a}{\longrightarrow} q \ \ \forall b > a.p \overset{b}{\not\longrightarrow}}{\theta(p) \overset{a}{\longrightarrow} \theta(q)} \qquad \mathrm{PRI\ II} \frac{p \overset{a^i}{\longrightarrow} q}{\theta(p) \overset{a^i}{\longrightarrow} \theta(q)}$$

Toda accion que comienza termina. Teorema Sean $u, u', r \in \mathrm{L}$, $a \in Act$, σ una secuencia de acciones $\in Act$, $i \in N$.

$$\forall u', r, a(u \overset{a}{\longrightarrow} u') \implies (\forall \sigma : u' \overset{\sigma}{\longrightarrow} r \text{ y } \forall i \in \{1 \ldots |\sigma|\}.\sigma(i) \neq a^i \implies r \overset{a^{|\sigma|+1}}{\longrightarrow})$$

La prioridad conmuta con la transformación natural de 2ST en STE
Sea $G \in$ 2ST, $G' \in$ STE

$MapSt : G \longrightarrow G'$ y $\mathrm{MapSt} \ \dfrac{p \overset{a}{\longrightarrow} q \ \ q \overset{a^1}{\longrightarrow} r}{p \overset{a}{\dashrightarrow} r} \qquad \mathrm{MapSt}' \ \dfrac{p \overset{\surd}{\longrightarrow} \delta}{p \overset{\surd}{\dashrightarrow} \delta}$

Prioridad para solo finales

$$\mathrm{PRI\ I} \frac{p \overset{a}{\longrightarrow} q}{\theta(p) \overset{a}{\longrightarrow} \theta(q)} \quad \mathrm{PRI\ II} \frac{p \overset{a^j}{\longrightarrow} q \ \ \forall b \succ a.p \overset{b^i}{\not\longrightarrow} \ j, i > 0}{\theta(p) \overset{a^j}{\longrightarrow} \theta(q)}$$

Prioridad para inicios entre si y finales entre si

$$\mathrm{PRI\ I} \frac{p \overset{a}{\longrightarrow} q \ \ \forall b > a.p \overset{b}{\not\longrightarrow}}{\theta(p) \overset{a}{\longrightarrow} \theta(q)} \quad \mathrm{PRI\ II} \frac{p \overset{a^j}{\longrightarrow} q \ \ \forall b \succ a.p \overset{b^i}{\not\longrightarrow} \ j, i > 0}{\theta(p) \overset{a^j}{\longrightarrow} \theta(q)}$$

Prioridad para inicios y finales mezclados

$$\mathrm{PRI\ I} \frac{p \overset{a}{\longrightarrow} q \ \ \forall b > a.p \overset{b}{\not\longrightarrow} \ \ \forall c \gg a.p \overset{c^i}{\not\longrightarrow} \ i > 0}{\theta(p) \overset{a}{\longrightarrow} \theta(q)} \quad \mathrm{PRI\ II} \frac{p \overset{a^j}{\longrightarrow} q \ \ \forall b \succ a.p \overset{b^i}{\not\longrightarrow} \ j, i > 0 \ \ \forall c \gg a.p \overset{c}{\not\longrightarrow}}{\theta(p) \overset{a^j}{\longrightarrow} \theta(q)}$$

El operador de prioridad es compatible con la bisimulación.Teorema
$\forall p, q, \theta, p \leftrightarrows q \implies \theta(p) \leftrightarrows \theta(q)$

$\mathrm{STE}^{\mathrm{R}} \ G \quad (S, i, \mathbf{A}, \dashrightarrow) \ \vdash_R p \overset{a}{\dashrightarrow} p'$

$$\mathrm{PAIT\ I} \frac{\vdash_R p \overset{a}{\dashrightarrow} p'}{\vdash_R p +_> q \overset{a}{\dashrightarrow} p'} \quad \mathrm{II} \frac{\vdash_R q \overset{a}{\dashrightarrow} q' \ \text{ y } \ \forall b \in R. \vdash_R p \overset{b}{\not\dashrightarrow}}{\vdash_R p +_> q \overset{a}{\dashrightarrow} q'}$$

$2\mathrm{ST}^{\mathrm{R}} \ G \quad (S, i, \mathbf{A}, \longrightarrow) \ \vdash_R p \overset{\alpha}{\dashrightarrow} p'$

$$\mathrm{PAIT} \frac{\vdash_R p \overset{\alpha}{\longrightarrow} p'}{\vdash_R p +_> q \overset{\alpha}{\longrightarrow} p'} \quad \mathrm{II} \frac{\vdash_R q \overset{\alpha}{\longrightarrow} q' \ \text{ y } \ \forall b \in R. \vdash_R p \overset{b}{\not\dashrightarrow}}{\vdash_R p +_> q \overset{\alpha}{\longrightarrow} q'}$$

El $|_{>s}$ definido sobre los 2STpermite dar prioridad a procesos que se ejecutan en paralelo y no quieren comunicarse y sólo permite que la segunda componente inicie una acción cuando la primer componente no puede iniciar ninguna

---

Ecuaciones de ambiente y funcional
$Env_x = \{b \in Ini(p). \vdash_{R \cup (Env_y \cap S)} p \overset{b}{\dashrightarrow}\} \quad Env_y = \{b \in Ini(q). \vdash_{R \cup (Env_x \cap S)} q \overset{b}{\dashrightarrow}\}$
$F(Y, X) \overset{\mathrm{def}}{=} \{\{b \in Ini(p). \vdash_{R \cup (S \cap Y)} p \overset{b}{\dashrightarrow}\}, \{b \in Ini(q). \vdash_{R \cup (S \cap X)} q \overset{b}{\dashrightarrow}\}\}$

Toda acción que comienza termina.Teorema Sean $u, u', r \in \mathrm{L}$, $a \in Act$, σ una secuencia de acciones $\in Act$, $i \in N$, R un ambiente.

$$\forall R, u, u', r, a(\vdash_R u \overset{a}{\longrightarrow} u') \implies (\forall R2, R3, \sigma : \vdash_{R2} u' \overset{\sigma}{\longrightarrow} r \text{ y } \forall i \in \{1 \ldots |\sigma|\}.\sigma(i) \neq a^i \implies \vdash_{R3} r \overset{a^{|\sigma|+1}}{\longrightarrow})$$

$+_>$ **y** $|_{>s}$ **son compatibles con la bisimulacion.**
$\forall p, q, r, t \ \ p \leftrightarrows q \ \ y \ \ r \leftrightarrows t \implies p +_> r \leftrightarrows q +_> t$
$\forall p, q, r, t \ \ p \leftrightarrows q \ \ y \ \ r \leftrightarrows t \implies p|_{>s} r \leftrightarrows q|_{>s} t$

Indice

Prioridades en un modelo de verdadero paralelismo

Abstract

En este trabajo se desarrolla un estudio sobre prioridades en sistemas de verdadero paralelismo mediante la introducción de operadores de prioridades en sistemas de transición que admiten la representación de ST estados. El estudio introduce dos tipos diferentes de prioridad. Uno basado en el orden entre lass acciones y otro basado en el orden entre subprocesos. Para el primer caso se dan cuatro definiciones alternativas del operador de prioridad. Se muestra que preservan ciertas propiedades importantes como que toda acción que comienza puede finalizar, y en particular que una de ellas es la extensión natural del operador de prioridades definido en modelos de interleaving. Para la prioridad basada en el orden entre subprocesos, se definen dos tipos de operadores de prioridad con objetivos diferentes. Uno permite que un subproceso que quiere comunicarse tenga prioridad sobre los demás subprocesos si la comunicación es posible. Este operador es definido sobre dos modelos: uno de verdadero paralelismo y otro de interleaving ya que su función es importante en ambos casos. El segundo operador permite que un subproceso tenga prioridad cuando no quiere comunicarse. Este es definido sólo en un modelo de verdadero paralelismo, ya que en los modelos de interleaving no resulta de interés y puede ser representado su comportamiento mediante la combinación de otros operadores. En todos los casos se muestra que los operadores definidos son compatibles con la bisimulación.

Por qué se necesita la prioridad

En este trabajo nos interesamos en la *semántica de sistemas reactivos* [Pnu85]. Estos sistemas (cuyos ejemplos clásicos son los sistemas de control industrial, los manejadores de redes de computadoras y los componentes de los sistemas operativos) tienen como vocación el mantener con el medio una cierta interacción.

No es adecuado entonces considerarlos simplemente como mecanizaciones del cálculo de una función o una relación, como es el caso de otros sistemas, que por ello los llamamos *funcionales* o *relacionales*. Los ejemplos típicos de estos últimos sistemas son los programas de cálculo numérico, los compiladores y todos aquellos cuya interacción con el medio se reduce a un único punto de entrada de datos y a uno de salida de resultados.

Un caso típico de sistemas reactivos lo representa el siguiente ejemplo:

Se tiene un controlador C que debe elegir no determinísticamente entre ejecutar la acción *tick* o la acción *tock*. Esta elección se repite continuamente hasta que un sensor T ejecute la accioń *shut_down*. Cuando esto ocurre, C debe dejar inmediatamente de ejecutar *tick* y *tock*. El controlador C podría ser una unidad de control que regula una reacción química en una planta. T sería un sensor que detecta exceso de temperatura. Cuando el nivel tope de

temperatura es sobrepasado, T impide de alguna manera que C siga trabajando. Esto podría escribirse de la siguiente manera:

$$\begin{array}{lcl} SYS & \stackrel{\text{def}}{=} & (C|_s T) \\ T & \stackrel{\text{def}}{=} & push\_button \ . \ shut\_down \\ C & \stackrel{\text{def}}{=} & tick.C + tock.C + shut\_down \\ S & = & \{shut\_down\} \end{array}$$

La definición de arriba indica que C puede hacer *tick* o *tock* reiteradas veces, o terminar. Pero cuando T, esté listo para hacer el *shut_down*, nada impide que C ejecute una serie de *tick* y *tock* extras debido al no determinismo del +. Esto representa un grave problema ya que es fundamental que inmediatamente luego de oprimir el botón de *shut_down*, el sistema deje de trabajar. Para solucionar este problema es necesario introducir la idea de prioridad. Esto puede hacerse de dos maneras diferentes:

- Diciendo que *la acción shut_down "es mayor"* que la acción *tick* y la acción *tock* y que cualquier otra acción del proceso. Decimos entonces que *shut_down "tiene prioridad"* sobre el resto de las acciones.
- Diciendo que *el subproceso shut_down "tiene prioridad"* de ejecución sobre los otros subprocesos. Es decir que en

  $$tick.C + tock.C + shut\_down$$

  el subtérmino que tiene prioridad es *shut_down*. (Esto puede escribirse: $tick.C + tock.C +_> shut\_down$)

En este trabajo definimos formalmente estos operadores sobre distintos sistemas. Evaluamos su comportamiento y según este comportamiento definimos importantes propiedades que los caracteriza.

Introducción

Los modelos más comunes para la semántica de sistemas reactivos están basados en los *Sistemas de Transiciones Etiquetados* (STE) [Kel76], que son grafos dirigidos donde los nodos representan los estados del sistema, y las flechas las trancisiones entre los estados, etiquetadas por acciones en las que se basa la interacción del sistema con el medio.

La popularidad de los STE es debida a su simplicidad, la posibilidad de representarlos gráficamente, la sencillez con que pueden definirse sobre ellos los constructores usuales de sistemas reactivos (composición secuencial, no determinista, paralela, operadores de comunicación y de prioridad, entre otros) y a la existencia de una técnica simple y elegante, la *Semántica Operacional Estructurada* [Plo81] para dar semántica sobre los STE a los lenguajes de programación (inclusive los no reactivos, ver [Hen90]). Muchos lenguajes simples, llamados *lenguajes de descripción de procesos* han sido propuestos para los sistemas reactivos, por ejemplo CCS [Mil80, Mil89], CSP [BHR84, Hoa85], ACP [BK85, BW90] y ATP [NRSV90, NS94]. La gran mayoría de ellos disponen de semántica sobre los STE.

En los STE, la ejecución de un programa puede verse como una sucesión de estados, y de flechas etiquetadas que unen esos estados, es decir, un camino en el grafo que parte del estado inicial y llega a uno de los posibles estados finales:

$$i \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} s_n$$

Este es el origen de la principal limitación en el uso de los STE: no se pueden representar aquí dos acciones desarrollándose al mismo tiempo, ya que se debe llegar a un estado (terminando una acción) para poder iniciar una nueva acción. Decimos por esto que los STE no pueden representar directamente el paralelismo. Aunque sí puede hacerlo indirectamente, representando la ejecución de a en paralelo con b como la secuencia a, b o bien la secuencia b, a. Por razones obvias llamamos de entrelazamiento (en ingles *interleaving*) a esta representación del paralelismo en los STE.

La principal limitación entonces de los STE como modelos de sistemas reactivos es que el paralelismo no puede representarse directamente, sino mediante entrelazamiento.

Hay muchos modelos que permiten representar el paralelismo sin utilizar el entrelazamiento, que son llamados en general *sistemas verdaderamente paralelos* (en inglés *truly concurrent*). Algunos de ellos son las Redes de Petri [Pet62, Rei85], las Estructuras de Eventos [Win87] y los Sistemas de Trancisiones Asíncronos [Bed87, Shi85].

En este trabajo nos interesamos en un modelo propuesto recientemente por *Gorrieri y Laneve* [GL95] que llamaremos aquí 2ST. Una de las principales ventajas de los 2ST es que son una variante sencilla de los ST. En realidad, en [GL95] y [BGG94], los autores manejan estos modelos como simples ST con ciertas propiedades sobre sus etiquetas, que son garantizadas a partir de una manera especial de construirlos. Estas propiedades les permiten representar el paralelismo sin necesidad de recurrir al entrelazamiento.

La idea esencial es que un 2ST permite expresar la noción elemental de *duración*, donde varias fases de una acción pueden ser ejecutadas entre su inicio y su final. Para ello cada acción será representada por su *inicio* y su *finalización* correspondiente. En ese sentido un 2ST puede verse como un STE donde las etiquetas de las flechas no son acciones $(a, b \ldots)$, sino inicios de acciones (que anotaremos como las acciones, $a, b \ldots$) y los finales correspondientes (que anotaremos con superíndices $a^i, b^j \ldots$, donde i y j sirven para representar cual es exactamente la acción que está terminando entre las acciones a y b iniciadas en el pasado).

Esto nos permite representar fácilmente ejecuciones donde las acciones a y b son ejecutadas en paralelo. Una de ellos es, por ejemplo

$$i \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{b^1} s_3 \xrightarrow{a^3} s_4$$

Al llegar al estado s_2, se han ejecutado los inicios de las acciones a y b, pero no el final de ninguna de ellas. Por tanto, *ambas acciones se están ejecutando en paralelo.* La tercera transición, etiquetada b^1 es el final de la acción b iniciada un paso atrás, mientras la transición etiquetada con a^3 es el fin de la acción a iniciada 3 pasos atrás. En este ejemplo los superíndices no parecen estrictamente necesarios, bastaría con poder diferenciar inicios de finales. Sin embargo, en el caso en que haya varias acciones a iniciadas, es imprescindible, cuando ocurre el final de una de ellas, saber de quien se trata: en ese caso el superíndice brinda la información necesaria.

Los operadores de prioridad forman una familia interesante entre los operadores sobre los STE. En este trabajo los hemos divididos en dos clases.

- Operadores de prioridad que se construyen a partir de un cierto orden entre las acciones, y su aplicacion provoca la poda de las alternativas de "menor prioridad" entre las transiciones que salen de cada estado.

  Estos operadores fueron propuestos por primera vez para modelos de los sistemas reactivos en [BBK86] sobre STE. Formalmente cada operador de prioridad está asociado a un orden parcial $\geq$ en el conjunto de las acciones. Cuando es aplicado a un STE, en cada estado son consideradas las etiquetas de todas las transiciones que parten de allí, y sólo aquellas etiquetadas por acciones maximales según $\geq$ son preservadas.

- Operadores de prioridad que se construyen a partir de un cierto orden entre los subprocesos. Una transición en este marco no es sólo la ejecución de una acción que permite ir de un estado a otro, sino que cada acción conoce el *ambiente* en el que se está ejecutando. Este *ambiente* será decisivo para saber si una transición está o no dentro del grafo.

Debido a que el paralelismo no puede representarse directamente en los STE, cuando se aplica prioridad entre dos acciones que se están ejecutando en forma paralela, el resultado será que la menos prioritaria deberá esperar a que la otra acción termine para poder iniciar su ejecución. Esto limita el espectro de representación de prioridad en un sistema verdaderamente paralelo.

Por ejemplo: Dos personas desean ser atendidas en una farmacia que cuenta con dos empleados. Una de ellas tiene el número de orden uno, y la otra el número dos. El empleado que primero se libere, comenzará a atender a la persona con el número uno, pues tiene prioridad sobre la otra. Si mientras el cliente con el número uno está siendo atendido el otro empleado se libera, podrá atender a la persona con el número dos, no siendo necesario esperar a que la otra termine de ser atendida. Incluso la segunda persona puede terminar primero, si realiza su compra más rápido.

Esta situación no es representable con las definiciones de prioridad conocidas hasta el momento. En este trabajo presentamos un concepto de prioridad que captutra la idea descripta en el ejemplo, definiendo un operador adecuado sobre el modelo 2ST, el operador θ. Lo que se hará será definir el orden parcial sobre el inicio o el final de las acciones o sobre ambos. Esto permitirá introducir distintas alternativas en la definición de prioridad.

A pesar de que el operador de prioridad de los STE no sirve para representar ciertas situaciones en los sistemas verdaderamente paralelos, cumple ciertas propiedades que sería deseable poder verificar en el operador de prioridad definido para los 2ST. Para esto definimos en este trabajo una función que permite traducir acciones divididas en inicio y final (acciones de los 2ST) en acciones indivisibles (acciones de los STE). Utilizando esta definición, se prueba que los operadores de prioridad conmutan. Es decir, es lo mismo aplicar prioridad en los 2ST y luego traducir a STE, que traducir a STE y luego aplicar prioridad. Pero esta conmutatividad no siempre es posible y depende de cómo se defina el orden parcial.

En este trabajo concluimos que cuando el orden parcial se define sobre los finales de las acciones, la conmutatividad no es posible.

Existe una propiedad deseable en los sistemas verdaderamente paralelos que no se puede obtener de los STE debido a la imposibilidad de representar verdadero paralelismo. Esta propiedad expresa que toda acción que comienza puede ser terminada. En este trabajo probamos que esto siempre es posible. Cuando se analiza esta propiedad para el operador de prioridad, en algunos casos se podrá decir que las acciones pueden ser finalizadas siempre y en cualquier momento. En otros, sólo que pueden ser terminadas sin poder determinar el momento exacto. Esta situación también dependerá del conjunto de acciones sobre el que se defina el orden parcial, llegando a la conclusión que cuando se define sobre el final de las acciones, no se puede determinar el momento exacto en el que la acción finaliza. Probamos también que el operador de prioridad θ es compatible con la bisimulación.

La prioridad analizada como un relación de orden sobre las acciones, obliga a que durante todo el proceso, se conserve el orden de prioridades establecido. Es decir, si definimos que la acción a es mayor que la acción b, en cualquier parte del proceso donde sea posible la ejecución de a y b, se elegirá a. A veces esto no es deseable. Una alternativa entonces, es definir la prioridad como una relación de orden entre subprocesos. Retomemos el ejemplo planteado en la sección *Por qué se necesita la prioridad*:

$$
\begin{array}{lcl}
SYS & \stackrel{\text{def}}{=} & (C|_s T) \\
T & \stackrel{\text{def}}{=} & push\_button \ . \ shut\_down \\
C & \stackrel{\text{def}}{=} & tick.C + tock.C + shut\_down \\
S & = & \{shut\_down\}
\end{array}
$$

La definición de arriba indica que C puede hacer *tick* o *tock* reiteradas veces, o terminar. Pero cuando T esté listo para hacer el *shut_down*, nada impide que C ejecute una serie de *tick* y *tock* extras debido al no determinismo del +.

Se necesita para resolver este problema un operador de prioridad que permita que ni bien se ejecuta el *shut_down*, C no pueda elegir hacer *tick* y *tock*.

Para esto es necesario que C conozca el *ambiente* donde se esta ejecutando, para poder saber que otro proceso está tratando de establecer comunicación y así poder decidir, con algún operador que lo permita, ejecutar *shut_down*. Debe entablarse una especie de intercambio de información entre los subprocesos.

En este trabajo definimos un sistema de transición que involucra la noción de ambiente. Este sistema STE^R, es una extensión de los STE. Aquí, la relación de transición no es sólo la ejecución de una acción que de un estado pasa a otro estado, sino *la ejecución de una acción bajo un ambiente de posibles acciones a ejecutar* que de un estado pasa a otro. La ejecución de un programa puede verse como una sucesión de estados, y de flechas etiquetadas que unen esos estados.Las etiquetas serán pares conformados con la acción a ejecutar y el ambiente donde se está ejecutando.

Definimos un sistema de ecuaciones que da la solución al intercambio de información que necesitan dos procesos que estan ejecutándose en paralelo para saber si hay intenciones de comunicación. Las soluciones de este sistema de ecuaciones pueden ser representadas como puntos fijos de un Funcional.

Definimos el *operador de alternativa con prioridad*, $+_>$ en este marco, que permite resolver el problema planteado. $+_>$ permitirá la ejecución de la segunda componente sólo en el caso que la primer componente no pueda hacer ninguna de las acciones que ofrece el ambiente. En este caso, hay un subproceso que desea comunicarse y por lo tanto, se da prioridad a la ejecución de la primer componente. Este mismo concepto es llevado a un modelo de verdadero paralelismo, $2ST^R$ que es una extensión de los 2ST, donde se incluye la noción de ambiente y se define el nuevo operador $+_>$.

Notemos que esta definicion de prioridad, es interesante tanto en modelos de verdadero paralelismo como en modelos que no son de verdadero paralelismo.

Lo que no se capturó con el $+_>$ es la prioridad entre subprocesos que no desean comunicarse (Ejemplo de la farmacia planteado en esta introducción). $+_>$ permite que un subproceso que desea comunicarse tenga prioridad sobre los demás subprocesos. Pero entre dos subprocesos que se están ejecutando en paralelo y no desean comunicarse, el $+_>$ no decide ningún orden de prioridad. Como vimos, este tipo de prioridad no tiene sentido en sistemas que no son de verdadero paralelismo, pero si en los $2ST^R$. Definimos en los $2ST^R$ un nuevo operador, paralelo con prioridad, $|_>$, que permita capturar esta noción de prioridad. Probaremos que en este marco se conserva la propiedad de que toda acción que comienza, termina. Probaremos que ambos operadores definidos son compatibles con la bisimulación tanto en STE^R como en $2ST^R$.

Nuestra contribución

- Definimos un operador de prioridad, basado en un orden entre las acciones, sobre el modelo 2ST. Llamamos al operador θ.
- Probamos en los 2ST que toda acción que comienza, puede ser finalizada y esto ocurre en el momento que se desee.
- Definimos una función que permite transformar 2STs que no contienen combinadores paralelos en STEs.
- Comprobamos que la transformación de 2STs en STEs conmuta. Es decir que los operadores en uno y otro sistema tienen comportamiento similar.
- Dimos 3 definiciones alternativas para el operador de prioridad en los 2ST
  - Definición de un operador de prioridad cuando sólo compiten finales de acciones.
  - Definición de un operador de prioridad cuando compiten inicios entre sí y finales entre sí.
  - Definición de un operador de prioridad cuando compiten inicios y finales de acciones mezclados.
- Comprobamos para las tres definiciones de prioridad alternativas que la propiedad *Toda acción que comienza, puede ser finalizada y esto sucede en cualquier momento* no se cumple.
- Comprobamos para las tres definiciones de prioridad alternativas que siempre existe algún camino para finalizar una acción que fue comenzada. Es decir, *Toda acción que*

comienza, puede ser finalizada. Esta finalización es en *algun momento* y ya no en el momento que se elija.

- Probamos que el operador de prioridad definido es compatible con la bisimulación.
- Definimos un sistema derivado de los STE que contiene la noción de ambiente necesaria para que cada proceso sepa las posibilidades de comunicación en cada momento. Llamamos a este sistema STE^R
- Definimos un sistema derivado de los 2ST que contiene la noción de ambiente necesaria para que cada proceso sepa las posibilidades de comunicación en cada momento. Llamamos a este sistema $2ST^R$
- Definimos un sistema de ecuaciones(Env_x, Env_y) que permite resolver el intercambio de información entre dos procesos que se están ejecutando en paralelo en un ambiente R determinado, para saber las posibilidades de ejecución de uno según las posibilidades de ejecución del otro y viceversa.
- Definimos un funcional para las ecuaciones y llegamos a la conclusión que el punto fijo del funcional es la solución de la ecuación y es exactamente lo que cada proceso puede hacer en un ambiente R dado.
- Definimos un operador de alternativa con prioridad, basado en un orden entre los procesos sobre el modelo STE^R. Llamamos al operador $+_>$.
- Probamos que $+_>$ es compatible con la bisimulación.
- Definimos un operador de alternativa con prioridad, basado en un orden entre los procesos sobre el modelo $2ST^R$. Llamamos al operador $+_>$.
- Probamos que el operador $+_>$ definido sobre los 2ST es compatible con la bisimulación.
- Definimos el operador paralelo con prioridad sobre los 2ST, para capturar la noción de prioridad que no captura $+_>$. Llamamos al operador $|_{>s}$.
- Probamos que $|_{>s}$ es compatible con la bisimulacion

Organización de este trabajo

El trabajo incluye dos papers que conforman el contenido de las Partes 1 y 2. El paper de la Parte 1, fue presentado y expuesto en las Jornadas de Informática e Investigación Operativa (Montevideo 1994) y en las Jornadas Argentinas de Informática e Investigación Operativa (1995). Debido a que ambos papers tratan sobre prioridad en modelos verdaderamente paralelos, hay información que se repite. La organización es la siguiente:

- **Parte 1. Operadores de prioridad basados en un orden sobre las acciones.**

  La sección 1 es la introducción a la prioridad basada en el orden entre acciones.

  La sección 2 introduce las características del lenguaje $\mathcal{L}$ sobre el cual se definirán los sistemas de transición en los que se trabajará.

La sección 3 introduce los conceptos básicos de sistemas de transición etiquetados y define el operador de prioridades en este marco.

En las sección 4 se definen los sistemas de transición ST y se da una primera definición del operador de prioridad en este contexto. Se definen dos propiedades importantes que cumplen los 2ST. Una de ellas indica que toda acción que comienza puede ser finalizada en el momento que se desee. La otra indica que en aquellos 2ST que no contienen combinadores paralelos, la aplicación de un operador en los 2ST tiene el mismo comportamiento que la del operador equivalente en los STE.

En la sección 5 se introducen tres definiciones alternativas del operador de prioridades sobre los 2ST comprobando que toda acción que comienza siempre puede ser finalizada, pero no en el momento que se desee.

En la seccíon 6 se define la noción de bisimulación sobre el operador de prioridad comprobando que es una congruencia.

Finalmente se establecen conclusiónes del trabajo realizado.

- **Parte 2. Operadores de prioridad basados en un orden entre procesos.**

  La sección 1 es la introducción a la prioridad basada en el orden entre procesos.

  En las sección 2 se definen el lenguaje $\mathcal{K}$ sobre el que se definirán los sistemas de transición.

  En la sección 3 se introducen los modelos STE^R y $2\mathrm{ST}^R$ que incluyen la noción de ambiente en las transiciones, necesaria para el intercambio de información entre los procesos. En 3.2, se muestra un caso de conflicto que según las reglas definidas, podrá ser evitado. En 3.3 se definen algunas propiedades para el ambiente y las ecuaciones que permiten el intercambio de información entre procesos en cada momento. Este intercambio puede ser mecanizado mediante la definición de un funcional para el sistema de ecuaciones que también es definido en 3.4.

  En la sección 4 se define la noción de bisimulación sobre el operador de prioridad definido comprobando que es una congruencia.

  En la sección 5 se define la semántica de $\mathcal{K}$ sobre los $2\mathrm{ST}^R$ y se dan las reglas que definen el comportamiento de los procesos. En 5.2 se amplía el lenguaje $\mathcal{K}$ a $\mathcal{K}^1$ introduciendo el operador $|_{>s}$ y se agregan las reglas necesarias para completar el comportamimento de los procesos.

  En la sección 6 se comprueba que la propiedad que cumplen los 2ST de que toda acción que comienza puede ser finaliza, se conserva en los $2\mathrm{ST}^R$

  En la sección 7 se define la noción de bisimulación sobre los operador de prioridad definidos comprobando que en todos los casos es una congruencia.

  Finalmente se establecen conclusiónes del trabajo realizado.

- **Conclusiones.** Se dan las conclusiones generales del trabajo realizado.

Parte 1

Operadores de prioridad basados en un orden entre las acciones

Operadores de prioridad basados en un orden entre las acciones

Abstract

En este trabajo se desarrolla un estudio sobre prioridades en sistemas de verdadero paralelismo mediante la introducción del operador de prioridades en sistemas de transición que admiten la representación de ST estados. Se define el operador θ que permitirá que una acción b que tiene *menor prioridad* que otra acción a, inicie su ejecución sólo en el caso que a no pueda ser iniciada. El estudio introduce cuatro definiciones alternativas para el operador θ. Se muestra que ellas preservan ciertas propiedades y en particular que una de ellas es la extensión natural del operador de prioridades definido en modelos de interleaving. Se muestra también que una vez iniciada una acción su finalización siempre es posible. Se define la noción de bisimulación sobre el operador definido, comprobando que es una congruencia.

1 Introducción

En este trabajo nos interesamos en la *semántica de sistemas reactivos* [Pnu85]. Estos sistemas (cuyos ejemplos clásicos son los sistemas de control industrial, los manejadores de redes de computadoras y los componentes de los sistemas operativos) tienen como vocación el mantener con el medio una cierta interacción.

No es adecuado entonces considerarlos simplemente como mecanizaciones del cálculo de una función o una relación, como es el caso de otros sistemas, que por ello los llamamos *funcionales* o *relacionales*. Los ejemplos típicos de estos últimos sistemas son los programas de cálculo numérico, los compiladores y todos aquellos cuya interacción con el medio se reduce a un único punto de entrada de datos y a uno de salida de resultados.

Los modelos más comunes para la semántica de sistemas reactivos están basados en los *Sistemas de Transiciones Etiquetados* (STE) [Kel76], que son grafos dirigidos donde los nodos representan los estados del sistema, y las flechas las trancisiones entre los estados, etiquetadas por acciones en las que se basa la interacción del sistema con el medio.

La popularidad de los STE es debida a su simplicidad, la posibilidad de representarlos gráficamente, la sencillez con que pueden definirse sobre ellos los constructores usuales de sistemas reactivos (composición secuencial, no determinista, paralela, operadores de comunicación y de prioridad, entre otros) y a la existencia de una técnica simple y elegante, la *Semántica Operacional Estructurada* [Plo81] para dar semántica sobre los STE a los lenguajes de programación (inclusive los no reactivos, ver [Hen90]). Muchos lenguajes simples, llamados *lenguajes de descripción de procesos* han sido propuestos para los sistemas reactivos, por ejemplo CCS [Mil80, Mil89], CSP [BHR84, Hoa85], ACP [BK85, BW90] y ATP [NRSV90, NS94]. La gran mayoría de ellos disponen de semántica sobre los STE.

En los STE, la ejecución de un programa puede verse como una sucesión de estados, y de flechas etiquetadas que unen esos estados, simplemente un camino en el grafo que parte del estado inicial:

$$i \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} s_n$$

Este es el origen de la principal limitación en el uso de los ST: no podemos representar aquí dos acciones desarrollándose al mismo tiempo, ya que se debe llegar a un estado (terminando una acción) para poder iniciar una nueva acción. Decimos por esto que los ST no pueden representar directamente el paralelismo. Aunque sí puede hacerlo indirectamente, representando la ejecución de a en paralelo con b como la secuencia a, b o bien la secuencia b, a. Por razones obvias llamamos de entrelazamiento (en ingles *interleaving*) a esta representación del paralelismo en los STE.

La principal limitación entonces de los STE como modelos de sistemas reactivos es que el paralelismo no puede representarse directamente, sino mediante entrelazamiento.

Hay muchos modelos que permiten representar el paralelismo sin utilizar el entrelazamiento, que son llamados en general *sistemas verdaderamente paralelos* (en inglés *truly concurrent*). Algunos de ellos son las Redes de Petri [Pet62, Rei85], las Estructuras de Eventos [Win87] y los Sistemas de Trancisiones Asíncronos [Bed87, Shi85]. En este trabajo nos interesamos en un modelo propuesto recientemente por *Gorrieri y Laneve* [GL95] que llamaremos aqui 2ST.

Una de las principales ventajas de los 2ST es que son una variante sencilla de los ST. En realidad, en [GL95] y [BGG94], los autores manejan estos modelos como simples ST con ciertas propiedades sobre sus etiquetas, que son garantizadas a partir de una manera especial de construirlos. Estas propiedades les permiten representar el paralelismo sin necesidad de recurrir al entrelazamiento.

La idea esencial es que un 2ST nos permite expresar la noción elemental de *duración*, donde varias fases de una acción pueden ser ejecutadas entre su inicio y su final. Para ello cada acción sera representada por su *inicio* y su *finalización* correspondiente. En ese sentido un 2ST puede verse como un STE donde las etiquetas de las flechas no son acciones $(a, b \ldots)$, sino inicios de acciones (que anotaremos como las acciones, $a, b \ldots$) y los finales correspondientes (que anotaremos con superíndices $a^i, b^j \ldots$, donde i y j representan cual es exactamente la acción que está terminando entre las acciones a y b iniciadas en el pasado).

Esto nos permite representar fácilmente ejecuciones donde las acciones a y b son ejecutadas en paralelo. Una de ellos es, por ejemplo

$$i \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{b^1} s_3 \xrightarrow{a^3} s_4$$

Al llegar al estado s_2, se han ejecutado los inicios de las acciones a y b, pero no el final de ninguna de ellas. Por tanto, *ambas acciones se están ejecutando en paralelo.* La tercera transición, etiquetada b^1 es el final de la acción b iniciada un paso atrás, mientras la transición etiquetada con a^3 es el fin de la acción a iniciada 3 pasos atrás. En este ejemplo los superíndices no parecen estrictamente necesarios, bastaría con poder diferenciar inicios de finales. Sin embargo, en el caso en que haya varias acciones a iniciadas, es imprescindible, cuando ocurre el final de una de ellas, saber de quien se trata: en ese caso el superíndice brinda la información necesaria.

Los operadores de prioridad forman una familia interesante entre los operadores sobre los STE. Se construyen a partir de un cierto orden en las acciones, y su aplicacion provoca la poda de las alternativas de "menor prioridad" entre las transiciones que salen de cada estado.

Estos operadores fueron propuestos por primera vez para modelos de los sistemas reactivos en [BBK86] sobre STE. Formalmente cada operador de prioridad está asociado a un orden parcial $\geq$ en el conjunto de las acciones. Cuando es aplicado a un STE, en cada estado son consideradas las etiquetas de todas las transiciones que parten de allí, y sólo aquellas etiquetadas por acciones maximales según $\geq$ son preservadas.

Debido a que el paralelismo no puede representarse directamente en los STE, cuando se aplica prioridad entre dos acciones que se están ejecutando en forma paralela, el resultado será que la menos prioritaria deberá esperar a que la otra acción termine para poder iniciar su ejecucion. Esto limita el espectro de representación de prioridad en un sistema verdaderamente paralelo. Por ejemplo: Dos personas desean ser atendidas en una farmacia que cuenta con dos empleados. Una de ellas tiene el número de orden uno, y la otra el número dos. El empleado que primero se libere, comenzará a atender a la persona con el número uno, pues tiene prioridad sobre la otra. Si mientras el cliente con el número uno está siendo atendido el otro empleado se libera, podrá atender a la persona con el número dos, no siendo necesario esperar a que la otra termine de ser atendida. Incluso la segunda persona puede terminar primero, si realiza su compra más rápido.

Esta situación no es representable con las definiciones de prioridad conocidas hasta el momento. En este trabajo se presenta un concepto de prioridad que captutra la idea descripta en el ejemplo, definiendo un operador adecuado sobre el modelo 2ST. Lo que se hará será definir el orden parcial sobre el inicio o el final de las acciones o sobre ambos. Esto permitirá introducir distintas alternativas en la definición de prioridad.

A pesar de que el operador de prioridad de los STE no sirve para representar ciertas situaciones en los sistemas verdaderamente paralelos, cumple ciertas propiedades que sería deseable poder verificar en el operador de prioridad definido para los 2ST. Esto pude hacerse de dos maneras: Enunciar la propiedad que se desea verificar y probar que se cumple, o tratar de establecer una compatibilidad entre ambos operadores de prioridad. En este trabajo se eligió la segunda alternativa. Para esto se define una función que permite traducir acciones divididas en inicio y final (acciones de los 2ST) en acciones indivisibles (acciones de los STE). Utilizando esta definición, se prueba que los operadores de prioridad conmutan. Es decir, es lo mismo aplicar prioridad en los 2STy luego traducir a STE, que traducir a STE y luego aplicar prioridad. Pero esta conmutatividad no siempre es posible y depende de cómo se defina el orden parcial. En este trabajo se concluye que cuando el orden parcial se define sobre los finales de las acciones, la conmutatividad no es posible.

Existe una propiedad deseable en los sistemas verdaderamente paralelos que no se puede obtener de los STE debido a la imposibilidad de representar verdadero paralelismo. Esta propiedad expresa que toda acción que comienza puede ser terminada. En este trabajo se prueba que esto siempre es posible. Cuando se analiza esta propiedad para el operador de prioridad, en algunos casos se podrá decir que las acciones pueden ser finalizadas siempre y en cualquier momento. En otros, sólo que pueden ser terminadas sin poder determinar el momento exacto. Esta situación también dependerá del conjunto de acciones sobre el que se defina el orden parcial, llegando a la conclusión que cuando se define sobre el final de las acciones, no se puede determinar el momento exacto en el que la acción finaliza.

En este trabajo probamos que el operador θ definido es compatible con la bisimulación.

El trabajo está organizado como sigue:

La sección 2 introduce las características del lenguaje $\mathcal{L}$ sobre el cual se definirán los sistemas de transición en los que se trabajará.

La sección 3 introduce los conceptos básicos de sistemas de transición etiquetados y define el operador de prioridades en este marco.

En las sección 4 se definen los sistemas de transición ST y se da una primera definición del operador de prioridad en este contexto. Se definen dos propiedades importantes que cumplen los 2ST. Una de ellas indica que toda acción que comienza puede ser finalizada en el momento que se desee. La otra indica que en aquellos 2ST que no contienen combinadores paralelos, la aplicación de un operador en los 2ST tiene el mismo comportamiento que la del operador equivalente en los STE.

En la sección 5 se introducen tres definiciones alternativas del operador de prioridades sobre los 2ST comprobando que toda acción que comienza siempre puede ser finalizada, pero no en el momento que se desee.

En la seccíon 6 se define la noción de bisimulación sobre el operador de prioridad comprobando que es una congruencia.

Finalmente se establecen conclusiónes del trabajo realizado.

2 El lenguaje

Es necesaria la definición de un lenguaje para poder representar los sistemas de transiciones. Este lenguaje que se llamará $\mathcal{L}$, es un lenguaje de descripción de procesos. Un término de $\mathcal{L}$ se define de la siguiente manera:

$$t ::= \quad a \mid a^1 \mid t \,.\, t' \mid t + t' \mid t \mid_s t' \mid [k]t \mid \varepsilon \mid \delta \mid \theta(t)$$

donde a es una acción o el inicio de una acción, dependiendo del sistema que se defina. $a \in \mathbf{A}$ que es el conjunto de acciones o eventos.

$t.t'$ es la composición secuencial de procesos.

$t + t'$ es la suma no determinista de procesos.

$t|_s t'$ es la ejecución en paralelo de procesos, siendo S el conjunto de acciones que establecen la comunicación.

$\theta(t)$ es el operador de prioridad que determina, cuando se debe elegir entre la ejecucion de dos acciones, cual de ellas es la mas prioritaria.

Cuando se observe una acción a con detalle, se hará referencia al *inicio de a* denotado como a y al *final de a* denotado como a^1. Cuando dos procesos que son mirados con detalle, se ejecutan en paralelo, el proceso que no avanza debe saber hace cuánto que está demorado. Esto es necesario para vincular unívocamente a un final de una acción con su inicio correspondiente. Entonces el término [2]t significa que t ha sido demorado 2 ejecuciones. Tanto a^1 como $[k]t$ son términos auxiliares que se incluyen para que la definición de los sistemas de transicón ST sea lo más sencilla y clara posible. Según estos dos criterio definimos :

- Términos buenos. Aquellos en los que no aparecen finalizaciones de acciones, ni retardos. Es decir no aparecen subtérminos de la forma a^1 o $[k]t$. Estos términos serán llamados $\mathcal{L}_b$

- Aquellos en los que aparecen finalizaciones de acciones y retardos. No son buenos pues no se puede saber a priori si a^i y $[1]t$ provienen de la evolución de otro proceso cuyos términos estan bien formados o es un término mal formado.

3 Sistemas de transiciones etiquetadas

Los sistemas de transiciones son utilizados para dar semántica a sistemas reactivos. En este trabajo se investigará sobre los STE(Sistemas de transiones etiquetadas y los 2ST (Sistemas de transición ST).

Definición 3.1 (Sistema de transición etiquetado) Un *sistema de transicion etiquetado* (STE), es una estructura $G = (S, i, \mathbf{A}, \dots\!\!\rightarrow)$ donde

- $S = \{i, u, s, \ldots\}$ es el conjunto de *estados* donde i es el *estado inicial*,
- $\mathbf{A} = \{a, b, e, \surd, \ldots\}$ es el conjunto de *eventos* o *acciones*, con una accion destacada $\surd$ que indica terminación satisfactoria.
- $\dots\!\!\rightarrow \subseteq S \times \mathbf{A} \times S$, es la *relación de transición* (escribimos $s \stackrel{e}{\dots\!\!\rightarrow} s_1$, por $(s, e, s_1) \in \dots\!\!\rightarrow$)

■

3.1 Semántica operacional de $\mathcal{L}$ en los STE

Los STE pueden representarse mediante los terminos buenos del lenguaje $\mathcal{L}$ y un conjunto de reglas que rigen su comportamiento y se detallan en la Tabla 1:

ACT I	$\dfrac{}{a \stackrel{a}{\dots\!\!\rightarrow} \surd}$		
ALT_{ste} I	$\dfrac{p \stackrel{a}{\dots\!\!\rightarrow} p'}{p + q \stackrel{a}{\dots\!\!\rightarrow} p'}$	ALT_{ste} II	$\dfrac{q \stackrel{a}{\dots\!\!\rightarrow} q'}{p + q \stackrel{a}{\dots\!\!\rightarrow} q'}$
SEQ_{ste} I	$\dfrac{p \stackrel{a}{\dots\!\!\rightarrow} p'}{p \cdot q \stackrel{a}{\dots\!\!\rightarrow} p' \cdot q}$	SEQ_{ste} II	$\dfrac{p \stackrel{\surd}{\dots\!\!\rightarrow} \delta y \qquad q \stackrel{a}{\dots\!\!\rightarrow} q'}{p.q \stackrel{a}{\dots\!\!\rightarrow} q'}$
PAR_{ste} I	$\dfrac{p \stackrel{a}{\dots\!\!\rightarrow} p \quad a \notin S}{p\|_s q \stackrel{a}{\dots\!\!\rightarrow} p'\|_s q}$	PAR_{ste} II	$\dfrac{q \stackrel{a}{\dots\!\!\rightarrow} q \quad \alpha \notin S}{p\|_s q \stackrel{a}{\dots\!\!\rightarrow} p\|_s q'}$
PAR_{ste} III	$\dfrac{p \stackrel{a}{\dots\!\!\rightarrow} p' \quad q \stackrel{a}{\dots\!\!\rightarrow} q' \quad \alpha \in S}{p\|_s q \stackrel{a}{\dots\!\!\rightarrow} p'\|_s q'}$		

Tabla 1. Reglas para los STE

3.2 Prioridad en los STE

Definición 3.2 (Operador de prioridad en los STE) Dado un STE $G = (S, i, \mathbf{A}, \dots\!\!\rightarrow)$, para cada orden parcial $\geq$ en el conjunto de acciones $\mathbf{A}$, se define al operador de prioridad $\theta : G \to G$ como

$$\text{PRI}_{\text{ste}} \quad \frac{p \overset{a}{\dots\!\!\rightarrow} q \;\; \forall b > a.p \overset{b}{\not\dots\!\!\rightarrow}}{\theta(p) \overset{a}{\dots\!\!\rightarrow} \theta(q)} \qquad \text{PRI}'_{\text{ste}} \quad \frac{p \overset{\surd}{\dots\!\!\rightarrow} \delta}{\theta(p) \overset{\surd}{\dots\!\!\rightarrow} \delta}$$

Formalmente deberíamos haber escrito $\theta_{\geq}$ en lugar de θ pero en general omitiremos el subíndice $\geq$ cuando quede claro en el contexto. ∎

4 Sistemas de transición ST

La prioridad provee un mecanismo de decisión entre dos o mas eventos en conflicto, imponiendo un orden en el tiempo cuando éstos se ejecutan en forma concurrente.

El concepto de prioridad conocido dice que si se tienen dos procesos ejecutandose en forma concurrente y uno es mas prioritario, éste se ejecutará primero. El otro sólo prodrá ejecutarse una vez que el primero haya terminado. Pero cuando tenemos acciones cuya evolución puede ser observada en el transcurso de su ejecución, resulta muy estricto y poco útil exigir que la acción más prioritaria termine para que la otra pueda comenzar. Por ejemplo: Dos personas desean ser atendidas en una farmacia que cuenta con dos empleados. Una de ellas tiene el número de orden 1 y la otra el número 2. El empleado que primero se libere comenzará a atender a la persona con el número uno, pues tiene prioridad sobre la otra. Si mientras el cliente con el número uno esta siendo atendido el otro empleado se libera, podra atender a la persona con el número 2 sin esperar a que la otra termine de ser atendida. Incluso la segunda persona puede terminar primero si realiza su compra mas rápido.

El objetivo de esta sección es definir un operador de prioridad que atrape esta intuición. La cuestión aquí es presentar los 2ST, el modelo donde esta noción es atrapable. Los 2ST estudian una versión de los STE donde cambia la noción de que las acciones son instantaneas e indivisibles por otra en donde tienen duración en el tiempo. No se asume más que cada acción es atómica y por lo tanto no observable en su evolución. Esto se logra considerando a las acciones divididas en fases ($Split_n$). Por lo tanto se necesita cambiar la noción de estado por una que exprese que sólo una parte de la acción se ha ejecutado. Los Sistemas de Transicion ST (2ST) son un refinamiento de los $Split_n$ ya que están divididos sólo en dos fases: principio y final. Cada final estará asociado sin ambiguedad a un único inicio y esto se determina con funciones auxiliares que asignan a cada final un superíndice $i > 0$ que indica cuantos pasos atrás se inicio la acción. Por ejemplo, a^3 indica que se está terminando una acción que comenzó 3 pasos atrás. Definimos formalmente un sistema de transición ST como:

Definición 4.1 (Sistemas de Transición ST) Un *sistema de transición ST* (2ST) es una estructura $G = (S, i, \mathbf{A}, \longrightarrow)$

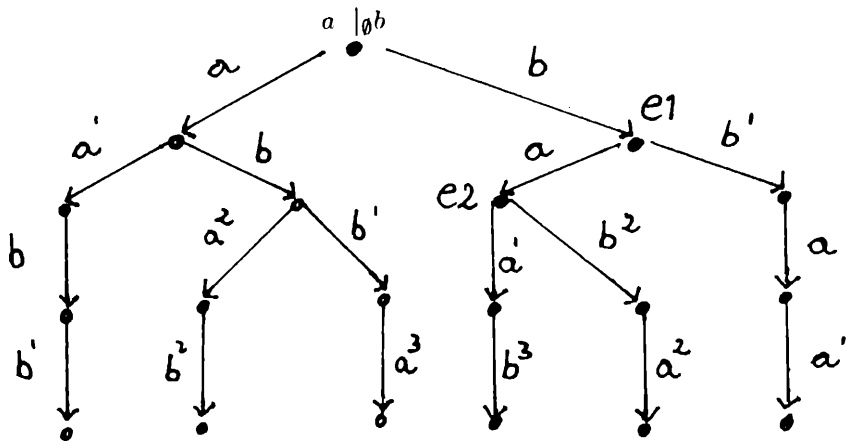
- $S = \{i, u, s, \dots\}$ es el conjunto de *estados* donde i es el *estado inicial*,

- $A = \{a, b, e, \ldots\}$ es el conjunto de *comienzo de acciones*, $A' = \{a^n | a \in A, n > 0\}$ es el conjunto de *terminacion de acciones*; ambos determinan el conjunto de *eventos* $\mathbf{A} = A \cup A' \cup \{\surd\}$ con un evento destacado $\surd$ que indica terminación satisfactoria.

- $\longrightarrow \subseteq S \times \mathbf{A} \times S$, es la *relación de transición.*

■

Ejemplo 4.1



En el estado $e1$ se puede iniciar a o terminar b (b^1). En el estado $e2$ se puede finalizar a (a^1) o b (b^2). El superíndice 2 de b indica que *dos pasos antes en el grafo se inició la acción b que está terminando en este momento.*

■

4.1 Semántica operacional de $\mathcal{L}$ en los 2ST

Los 2ST pueden definirse mediante los terminos buenos y dudosos del lenguaje $\mathcal{L}$.Siguiendo la técnica de Plotkin [Plo81], el conjunto de reglas que define el comportamiento de los operadores está dado en la tabla 1 donde $\alpha \in A \cup A'$ y

$$
\begin{array}{lcl}
name(a) & = & a \\
name(a^n) & = & a \\
name(\surd) & = & \surd \\
inc(n) & = & n + 1
\end{array}
\qquad
\begin{array}{lcll}
f(a, n) & = & a & \\
f(a^n, m) & = & a^n & \text{si } n > m \\
f(a^n, m) & = & a^{n+1} & \text{si } n \leq m
\end{array}
$$

ACT I	$\dfrac{}{a \xrightarrow{a} a^1}$	ACT II	$\dfrac{}{a^1 \xrightarrow{a^1} \varepsilon}$
ALT I	$\dfrac{p \xrightarrow{\alpha} p'}{p + q \xrightarrow{\alpha} p'}$	ALT II	$\dfrac{q \xrightarrow{\alpha} q'}{p + q \xrightarrow{\alpha} q'}$
SEQ I	$\dfrac{p \xrightarrow{\alpha} p'}{p \cdot q \xrightarrow{\alpha} p' \cdot q}$	SEQ II	$\dfrac{p \xrightarrow{\surd} \delta y \qquad q \xrightarrow{\alpha} q'}{p.q \xrightarrow{\alpha} q'}$
PAR I	$\dfrac{p \xrightarrow{\alpha} p \;\; name(\alpha) \notin S}{p\|_s q \xrightarrow{\alpha} p'\|_s [1]q}$	PAR II	$\dfrac{q \xrightarrow{\alpha} q \;\; name(\alpha) \notin S}{p\|_s q \xrightarrow{\alpha} [1]p\|_s q'}$
PAR III	$\dfrac{p \xrightarrow{\alpha} p' \;\; q \xrightarrow{\alpha} q' \;\; name(\alpha) \in S}{p\|_s q \xrightarrow{\alpha} p'\|_s q'}$		
DELAY	$\dfrac{p \xrightarrow{\alpha} p'}{[k]p \xrightarrow{f(\alpha,k)} [inc(k)]p'}$		
PRI I	$\dfrac{p \xrightarrow{a} q \;\; \forall b > a.p \not\xrightarrow{b}}{\theta(p) \xrightarrow{a} \theta(q)}$	PRI II	$\dfrac{p \xrightarrow{a^i} q}{\theta(p) \xrightarrow{a^i} \theta(q)}$
PRI III	$\dfrac{p \xrightarrow{\surd} \delta}{\theta(p) \xrightarrow{\surd} \delta}$		

Tabla 2. Reglas para 2ST

Las únicas reglas que se detallarán son las del operador de prioridad.

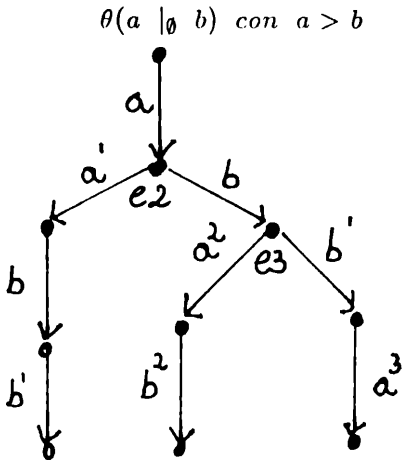
Dado un 2ST $G = (S, i, \mathbf{A}, \longrightarrow)$, para cada orden parcial $\geq$ en el conjunto de acciones $\mathbf{A} = A \cup A'$, se define el operador $\theta : G \to G$ de acuerdo a las reglas PRI I, PRI II y PRI III.

PRI I dice que si un proceso p puede comenzar una acción a y no existe ninguna otra acción que según el orden parcial $\geq$ sea mayor que a, entonces al aplicar θ a p , se ejecutará el inicio de a.

PRI II indica que no existe competencia entre los finales. $\geq$ solo actúa sobre inicios.

PRI III dice que si un proceso p termina exitosamente, al aplicar θ, también se obtendrá una terminación existosa.

Ejemplo 4.2



Cuando a y b se inician, como $a > b$, tiene prioridad el inicio de a. Entonces se anula la rama donde b puede comenzar. En el estado e2 se puede elegir entre finalizar a o comenzar b pues el final de una acción no compite con el inicio de otra. La misma situación ocurre en el estado e3 pues los finales no compiten entre si. (se puede comparar este gráfico con el ejemplo 4.1 para ver las ramas que faltan). ∎

4.2 Toda acción que comienza termina

Cuando se presantaron los 2ST al comienzo de esta sección, se dijo que cada inicio tiene asociado un único final. Lo que no se dijo es que en los 2ST definidos por $\mathcal{L}$ este final siempre es alcanzable y exactamente en el momento que se desee. En el gráfico de ejemplo 4.2 se puede observar que por cualquier rama que tome, todas las acciones que comenzaron llegan al final y cada rama provee distintos momentos para terminar cada acción, lo que permite poder decidir en qué momento se desea terminar.

Formalmente este concepto se puede expresar de la siguiente manera:

Teorema 4.1

Sean u, u',r $\in L$, $a \in Act$, σ *una secuencia de acciones* $\in Act$, $i \in N$.

$$\forall u', r, a(u \xrightarrow{a} u') \implies (\forall \sigma : u' \xrightarrow{\sigma} r \ y \ \forall i \in \{1 \ldots |\sigma|\}.\sigma(i) \neq a^i \implies r \xrightarrow{a^{|\sigma|+1}})$$

Prueba.

Dada la propiedad:

P(u) =

$$\forall u', r, a(u \xrightarrow{a} u') \implies (\forall \sigma : u' \xrightarrow{\sigma} r \ y \ \forall i \in \{1 \ldots |\sigma|\}.\sigma(i) \neq a^i \implies r \xrightarrow{a^{|\sigma|+1}})$$

se probará ∀ u P(u) por inducción sobre la estructura de u.

- Casos base:
  - u = a. P(a) se cumple
  - u = a^1. $a^1 \not\xrightarrow{a}$. Por lo tanto $P(a^1)$.

- u = δ. $\delta \xrightarrow{a} \not\rightarrow$. Por lo tanto $P(\delta)$.
- u = ϵ. $\epsilon \xrightarrow{a} \not\rightarrow$. Por lo tanto ϵ).

- $u = p + q$.

  $P(p)\ y\ P(q) \quad \Longrightarrow \quad P(p+q)$

$$p + q \xrightarrow{a} u' \xrightarrow{\sigma} r$$

$$sii$$

$$p \xrightarrow{a} u' \xrightarrow{\sigma} r \qquad porALT1$$

$$o$$

$$q \xrightarrow{a} u' \xrightarrow{\sigma} r \qquad porALT2$$

Ya que son las unicas posibles evoluciones de $p + q$. Como p y q cumplen la propiedad, en ambos casos

$$r \xrightarrow{a|\sigma|+1} .$$

Por lo tanto, P($p + q$)

- $u = p.q$

  $P(p)\ y\ P(q) \quad \Longrightarrow \quad P(p.q)$

Lema 4.2 *Si en una secuencia de procesoc p.q , p inicia la acción a , todas las demás acciones que se ejecuten antes de la finalización de a, serán realizadas por p.*

$$p.q \xrightarrow{a} u' \xrightarrow{\sigma} r$$

$$sii$$

$$(\xrightarrow{\surd} \delta p) \qquad y \quad q \xrightarrow{a} q' \xrightarrow{\sigma} p'' \qquad porSEQ2$$

$$o$$

$$p \xrightarrow{a} p' \xrightarrow{\sigma} p'' \qquad por\ SEQ1\ y\ Lema \quad u' = p' \cdot q \quad r = p'' \cdot q$$

- Cuando se aplica SEQ2, es el caso P(q).Por lo tanto se cumple.

– Cuando se aplica SEQ1, p cumple el lema y HI. Por lo tanto

$$p \xrightarrow{a} p' \xrightarrow{\theta} p'' \xrightarrow{a|\sigma|+1}$$

Aplicando SEQ1

$$p.q \xrightarrow{a} p'.q \xrightarrow{\theta} p''.q \xrightarrow{a|\sigma|+1}$$

$$p.q \xrightarrow{a} p'.q \xrightarrow{\theta} r \xrightarrow{a|\sigma|+1}$$

- $u = \theta(p)$

  $P(p)\ y\ P(q) \quad \Longrightarrow \quad P(\theta(p))$

$$\theta(p) \xrightarrow{a} \theta(p')$$

$$sii$$

$$p \xrightarrow{a} p' \quad y \quad \forall b > a p \not\xrightarrow{b}$$

Si

$$\theta(p') \xrightarrow{\sigma} \theta(r)$$

es porque todas las acciones en σ son maximales. Por lo tanto, aplicando PRI I y PRI II

$$p' \xrightarrow{\sigma} r$$

.

P(p) se cumple por HI. Por lo tanto:

$$r \xrightarrow{a|sigma|+1}$$

$$\Longrightarrow$$

$$\theta(r) \xrightarrow{a|sigma|+1} \ Por\ PRI\ II$$

Por lo tanto se cumple P(θ(p)).

- $u = p|_s q$

  $P(p)\ y\ P(q) \quad \Longrightarrow \quad P(p|_s)$

$$p|_s q \xrightarrow{a} u'$$

$$sii$$

$$
\begin{array}{llllllll}
1 & p \xrightarrow{a} p' & & & a \notin S & y & u' = p'|_s[1]q \\
2 & q \xrightarrow{a} q' & & & a \notin S & y & u' = [1]p|_s q' \\
3 & p \xrightarrow{a} p' & y & q \xrightarrow{a} q' & a \in S & y & u' = p'|_s q'
\end{array}
$$

Sea σ = b.σ

$$u' \xrightarrow{\sigma} r$$

$$sii$$

$$\exists u''.u' \xrightarrow{b} u'' \xrightarrow{\sigma'} r$$

$$u' \xrightarrow{b} u''$$

$$sii$$

$$
\begin{array}{llllllll}
p' \xrightarrow{a} p'' & & & & a \notin S & y & u' = p'|_s[1]q & u'' = p''|_s[2]q \\
[1]q \xrightarrow{a} [2]q' & & & & a \notin S & y & u' = p'|_s[1]q' & u'' = [1]p'|_s[2]q' \\
p' \xrightarrow{a} p'' & y & [1]q \xrightarrow{a} [2]q' & & a \in S & y & u' = p'|_s[1]q & u'' = p''|_s[1]q'
\end{array}
$$

Los mismos avances se hacen con 2 y 3. Repitiendo la operacion con σ' = b_1 .σ" ... σ^n = b_n, vemos que las unicas posibles evoluciones del operador paralelo son evoluciones de p'^i o de q'^i o de ambas. Sea σ_p todas las acciones de σ realizadas por p para que u' se convierta en r. Sea σ_q todas las acciones de σ realizadas por q' para que u' se convierta en r.

p y q cumplen la propiedad. Es decir

$$p \xrightarrow{a} p' \xrightarrow{\sigma_p} r_p \overset{a|sigma_p|+1}{\longrightarrow}$$

$$q \xrightarrow{a} q' \xrightarrow{\sigma_q} r_q \overset{a|sigma_q|+1}{\longrightarrow}$$

Como σ es una combinacion de σ_p y σ_q, se puede decir que

$$u' \xrightarrow{\sigma} r \overset{a|sigma|+1}{\longrightarrow}$$

- $u = [1]p$

  $P(p) \implies P([1]p)$

$$[1]p \xrightarrow{a} u'$$

$$sii$$

$$p \xrightarrow{a} p'$$

Como P(p), $\exists \sigma.p' \xrightarrow{\sigma} r \overset{a|\sigma|+1}{\longrightarrow}$

Aplicando DELAY

$$[1]p \xrightarrow{a} [2]p' \xrightarrow{\sigma} r \overset{a|sigma|+1}{\longrightarrow}$$

Por lo tanto $P([1]p)$.

■

4.3 La prioridad conmuta con la transformación natural de 2ST en STE

Se ha presentado una definición para un operador de prioridad sobre los 2ST, inspirado en uno ya definido sobre los STE. Es deseable que, por ejemplo, en aquellos 2ST que no contienen combinadores paralelos, y que son, por lo tanto, esencialmente STE, la aplicación de ambos operadores devuelva el mismo resultado. El objetivo de esta sección es presentar un resultado formal que nos asegura propiedades como esta y justifica por lo tanto la utilización del término "prioridad" para designar el operador definido.

Este resultado se basa en la posibilidad de establecer una función, aquí llamada MapSt, que devuelva para cada 2ST aquel STE que se comporta de una manera similar, pero sin tomar provecho del paralelismo. MapSt está definida de la siguiente manera:

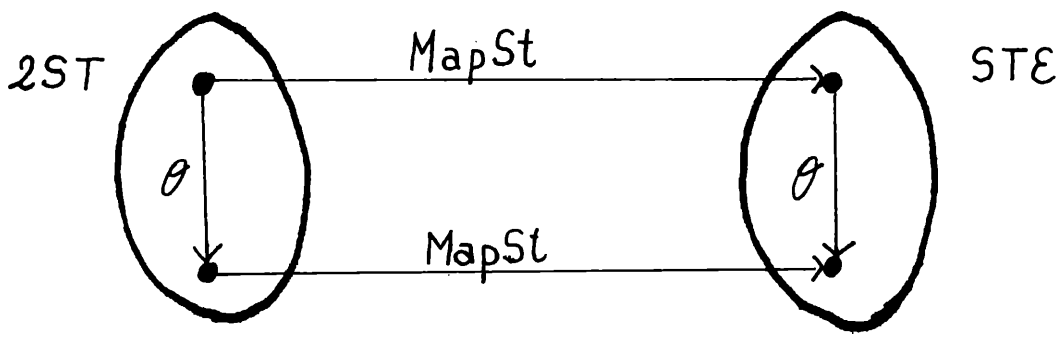
Sea $G \in$ 2ST, $G' \in$ STE

$MapSt : G \longrightarrow G'$ *y*
$MapSt(< S, i, \mathbf{A}, \longrightarrow >) =< S', i', \mathbf{A}', \cdots\cdots\blacktriangleright >$
donde $S = \mathcal{L}$ *y* S' = terminos buenos de $\mathcal{L}$
$i \in$ terminos buenos de ℓ y $i' = i$
$\mathbf{A}_{st2}$ = es el dado por la definición 4.1 y $\mathbf{A}_{ste}$ = es el dado por la definición 3.1.
$\longrightarrow$ queda definida por las reglas de la Tabla 2
$\cdots\cdots\blacktriangleright$ queda definida por las siguientes reglas

$$\text{MapSt} \quad \frac{p \xrightarrow{a} q \quad q \xrightarrow{a^1} r}{p \overset{a}{\cdots\cdots\blacktriangleright} r} \qquad \text{MapSt}' \quad \frac{p \xrightarrow{\surd} \delta}{p \overset{\surd}{\cdots\cdots\blacktriangleright} \delta}$$

Como vemos, en MapSt se mantienen aquellas acciones cuyo final se ejecuta inmediatamente después de su inicio.

Para formalizar la noción de que los operadores en uno y otro sistema representen la misma intuición, mostraremos que cada operador conmuta con MapSt. O dicho de otra manera, con los terminos buenos de $\mathcal{L}$, las reglas de la tabla 2 (2ST) y la funcion MapSt se pueden construir todas las reglas de los STE. De la misma manera, con los terminos buenos de $\mathcal{L}$, las reglas de la tabla 1 (STE) y la funcion MapSt, se pueden construir las reglas de los 2ST. Con esto se está garantizando que los operadores en uno y otro sistema se comportan en forma similar.



Esto se demuestra formalmente con el siguiente teorema.

Teorema 4.3 .
$\forall p, q \in$ *términos buenos de* L

1. $\text{MapSt}(\theta(p)) = \theta(\text{MapSt}(p))$
2. $\text{MapSt}(p + q) = \text{MapSt}(p) + \text{MapSt(q)}$
3. $\text{MapSt}(p \cdot q) = \text{MapSt}(p) \cdot \text{MapSt(q)}$
4. $\text{MapSt}(p|q) = \text{MapSt}(p)|\text{MapSt(q)}$

Prueba. Teninendo en cuenta el teorema 4.1 sabemos que para todo p, existen q y r tal que $p \xrightarrow{a} q \xrightarrow{a^1} r$. Entonces,

1. Por un lado tenemos que

$$\text{MapSt}\,\frac{\text{PRI I}\,\dfrac{p \xrightarrow{a} q \quad \forall b > a.p \overset{b}{\not\longrightarrow}}{\theta(p) \xrightarrow{a} \theta(q)} \qquad \text{PRI II}\,\dfrac{q \xrightarrow{a^1} r}{\theta(q) \xrightarrow{a^i} \theta(r)}}{\theta(p) \overset{a}{\dashrightarrow} \theta(r)}$$

   y por el otro

$$\text{PRI}_{\text{STE}}\,\frac{\text{MapSt}\,\dfrac{p \xrightarrow{a} q \quad q \xrightarrow{a^1} r}{p \overset{a}{\dashrightarrow} q} \qquad \text{MapSt}\,\dfrac{\forall b > a.p \overset{b}{\not\longrightarrow}}{\forall b > a.p \overset{b}{\not\dashrightarrow}}}{\theta(p) \overset{a}{\dashrightarrow} \theta(r)}$$

   Es decir, se obtiene el mismo resultado aplicando las reglas de los STE que aplicando las reglas de los 2ST + MapSt (la flecha definida por las dos reglas de MapSt, tiene el mismo comportamiento que las reglas de la Tabla 1).

2. Por un lado tenemos que

$$\text{MapSt}\,\frac{\text{ALT I}\,\dfrac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad p' \xrightarrow{a^i} r}{p + q \overset{a}{\dashrightarrow} r}$$

   y por el otro

$$\text{ALT 1}_{\text{STE}}\,\frac{\text{MapSt}\,\dfrac{p \xrightarrow{a} p' \quad p' \xrightarrow{a^1} r}{p \overset{a}{\dashrightarrow} r}}{p + q \overset{a}{\dashrightarrow} r}$$

   De manera similar se prueba aplicando ALT2

3. Por un lado tenemos que

$$\text{MapSt}\,\frac{\text{SEQ I}\,\dfrac{p \xrightarrow{a} p'}{p \cdot q \xrightarrow{a} p' \cdot q} \qquad \text{SEQ I}\,\dfrac{p' \xrightarrow{a^1} p''}{p' \cdot q \xrightarrow{a^i} r = p'' \cdot q}}{p \cdot q \overset{a}{\dashrightarrow} r}$$

   y por el otro

$$\text{SEQ I}_{\text{STE}}\dfrac{\text{MapSt}\dfrac{p \xrightarrow{a} p' \;\; p' \xrightarrow{a^1} p''}{p \overset{a}{\dashrightarrow} p''}}{p.q \overset{a}{\dashrightarrow} r = p'' \cdot q}$$

De manera similar se prueba aplicando SEQ2

4. Por un lado tenemos que

$$\text{MapSt}\dfrac{\text{PAR I}\dfrac{p \xrightarrow{a} p' \;\; a \notin S}{p|_s q \xrightarrow{a} p'|_s[1]q} \qquad \text{PAR I}\dfrac{p' \xrightarrow{a^1} p''}{p'|_s[1]q \xrightarrow{a^i} r = p''|_s[2]q}}{p|_s q \overset{a}{\dashrightarrow} r}$$

y por el otro

$$\text{PAR I}_{\text{STE}}\dfrac{\text{MapSt}\dfrac{p \xrightarrow{a} p' \;\; p' \xrightarrow{a^1} p''}{p \overset{a}{\dashrightarrow} p''}}{p|_s q \overset{a}{\dashrightarrow} r = p''|_s q}$$

De manera similar se prueba aplicando SEQ2

■

5 Otras definiciones para el operador de prioridades

Hasta ahora fue analizado el comportamiento del operador de prioridad aplicando el orden parcial sobre el inicio de las acciones. Pero también se podría querer aplicar prioridad en el momento que dos acciones están finalizando. Por ejemplo: Dos personas entran al un edificio y deben tomar un ascensor para llegar hasta sus departamentos. Tomar el ascensor implica:

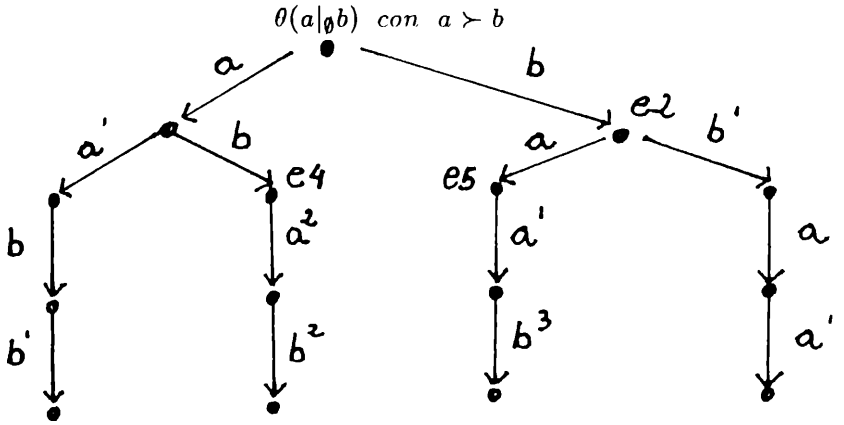
- Subir al ascensor.
- Llegar hasta el piso deseado y bajar del ascensor.

Ambas personas puden iniciar la subida a la vez. Pero para bajar, habrrá una que tendrá prioridad sobre la otra; obviamente, la que va al piso mas cercano. Incluso si mientras el ascensor está en camino sube otra persona y va a un piso anterior al de las otras dos, concluira el viaje primero, apesar de haberlo empezado mas tarde. Esta situación se formaliza de la siguiente manera:

5.1 Reglas de prioridad cuando sólo compiten finales

$$\text{PRI I}\quad \frac{p \xrightarrow{a} q}{\theta(p) \xrightarrow{a} \theta(q)} \qquad \text{PRI II}\quad \frac{p \xrightarrow{a^j} q \;\; \forall b \succ a.p \overset{b^i}{\not\longrightarrow} \; j, i > 0}{\theta(p) \xrightarrow{a^j} \theta(q)} \qquad \text{PRI III}\quad \frac{p \xrightarrow{\surd} \delta}{\theta(p) \xrightarrow{\surd} \delta}$$

Ejemplo 5.1



En el estado $e4$ o $e5$, se puede elegir entre finalizar a o b.Como $a \succ b$, se anula la rama de finalización de b. En $e2$ se puede elegir entre iniciar a o finalizar b pues los inicios no compiten con los finales. ∎

Otra posibilidad para aplicar prioridad es hacerlo sobre los inicios y también sobre los finales de las acciones. En el ejemplo de la farmacia, supongamos que el cobro de la compra se realiza por caja. Es decir comprar en la farmacia implica:

- Realizar la compra.
- Pagar la compra

Si ambas personas terminan al mismo tiempo su compra, llegarán juntas a la caja y nuevamente habrá que establecer un criterio de prioridad para ver a quien se le cobra primero. Este criterio no tiene por que ser igual al que se utilizo para realizar la compra.

Ello implica que pueden usarse dos ordenes parciales diferentes sobre las acciones, uno para los inicios y otro para los finales.

Esta situación se formaliza de la siguiente manera:

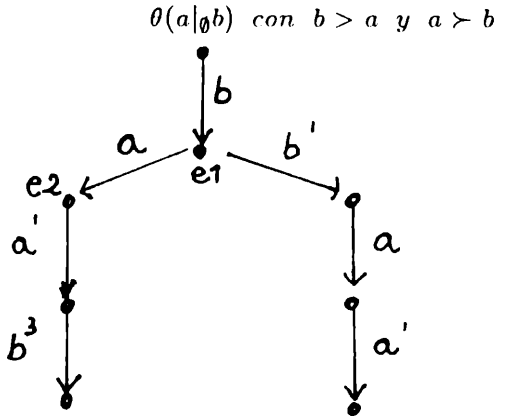
5.2 Reglas de prioridad cuando compiten inicios entre sí y finales entre sí

PRI I
$$\frac{p \xrightarrow{a} q \quad \forall b > a . p \xrightarrow{b} \not\rightarrow}{\theta(p) \xrightarrow{a} \theta(q)}$$

PRI II
$$\frac{p \xrightarrow{a^j} q \quad \forall b \succ a . p \xrightarrow{b^i} \not\rightarrow \quad j,i > 0}{\theta(p) \xrightarrow{a^j} \theta(q)}$$

PRI III
$$\frac{p \xrightarrow{\surd} \delta}{\theta(p) \xrightarrow{\surd} \delta}$$

Ejemplo 5.2



En el estado inicial se corta la rama que inicia a, ya que $b > a$. En $e2$, se corta la rama que finaliza b pues la finalización de a tiene prioridad sobre la de b ($a \succ b$). En $e1$ se puede elegir entre el final de b o el inicio de a ya que no compiten inicios con finales. ■

Como se puede ver en PRI II, los ordenes parciales son distintos.

Aún se puede analizar otra posibilidad y es la competencia entre un inicio y un final. Supongamos que en la farmacia del ejemplo, la persona que atiende es la misma que cobra. En el momento de cobrar al cliente que está siendo atendido, llega otro cliente a la farmacia con una herida que debe ser curada de inmediato. La persona que atiende deberá establecer un criterio de prioridad entre el cobro y la atencion del nuevo cliente. En el ejemplo del ascensor, supongamos que mientras una persona inició su viaje, otra persona quiere subir en un piso intermedio, pero va hacia abajo. Para subir al ascensor, deberá esperar que la otra persona primero llegue al piso deseado y recién luego podrá utilizarla para descender. Como vemos el criterio de prioridad utilizado en uno y otro caso, es diferente. Ello implica que pueden usarse tres órdenes parciales diferentes sobre las acciones, uno para los inicios, otro para los finales y un tercero para comparar los inicios con lo finales.

Esta situación se formaliza de la siguiente manera:

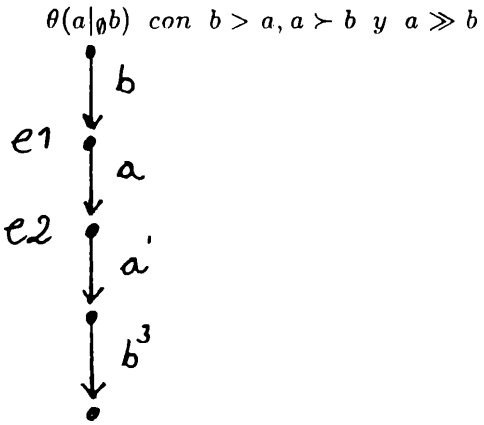
5.3 Reglas de prioridad cuando compiten inicios y finales

PRI I
$$\frac{p \xrightarrow{a} q \quad \forall b > a.p \not\xrightarrow{b} \quad \forall c \gg a.p \not\xrightarrow{c^i} \quad i > 0}{\theta(p) \xrightarrow{a} \theta(q)}$$

PRI II
$$\frac{p \xrightarrow{a^j} q \quad \forall b \succ a.p \not\xrightarrow{b^i} \quad j,i > 0 \quad \forall c \gg a.p \not\xrightarrow{c}}{\theta(p) \xrightarrow{a^j} \theta(q)}$$

PRI III
$$\frac{p \xrightarrow{\surd} \delta}{\theta(p) \xrightarrow{\surd} \delta}$$

Ejemplo 5.3



En el estado inicial se corta la rama que inicia a pues $b > a$. En $e1$ se corta la rama que finaliza b pues el inicio de a tiene prioridad sobre el final de b ($a \gg b$). En $e2$ se corta la rama que finaliza b pues el final de a tiene prioridad sobre el final de b($a \succ b$). ∎

Si se observan los ejemplos con antención, se puede ver que cuando compiten finales, ya no se puede afirmar que una acción iniciada, puede ser terminada en el momento que se desee. En los ejemplos 5.1 y 5.2 vemos que luego de iniciadas a y b, b no puede ser finalizada inmediatamente(es decir, no existe en el grafo ningún b^1).

Lo que sí se puede afirmar cuando compiten finales, es que siempre existe algún camino que llevara al final de cualquier acción comenzada, pero el final no será en el momento que se elija, sino en algún momento. En los ejemplos 5.1, 5.2 y 5.3 vemos que si bien no existe b^1, siempre existe un camino que conduce a un b^i, tal que la acción b que fue iniciada, puede ser finalizada.

Esto se formaliza con el teorema 5.2.

Lema 5.1 *Para todo término p, y para todo σ, $p \xrightarrow{\sigma}$ implica $|\sigma| < f(p)$, donde $f(p)$ está dado por el doble de la cantidad de acciones que ocurren en p.*

Teorema 5.2 *Si $p \xrightarrow{a} p'$ entonces $\forall \sigma.\exists \rho.p' \xrightarrow{\sigma} p'' \xrightarrow{\rho} r$ y $\forall i \in \{1..|\sigma|\}.\sigma(i) \neq a^i$ y $\forall j \in \{1..|\rho|\}.\rho(i) \neq a^{j+|\sigma|}$ implica $r \xrightarrow{a^{|\sigma\rho|+1}} r'$.*

Prueba. Por inducción en la estructura de p. Consideraremos sólo el caso $p \equiv \theta(q)$.

$$\theta(q) \xrightarrow{a} \theta(q') \xrightarrow{\sigma} \theta(q'')$$

implica, por PRI I y PRI II

$$q \xrightarrow{a} q' \xrightarrow{\sigma} q''$$

y por hipótesis inductiva existe σ_1

$$q \xrightarrow{a} q' \xrightarrow{\sigma} q'' \xrightarrow{\sigma_1} r_1 \xrightarrow{a^{n_1}}$$

con $n_1 = |\sigma\sigma_1| + 1$.

Aquí se presentan dos casos:

Caso 1. Si $q'' \xrightarrow{\sigma_1} r_1 \xrightarrow{a^{n_1}}$ es un camino maximal, entonces usando PRI I y PRI II, inmediatamente se obtiene,

$$\theta(q'') \xrightarrow{\sigma_1} \theta(r_1) \xrightarrow{a^{n_1}}$$

Caso 2. Si σ_1 no es maximal, entonces consideramos el prefijo σ_1^{ini} mas largo de σ_1 que este formado por acciones maximales. Podemos escribir, entonces, $\sigma_1 a^{n_1} = \sigma_1^{ini}\sigma_1^{fin}a^{n_1}$, y considerar

$$q \xrightarrow{a} q' \xrightarrow{\sigma} q'' \xrightarrow{\sigma_1^{ini}} \xrightarrow{b_{max_1}} p_1''$$

donde b_{max_1} es una acción maximal, justamente una de aquellas que hace que σ_1^{ini} sea el prefijo maximal más largo. Por hipótesis inductiva

$$q \xrightarrow{a} q' \xrightarrow{\sigma\sigma_1^{ini}b_{max_1}} p_1'' \xrightarrow{\sigma_2} r_2 \xrightarrow{a^{n_2}}$$

con $n_2 = |\sigma\sigma_1^{ini}b_{max_1}\sigma_2| + 1$, y entonces el mismo razonamiento se vuelve a aplicar.

Notar que $\sigma\sigma_1^{ini}b_{max_1}$ es estrictamente mas largo que σ ya que, si bien σ_1^{ini} puede ser vacío, b_{max_1} siempre es una acción

Eso basta para garantizar que en alguna iteración se usará el caso 1. Aumentando el largo del camino maximal en cada paso se llegará eventualmente al caso donde

$$|\sigma\sigma_1^{ini}b_{max_1}\sigma_2^{ini}b_{max_2}\ldots\sigma_n^{ini}b_{max_n}|$$

mida $f(n) - 1$, es decir uno menos que el camino más largo. Este camino debe poder completarse inmediatamente con a^{n+1}, ya que en otro caso la hipótesis inductiva no se cumpliría. Entonces ese camino puede completarse con a^{n+1} que es la acción maximal (y única en este caso). Por lo tanto ese camino genera uno de la forma

$$\theta(q) \xrightarrow{a} \theta(q') \xrightarrow{\sigma} \theta(q'') \longrightarrow \ldots \xrightarrow{a^{n_1}}$$

■

Se concluye de esta propiedad que no existe conmutatividad con STE ya que la función de MapST solo mapea aquellas acciones que finalizan ni bien fueron iniciadas y como ya se dijo, esto no siempre pasa cuando compiten los finales.

6 Bisimulacion como congruencia para los 2ST

Diversas relaciones de equivalencia han sido definidas en el marco de los sistemas de transiciónes etiquetadas y el los sistemas de transición ST. En este trabajo nos interesamos en la relación de equivalencia *Bisimulacion* entre procesos. Se dice que dos procesos son bisimilares cuando toda acción que es ejecutada por uno, puede ser ejecutada por el otro, transformándose en procesos bisimilares. Este juego de imitación de un proceso por otro se mantiene así a lo largo de las transiciones. Además el rol del imitado y del imitador pueden intercambiarse en cada instante. Formalmente se define bisimulación de la siguiente manera:

Definición 6.1 (Bisimulación) Sea $a \in \mathbf{A}$ p y $q \in \mathcal{K}$. Una *bisimulacion* es una relación de equivalencia binaria R sobre procesos que satisface :

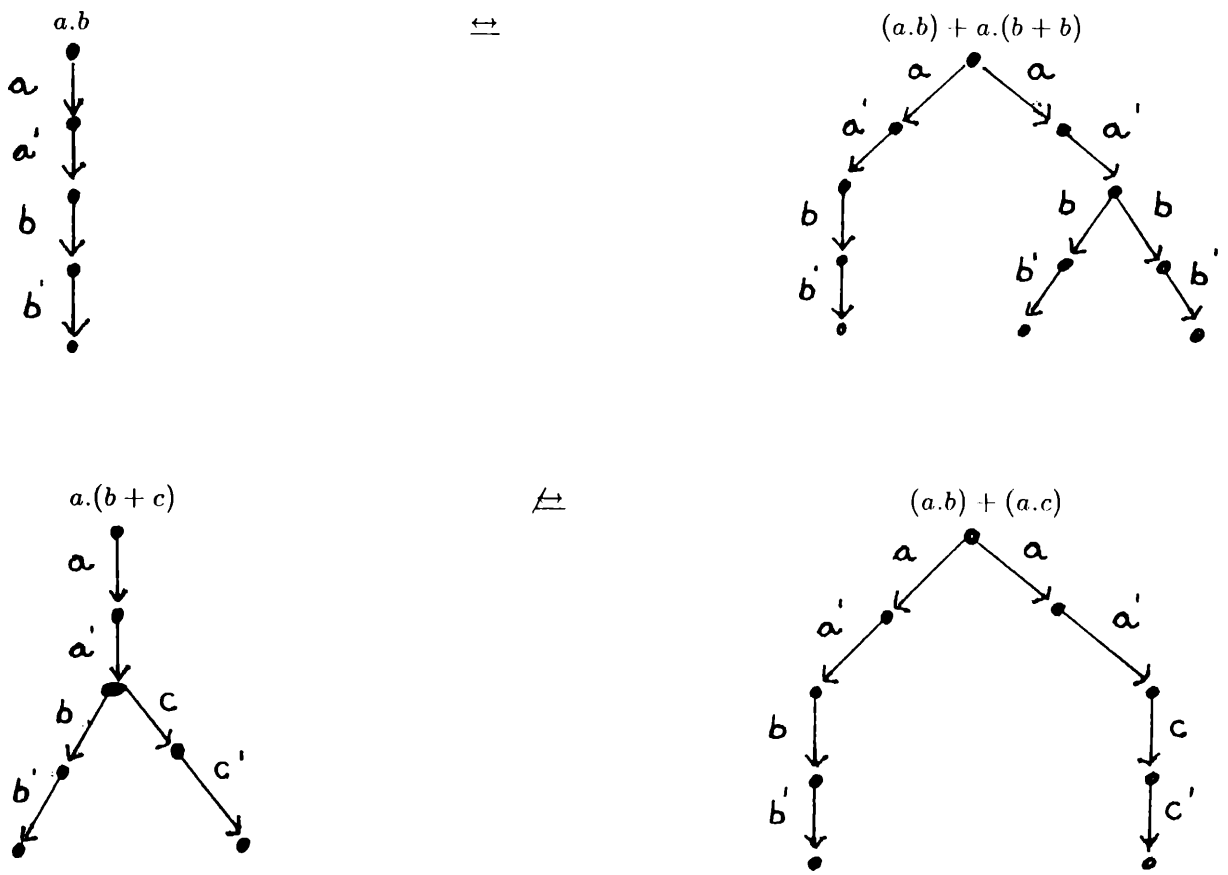
- $$pRq \;\; y \;\; p \xrightarrow{\alpha} p' \implies \exists q' : \; q \xrightarrow{\alpha} q' \;\; y \;\; p'Rq'$$

- $$pRq \;\; y \;\; q \xrightarrow{\alpha} q' \implies \exists p' : \; p \xrightarrow{\alpha} p' \;\; y \;\; p'Rq'$$

Dos procesos p y q son *bisimilares* ($p \leftrightarrows q$) si existe una bisimulación R con p R q.

■

Ejemplo 6.1



■

Se define congruencia de la siguiente manera.

Definición 6.2 (Congruencia) Sea una relación de equivalencia $\approx$ sobre un conjunto X y una funcion $f : x \longrightarrow x$. Se dice que $\approx$ *es una congruencia para* f (o que f *es compatible con* $\approx$) si y sólo si

$$\forall x, y \in X . x \approx y \implies f(x) \approx f(y)$$

■

6.1 Bisimulacion en los 2ST

El operador de prioridad es compatible con la bisimulación.

Teorema 6.1 $\forall p, q, \theta, p \leftrightarrows q \implies \theta(p) \leftrightarrows \theta(q)$

Sea R: $p \leftrightarrows q$. Se define S como

$$S : \theta(p)\ S\ \theta(q) \quad sii \quad p\ R\ q$$

Lema 6.2 *Lema Previo*

$$p \leftrightarrows q \implies \forall b \in \mathbf{A} p \overset{b}{\not\longrightarrow} \leftrightarrow q \overset{b}{\not\longrightarrow}$$

Prueba. Aplicando la definicion de bisimulación.

$$(\exists R.p \xrightarrow{\alpha} p' \ \ y \ \ p\ R\ q \implies \exists q'.q \xrightarrow{\alpha} q' \ \ y \ \ p'\ R\ q')$$

$\Longrightarrow$

$$\exists S.\theta(p) \xrightarrow{\alpha} \theta(p') \ \ y \ \ \theta(p)\ S\ \theta(q) \implies \exists q'.\theta(q) \xrightarrow{\alpha} \theta(q') \ \ y \ \ \theta(p')\ S\ \theta(q'))$$

- $\alpha = a$

$$\theta(p) \xrightarrow{a} \theta(p')$$

$$sii$$

$$p \xrightarrow{a} p' \ \ y \ \ \forall b > a.p \overset{b}{\not\longrightarrow} \quad (porPRI - I)$$

Como pRq,

$$\exists q'.q \xrightarrow{a} q' y p' R q'$$

Ademas por Lema

$$\forall b > a\ q \overset{b}{\not\longrightarrow}$$

Por lo tanto, aplicando PRI-I

$$\theta(q) \xrightarrow{a} \theta(q')$$

y por definición de S

$$\theta(p') \ \ S \ \ \theta(q')$$

PRI I es la unica regla aplicable. Por lo tanto se cumple el teorema para $\alpha = a$

- $\alpha = a^i$

$$\theta(p) \xrightarrow{a^i} \theta(p')$$

$$sii$$

$$p \xrightarrow{a^i} p' \ \ (porPRI - II)$$

Como p R q, $\exists q' \ . \ q \xrightarrow{a^i} q'$ y p' R q'

Aplicando PRI II

$$\theta(q) \xrightarrow{a^i} \theta(q')$$

$$y$$

$$\theta(p') \ \ S \ \ \theta(q') \ \ sii \ \ p' \ \ R \ \ q'$$

Por lo tanto se cumple el teorema.

■

7 Conclusiones

En este trabajo se ha explorado la definición de operadores de prioridad en un modelo verdaderamente paralelo. Los operadores de prioridad son bien conocidos en la literatura, y en los STE se representan como la eliminación, entre las transiciones que salen de cada estado, de aquellas cuyas etiquetas no son maximales en el orden parcial de prioridad entre las acciones. En ese sentido, las transiciones "compiten", y sólo aquellas que tienen etiquetas maximales de acuerdo al preorden dado son preservadas.

Se ha trabajado aquí sobre los 2ST, un modelo muy cercano a los STE, pero donde las acciones están divididas en inicio y final. Se propusieron cuatro definiciones diferentes del operador de prioridad θ, según la competencia entre las transiciones involucre los inicios y/o los finales de las transiciones.

Se ha prestado especial interés a la más simple entre ellas, donde sólo los inicios de las acciones compiten entre ellos. Se mostró allí que se cumple la propiedad esencial de los 2ST, aquella que dice que todo las transiciones que se inician, para cualquier evolución del sistema, terminan en algún momento. La propiedad es inclusive algo más fuerte: toda acción no terminada puede finalizar en cualquier momento que se decida. Esto permite dar una transformación natural de los 2STen los STE.

En este trabajo se definió una transformación natural de los 2ST en los STE , para aquellos 2ST que no contienen operadores paralelos. Se comprobó que la definición de prioridad dada conmuta con esta transformación. Eso asegura en algun sentido que la operación de prioridad definida representa la misma intuición que la definida en los STE.

Se ha estudiado también con cuidado el caso donde sólo los finales compiten. En este caso, la propiedad que se encuentra es algo mas complicada: para toda transición iniciada, toda evolución ulterior del sistema que no la termine puede completarse con una evolución (que eventualmente involucra varios pasos) de manera que la transición pueda finalizarse. En este sentido este operador es mas complejo que el anterior. Nos permite llegar a 2ST cuya traducción en terminos de STE no es directa.

Se han estudiado las propiedades de congruencia que cumple el operador θ definido cuando compiten sólo inicios, comprobando que es compatible con la bisimulación.

Parte 2

Operadores de prioridad basados en un orden entre procesos

Operadores de prioridad basados en un orden entre procesos

Abstract

En este trabajo se estudian las prioridades en sistemas de verdadero paralelismo mediante la introducción de operadores de prioridad que actuan sobre los procesos. En base a la definición del operador de alternativa con prioridad propuesto en [CW91], se incluye uno de comportamiento similir en una extensión de los Sistemas de Transiciones STE y ST. El nuevo operador permite la ejecucioń de su segunda componente sólo cuando el ambiente no propone una acción que la primer componente pueda ejecutar. Se da una nueva semántica operacional donde cada transición tiene registro de las acciones que el ambiente propone. El concepto de acciones que un ambiente propone está basada en la de CCS. Se incluye además un segundo operador de prioridad, el operador paralelo con prioridad , que en caso de no haber comunicación entre los procesos, sólo permitirá la ejecución de la segunda componente, si la primer componente no puede iniciar ninguna acción. Se define la noción de bisimulación sobre los operadores definidos comprobando que es una congruencia.

1 Introducción

En este trabajo nos interesamos en la *semántica de sistemas reactivos* [Pnu85]. Estos sistemas (cuyos ejemplos clásicos son los sistemas de control industrial, los manejadores de redes de computadoras y los componentes de los sistemas operativos) tienen como vocación el mantener con el medio una cierta interacción.

No es adecuado entonces considerarlos simplemente como mecanizaciones del cálculo de una función o una relación, como es el caso de otros sistemas, que por ello los llamamos *funcionales* o *relacionales*. Los ejemplos típicos de estos últimos sistemas son los programas de cálculo numérico, los compiladores y todos aquellos cuya interacción con el medio se reduce a un único punto de entrada de datos y a uno de salida de resultados.

Los modelos más comunes para la semántica de sistemas reactivos están basados en los *Sistemas de Transiciones Etiquetados* (STE) [Kel76], que son grafos dirigidos donde los nodos representan los estados del sistema, y las flechas las trancisiones entre los estados, etiquetadas por acciones en las que se basa la interacción del sistema con el medio.

En los STE, la ejecución de un programa puede verse como una sucesión de estados, y de flechas etiquetadas que unen esos estados, es decir, un camino en el grafo que parte del estado inicial y llega a uno de los posibles estados finales:

$$i \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} s_n$$

Una variante en los STE es incluir en cada transición la información sobre las formas de comunicación que estan ofreciendo otros subprocesos. Es decir, cada acción se ejecuta

en un ambiente que propone una serie de acciones para comunicarse con otros subprocesos. Según esta información, cada subproceso elige qué hacer. Cuando dos procesos se ejecutan en paralelo debe establecerse una especie de intercambio de información entre ellos que permite saber qué es lo que puede hacer uno según la propuesta del otro. Esta información estará contenida en el ambiente. En este trabajo definimos un sistema de transición que contiene la noción de ambiente. Este sistema, que llamaremos STE^R, es una extensión de los STE. Aquí, la relación de transición no es sólo la ejecución de una acción que de un estado pasa a otro estado, sino *la ejecución de una acción bajo un ambiente de posibles acciones a ejecutar* que de un estado pasa a otro. La ejecución de un programa puede verse como una sucesión de estados, y de flechas etiquetadas que unen esos estados.Las etiquetas serán pares conformados con la acción a ejecutar y el ambiente donde se está ejecutando. Se define un sistema de ecuaciones que da la solución al intercambio de información que necesitan dos procesos que están ejecutándose en paralelo para saber si hay intenciones de comunicación. Este sistema de ecuaciones puede ser mecanizado mediante la definición de un Funcional. Ya no será necesaria la evaluación exhaustiva de todos los casos para llegar al intercambio adecuado.

La principal limitación en el uso de los STE^R es que no se pueden representar aquí dos acciones desarrollándose al mismo tiempo, ya que se debe llegar a un estado (terminando una acción) para poder iniciar una nueva acción. Decimos por esto que los STE^R no pueden representar directamente el paralelismo. Aunque sí puede hacerlo indirectamente, representando la ejecución de a en paralelo con b como la secuencia a, b o bien la secuencia b, a. Por razones obvias llamamos de entrelazamiento (en ingles *interleaving*) a esta representación del paralelismo en los STE.

Hay muchos modelos que permiten representar el paralelismo sin utilizar el entrelazamiento, que son llamados en general *sistemas verdaderamente paralelos* (en inglés *truly concurrent*). Algunos de ellos son las Redes de Petri [Pet62, Rei85], las Estructuras de Eventos [Win87] y los Sistemas de Trancisiones Asíncronos [Bed87, Shi85]. En este trabajo nos interesamos en un modelo propuesto recientemente por *Gorrieri y Laneve* [GL95] que llamaremos aqui 2ST.

La idea esencial es que un 2ST permite expresar la noción elemental de *duración*, donde varias fases de una acción pueden ser ejecutadas entre su inicio y su final. Para ello cada acción sera representada por su *inicio* y su *finalización* correspondiente. En ese sentido un 2ST puede verse como un STE donde las etiquetas de las flechas no son acciones $(a, b \ldots)$, sino inicios de acciones (que anotaremos como las acciones, $a, b \ldots$) y los finales correspondientes (que anotaremos con superíndices $a^i, b^j \ldots$, donde i y j representan cual es exactamente la acción que está terminando entre las acciones a y b iniciadas en el pasado).

Esto nos permite representar fácilmente ejecuciones donde las acciones a y b son ejecutadas en paralelo. Una de ellos es, por ejemplo

$$i \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{b^1} s_3 \xrightarrow{a^3} s_4$$

Al llegar al estado s_2, se han ejecutado los inicios de las acciones a y b, pero no el final de ninguna de ellas. Por tanto, *ambas acciones se están ejecutando en paralelo.* La tercera transición, etiquetada b^1 es el final de la acción b iniciada un paso atrás, mientras la transición etiquetada con a^3 es el fin de la acción a iniciada 3 pasos atrás. En este ejemplo los superíndices no parecen estrictamente necesarios, bastaría con poder diferenciar inicios de finales. Sin

embargo, en el caso en que haya varias acciones a iniciadas, es imprescindible, cuando ocurre el final de una de ellas, saber de quien se trata: en ese caso el superíndice brinda la información necesaria.

En este trabajo definimos un sistema de transición que es una extensión de los 2ST y que contiene la noción de ambiente. Este sistema que llamaremos $2ST^R$ tendra todas las ventajas de los 2ST ya que se puede representar el verdadero paralelismo, más la información del ambiente en cada transición. Es decir, cada acción que se ejecuta, conoce qué subprocesos quieren establecer comunicación y de qué manera.

Los operadores de prioridad forman una familia interesante entre los operadores sobre los ST. En este trabajo desarrollamos el estudio de operadores de prioridad que se construyen a partir de un cierto orden entre los subprocesos. Lo que intenta resolver la prioridad entre procesos es un caso como el siguiente:

Se tiene un controlador C que debe elegir no determinísticamente entre ejecutar la acción *tick* o la acción *tock*. Esta elección se repite continuamente hasta que un sensor T ejecute la accioń *shut_down*. Cuando esto ocurre, C debe dejar inmediatamente de ejecutar *tick* y *tock*. El controlador C podría ser una unidad de control que regula una reacción química en una planta. T sería un sensor que detecta exceso de temperatura. Cuando el nivel tope de temperatura es sobrepasado, T impide de alguna manera que C siga trabajando. Esto podría escribirse de la siguiente manera:

$$
\begin{array}{lcl}
SYS & \stackrel{\text{def}}{=} & (C|_S T) \\
T & \stackrel{\text{def}}{=} & push\_button \ . \ shut\_down \\
C & \stackrel{\text{def}}{=} & tick.C + tock.C + shut\_down \\
S & = & \{shut\_down\}
\end{array}
$$

La definición de arriba indica que C puede hacer *tick* o *tock* reiteradas veces, o terminar. Pero cuando T esté listo para hacer el *shut_down*, nada impide que C ejecute una serie de *tick* y *tock* extras debido al no determinismo del $+$.

Se necesita para resolver este problema un operador de prioridad que permita que ni bien se ejecuta el *shut_down*, C no pueda elegir hacer *tick* y *tock*. Hay un subproceso que quiere comuinicarse y necesita tener prioridad sobre los otros subprocesos.

Para esto es necesario que C conozca el *ambiente* donde se está ejecutando, para poder saber que otro proceso está tratando de establecer comunicación y así poder decidir, con algún operador que lo permita, ejecutar *shut_down*. En este trabajo definimos el *operador de alternativa con prioridad*, $+_>$ sobre los STE^R. $+_>$ permitirá la ejecución de la segunda componente sólo en el caso que la primer componente no pueda hacer ninguna de las acciones que ofrece el ambiente. En este caso, hay un subproceso que desea comunicarse y por lo tanto, se da prioridad a la ejecución de la primer componente. También definimos este concepto en un modelo de verdadero paralelismo, $2ST^R$.

Veremos que esta definicion de prioridad, es interesante tanto en modelos de verdadero paralelismo como en modelos que no son de verdadero paralelismo. Lo que no se capturó con el $+_>$ es la noción de prioridad que permite que exista un orden de ejecución entre dos subprocesos que no desean comunicarse . Por ejemplo:

Dos personas desean ser atendidas en una farmacia que cuenta con dos empleados. Una de ellas tiene el número de orden uno, y la otra el número dos. El empleado que primero se libere,

comenzará a atender a la persona con el número uno, pues tiene prioridad sobre la otra. Si mientras el cliente con el número uno está siendo atendido el otro empleado se libera, podrá atender a la persona con el número dos, no siendo necesario esperar a que la otra termine de ser atendida. Incluso la segunda persona puede terminar primero, si realiza su compra más rápido. El $+_>$ no alcanza para representar este caso pues decide entre uno u otro subproceso,

pero no permite ejecutar ambos. Definimos en los $2ST^R$ un nuevo operador, paralelo con

prioridad, $|_>$, que permita capturar esta noción de prioridad Probaremos que en este marco se conserva la propiedad que cumplen los 2ST (probadsas en la Parte 1) de que toda acción que comienza termina. Probaremos que ambos operadores definidos son compatibles con la bisimulación tanto en STE^R como en $2ST^R$.

El trabajo está organizado de la siguiente manera:

En la sección 2 se introduce el lenguaje $\mathcal{K}$ sobre el que se definirán los sistemas de transición.

En la sección 3 se introducen los modelos STE^R y $2ST^R$ que incluyen la noción de ambiente en las transiciones, necesaria para el intercambio de información entre los procesos. En 3.2, se muestra un caso de conflicto que según las reglas definidas, podrá ser evitado. En 3.3 se definen algunas propiedades para el ambiente y las ecuaciones que permiten el intercambio de información entre procesos en cada momento. Este intercambio puede ser mecanizado mediante la definición de un funcional para el sistema de ecuaciones que también es definido en 3.4.

En la sección 4 se define la noción de bisimulación sobre el operador de prioridad definido comprobando que es una congruencia.

En la sección 5 se define la semántica de $\mathcal{K}$ sobre los $2ST^R$ y se dan las reglas que definen el comportamiento de los procesos. En 5.2 se amplía el lenguaje $\mathcal{K}$ a $\mathcal{K}^1$ introduciendo el operador $|_{>s}$ y se agregan las reglas necesarias para completar el comportamimento de los procesos.

En la sección 6 se comprueba que la propiedad que cumplen los 2ST de que toda acción que comienza puede ser finaliza, se conserva en los $2ST^R$

En la sección 7 se define la noción de bisimulación sobre los operador de prioridad definidos comprobando que en todos los casos es una congruencia.

Finalmente se establecen conclusiónes del trabajo realizado.

2 El lenguaje

El lenguaje que se utilizará para describir los procesos, se llamará $\mathcal{K}$. Un término de $\mathcal{K}$ se define de la siguiente manera:

$$t ::= a \mid a^1 \mid t.t' \mid t + t' \mid t|_s t' \mid [k]t \mid \varepsilon \mid t +_> t'$$

donde a es una acción o el inicio de una acción. $a \in \mathbf{A}$ que es el conjunto de acciones o eventos.

$t.t'$ es la composición secuencial de procesos.

$t + t'$ es la suma no determinista de procesos.

$t|_s t'$ es la ejecución en paralelo de procesos, siendo S el conjunto de acciones que establecen la comunicación.

$t +_> t'$ es la suma con priordad donde la segunda componente sólo podrá ejecutar una de sus acciónes cuando la primera no pueda hacerlo. Para saber cuándo un proceso puede ejecutar una acción es necesaria la noción de ambiente que será introducida en la próxima sección.

Tanto a^i como $[1]t$ son términos auxiliares que se incluyen para que la definición de los sistemas de transicón ST sea lo más sencilla y clara posible. Según estos dos criterio definimos :

- Términos buenos. Aquellos en los que no aparecen finalizaciones de acciones, ni retardos. Es decir no aparecen subtérminos de la forma a^i o $[1]t$.Estos términos serán llamados $\mathcal{L}_b$
- Aquellos en los que aparecen finalizaciones de acciones y retardos. No se consideran buenos pues no se puede saber a priori si a^i y $[1]t$ provienen de la evolución de otro proceso cuyos términos estan bien formados o es un término mal formado.

3 STE^R

Los sistemas de transiciónes Etiquetadas STE^R son una extensión de los STE. En estos sistemas, la relación de transición no es sólo la ejecución de una acción que de un estado pasa a otro estado, sino *la ejecución de una acción bajo un ambiente de posibles acciones a ejecutar* que de un estado pasa a otro.

Definición 3.1 (Sistema de transición STE^R) Un STE^R, es una estructura $G = (S, i, \mathbf{A}, \dashrightarrow)$ donde

- $S = \{i, u, s, \ldots\}$ es el conjunto de *estados* donde i es el *estado inicial*,
- $\mathbf{A} = \{a, b, e, \surd, \ldots\}$ es el conjunto de *eventos* o *acciones*, con una accion destacada $\surd$ que indica terminación satisfactoria.
- $\dashrightarrow \subseteq S \times \mathcal{P}(\mathbf{A}) \times \mathbf{A} \times S$, es la *relación de transición* y se escribe

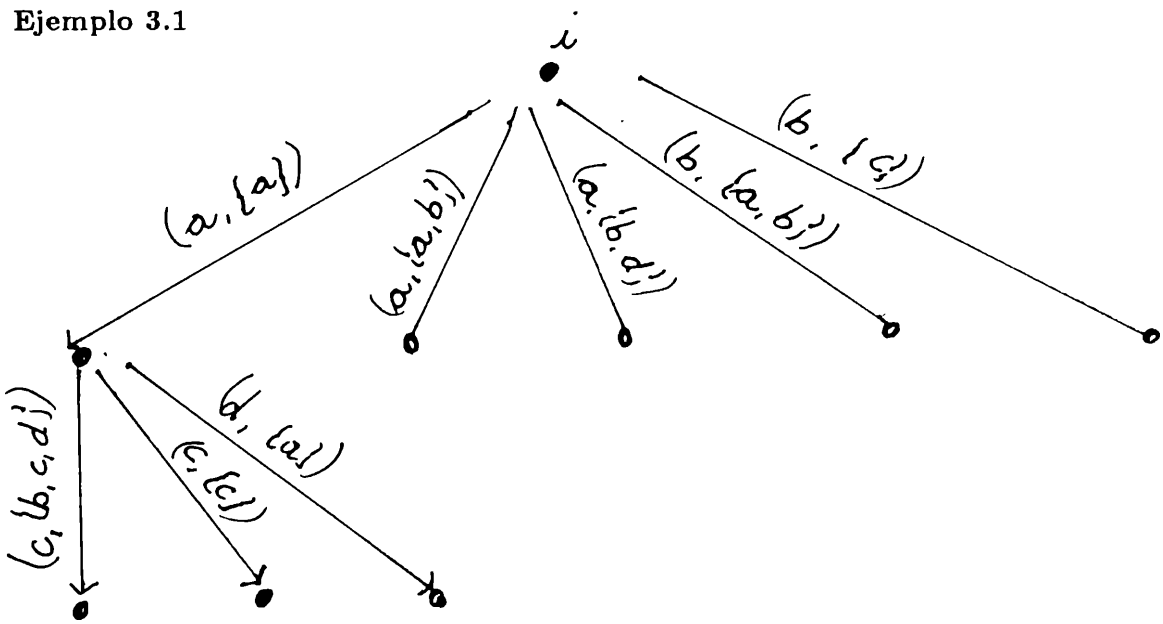
$$\vdash_R p \overset{a}{\dashrightarrow} p'$$

para decir que en el ambiente que *propone* ejecutar las acciones R el proceso p puede hacer la acción a y volverse p'. De este ambiente depende si una acción puede o no ser ejecutada. El ambiente R proporciona en cada momento las acciones mediante las cuales un proceso puede comunicarse.

■

Intuitivamente se podría decir que en los STE clásicos, de un estado parten flechas con las acciones que se pueden ejecutar, habiendo una flecha para cada acción. Todas las acciones que figuran en el grafo son posibles de ejecutar. En los STE^R cada transición será un par formado por la acción a ejecutar y el ambiente donde se ejecutará. La cantidad de flechas que salen de un estado será una por cada acción en cada ambiente donde sea posible su ejecución. La posible ejecución de una acción dependerá del ambiente. Aquí no se puede decir que si la acción figura en el grafo, su ejecución siempre es posible.

Ejemplo 3.1



Se puede observar que partiendo del estado inicial, la ejecución de a es posible en los ambiente {a}, {a, b}, {b, d}, y no es posible en el ambiente {c}, ya que no hay una flecha en el grafo que lo indique.

■

3.1 Semántica operacional de $\mathcal{K}$ sobre los STE^R

El comportamiento de los STE^R se formaliza con las siguientes reglas:

ACT $\dfrac{}{\vdash_R a \xrightarrow{a} \surd}$	
ALT I $\dfrac{\vdash_R p \xrightarrow{a} p'}{\vdash_R p + q \xrightarrow{a} p'}$	ALT II $\dfrac{\vdash_R q \xrightarrow{a} q'}{\vdash_R p + q \xrightarrow{a} q'}$
PRIALT I $\dfrac{\vdash_R p \xrightarrow{a} p'}{\vdash_R p +_> q \xrightarrow{a} p'}$	PRIALT II $\dfrac{\vdash_R q \xrightarrow{a} q' \;\; y \;\; \forall b \in R. \vdash_R p \not\xrightarrow{b}}{\vdash_R p +_> q \xrightarrow{a} q'}$
SEQ I $\dfrac{\vdash_R p \xrightarrow{a} p'}{\vdash_R p \cdot q \xrightarrow{a} p' \cdot q}$	SEQ II $\dfrac{\vdash_R p \xrightarrow{\surd} \delta y \;\; \vdash_R q \xrightarrow{a} q'}{\vdash_R p.q \xrightarrow{a} q'}$
PAR I $\dfrac{\vdash_{R \cup (Y \cap S)} p \xrightarrow{a} p \;\; a \notin S}{\vdash_R p\|_s q \xrightarrow{a} p'\|_s q}$	PAR II $\dfrac{\vdash_{R \cup (X \cap S)} q \xrightarrow{a} q \;\; a \notin S}{\vdash_R p\|_s q \xrightarrow{a} p\|_s q'}$
PARIII $\dfrac{\vdash_{R \cup (Y \cap S)} p \xrightarrow{a} p' \;\; \vdash_{R \cup X)} q \xrightarrow{a} q' \;\; a \in S}{\vdash_R p\|_s q \xrightarrow{a} p'\|_s q'}$	X, Y *satisfacen las ecuaciones* Env_x, Env_y

*Tabla*1 : *Reglas para* STE^R

$$Env_x = \{b \in Ini(p). \vdash_{R \cup (Env_y \cap S)} p \xrightarrow{b}\}$$
$$Env_y = \{b \in Ini(q). \vdash_{R \cup (Env_x \cap S)} q \xrightarrow{b}\}$$

La regla PRIALT I dice que siempre que p pueda ejecutar la acción a en el ambiente R, $p +_> q$ también puede hacerlo.

La regla PRIALT II dice que sólo en el caso que p no pueda ejecutar ninguna de las acciones propuestas por R y q pueda ejecutar a en R , entonces $p +_> q$ podrá ejecutar a. Si p puede ejecutar alguna acción propuesta por R, significa que el ambiente está queriendo comunicarse, y p tiene prioridad para hacerlo. Un ejemplo donde q no puede avanzar es:

$$\vdash_{\{a\}} (a + b.c) \;\; +_> d$$

$$\vdash_{\{b\}} (a + b.c) \ +_> d$$

$$\vdash_{\{a,b\}} (a + b.c) \ +_> d$$

En el primer caso el ambiente propone comunicación por medio de la acción a. En el segundo por medio de la acción b. En el tercero, por medio de las acciónes a o b.

Un ejemplo donde q puede avanzar es:

$$\vdash_{\{\emptyset\}} (a + b.c) \ +_> d$$

$$\vdash_{\{c,d\}} (a + b.c) \ +_> d$$

$$\vdash_{\{a\}} (a \mid_{\{a\}} (b.a)) \ +_> d$$

En el primer caso el ambiente no propone comunicación. En el segundo propone comunicación por medio de las acciónes b o c. En el tercero, por medio de la acción a. p no puede ejecutar ninguna de las acciones que propone el ambiente para comunicarse. Por lo tanto q puede avanzar.

La regla PAR I dice que si en el ambiente compuesto por las acciones de R más las acciones que ofrece q para comunicarse (Y) , p puede hacer a, $p|_S q$ puede hacer a. Las acciones que ofrece q para comunicarse, deben ser aquellas que q puede hacer en el ambiente R mas lo que propone p para comunicarse (X). De este conjunto sólo interesan las que pertenecen a S (Y $\cap$ S; X $\cap$ S) ya que son las que permiten la comunicación. De aquí la definición de las ecuaciones Env_x, Env_y.

Ejemplo:

$$\vdash_{\{\emptyset\}} (a \mid_\emptyset (b +_> c)) \ \mid_{\{a,b\}} \ (a +_> b) \ \overset{c?}{\cdots\cdots\rightarrow}$$

Este proceso podrá ejecutar c si p puede hacerlo aplicando PAR I, ya que c $\notin$ S. Para eso hay que analizar si

$$\vdash_{RU(Y \cap S)} p \ \overset{c}{\cdots\cdots\rightarrow}$$

Puede observarse en este ejemplo que si q hiciera su oferta en el ambiente R, es decir, sin tener en cuenta la oferta de p para comunicarse, se tendría:

$$\vdash_{R \, U \, \{a,b\}} p \ \overset{c}{\cdots\cdots\rightarrow}$$

ya que {a, b} son las acciones por medio de las que se puede comunicar q en R. De esta manera, aplicando PAR II y luego PRIALT II, se llegará a la conclusión de que la ejecución de c no es posible. Si se analiza lo que puede hacer q en el ambiente R U la propuesta de p, la propuesta de q será sólo {a} ya que habrá que analizar:

$$\{b \in Ini(q). \vdash_{R \cup (X \cap S)} q \ \overset{b}{\cdots\cdots\rightarrow}\}$$

$$\{b \in Ini(q). \vdash_{R \cup \{a,b\}} a +_> b \ \overset{b}{\cdots\cdots\rightarrow}\}$$

Aplicando PRIALT I y PRIALT II el conjunto resultante es {a}.

En este caso, la ejecución de c es posible. Esto es debido a que se tuvo en cuenta la propuesta de p para obtener la propuesta de q.

La regla PAR II debe interpretarse de manera similar a PAR I.

PAR III dice que si p puede realizar la acción $a \in S$ si en el ambiente R U las acciones que ofrece q para comunicarse y q puede realiar esa misma acción en el ambiente R U las acciones que ofrece p para comunicarse, entonces ambos procesos pueden comunicarse ejecutando a.

Por ejemplo

$$\vdash_R (a+ > b) \quad |_{\{a,b\}} \quad (a + b)$$

p y q pueden comunicarse a través de a ya que

$$\vdash_{RU(Y \cap S)} p \overset{a}{\cdots\cdots\rightarrow} y \ \vdash_{RU(X \cap S)} q \overset{a}{\cdots\cdots\rightarrow}$$

pero no pueden comunicarse a través de b ya que

$$\vdash_{RU(Y \cap S)} p \overset{b}{\cdots\not\rightarrow}$$

3.2 Un caso de conflicto

Sería deseable que según las reglas dadas, $a +_> b \ |_{\{a,b\}} \ b +_> a$ no avance.

$Sea \ P = p \ |_s \ q$ donde

$p = a +_> b$ y $q = b +_> a$ y $S = \{a, b\}$

$\vdash_R p|_{\{a,b\}} q \cdots\cdots\rightarrow$

sii

1 $\vdash_{R \cup (Y \cap S)} p \overset{a}{\cdots\cdots\rightarrow} y \ \vdash_{R \cup (X \cap S)} q \overset{a}{\cdots\cdots\rightarrow}$ $(PARIII)$

2 $\vdash_{R \cup (Y \cap S)} p \overset{b}{\cdots\cdots\rightarrow} y \ \vdash_{R \cup (X \cap S)} q \overset{b}{\cdots\cdots\rightarrow}$ $PARIII)$

3 $\vdash_{R \cup (Y \cap S)} p \overset{c}{\cdots\cdots\rightarrow} c \neq a, b$ $(PARI)$

4 $\vdash_{R \cup (X \cap S)} q \overset{c}{\cdots\cdots\rightarrow} c \neq a, b$ $(PARII)$

Las opciones 3 y 4 no se pueden aplicar pues todas las acciones pertenecen a S.

- 1

$$\vdash_{\{a,b\}} b +_> a \overset{a}{\cdots\cdots\rightarrow}$$

$$sii$$

$$\vdash_{\{a,b\}} a \overset{a}{\cdots\cdots\rightarrow} y \ \forall c \in \{a, b\}. \vdash_{a,b} b \overset{c}{\cdots\not\rightarrow} (PRIALTII)$$

Pero $\vdash_{a,b} b \overset{b}{\cdots\cdots\rightarrow}$. Por lo tanto con la opcion 1 no se puede avanzar

- 2

$$\vdash_{\{a,b\}} a +_> b \overset{b}{\cdots\cdots\rightarrow}$$

$$sii$$

$$\vdash_{\{a,b\}} b \overset{b}{\cdots\cdots\rightarrow} y \ \forall c \in \{a, b\}. \vdash_{a,b} a \overset{c}{\cdots\not\rightarrow} (PRIALTII)$$

Pero $\vdash_{a,b} a \overset{a}{\cdots\cdots\rightarrow}$. Por lo tanto con la opcion 2 no se puede avanzar

Como 1 , 2, 3 y 4 son las unicas posibles evoluciones de P, se pude concluir que P no avanza.

3.3 El ambiente

El ambiente proporciona a cada proceso información sobre los demás procesos que se están ejecutando en ese mismo momento. Esto resulta imprescinible para poder saber si se está tratando de establecer comunicación, sobre todo si dicha comunicación tiene prioridad sobre las demás acciones. Es oportuno destacar que cuanto más grande es el ambiente, más posibilidades hay de comunicación. Es decir, es más complejo para un proceso ejecutar una acción de manera independiente. Según estas consideraciones, hay dos propiedades que se pueden asegurar:

- En un proceso que no contenga operadores paralelos todos los términos y subtérminos se ejecutarán en el mismo ambiente ya que es el operador paralelo el que provoca el cambio de ambiente.

  Prueba. Cuando se contruye la prueba de una derivación, se consideran distintos subtérminos en ambientes que pueden ser diferentes. Mirando las reglas, se puede verificar que las únicas que introducen cambios en el ambiente son PAR I, PAR II y PAR III. Al no haber combinadores paralelos, estas reglas no se utilizarán.Por lo tanto el ambiente siempre es el mismo. ∎

- Un proceso que es capaz de ejecutar una acción en un ambiente R determinado, puede ejecutar esa misma acción en cualquier subconjunto de R. Esto es debido a que un subconjunto de R está proponiendo menos posibilidades de comunicación que R. Si un proceso es capaz de ejecutar la acción a en el conjunto más grande de posibles comunicaciones a establecer, también podra hacerlo en cualquier subconjunto de este. Esta propiedad queda formalizada con el siguiente teorema

  Teorema 3.1 *Sea p un proceso. R un ambinete.* $R_i \subseteq \mathcal{P}(R)$

  $$\forall p, R. \vdash_R p \xrightarrow{a} p' \implies \vdash_{R_i} p \xrightarrow{a} p'$$

  No se cumple la propiedad inversa. Es decir, si un proceso es capaz de ejecutar una acción en un ambiente R, no se puede asegurar que pueda hacerlo en $R \cup R_2$.

  Prueba.

  $$\vdash_\emptyset a +_> b \xrightarrow{b}$$

  pero

  $$\vdash_{\{a\}} a +_> b \not\xrightarrow{b}$$

  ∎

3.4 Las ecuaciónes de ambiente Env_x, Env_y

Cuando dos procesos actúan en paralelo, cada uno integra el ambiente donde el otro se ejecuta. Esta situación es la planteada mediante las ecuaciones

$$\begin{aligned} Env_x &= \{b \in Ini(p).\vdash_{R\cup(Env_y\cap S)} p \overset{b}{\cdots\!\!\rightarrow}\} \\ Env_y &= \{b \in Ini(q).\vdash_{R\cup(Env_x\cap S)} q \overset{b}{\cdots\!\!\rightarrow}\} \end{aligned}$$

Cuando (X, Y) es una solución de estas ecuaciones, p puede hacer cualquier acción de X a condición de que q pueda hacer cualquier acción de Y. A su vez, q puede hacer cualquier acción de Y a condición de que p pueda hacer cualquier acción de X. Hay un acuerdo entre p y q de lo que cada uno puede hacer en el ambiente ofrecido por el otro. Observando las ecuaciones se puede ver que las posibles soluciones serán siempre parejas (A, B) tal que A $\subseteq$ Ini(p) y B $\subseteq$ Ini(q). Dada esta representación ecuacional es necesario preguntarse: Es sencillo chequear si algo es solución de ellas? Cómo hacerlo?.

El conjunto de soluciones factibles de la ecuación es finito, es el conjunto de pares (A, B) tales que A $\subseteq$ Ini(p) y B $\subseteq$ Ini(q), y estos dos son siempre finitos. La búsqueda de la solución puede hacerse entonces por revisión exhaustiva. Por lo tanto, resolver estas ecuaciones es computable. La pregunta que surge es si se podría encontrar una solución con formato de regla que permita encontrar el conjunto sin una búsqueda exhaustiva del mismo. Una primera aproximación a esto, sería la definición de un funcional para el sistema de ecuaciones:

$$F(Y,X) \overset{\text{def}}{=} \{\{b \in Ini(p).\vdash_{R\cup(S\cap Y)} p \overset{b}{\cdots\!\!\rightarrow}\}, \{b \in Ini(q).\vdash_{R\cup(S\cap X)} q \overset{b}{\cdots\!\!\rightarrow}\}\}$$

El punto fijo del funcional es la solución de la ecuación y es exactamente lo que cada proceso puede hacer en un ambiente R dado. Por esto se podría ver si un proceso determinado puede ejecutar la acción a de dos maneras:

- Aplicando las reglas.
- Viendo si a esta en el conjunto devuelto por el funcional.

Por ejemplo:

$$\vdash_{\{\oslash\}} (a \mid_{\oslash} (b +_{>} c)) \mid_{a,b} (a +_{>} b) \overset{c?}{\cdots\!\!\rightarrow}$$

Se parte de $(\oslash, \oslash)$.El primer paso es

$$F(\oslash, \oslash)$$

$$=$$

$$(\{a, b, c\}, \{a, b\})$$

Es decir, que cuando q no propone nada para comunicarse, p puede hacer {a, b, c}. Y cuando p no propone nada para comunicarse q puede hacer {a, b} El proximo paso es

$$F(\{a, b\}, \{a, b, c\})$$

$$=$$

$$(\{a, b\}, \{a\})$$

Es decir, que cuando q propone {a, b}, p puede hacer {a, b}. Y cuando p propone {a, b, c}, q puede hacer {a}

El proximo paso es

$$F(\{a\}, \{a, b\})$$

$$=$$

$$(\{a, b, c\}, \{a\})$$

El proximo paso es

$$F(\{a\}, \{a, b, c\})$$

$$=$$

$$(\{a, b, c\}, \{a\}$$

Aquí se obtuvo el punto fijo, y se sabe que p puede hacer exactamente las acciones {a, b, c} y q {a}. Por lo tanto la rta a

$$\vdash_{\{\varnothing\}} (a \mid_{\varnothing} (b +_> c)) \mid_{a,b} (a +_> b) \overset{c?}{\dashrightarrow}$$

es sí.

Lo que falta probar para poder decir que el funcional puede ser usado como mecanismo de solución es verificar siempre se llegará al punto fijo y que si no se llega es porque el proceso considerado no tiene solución.

Po ejemplo:

$$(a +_> b) +_> c \mid_{\{a,b\}} (b +_> a) +_> d$$

Si se aplica el funcional reiteradas veces para obtener el punto fijo se obtiene

$$\begin{array}{lcl}
F(\varnothing, \varnothing) & = & (\{a, b, c\}, \{a, b, d\}) \\
F(\{a, b, d\}, \{a, b, c\}) & = & (\{a\}, \{b\}) \\
F(\{b\}, \{a\}) & = & (\{a, b\}, \{a, b\}) \\
F(\{a, b\}, \{a, b\}) & = & (\{a\}, \{b\}) \\
F(\{b\}, \{a\}) & = & (\{a, b\}, \{a, b\}) \\
F(\{a, b\}, \{a, b\}) & = & (\{a\}, \{b\}) \\
F(\{b\}, \{a\}) & = & (\{a, b\}, \{a, b\}) \\
F(\{a, b\}, \{a, b\}) & = & (\{a\}, \{b\})
\end{array}$$

Es decir, se hace un intercambio permanente entre p y q, no permitiendo nunca la ejecución de c y d . Aunque hay subprocesos que causan la traba del proceso, intuitivamente se desearía que se pudieran ejecutar c y d.

Algo parecido ocurre con

$$(a +_> b) + c \mid_{\{a,b\}} (b +_> a) + d$$

$$
\begin{array}{lcl}
F(\emptyset,\emptyset) & = & (\{a,b,c\},\{a,b,d\}) \\
F(\{a,b,d\},\{a,b,c\}) & = & (\{a,c\},\{b,d\}) \\
F(\{b,d\},\{a,c\}) & = & (\{a,b,c\},\{a,b,d\}) \\
F(\{a,b,d\},\{a,b,c\}) & = & (\{a,c\},\{b,d\})
\end{array}
$$

En este caso el intercambio es permanente, pero siempre está presente la posibilidad de ejecutar c y d. El problema aquí es que no se llega al punto fijo. Una solución a esto sería que cuando se entra en ciclos, se puede decir que un proceso puede realizar una acción c si esta acción es realizable en cualquier etapa del ciclo.

En este trabajo no vamos a profundizar más en el Funcional. Se tiene un camino para hallar la solución que es la búsqueda exhaustiva. El funcional aportaría un mecanismo para evitar esta búsqueda. Las consideraciones para poder asegurar que este mecanismo es bueno son muchas y exceden los objetivos de este trabajo.

4 Bisimulación como congruencia sobre los STE^R

En este trabajo nos interesamos en la relación de equivalencia *Bisimulacion* entre procesos. Se dice que dos procesos son bisimilares cuando toda acción que es ejecutada por uno, puede ser ejecutada por el otro, transformándose en procesos bisimilares. Este juego de imitación de un proceso por otro se mantiene así a lo largo de las transiciones. Además el rol del imitado y del imitador pueden intercambiarse en cada instante. Formalmente se define bisimulación de la siguiente manera:

Definición 4.1 (Bisimulación) Sea $a \in \mathbf{A}$ p y $q \in \mathcal{K}$. Una *bisimulacion* es una relación de equivalencia binaria R sobre procesos que satisface :

- $$pRq \;\; y \;\; p \xrightarrow{a} p' \implies \exists q' : \; q \xrightarrow{a} q' \;\; y \;\; p'Rq'$$
- $$pRq \;\; y \;\; q \xrightarrow{a} q' \implies \exists p' : \; p \xrightarrow{a} p' \;\; y \;\; p'Rq'$$

Dos procesos p y q son *bisimilares* ($p \leftrightarrows q$) si existe una bisimulación R con p R q.

■

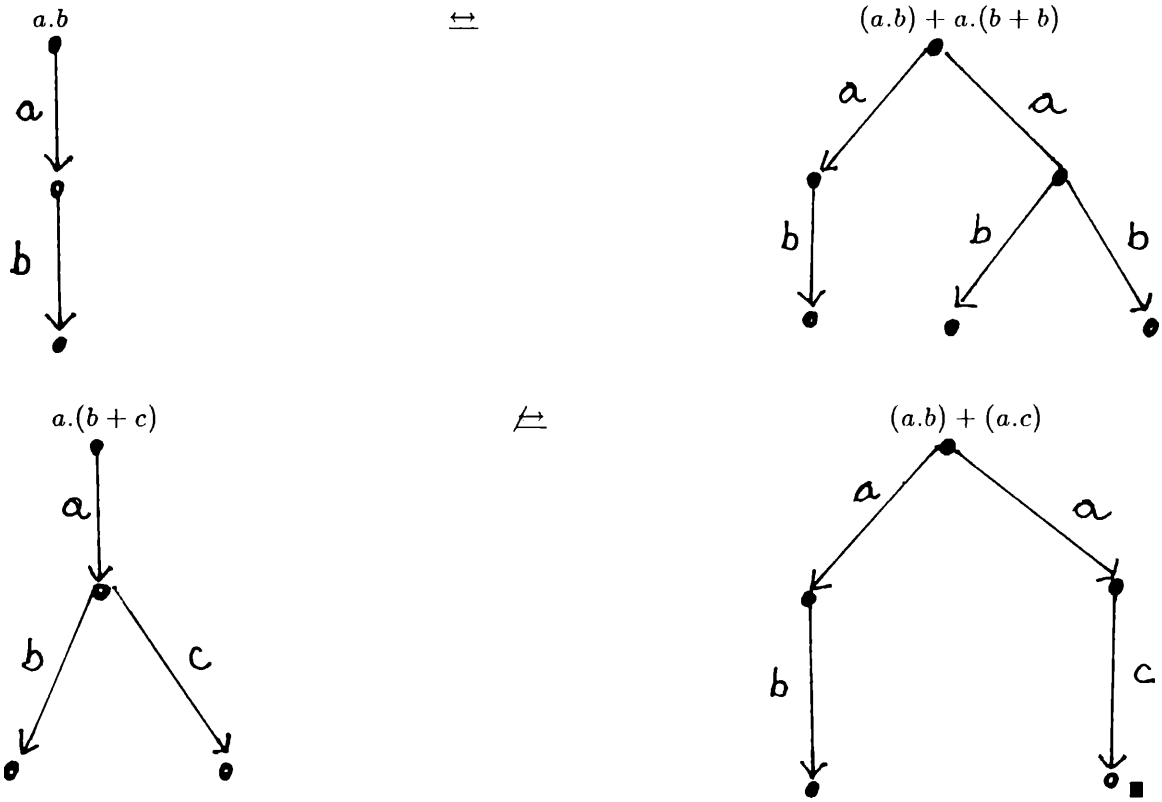
Para poder decir que dos procesos son bisimilares deben serlo en cualquier ambiente. Por ejemplo, no se puede decir que $a +_> b$ es bisimilar a $a + b$ ya que

$$\vdash_\emptyset a +_> b \leftrightarrows \vdash_\emptyset a + b$$

pero

$$\vdash_{\{a\}} a +_> b \not\leftrightarrows \vdash_{\{a\}} a + b$$

Ejemplo 4.1



Se define congruencia de la siguiente manera.

Definición 4.2 (Congruencia) Sea una relación de equivalencia $\approx$ sobre un conjunto X y una funcion $f : x \longrightarrow x$. Se dice que $\approx$ *es una congruencia para* f (o que f *es compatible con* $\approx$) si y sólo si

$$\forall x, y \in X.x \approx y \implies f(x) \approx f(y)$$

■

4.1 El operador de alternativa con prioridad es compatible con la bisimulacion.

Teorema 4.1 *Sean p, q, r, t procesos.*

$\forall p, q, r, t \;\; p \leftrightarrows q \;\; y \;\; r \leftrightarrows t \implies p +_> r \leftrightarrows q +_> t$

Lema 4.2 *Lema Previo*

$$p \leftrightarrows q \quad \Longrightarrow \quad \forall R.(\{b \in Ini(p).\vdash_R p \xrightarrow{b}\} = \{b \in Ini(q).\vdash_R q \xrightarrow{b}\})$$

Prueba. Aplicando la definición de bisimulación.

Sea BR: $p \leftrightarrows q$ y $r \leftrightarrows t$.

Se define BS como

$$BS : p +_> r \;\; BS \;\; q +_> t \;\; sii \;\; p \;\; BR \;\; q \;\; y \;\; r \;\; BR \;\; t$$

$$\exists BR.((\vdash_R p \xrightarrow{a} p' \;\; y \;\; p \; BR \; q \implies \exists q'.\vdash_R q \xrightarrow{a} q' \;\; y \;\; p' \; BR \; q')y$$

$$(\vdash_R r \xrightarrow{a} r' \;\; y \;\; r \; BR \; t \implies \exists t'.\vdash_R t \xrightarrow{a} t' \;\; y \;\; r' \; BR \; t'))$$

$\implies$

$$\exists BS.(\vdash_R p +_> r \xrightarrow{a} w \;\; y \;\; p +_> r \; BS \; q +_> t \implies \exists w'.\vdash_R q +_> t \xrightarrow{a} w' \;\; y \;\; w \; BS \; w')$$

$$\vdash_R p +_> r \xrightarrow{a} w$$

$$sii$$

$$\begin{array}{llll} A & \vdash_R p \xrightarrow{a} p' & porPRIALTI & y \;\; w = p' \\ B & \vdash_R r \xrightarrow{a} r' \;\; y \;\; \forall b \in R \vdash_R p \not\xrightarrow{b} & porPRIALTII & y \;\; w = r' \end{array}$$

- A

  p BR q por HI. Por lo tanto

$$\exists q'.\vdash_R q \xrightarrow{a} q' \;\; y \;\; p' \; BR \; q'$$

  Aplicando PRIALT I

$$\vdash_R q +_> t \xrightarrow{a} q' = w'$$

$$w \;\; BS \;\; w'$$

$$sii$$

$$p' \;\; BR \;\; q' \;\; y \;\; r \;\; BR \;\; t \;\; (porHI)$$

  Por lo tanto

$$p' \;\; BS \;\; q'$$

- B

  r BR t por HI. Por lo tanto

$$\exists t'. \vdash_R t \xrightarrow{a} t' \;\; y \;\; r' \; BR \; t'$$

  Por LEMA $\{b \in Ini(p) \vdash_R p \xrightarrow{b}\} = \{b \in Ini(q) \vdash_R q \xrightarrow{b}\}$.

$$\Longrightarrow$$

$$\{b \in R. \vdash_R p \not\xrightarrow{b}\} = \{b \in R. \vdash_R q \not\xrightarrow{b}\}$$

  Aplicando PRIALT II

$$\vdash_R q +_> t \xrightarrow{a} t' = w'$$

$$w \;\; BS \;\; w'$$

$$sii$$

$$p \;\; BR \;\; q \;\; y \;\; r' \;\; BR \;\; t' \;\; (porHI)$$

  Por lo tanto

$$\vdash_R r' \;\; BS \;\; t'$$

■

5 Operador de alternativa con prioridad en un sitema de verdadero paralelismo

En esta sección se introducirá el operador de alternativa con prioridad en un sistema de verdadero paralelismo. Este sistema, $2ST^{+>}$ es una extensión de los 2ST. El lenguaje sobre el que se definen los términos es $\mathcal{K}$, que fue introducido en la sección 2. Se definirán las reglas que rigen el comportamiento de los procesos y se comprobará que el operador de alternativa con prioridad también es compatible con la bisimulación en este marco. Se comprobará también la propiedad que cumplen los 2ST de que todo proceso que comienza, pude ser finalizado.

Definición 5.1 (Sistema de transición $2ST^R$) Un $2ST^{+>}$, es una estructura $G = (S, i, \mathbf{A}, \longrightarrow)$ donde

- $S = \{i, u, s, \ldots\}$ es el conjunto de *estados* donde i es el *estado inicial*,
- $A = \{a, b, e, \surd, \ldots\}$ es el conjunto de *comienzo de acciones*, $A' = \{a^n | a \in A, n > 0\}$ es el conjunto de *terminación de acciones*; ambos determinan el conjunto de *eventos* $\mathbf{A} = A \cup A' \cup \{\surd\}$ con un evento destacado $\surd$ que indica terminación satisfactoria.

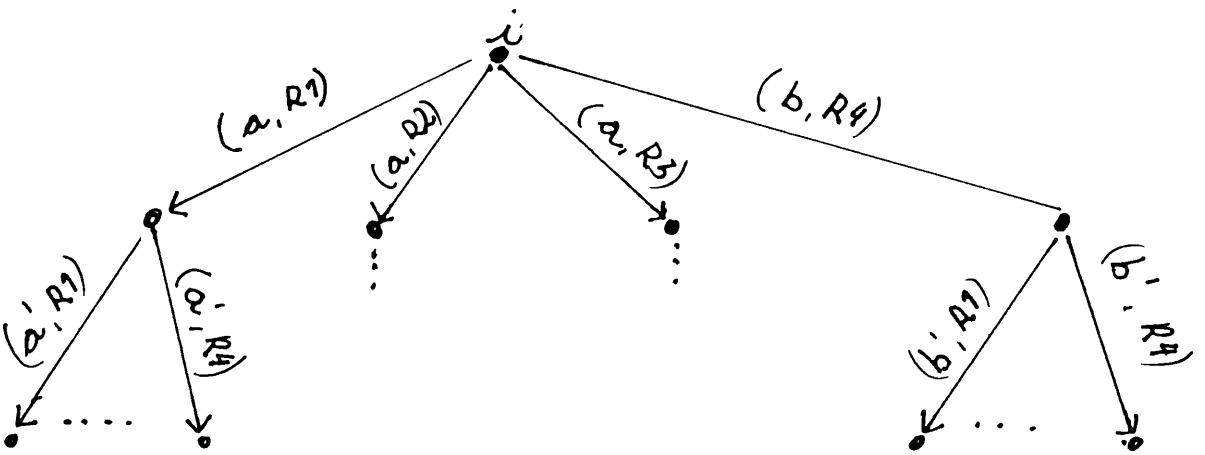
- $\longrightarrow \subseteq S \times \mathcal{P}(\mathbf{A}) \times \mathbf{A} \times S$, es la *relación de transición* y se escribe

$$\vdash_R p \xrightarrow{\alpha} p'$$

para decir que en el ambiente que *propone* ejecutar las acciones R el proceso p puede hacer la acción α y volverse p'. El ambiente R proporciona en cada momento las acciones mediante las cuales un proceso puede comunicarse. De este ambiente depende si una acción puede o no ser ejecutada.

■

Intuitivamente se puede ver que en los 2ST clásicos, de un estado parten flechas con las acciones que se pueden ejecutar, habiendo una flecha para cada acción. Todas las acciones que figuran en el grafo son posibles de ejecutar. En los $2ST^R$ cada transición será un par formado por la acción a ejecutar y el ambiente donde se ejecutará. La cantidad de flechas que salen de un estado será una por cada acción en cada ambiente donde sea posible su ejecución. La posible ejecución de una acción dependerá del ambiente. Aquí no se puede decir que si la acción figura en el grafo, su ejecución siempre es posible. Lo que sí podemos decir es que el final de una acción puede ser ejecutado en cualquier ambiente. Esto nos asegura que una vez iniciada una acción, podrá ser finalizada siempre.



La acción a sólo puede ser iniciada en los ambientes R1, R2 y R3. El inicio de b sólo en R4. Los finales de a y b se ejecutan en todos los ambientes.

5.1 Semántica operacional de $\mathcal{K}$ sobre los $2ST^R$

El comportamiento de los $2ST^R$ se formaliza con las siguientes reglas, donde $\alpha \in \mathbf{A}$ y

$name(a)$	$=$	a	$f(a,n)$	$=$	a
$name(a^n)$	$=$	a	$f(a^n,m)$	$=$	a^n si $n > m$
$name(\surd)$	$=$	$\surd$	$f(a^n,m)$	$=$	a^{n+1} si $n \leq m$
$inc(n)$	$=$	$n+1$			

ACT I $\dfrac{}{\vdash_R a \xrightarrow{a} \surd}$	ACT II $\dfrac{}{\vdash_R a^1 \xrightarrow{a^1} \surd}$
ALT I $\dfrac{\vdash_R p \xrightarrow{\alpha} p'}{\vdash_R p + q \xrightarrow{\alpha} p'}$	ALT II $\dfrac{\vdash_R q \xrightarrow{a} q'}{\vdash_R p + q \xrightarrow{a} q'}$
PRIALT I $\dfrac{\vdash_R p \xrightarrow{\alpha} p'}{\vdash_R p +_> q \xrightarrow{\alpha} p'}$	PRIALT II $\dfrac{\vdash_R q \xrightarrow{\alpha} q' \;\; y \;\; \forall b \in R. \vdash_R p \xrightarrow{b} \not\rightarrow}{\vdash_R p +_> q \xrightarrow{\alpha} q'}$
SEQ I $\dfrac{\vdash_R p \xrightarrow{\alpha} p'}{\vdash_R p \cdot q \xrightarrow{\alpha} p' \cdot q}$	SEQ II $\dfrac{\vdash_R p \xrightarrow{\surd} \delta y \;\; \vdash_R q \xrightarrow{\alpha} q'}{\vdash_R p.q \xrightarrow{\alpha} q'}$
PAR I $\dfrac{\vdash_{R \cup (Y \cap S)} p \xrightarrow{\alpha} p \;\; name(a) \notin S}{\vdash_R p \mid_s q \xrightarrow{\alpha} p' \mid_s [1]q}$	PAR II $\dfrac{\vdash_{R \cup (X \cap S)} q \xrightarrow{\alpha} q \;\; name(a) \notin S}{\vdash_R p \mid_s q \xrightarrow{\alpha} [1]p \mid_s q'}$
PARIII $\dfrac{\vdash_{R \cup (Y \cap S)} p \xrightarrow{\alpha} p' \;\; \vdash_{R \cup (X \cap S)} q \xrightarrow{\alpha} q' \;\; name(a) \in S}{\vdash_R p \mid_s q \xrightarrow{\alpha} p' \mid_s q'}$	X, Y *satisfacen las ecuaciones* Env_x, Env_y
DELAY $\dfrac{\vdash_R p \xrightarrow{\alpha} p'}{\vdash_R [k]p \xrightarrow{f(\alpha,k)} [inc(k)]p'}$	

Tabla2 : Reglas para $2ST^R$

La definición de las ecuaciones Env_x, Env_y es la definida en la sección 3.4.

El comportamiento de las reglas es similar al de las STE^R, solo que aquí las acciones se dividen en inicios y finales.

El operador de alternativa con prioridad, sólo permite que un proceso tenga prioridad si el ambiente está queriendo comunicarse a través de dicho proceso. Pero no se puede representar el ejemplo de la farmacia (planteado en la introducción).Es decir, cuando se quiere establecer prioridad entre dos procesos que no quieren comunicarse, no alcanzan los elementos que tenemos. Esto es muy sencillo de ver ya que el operador de alternativa con prioridad justamente hace que un proceso tenga prioridad si el ambiente quiere comunicarse con él pero no cuando dos procesos actúan en forma independiente. El $+_>$ elige entre dos procesos descartando a uno. No permite que uno comience primero y luego lo haga el otro.

Para abarcar este concepto se introduce en los $2ST^R$ un nuevo operador, el paralelo con

prioridad. Se extiende el lenguaje $\mathcal{K}$ incluyendo el operador y se agregan reglas para regular su comportamiento.

5.2 Redefinición de $\mathcal{K}$ para incluir otra noción de prioridad

Se llamará al nuevo lenguaje $\mathcal{K}^1$.Un término de $\mathcal{K}^1$ se define como:

$$t ::= a \mid a^1 \mid t.t' \mid t + t' \mid t|_s t' \mid [k]t \mid \varepsilon \mid t +_> t' \mid t|_{>_s} t'$$

$t \mid_{>_s} t'$ es la ejecución en paralelo de dos procesos, donde el comienzo de las acciones de t, tiene prioridad sobre el comienzo de las acciones de t'. Es decir, el proceso de la derecha podrá iniciar una acción sólo si el proceso de la izquierda no puede iniciar ninguna acción.

5.3 Semántica operacional de $\mathcal{K}^1$ sobre los $2\mathrm{ST}^R$

PRIPAR I

$$\frac{\vdash_{R\cup(Y\cap S)} p \xrightarrow{\alpha} p' \quad name(a) \notin S}{\vdash_R p|_{>_s} q \xrightarrow{\alpha} p'|_{>_s}[1]q}$$

PRIPAR II_{ini}

$$\vdash_{R\cup(X\cap S)} q \xrightarrow{a} q \quad y$$

$$\forall b \notin S. \vdash_{R\cup(Y\cap S)} p \xrightarrow{b} \quad y$$

$$\frac{\forall b \in S. \vdash_{R\cup(Y\cap S)} p \xrightarrow{b} \implies \vdash_{R\cup(X\cap S)} q \xrightarrow{b} \quad y \quad name(a) \notin S}{\vdash_R p|_{>_s} q \xrightarrow{a} [1]p|_{>_s} q'}$$

PRIPAR II_{fin}

$$\frac{\vdash_{R\cup(Y\cap S)} q \xrightarrow{a^i} q' \quad name(a) \notin S; i > 0}{\vdash_R p|_{>_s} q \xrightarrow{a^i} [1]p|_{>_s} q}$$

PRIPAR III

$$\frac{\vdash_{R\cup(Y\cap S)} p \xrightarrow{\alpha} p' \quad y \quad \vdash_{R\cup(X\cap S)} q \xrightarrow{\alpha} q' \quad name(a) \in S}{\vdash_R p|_{>_s} q \xrightarrow{\alpha} p'|_s q'}$$

$$X, Y satisfacen \ las \ ecuaciones \ Env_x, Env_y$$

Tabla 3. Reglas para $2\mathrm{ST}^R$

El comportamiento de los $2ST^R$ segun el lenguaje $\mathcal{K}^1$ se formaliza con las reglas de la Tabla 2 y de la Tabla 3.

La regla PRIPAR I indica que siempre que p pueda hacer una acción α que no es de sincronización, $p \mid_{>s} q$ puede hacer α.

PRIPAR II_{ini} indica que q puede iniciar una acción a que no es de sincronización, sólo en el caso que p no pueda iniciar ninguna acción que no sea de sincronización (es decir que no pertenezca al conjunto S) y si p puede iniciar una acción β que es de sincronización, q no pueda hacer β.

PRIPAR II_{fin} indica que siempre que q pueda finalizar una acción, $p \mid_{>s} q$ también puede finalizar esa acción.

PRIPAR III es la regla que permite sincronización. p y q pueden sincronizarse a través de α si ambas pueden hacer α en los ambientes correspondientes.

6 Toda acción que comienza termina

Se sabe que en los 2ST definidos por $\mathcal{L}$ toda acción que comienza tiene asociado un único final y este final es alcanzable en el momento que se desee. Ahora se comprobará que los $2ST^R$ definidos por $\mathcal{K}^1$ y las reglas de las Tablas 2 y 3, también cumplen esta propiedad.

Este concepto se puede expresar formalmente de la siguiente manera:

Teorema 6.1

Sean u, u',r $\in L$, $a \in Act$, σ *una secuencia de acciones* $\in Act$, $i \in N$, R *un ambiente.*

$$\forall R, u, u', r, a(\vdash_R u \xrightarrow{a} u') \implies (\forall R2, R3, \sigma : \vdash_{R2} u' \xrightarrow{\sigma} r \ y \ \forall i \in \{1 \ldots |\sigma|\}.\sigma(i) \neq a^i \implies \vdash_{R3} r \xrightarrow{a^{|\sigma|+1}})$$

Prueba.

Dada la propiedad:

P(u) =

$$\forall R, u', r, a(\vdash_R u \xrightarrow{a} u') \implies (\forall R2, R3, \sigma : \vdash_R 2u' \xrightarrow{\sigma} r \text{ y } \forall i \in \{1 \ldots |\sigma|\}.\sigma(i) \neq a^i \implies \vdash_{R3} r \xrightarrow{a^{|\sigma|+1}})$$

se probará $\forall$ u P(u) por inducción sobre la estructura de u.

- Casos base:
  - u = a. P(a) se cumple
  - u = a^1. $\vdash_R a^1 \not\xrightarrow{a}$. Por lo tanto $P(a^1)$.
  - u = δ. $\vdash_R \delta \not\xrightarrow{a}$. Por lo tanto $P(\delta)$.
  - u = ϵ. $\vdash_R \epsilon \not\xrightarrow{a}$. Por lo tanto ϵ).
- $u = p + q$.

  $P(p) \ y \ P(q) \implies P(p+q)$

$$\vdash_R p + q \xrightarrow{a} u' \xrightarrow{\sigma} r$$

sii

$$\vdash_R p \xrightarrow{a} u' \qquad porALT1$$

$$o$$

$$\vdash_R q \xrightarrow{a} u' \qquad porALT2$$

Ya que son las únicas posibles evoluciones de $p + q$. Como p y q cumplen la propiedad, en ambos casos

$$\vdash_{R2} u' \xrightarrow{\sigma} r \;\; y \;\; \vdash_{R3} r \xrightarrow{a|\sigma|+1}$$

.

Por lo tanto, P($p + q$)

- $u = p +_> q$.

  $P(p)$ y $P(q)$ $\Longrightarrow$ $P(p +_> q)$

$$\vdash_R p +_> q \xrightarrow{a} u' \xrightarrow{\sigma} r$$

$$sii$$

$$\vdash_R p \xrightarrow{a} u' \qquad porPRIALT1$$

$$o$$

$$\vdash_R q \xrightarrow{a} u' \;\; y \;\; \forall b \in R \vdash_R p \not\xrightarrow{b} \qquad porPRIALT2$$

Ya que son las únicas posibles evoluciones de $p + q$. Como p y q cumplen la propiedad, en ambos casos

$$\vdash_{R2} u' \xrightarrow{\sigma} r \;\; y \;\; \vdash_{R3} r \xrightarrow{a|\sigma|+1}$$

.

Por lo tanto, P($p +_> q$)

- $u = p|_s q$

  $P(p)$ y $P(q)$ $\Longrightarrow$ $P(p|_s)$

$$\vdash_R p|_s q \xrightarrow{a} u'$$

sii

1	$\vdash_{R\cup(Y\cap S)} p \xrightarrow{a} p'$		$a \notin S$	y	$u' = p'|_s[1]q$
2	$\vdash_{R\cup(X\cap S)} q \xrightarrow{a} q'$		$a \notin S$	y	$u' = [1]p|_s q'$
3	$\vdash_{R\cup(Y\cap S)} p \xrightarrow{a} p'$	$\vdash_{R\cup(X\cap S)} q \xrightarrow{a} q'$	$a \in S$	y	$u' = p'|_s q'$

Sea σ = b.σ'

$$u' \xrightarrow{\sigma} r$$

$$sii$$

$$\exists u'', R_{u''}. \vdash_{R2} u' \xrightarrow{b} u'' \;\; y \;\; \vdash_{R_{u''}} u'' \xrightarrow{\sigma'} r$$

$$\vdash_{R2} u' \xrightarrow{b} u''$$

sii

$\vdash_{R2 \cup (Y \cap S)} p' \xrightarrow{b} p''$ $\quad a \notin S \quad u' = p'|_s[1]q \quad u'' = p''|_s[2]q$

$\vdash_{R2 \cup (X \cap S)} [1]q \xrightarrow{b} [2]q'$ $\quad a \notin S \quad u' = p'|_s[1]q \quad u'' = [1]p'|_s[2]q'$

$\vdash_{R2 \cup (Y \cap S)} p' \xrightarrow{a} p''$ $\quad y \vdash_{R2 \cup (X \cap S)} [1]q \xrightarrow{a} [2]q' \quad a \in S \quad u' = p'|_s[1]q \quad u'' = p''|_s[2]q'$

Lo mismos avances se hacen con 2 y 3. Repitiendo la operacion con σ' = b_1 .σ" ... σ^n = b_n, vemos que las únicas posibles evoluciones del operador paralelo son evoluciones de p'^i o de q'^i o de ambas. Sea σ_p todas las acciones de σ realizadas por p para que u' se convierta en r. Sea σ_q todas las acciones de σ realizadas por q para que u' se convierta en r.

p y q cumplen la propiedad. Es decir

$$\vdash_{R \cup (Y \cap S)} p \xrightarrow{a} p' \;\; \vdash_{R2 \cup (Y_1 \cap S)} p' \xrightarrow{\sigma_p} r_p \;\; \vdash_{R3 \cup (Y_2 \cap S)} r_p \overset{a|sigma_p|+1}{\longrightarrow}$$

$$\vdash_{R \cup (X \cap S)} q \xrightarrow{a} q' \;\; \vdash_{R2 \cup (X_1 \cap S)} q' \xrightarrow{\sigma_q} r_q \;\; \vdash_{R3 \cup (X_2 \cap S)} r_q \overset{a|sigma_q|+1}{\longrightarrow}$$

Como σ es una combinacion de σ_p y σ_q, se puede decir que $\vdash_{R2} u' \xrightarrow{\sigma} r \;\; \vdash_{Rx} r \overset{a|sigma|+1}{\longrightarrow}$

- $u = p|_{>s}q$. Similar a $p|_s q$, aplicando PRIPAR
- $u = [1]p$

  $P(p) \implies P([1]p)$

$$\vdash_R [1]p \xrightarrow{a} u'$$

$$sii$$

$$\vdash_R p \xrightarrow{a} p'$$

p cumple la HI. Por lo tanto:

$$\vdash_{R2} p' \xrightarrow{\sigma} r \;\; \vdash_{R3} r \overset{a|\sigma|+1}{\longrightarrow}$$

Aplicando DELAY

$$\vdash_R [1]p \xrightarrow{a} [2]p' \;\; \vdash_{R2} [2]p' \xrightarrow{\sigma} r \;\; \vdash_{R3} r \overset{a|sigma|+1}{\longrightarrow}$$

Por lo tanto $P([1]p)$.

■

7 Bisimulación y congruencia para los $2ST^R$

7.1 El operador de alternativa con prioridad es compatible con la bisimulacion.

Teorema 7.1 *Sean p, q, r, t procesos.*
$\forall p,q,r,t \;\; p \leftrightarrows q \;\; y \;\; r \leftrightarrows t \implies p +_> r \leftrightarrows q +_> t$

Prueba. Aplicando la definición de bisimulación.
Sea BR: $p \leftrightarrows q$ y $r \leftrightarrows t$.
Se define BS como

$$BS : p +_> r \;\; BS \;\; q +_> t \;\; sii \;\; p \; BR \; q \;\; y \;\; r \; BR \; t$$

$$\exists BR.((\vdash_R p \xrightarrow{\alpha} p' \;\; y \;\; p \; BR \; q \implies \exists q'.\vdash_R q \xrightarrow{\alpha} q' \;\; y \;\; p' \; BR \; q')y$$

$$(\vdash_R r \xrightarrow{\alpha} r' \;\; y \;\; r \; BR \; t \implies \exists t'.\vdash_R t \xrightarrow{\alpha} t' \;\; y \;\; r' \; BR \; t'))$$

$\implies$

$$\exists BS.(\vdash_R p +_> r \xrightarrow{\alpha} w \;\; y \;\; p +_> r \; BS \; q +_> t \implies \exists w'.\vdash_R q +_> t \xrightarrow{\alpha} w' \;\; y \;\; w \; BS \; w')$$

La prueba es similar a la de los STE^R. ∎

7.2 El operador de paralelo con prioridad es compatible con la bisimulacion

Teorema 7.2 *Sean p, q, r, t procesos.*
$\forall p,q,r,t \;\; p \leftrightarrows q \;\; y \;\; r \leftrightarrows t \implies p|_{>s}r \leftrightarrows q|_{>s}t$

Lema 7.3 *Si $p \leftrightarrows q$ y $r \leftrightarrows t$ las ecuaciones asociadas a $p|_{>s}r$ y $q|_{>s}t$ tienen exactamente las mismas soluciones.*
Sean (X, Y) las soluciones de $p|_{>s}r$ y (X_1, Y_1) las soluciones de $q|_{>s}t$
$\forall p,q,r,t \;\; p \leftrightarrows q \;\; yr \leftrightarrows t \implies X = X_1 \;\; y \;\; Y = Y_1$
$\{b \in Ini(p).\vdash_{R\cup(Y\cap S)} p \xrightarrow{b}\} = \{b \in Ini(q).\vdash_{R\cup(Y_1\cap S)} q \xrightarrow{b}\}$
$\{b \in Ini(r).\vdash_{R\cup(X\cap S)} r \xrightarrow{b}\} = \{b \in Ini(t).\vdash_{R\cup(X_1\cap S)} t \xrightarrow{b}\}$

Prueba.
Aplicando la definicion de bisimulación.
Sea BR: $p \leftrightarrows q$ y $r \leftrightarrows t$.
Se define BS como

$$BS : p \mid_{>s} r \;\; BS \;\; q \mid_{>s} t \;\; sii \;\; p \; BR \; q \;\; y \;\; r \; BR \; t$$

$$\exists BR.((\vdash_R p \xrightarrow{\alpha} p' \;\; y \;\; p \; BR \; q \implies \exists q'.\vdash_R q \xrightarrow{\alpha} q' \;\; y \;\; p' \; BR \; q')y$$

$$(\vdash_R r \xrightarrow{\alpha} r' \;\; y \;\; r \; BR \; t \implies \exists t'. \vdash_R t \xrightarrow{\alpha} t' \;\; y \;\; r' \; BR \; t'))$$

$\Longrightarrow$

$$\exists BS.(\vdash_R p|_{>s} r \xrightarrow{\alpha} w \;\; y \;\; p|_{>} r \; BS \; q|_{>s} t \implies \exists w'. \vdash_R q|_{>s} t \xrightarrow{\alpha} w' \;\; y \;\; w \; BS \; w')$$

$$\vdash_R p|_{>s} r \xrightarrow{\alpha} w$$

$$sii$$

- A. Por PRIPAR I y w = $p' \;|_{>s}\; [1]q$

$$\vdash_{R \cup (Y \cap S)} p \xrightarrow{a} p'$$

- B. Por PRIPAR II_{ini} y w = $[1]p \;|_{>s}\; r'$

$$\vdash_{R \cup (X \cap S)} r \xrightarrow{a} r';\; y$$

$$\forall b \notin S. \vdash_{R \cup (Y \cap S)} p \xrightarrow{b} \not\rightarrow \;\; y$$

$$\forall b in S. \vdash_{R \cup (Y \cap S)} p \xrightarrow{b} \implies \vdash_{R \cup (X \cap S)} r \xrightarrow{b} \not\rightarrow \;\; y \;\; a \notin S$$

- C. Por PRIPAR II_{fin} y w = $[1]p \;|_{>s}\; r'$

$$\vdash_{R \cup (Y \cap S)} q \xrightarrow{a^i} q'$$

- D. Por PRIPAR III y w = $p' \;|_{>s}\; q'$

$$\vdash_{R \cup (Y \cap S)} p \xrightarrow{\alpha} p' \;\; y$$

$$\vdash_{R \cup (X \cap S)} q \xrightarrow{\alpha} q' \;\; y$$

- A

  Por HI

$$\exists q'. \vdash_{R \cup (Y \cap S)} q \xrightarrow{\alpha} q' \;\; y \;\; p' \; BR \; q'$$

  Por lema $p|_{>s}q$ tiene las mismas soluciones para las ecuaciones X, Y que $q|_{>s}t$. Por lo tanto se puede aplicar PRIPAR I.

$$\vdash_R q|_{>s} t \xrightarrow{\alpha} q'|_{>s}[1]t = w'$$

$$w \;\; BS \;\; w'$$

$$sii$$

$$p' \;\; BR \;\; q' \;\; y \;\; [1]r \;\; BR \;\; [1]t$$

  Se sabe que por HI p' BR q' y r BR t . Ademas el operador DELAY es compatible con la bisimulacion. Por lo tanto

$$w \;\; BS \;\; w'$$

- B

  r BR t y p BR q por HI.

$$\exists t'.\vdash_{R\cup(X\cap S)} t \xrightarrow{a} t';\ y$$

$$\forall b \notin S.\vdash_{R\cup(Y\cap S)} q \xrightarrow{b} \not\quad y$$

$$\forall b \in S.\vdash_{R\cup(Y\cap S)} q \xrightarrow{b} \implies \vdash_{R\cup(X\cap S)} t \xrightarrow{b} \not\quad y \quad a \notin S$$

  Por lema se sabe que los conjuntos X, Y tienen las mismas soluciones para $p|_{<_s r}$ que para $q|_{>_s t}$. Por lo tanto se puede aplicar PRIPAR II_{ini}.

$$\vdash_R q|_{>_s t} \xrightarrow{a} [1]q|_{>_s t'} \ = \ w' \ y \ r' \ BR \ t'$$

$$w \ BS \ w'$$

$$sii$$

$$[1]p \ BR \ [1]q \ \ y \ \ r' \ BR \ t' \ \ (por HI)$$

  Por lo tanto

$$w \ BS \ w'$$

- C

  Similar a A y B.

- D

  Similar a A y B.

■

8 Conclusiones

En este trabajo se ha explorado la definición de operadores de prioridad en modelos que permiten representar el verdadero paralelismo. La prioridad ha sido analizada como una relación de orden entre subprocesos.

Se definieron operadores de prioridad tanto en un modelo que permite representar veradero paralelismo como en uno que no. La idea de este operador es permitir que una acción de sincronización entre dos procesos que se están ejecutando en paralelo, tenga prioridad sobre el resto de las acciones. Por lo tanto este tipo de prioridad tiene sentido tanto en modelos de verdadero paralilesmo como en modelos que no lo son. Para poder representar esta noción de prioridad es necesario que cada proceso conozca las posibles maneras de comunicarse con otros procesos(ambiente).

Definimos en este trabajo dos modelos que incluyen la noción de ambiente. Uno de ellos, el STE^R, es una extensión de STE pero con reglas que permiten saber en cada momento en qué ambiente se está ejecutando un proceso. El otro, $2ST^R$, es una extensión de los 2ST. Aquí las accione están divididas en inicios y finales y también se puede saber en cada momento las posibilidades de comunicación que tiene un proceso. Cuando dos procesos se ejecutan en paralelo en alguno de estos modelos, necesitan saber qué acciones puede ejecutar el otro proceso para decidir qué acciones ejecutar. Es decir existe una especie de intercambio de información entre los procesos.

Definimos en este trabajo un sistema de ecuaciones que permite resolver este intercambio. Mediante la inclusión de un funcional definimos también una manera de mecanizar este sistema de ecuaciones, para que la búsqueda del conjunto solución no deba ser por análisis exhaustivo.

En este marco definimos el operador de prioridad $+_>$, que permite la ejecución de la segunda componente sólo en caso que el ambiente no proponga la ejecución de la primer componente. Si el ambiente propone la ejecución de la primer componente, significa que otro subproceso se quiere comunicar mediante esa acción y esa comunicación tiene prioridad gracias al operador $+_>$.

La prioridad definida por este operador no alcanza cuando se quiere definir prioridad entre dos procesos que no desean comunicarse. Por lo tanto incluimos en los $2ST^R$ un nuevo operador de prioridad $|_{>s}$. Este operador permite que el proceso de la derecha inicie la ejecución de una acción sólo en el caso que el proceso de la izquierda no pueda iniciar ninguna acción.

Comprobamos en este trabajo que para los $2ST^R$ se conserva la propiedad que cumplen los 2ST que toda acción que comienza, termina.

Se han estudiado las propiedades de congruencia que cumplen los operadores definidos con respecto a la Bisimulación, llegando a la conclusión que el operador propuesto es compatible con la bisimulación.

Conclusiones

En este trabajo se ha explorado la definición de operadores de prioridad en modelos que permiten representar el verdadero paralelismo. La prioridad ha sido analizada desde dos perspectivas diferentes. En una de ellas se estableció una relación de orden entre las acciones y en la otra, una relación de orden entre subprocesos.

Los operadores de prioridad que establecen un orden entre las acciones en los STE se representan como la eliminación, entre las transiciones que salen de cada estado, de aquellas cuyas etiquetas no son maximales en el orden parcial de prioridad entre las acciones. En ese sentido, las transiciones "compiten", y sólo aquellas que tienen etiquetas maximales de acuerdo al preorden dado son preservadas.

Se ha trabajado aquí sobre los Sistemas de transición ST (2ST), un modelo muy cercano a los STE, pero donde las acciones están divididas en inicio y final. Se propusieron cuatro definiciones diferentes del operador de prioridad θ, según la competencia entre las transiciones involucre los inicios y/o los finales de las mismas.

Se ha prestado especial interés a la más simple de ellas, donde sólo los inicios de las acciones compiten. Se mostró allí que se cumple la propiedad esencial de los 2ST, aquella que dice que todo las transiciones que se inician, para cualquier evolución del sistema, terminan en algún momento. La propiedad es inclusive algo más fuerte: toda acción no terminada puede finalizar en cualquier momento que se decida. Esto permite dar una transformación natural de los 2ST en los STE. Se ha probado que la definición de prioridad conmuta con esta transformación. Eso asegura en algun sentido que la operación de prioridad definida representa la misma intuición que la definida en los STE.

Se ha estudiado también con cuidado el caso donde sólo los finales compiten. En este caso, la propiedad que se encuentra es algo mas complicada: para toda transición iniciada, toda evolución ulterior del sistema que no la termine puede completarse con una evolución (que eventualmente involucra varios pasos) de manera que la transición pueda finalizarse. En este sentido este operador es mas complejo que el anterior. Nos permite llegar a 2ST cuya traducción en terminos de STE no es directa.

Se han estudiado las propiedades de congruencia que cumple el operador definido con respecto a la Bisimulación, llegando a la conclusión que el operador propuesto es compatible con la bisimulación.

Los operadores de prioridad que establecen un orden entre subprocesos se definieron tanto en un modelo que permite representar veradero paralelismo como en uno que no. La idea de este operador es permitir que una acción de sincronización entre dos procesos que se están ejecutando en paralelo, tenga prioridad sobre el resto de las acciones. Por lo tanto este tipo de prioridad tiene sentido tanto en modelos de verdadero paralelesmo como en modelos que no lo son. Para representar esta noción de prioridad es necesario que cada proceso *conozca* las posibles maneras de comunicarse con otros procesos (lo que representamos en un ambiente que permite establecer un *intercambio de informaciń entre dos procesos*).

Definimos en este trabajo dos modelos que incluyen la noción de ambiente. Uno de ellos,

el STE^R, es una extensión de STE pero con reglas que permiten saber en cada momento en qué ambiente se está ejecutando un proceso. El otro, $2ST^R$, es una extensión de los 2ST. Aqui las accione están divididas en inicios y finales y también se puede saber en cada momento las posibilidades de comunicación que tiene un proceso.

Definimos en este trabajo un sistema de ecuaciones que permite resolver este intercambio.

En este marco definimos el operador de prioridad $+_>$ que resuelve el concepto de prioridad planteado. El operador $+_>$ no permite decidir prioridad entre dos subprocesos que no desean comunicarse. Por lo tanto incluimos en los $2ST^R$ un nuevo operador de prioridad $|_{>s}$, que sí lo hace, comprobando también en este caso que se conserva la propiedad que toda acción que comienza, termina.

Planteamos una mecanización para hallar la solución de las ecuaciones que permiten el intercambio de información entre procesos, mediante la introducción de un funcional. Introdujimos todos los inconvenientes que surgen a partir de esta definición y por los cuales no podemos asegurar que el funcional sirva realmente como mecanismo de solución. Mucho falta investigar al respecto. Este tema será retomado posiblemente en futuros trabajos.

Se han estudiado las propiedades de congruencia que cumplen los operadores definidos con respecto a la Bisimulación, mostrando que el operador propuesto es compatible con la bisimulación.

References

[BBK86] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamentae Informaticae*, IX(2):127–168, 1986.

[Bed87] M.A. Bednarczyc. *Categories of asynchronous systems.* PhD thesis, University of Sussex, 1987.

[BGG94] N Busi, R. van Glabbeek, and R. Gorrieri. Axiomatising ST-bisimulation equivalence. In E.-R. Olderog, editor, *Proceedings of PROCOMET'94, IFIP 2 Working Conference*, San Miniato, 1994. North-Holland.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, July 1984.

[BK85] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process algebra.* Cambridge University Press, 1990.

[GL95] R Gorrieri and C Laneve. Split and st bisimulation semantcs. *Information and Computation*, 1995. To appear.

[GV87] R. van Glabbeek and F. Vaandrager. Petri nets models for algebraic theories of concurrency. In J.W. de Bakker, A.J. Nijman, and P. Treleaven, editors, *Proceedings of PARLE conference*, volume II, pages 224–242, Eindhoven, 1987. LNCS 259, Springer-Verlag.

[Hen90] Mathew Hennessy. *The semantics of programming languages: an elementary introduction using structural operational semantics.* Jhon Wiley & Sons, Sussex, 1990.

[Hoa85] C.A.R. Hoare. *Communicating sequential process.* Prentice Hall, 1985.

[Kel76] R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 8(19):371–384, 1976.

[Mil80] Robin Milner. *A calculus of communicating systems*, volume 92 of *LNCS.* Springer-Verlag, 1980.

[Mil89] Robin Milner. *Communication and concurrency.* Prentice Hall, 1989.

[NRSV90] X. Nicollin, J.-L. Richier, J. Sifakis, and J. Voiron. ATP: an algebra for timed processes. In M. Broy and C.B. Jones, editors, *Proceedings of the IFIP TC 2 Working Conference on Programming Concepts and Methods*, Sea of Gallilee,Israel, April 1990. North-Holland.

[NS94] X. Nicollin and J. Sifakis. The algebra of timed processes, ATP: theory and application. *Information and Computation*, 114:131–178, 1994.

[Pet62] C.A. Petri. *Kommunikation mit Automaten.* PhD thesis, University of Bonn, 1962.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI-FN-19, Computer Science Department, University of Århus, 1981.

[Pnu85] Amir Pnueli. Linear and branching structures in the semantic and logic of the reactive systems. In W. Brauer, editor, *Proceedings 12th. ICALP*, pages 15–32, Nafplion, 1985. LNCS 194, Springer-Verlag.

[Rei85] W. Reissig. *Petri Nets, An Introduction.* Springer-Verlag, EATCS Monograph on Thoeretical Computer Sciences, 1985.

[Shi85] V.R. Shields. Concurrent machines. *The Computer Journal*, 28(5):449–465, 1985.

[Win87] Glynn Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozemberg, editors, *Proceedings of Advances in Petri Nets 1986 (Part II)*, pages 325–392, Bad Honnef, 1987. LNCS 255, Springer-Verlag.

[CW91] Juanito Camilleri y Glyn Winskel. CCS with Priority Choice. *CH3095-4/91/0000/0246*, 1991 IEEE.