UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS MATEMÁTICAS

Departamento de Sistemas Informáticos y Programación

SEMÁNTICAS PARA ÁLGEBRAS DE PROCESOS ESTOCÁSTICOS NO MARKOVIANOS

MEMORIA PRESENTADA PARA OPTAR AL GRADO DE DOCTOR POR

Natalia López Barquilla

Bajo la dirección del Doctor:

Manuel Núñez García

Madrid, 2003

ISBN: 84-669-1801-9

Semánticas para álgebras de procesos estocásticos no markovianos

TESIS DOCTORAL

Autora: Natalia López Barquilla
Departamento de Sistemas Informáticos y Programación
Universidad Complutense de Madrid (España)

Director: Manuel Núñez García

Semánticas para álgebras de procesos estocásticos no markovianos

Memoria presentada para obtener el grado de

Doctora en Matemáticas por

Natalia López Barquilla

Dirigida por el profesor

Manuel Núñez García

Departamento de Sistemas Informáticos y Programación

Facultad de Matemáticas

Universidad Complutense de Madrid

Enero 2003

a mis padres
a Ricar y Rosa
a mis abuelos
a Juanjo

Agradecimientos

Quisiera comenzar el presente trabajo expresando mi gratitud a todas aquellas personas que han contribuido directa o indirectamente en el desarrollo del mismo. Primeramente me gustaría agradecerle a Manuel Núñez sus enseñanzas durante estos últimos cinco años, así como la dirección de esta tesis. No sólo ha contestado pacientemente mis numerosas preguntas sobre todo lo relacionado con ambos tipos de *ciencia*, sino que siempre me ha brindado su apoyo y amistad. Realmente me costaría trabajo decidir por cuál de sus dos facetas (director y amigo) quiero darle más las gracias, pero si tuviera que decidir me quedaría con su amistad.

A David de Frutos y Fernando Rubio, coautores de dos articulos que se presentan en esta tesis, les doy las gracias por dejarme trabajar con ellos y enseñarme tantas cosas, por lo que espero poder seguir trabajando a su lado. Por otra parte le quiero agradecer a David su honestidad cuando hace unos años decidí quedarme en la universidad, así como por la acogida dentro de su grupo. También le doy las gracias por la lectura tan minuciosa que ha realizado de este trabajo, así como por sus siempre útiles comentarios al respecto. Mis agradecimientos a Luis Llana no necesitan justificación: Luis, gracias por ser como eres y por estar siempre cerca.

La suerte de tener un compañero de despacho como Fernando es que ha soportado estoicamente mis decisiones prácticamente unilaterales sobre la temperatura del despacho, la música, etc. y siempre me ha ayudado, incluso cuando no lo he pedido, como por ejemplo escribiendo esta frase de los agradecimientos.

Por otra parte me gustaría agradecer a los miembros del tribunal por haber aceptado leerse las páginas que vienen a continuación. Gracias a todos, ¡¡¡¡Grazie Mille Mario!!!! Asimismo, me gustaría agradecerle a Javier Gómez Gil su apoyo incondicional desde que empezó mi aventura dentro de la universidad, que le ha llevado incluso a pertenecer al tribunal de esta tesis. También me gustaría agradecer el *sabio consejo* que me dio Martín Abadi y que siempre recordaré.

A Albertoe, Mercedes, Olga, Alberto e Ismael les agradezco su amistad. En particular,

debo darle las gracias a Albertoe por arreglar todo lo arreglable y ofrecer siempre su ayuda, y a Ismael por sorprenderme constantemente con sus ideas y con sus comentarios sobre el despacho S221. Quiero agradecer a Belén que me soportara con buena cara la época en que mi ordenador estuvo roto, y que me animara en todo momento. Asimismo, agradezco a Mercedes por sus agradables visitas y sus muestras de interés con sus *¿qué tal va todo?*.

Por último me gustaría agradecer a los miembros del *versatile group*, antes conocido como *disperse group*, que me hayan aceptado como miembro y me hayan dejado trabajar con ellos dándome su amistad. Escribir artículos con vosotros es algo que merece la pena.

Quiero dedicar esta tesis a toda mi familia (tanto a la de siempre como a la nueva), en particular a mis padres por todo, a Ricar por ser mi ejemplo a seguir, a Rosa por convertirse en mi hermana y a mi abuelito por todo el tiempo que me ha ahorrado de lágrimas ayudándome a llorar para acabar antes.

Por último quiero dedicársela a la persona que ha estado desde el principio a mi lado: Juanjo, gracias por querer vivir conmigo incluso estos últimos meses de locura, por creer en mí mucho más que yo misma y por tu amor infinito.

Índice general

Capítulo 1

Introducción

Recuerdo con especial satisfacción el día del mes de julio de 1994 en el que aprobé la asignatura de *Cálculo de Probabilidades y Estadística* que se cursaba en el segundo año de la licenciatura de Matemáticas. En aquel momento pensé que sería mi primer y último contacto con conceptos tales como probabilidades, variables aleatorias, funciones de distribución de probabilidad, etc. Cómo iba a pensar entonces que cuando decidí empezar mis estudios de tercer ciclo, matriculándome en una asignatura llamada *Álgebras de Procesos Probabilísticos*, más comúnmente conocida como "curso de Manolo", toda mi investigación, y por tanto esta tesis, iba a utilizar algunos de los conceptos de aquel curso que tanta alegría me dio finalizar.

En los cursos de doctorado estudié, entre otras cosas, los principales modelos formales para especificar procesos concurrentes. Es decir, formalismos adecuados para modelar procesos de la vida cotidiana a través de un cierto lenguaje y cómo dar a continuación un significado a las distintas *frases* que se podían componer con esos lenguajes. Utilizamos las álgebras de procesos para definirlos y dependiendo de qué características tuvieran los constructores del mismo, así serían adecuados para describir un tipo de procesos u otro. Inicialmente, los lenguajes describían únicamente las acciones que ejecutaban los procesos y su comportamiento ante determinados entornos. Sin embargo, según se fueron desarrollando estudios sobre ellos, se vio la necesidad de incluir información de otro tipo, de modo que aparecieron una serie de extensiones de las álgebras de procesos clásicas. En los cursos de tercer ciclo estudié también las principales extensiones probabilísticas y/o temporales que habían aparecido hasta ese momento en la literatura. Sin duda alguna eran temas interesantes, pero en 1997 estaban demasiado trillados como para encontrar un área dentro de ese tema lo suficientemente virgen. Sin embargo, por aquel entonces ya existían una serie de modelos formales donde la información probabilística y la temporal estaban estrechamente ligadas. Estos modelos ser-

vían para especificar un tipo concreto de procesos denominados *Procesos Estocásticos*. Este tema, más novedoso que sus predecesores, era aún un campo abierto en el que investigar, de modo que decidimos que podría ser un buen tema sobre el que centrar mi investigación.

El objetivo de esta tesis es aportar una serie de resultados a un tema aún joven, en el cual existen aún grandes nichos de investigación. Cuando se trabaja en un tema bastante desarrollado, como es el caso de las álgebras de procesos clásicas y sus principales extensiones probabilísticas y/o temporales, donde se conocen las principales consecuencias de cualquier decisión de diseño que se pueda tomar a la hora de definir un modelo, es lógico centrarse en un tema o modelo muy concreto y desarrollar todo un trabajo basado en él. Este tipo de investigación se podría denominar *recorrido* o *estudio en profundidad* de un tema. Este es el caso de tesis doctorales como la de Manuel Núñez [Núñ96], dedicada a desarrollar una teoría completa sobre un modelo concreto con información probabilística, o como la de Luis Llana [Lla96] dedicada al desarrollo de una teoría sobre un modelo temporal. En estos trabajos se define inicialmente un lenguaje con el que representar los procesos y a partir de ese punto se desarrolla una semántica para ese lenguaje, estudiando exclusivamente las propiedades y características de ese modelo semántico. Sin embargo, cuando empezamos el estudio de los modelos formales para procesos estocásticos, este tema estaba muy poco desarrollado (y todavía lo sigue estando, en algunos campos). Por ese motivo decidimos abordarlo con un *estudio en anchura*. Este tipo de estudio nos ofrece la posibilidad de definir varios modelos y distintas semánticas sobre ellos, de forma que podemos estudiar sus propiedades encontrando las principales ventajas al igual que los principales inconvenientes de cada una de las decisiones de diseño tomadas. Como contrapartida, este tipo de estudio no nos permite profundizar en los modelos y semánticas que hemos definido tanto como desearíamos.

Afortunadamente, terminar una tesis no significa finalizar con el tema que se desarrolla en la misma, ni finalizar la publicación de los resultados que se puedan conseguir a partir de lo ya obtenido. De hecho, es mi intención profundizar en un futuro inmediato en aquellos temas que se han tratado de forma somera en el trabajo que constituye mi tesis doctoral.

1.1. Estado del arte

Se ha mostrado de forma profusa durante los últimos veinte años que las álgebras de procesos [Hoa85, Hen88, Mil89, BW90, BPS01] son un mecanismo poderoso para especificar el comportamiento funcional de sistemas distribuidos y concurrentes. Además de la continua aparición de lenguajes que describieran mejor los procesos a estudiar, ha habido un estudio intenso de las semánticas que mejor podrían capturar comportamientos equivalentes de

procesos sintácticamente distintos. Sin embargo, los primeros modelos algebraicos no eran capaces de representar adecuadamente sistemas en los que la información cuantitativa (como la temporal o la probabilística) juega un papel fundamental. Por ejemplo, si se especifica un canal de comunicación defectuoso sin utilizar una estimación probabilística sobre la media de fallos, todo lo que se puede saber es si el mensaje llega o no. Por el contrario, si especificamos esa probabilidad, y repetimos el envío del mensaje, se puede probar que, con probabilidad 1, el mensaje llegará. De igual modo, podríamos estar interesados no sólo en el hecho de que el mensaje llegue, sino en el hecho de que llegue antes de un cierto tiempo t con una cierta probabilidad (digamos $1-\epsilon$). Por este motivo, han aparecido varias extensiones de álgebras de procesos probabilísticas, temporales e, incluso, probabilístico-temporales. Sin embargo, existen sistemas en los que incluir únicamente información probabilística o temporal no es suficiente, y ni siquiera incluyendo ambos tipos de información es posible modelarlos. Éste es el caso de los procesos en los cuales se quiere especificar que la probabilidad de ejecutar una acción varía según va pasando el tiempo. Por ejemplo, el evento de coches llegando a una gasolinera sigue una distribución de Poisson.

Una evolución natural de las álgebras de procesos probabilísticas y/o temporales a dado lugar a las *álgebras de procesos estocásticas* (p.ej. [GHR93, ABC$^+$94, Hil96, Her98, BG98, PCVC00, HS00, LN00, HHK02, BG02b]). En estos lenguajes, la información temporal se incrementa con algún tipo de información probabilística. Por ejemplo, podemos especificar un sistema que espera recibir un mensaje con probabilidad $\frac{1}{2}$ en el intervalo $(0,1]$, con probabilidad $\frac{1}{4}$ en $(1,2]$, y así sucesivamente. Este mecanismo representa una ventaja importante con respecto a los procesos temporales donde solamente podríamos especificar que el mensaje llegará en el intervalo $(0,\infty)$.

En esta sección vamos a presentar un breve estado del arte en las tres extensiones de las álgebras de procesos que utilizaremos a lo largo de esta tesis. Empezaremos con un resumen de los modelos probabilísticos existentes, a continuación pasaremos a los modelos con información temporal, y terminaremos este repaso con los principales modelos en los que se ha combinado información probabilística y temporal dando lugar a modelos estocásticos.

1.1.1. Modelos probabilísticos

A la hora de definir un álgebra de procesos hay que tomar dos decisiones. En primer lugar, debemos decidir el mecanismo que se utilizará para modelar la *elección* entre un conjunto de acciones posibles. En segundo lugar, tenemos que definir la *semántica* más adecuada para dar significado a los procesos. En el caso de las álgebras de procesos con información

probabilística estas dos decisiones son más complicadas que en el caso general. En el caso de la elección, existen más opciones entre las que decidirse. Primero debemos elegir entre las versiones probabilísticas y no probabilísticas de los operadores de elección de CCS o CSP. Adicionalmente, podemos obtener distintos modelos dependiendo de cómo se incluya la información probabilística. Por ejemplo, si tenemos una única distribución de probabilidad para resolver las elecciones, o diferentes distribuciones, una para cada tipo de acción. Por otro lado, como cabe esperar, la incorporación de información probabilística complica el estudio de cualquier semántica.

La investigación en el campo de los modelos probabilísticos no es nueva. Realmente, los primeros trabajos empezaron con los autómatas probabilísticos [Rab63] en los años 60. Durante la década de los 80 aparecieron diferentes trabajos extendiendo los marcos existentes, principalmente lógicas, con nociones de probabilidad (p.ej. [FH82, Koz83, HS84, Var85, CY88, JP89]). Sin embargo, ha sido en los últimos quince años cuando han aparecido diferentes modelos para procesos probabilísticos, con diferentes lenguajes, diferentes interpretaciones de la información probabilística, etc. Los primeros trabajos relacionados con sistemas de transiciones etiquetados y con información probabilística aparecieron a finales de los años 80. En [LS89, LS91], Larsen y Skou dieron una extensión probabilística de la noción de bisimulación fuerte. Caracterizaron esta relación utilizando una semántica de *testing*[1] donde el poder de las pruebas se incrementaba con respecto a las pruebas definidas en [dNH84, Hen88]. Concretamente, permitían la utilización de múltiples copias de los procesos probados de modo que el experimentador podía experimentar con una copia cada vez. Mostraron que si dos procesos no son bisimilares (fuertes) probabilísticos entonces existe una prueba que los distingue con una cierta probabilidad $1 - \epsilon$, donde ϵ es arbitrariamente pequeño. Esta primera noción de bisimulación se ha convertido en la noción estándar de bisimulación fuerte probabilística. Siguiendo este primer modelo, Bloom y Meyer [BM89] introdujeron una noción de bisimulación probabilística en términos de *probabilizaciones* de procesos no probabilísticos. Mostraron que si dos sistemas de transiciones etiquetados eran bisimilares (sin información probabilística) entonces se podían asignar pesos para obtener nuevos sistemas que no pudieran distinguirse con una noción general de *testing* probabilístico. El primer *álgebra de procesos probabilístico* aparece en 1990. En [GJS90] se presenta una versión probabilística de SCCS [Mil83], denominada PCCS. En dicho trabajo reemplazan el operador no determinista de SCCS por un operador de elección probabilístico n-ario. La

[1] A lo largo de este trabajo hemos preferido mantener el término original *testing* al considerar que, dependiendo del contexto, se deberían utilizar distintos términos en castellano. Ello podría llevar a pensar que estamos hablando de cosas distintas a pesar de que nos refiramos a un único concepto.

sintaxis utilizada es $\sum_{i \in I}[p_i]P_i$, de modo que las acciones que ofrece cada proceso P_i se eligen teniendo en cuenta los pesos p_i asociados con cada P_i.

Como ya hemos comentado anteriormente, una de las decisiones de diseño que se deben tomar cuando se define un álgebra de procesos probabilísticos es la forma de incluir la información probabilística con respecto a las acciones habituales. En [GSST90, GSS95] van Glabbeek et al. presentan tres modelos de procesos probabilísticos basados en el lenguaje PCCS, denominados modelo *reactivo*, *generativo* y *estratificado*. Mientras que los dos primeros modelos han sido ampliamente utilizados a la hora de definir procesos probabilísticos, el modelo estratificado ha recibido menos atención (posiblemente por su complejidad).

El *modelo reactivo* apareció por primera vez en el trabajo de Larsen y Skou en [LS89] para sistemas de transiciones etiquetados. Sin embargo, en [GSST90] se considera esta interpretación de las probabilidades en el contexto de las álgebras de procesos probabilísticos. En este modelo se considera que los procesos *reaccionan* al estímulo dado por su entorno. Es decir, el entorno ofrece una única acción y el proceso elige entre las acciones de ese tipo considerando sus probabilidades asociadas. Por tanto, no hay una relación probabilística entre acciones distintas. En el *modelo generativo* el entorno puede ofrecer simultáneamente varias acciones y el proceso debe elegir entre ellas considerando la distribución de probabilidad asociada con estas acciones. En este modelo hay una relación probabilística entre las diferentes acciones. En este caso la suma de las probabilidades asociadas con todas las acciones debe sumar 1, existiendo en cada paso de ejecución una *redistribución* de probabilidades entre aquellas acciones ofrecidas por el entorno. En resumen, la principal diferencia entre el modelo reactivo y el generativo es que mientras en el generativo hay una única distribución de probabilidad, en el modelo reactivo hay una distribución de probabilidad para cada una de las diferentes acciones. En el *modelo estratificado* la interpretación de las probabilidades es similar a la del modelo generativo, pero se mantiene la probabilidad de la ramificación del proceso, es decir, la redistribución de las probabilidades se hace de manera *local*.

Aunque la mayoría de los modelos para procesos probabilísticos están basados en una de las interpretaciones previas de probabilidades, hay algunos trabajos que combinan más de una de estas interpretaciones. Este es el caso de la extensión probabilística de los autómatas de E/S [LT87] presentado en [WSS94, WSS97]. La idea es que el entorno puede controlar las acciones de entrada diciendo cuáles se ejecutan. De ese modo, el proceso sería reactivo para las acciones de entrada. Por el contrario, el entorno no puede controlar ni las acciones de salida y ni las acciones internas, por lo que lo apropiado es una única distribución de probabilidad para resolver la elección entre este tipo de acciones. Así, el modelo es generativo

para las acciones internas y de salida. Además, no existe relación probabilística entre las diferentes acciones de entrada o entre éstas y las de salida, por lo que el proceso decidirá de forma no determinista si ejecuta una acción ofrecida por el entorno, una acción de salida o una acción interna. Una interpretación similar para el tratamiento de las acciones se ha utilizado recientemente en [BA01, BA03] para definir un modelo algebraico donde se trata información temporal, además se especifica la velocidad de avance de los procesos por medio de la inclusión de un operador de paralelo probabilístico.

La mayoría de las extensiones probabilísticas que aparecen en la literatura están basadas en CCS o, de forma equivalente, en sistemas de transiciones etiquetados probabilísticos. Es decir, se considera usualmente un lenguaje con un único operador de elección al que se le añade información probabilística. Sin embargo, existen otras extensiones de modelos estilo CCS que combinan operadores de elección probabilísticos y otros no probabilísticos. Este es el caso, por ejemplo, del álgebra de procesos descrita en [YL92]. Este modelo tiene dos operadores de elección: el operador habitual de elección de CCS no probabilístico y un operador de elección *puramente* probabilístico. Este último es una versión probabilística del operador de elección interna de CSP.

Las extensiones basadas en CSP son más escasas. En [Low95, NdFL95] se extienden los dos operadores de elección con probabilidades. Por un lado, se dispone de una elección externa probabilística que se comporta como el operador de elección probabilístico de CCS en aquellas situaciones donde no hay acciones internas involucradas en la elección. Por otro lado, una elección interna probabilística, de modo que $P \oplus_p Q$ es equivalente al proceso probabilístico en CCS $\tau\,;a +_p \tau\,;b$. En el caso de CSP, hay también propuestas que incluyen operadores probabilísticos y operadores no deterministas. En [Sei95] el operador de elección externa es no probabilístico pero está parametrizado con un conjunto de trazas. Este parámetro debe considerarse como un planificador para resolver la elección no determinista. En [CdFV97] los autores introducen un lenguaje en el que combinan una elección externa (no probabilística) y una elección interna probabilística. Por último, en [CCVP01, CCV+03] se consideran las versiones no probabilísticas de los operadores de CSP y una elección probabilística interna.

En conclusión, se debe tener en cuenta que por encima del tipo de operadores de elección que se utilice para el lenguaje está el hecho de considerar si todas las elecciones están cuantificadas (probabilísticamente). De este modo, los modelos probabilísticos se clasifican básicamente en dos grupos: modelos puramente probabilísticos y modelos probabilísticos no deterministas. En el primer caso las elecciones siempre están cuantificadas (véase p.ej. [CSZ92, NdFL95, BH97]). En el segundo caso, algunas elecciones están cuantifi-

cadas (normalmente son elecciones internas puras probabilísticas) y otras no (véase p.ej. [HJ89, YL92, PLS00, BS01]).

Bisimulaciones probabilísticas

La definición de la bisimulación fuerte probabilística no ha cambiado desde que se definió por primera vez en [LS89, LS91]. En el marco probabilístico, es necesario calcular las probabilidades de alcanzar una determinada clase de equivalencia. Si consideramos dos procesos probabilísticos, además de pedir que el segundo proceso sea capaz de imitar al primero, necesitamos pedir que lo haga con la misma probabilidad. Por ello, hay que considerar todas las formas posibles de *imitar* la ejecución de una acción y sumar todas las probabilidades que tienen asociadas esas ejecuciones. Es más, se deben considerar todas las posibilidades asociadas con la evolución a procesos equivalentes.

A diferencia de lo que ocurre con la extensión probabilística de bisimulación fuerte, no hay un acuerdo acerca de lo que debe ser una *buena* definición de bisimulación débil probabilística. El problema radica en que una vez que se ha incluido información probabilística, no se puede considerar una abstracción simple de las acciones internas (como en el caso no probabilístico) debido a que hay que tener en cuenta las probabilidades de ejecutar esas acciones internas. De modo que las distintas interpretaciones de *cómo abstraer* las transiciones internas producen nociones divergentes de bisimulación débil probabilística. La primera definición de bisimulación débil probabilística no apareció hasta 1997 [BH97]. Desgraciadamente, dicha equivalencia no es completamente satisfactoria, apareciendo posteriormente otras nociones que refinaban de alguna manera esta formulación (por ej. [PLS00, AB01, BS01]).

Semánticas de testing

Cuando se quiere definir una semántica de *testing* probabilístico la principal pregunta que hay que contestar es: *¿Con qué probabilidad pasa el proceso la prueba?* Dependiendo del modelo con el que trabajemos, existen dos posibilidades. En el caso de los procesos puramente probabilísticos, la probabilidad con la que un proceso pasa una prueba viene dada por la suma de las probabilidades asociadas a las computaciones que han finalizado con éxito. Así, dos procesos son equivalentes si pasan cualquier prueba con la misma probabilidad. En el caso de los procesos probabilísticos no deterministas, no se obtiene una única probabilidad de pasar una prueba, debido a la presencia de no determinismo en algunas de las elecciones. En este caso existen diferentes posibilidades. Por ejemplo, podemos considerar que dos procesos son equivalentes si las probabilidades mínimas/máximas con las que pasan cualquier prueba

son las mismas. A pesar de que la mayor parte de las extensiones probabilísticas se han definido considerando semánticas de bisimulación, existen numerosos trabajos dedicados a este tema (p.ej. [Chr90a, YL92, CSZ92, NdF95, NCS98, CDSY99, Núñ02]).

1.1.2. Modelos temporales

Las principales teorías algebraicas se han extendido con información temporal de distintos modos. En primer lugar, podemos distinguir entre escalas de tiempo relativo y tiempo absoluto, en las cuales el tiempo puede medirse como continuo o como discreto. Además, la ejecución de acciones y el paso del tiempo pueden aparecer combinados o separados.

Las extensiones en las que se mide el tiempo de manera continua suelen denominarse versiones en *tiempo real*. Por otro lado, la medición de tiempo en una escala discreta de tiempo no significa que la ejecución de acciones esté restringida a puntos de tiempo discreto. Habitualmente, lo que se hace es dividir el tiempo en intervalos y las acciones se podrán ejecutar dentro de los intervalos de tiempo que tengan asignados. Así, las versiones de tiempo discreto permiten considerar sistemas a un nivel más abstracto que las versiones de tiempo real puesto que el tiempo se puede medir con precisión finita. Además, en la práctica, todas las computadoras utilizan relojes discretos para medir el tiempo, de modo que la duración de las acciones siempre es múltiplo del correspondiente ciclo de reloj. En ambos casos, si consideramos tanto tiempo continuo como tiempo discreto, el tiempo relativo puede incluirse en un marco con tiempo absoluto mediante la inclusión de un operador de abstracción para tratar el tiempo relativo.

Durante los años 90 y a finales de los 80, aparecieron las principales extensiones temporales de las álgebras de procesos. Algunas de las más conocidas son las extensiones de los lenguajes ACP [BB93], CSP [RR88, NS91, Sch95, LdFN96, RR99, LdF99], CCS [MT90, Yi90, HR91], LOTOS [QdFA93, QdFML94, LL97] y también para autómatas de E/S [AD94, SVD01]. La principal idea subyacente a este tipo de extensiones consiste en ampliar el lenguaje con información temporal. Esta información puede incluirse de dos formas, dependiendo de si van separados o combinados los tiempos y las acciones. Separando ambas ejecuciones tendríamos que se considera que las acciones se ejecutan de forma instantánea y que existen transiciones de tiempo cuyo significado es que el proceso que la ejecuta deja pasar la cantidad de tiempo especificada. Algunos ejemplos de este tipo de modelos son [NS91, Yi91]. En el caso de la combinación de acciones y tiempo tendríamos un único tipo de transiciones cuyo significado sería dejar pasar una cantidad determinada de tiempo para después poder ejecutar la acción. Este tipo de inclusión de la información temporal se puede encontrar, por

ejemplo, en [QdFML94, Sch95].

Una distinción que existe entre distintos modelos de álgebras de procesos temporales es la interpretación que se hace del momento en el cual se puede ejecutar una acción. Existe una interpretación en la que una acción *puede* ejecutarse después de un cierto retardo, pero también puede sufrir un nuevo retardo antes de ejecutarse. Este tipo de interpretación aparece cuando tenemos que una acción debe sincronizar con otra (o con el entorno), dado que en ese caso debe esperar hasta que la otra acción esté también disponible. El otro tipo de interpretación consiste en considerar que una acción *debe* ejecutarse tan pronto como esté disponible. En este caso, los procesos no pueden dejar pasar el tiempo sin hacer nada. Es razonable aplicar este segundo tipo de interpretación a acciones que un sistema pueda ejecutar de manera autónoma. Por ello, habitualmente se aplica sólo a acciones internas debido a que no existe posibilidad de interactuar con el entorno o con otro proceso para su ejecución. Por tanto, no hay ningún motivo para que se retrase su ejecución. Esta interpretación en la que las acciones internas se ejecutan de manera urgente se conoce como *máximo progreso*, *retardo mínimo* o *urgencia de* τ. La mayoría de los trabajos anteriormente citados cumplen esta propiedad. No obstante, existen también algunos trabajos que no la mantienen (véase p.ej. [MT90, NS94]).

Además de la decisión sobre la urgencia o no de las acciones, existen otro tipo de características que pueden cumplir o no los modelos para procesos temporales. A continuación destacamos las más relevantes:

Determinismo en tiempo. Mediante una evolución temporal, un proceso no puede alcanzar dos estados que no sean iguales. Es decir, si un proceso deja pasar una cantidad concreta de tiempo, alcanzará siempre el mismo proceso.

Suma de tiempos. Para asegurarse de la corrección de la noción de tiempo, habitualmente se requiere que si un proceso puede retrasarse $t + t'$ unidades de tiempo, entonces puede retrasarse inicialmente t unidades y después otras t', y viceversa. En ambos casos el comportamiento resultante debe ser el mismo.

Paciencia. Si un proceso no puede ejecutar acciones internas entonces puede dejar pasar el tiempo, es decir, un proceso puede esperar indefinidamente hasta que pueda comunicarse. También existen trabajos que no cumplen esta condición, por ejemplo [MT90, NS94]. Habitualmente, si un modelo no cumple la característica del progreso máximo, entonces tampoco suele cumplir esta propiedad.

Persistencia. En algunos modelos algebraicos existe la característica de que si un proceso

puede ejecutar una acción entonces el paso del tiempo no puede provocar que esa acción no está disponible.

1.1.3. Modelos estocásticos

Como ya se comentó al principio de esta sección, la necesidad de modelar sistemas en los que las acciones sufrían retardos durante unos periodos de tiempo de naturaleza estocástica provocó la aparición de las álgebras de procesos estocásticos. Es a principios de los años 90 cuando aparece en [GHR93] el primero de dichos marcos, donde se define TIPP (TImed Process and Performance analysis) que es una extensión de CSP. Por otra parte, en [Hil93] aparece otra versión de CSP con información estocástica (PEPA: Performance Evaluation Process Algebra) donde se asocia a cada acción una variable aleatoria que representa su duración. Otras propuestas que aparecieron poco después son una extensión del lenguaje LOTOS [ABC+94], la de Buchloz [Buc94] y el lenguaje EMPA definido en [BDG94, BG98]. Estos primeros trabajos están desarrollados para distribuciones exclusivamente exponenciales. Las álgebras de procesos estocásticos que trabajan con este tipo de restricción se denominan *álgebras de procesos markovianos*, siendo las no markovianas todas aquellas que utilizan distribuciones de probabilidad generalizadas. Las primeras aproximaciones con distribuciones generales aparecieron en [BKLL95, KBLL96]. Sin embargo, estos modelos carecen de operadores de paralelo, pues la definición de este operador es el principal problema (como veremos más detenidamente en esta misma sección) cuando se incluyen distribuciones generalizadas. En [HS95, HS00] aparece una primera semántica de *interleaving*[2] para procesos no markovianos, pero su principal inconveniente es que la estructura semántica que resulta es infinita. A finales de los 90 aparecen dos trabajos en los que se trata el problema de las semánticas de interleaving para distribuciones generalizadas. En primer lugar, en [DKB98a, DKB98b] se resuelve el problema mediante la utilización de relojes que van contando el tiempo que falta hasta que una acción se pueda ejecutar. En segundo lugar, en [BBG98, BG99, BG02a] se dividen las acciones estocásticas en dos, el inicio y la terminación de la misma.

Al igual que en el caso de las álgebras de procesos temporales, existen dos formas de especificar la información estocástica (temporal). Una opción es incluir esa información en cada una de las acciones y la otra opción consiste en especificar los retardos aleatorios separados del resto de acciones. En el primer caso, una expresión como (a, ξ) ; P indica que la probabilidad de ejecutar a antes de que transcurra un tiempo t es igual a la probabilidad

[2]Utilizamos el termino en inglés pues entendemos que en castellano no existe ninguna palabra con el mismo significado.

de que la variable aleatoria[3] ξ tome valores menores o iguales que t. Después de ejecutar a, el proceso se comporta como P. Por ejemplo, si ξ está distribuida uniformemente sobre el intervalo $[1,2]$ la probabilidad de ejecutar a antes de tiempo $\frac{2}{3}$ es igual a 0, la probabilidad de ejecutar a antes de tiempo $\frac{3}{2}$ es igual a $\frac{1}{2}$, y así sucesivamente. Ejemplos de este tipo de lenguajes son [Hil96, BG98, LN00]. En el segundo caso, una expresión como $\xi\,;P$ indica que el proceso se retrasará de acuerdo con ξ y después se comportará como P. De nuevo, si ξ está distribuida uniformemente sobre el intervalo $[1,2]$, P empezará antes de tiempo $\frac{5}{4}$ con probabilidad $\frac{1}{4}$, empezará antes de tiempo 2 con probabilidad 1, etc. Ejemplos de este tipo de lenguajes son [Her98, DKB98a, LN01, HHK02, BG02b]. En este trabajo presentaremos modelos para cada una de las dos posibilidades. El primer modelo estocástico que presentaremos tendrá información estocástica asociada a las acciones, pero en el resto de capítulos los modelos que definiremos tendrán esta información estocástica separada de las acciones.

Como ya hemos remarcado, exceptuando algunos trabajos relativamente recientes, la mayoría de los modelos estocásticos consideran que las distribuciones están restringidas a ser exponenciales. Como veremos más adelante, esta restricción simplifica algunos problemas que aparecen cuando se consideran distribuciones generalizadas. Por ejemplo, el cálculo de algunas cantidades de interés, como son las probabilidades de alcanzabilidad, se puede realizar eficientemente utilizando métodos ya conocidos sobre cadenas de Markov. Además, la definición de la semántica operacional del lenguaje es normalmente más simple que la correspondiente a un lenguaje que permite distribuciones generalizadas. Sin embargo, esta restricción no permite especificar adecuadamente algunos tipos de sistemas en los que las distribuciones de tiempo no son exponenciales.

Algo que tienen en común ambos tipos de álgebras son las distintas políticas existentes para la resolución de las elecciones entre acciones estocásticas. En el caso de que las elecciones sean no deterministas se utiliza la denominada *política de carreras*. Dicha política, como indica su nombre, consiste en resolver la elección en favor de la variable aleatoria más rápida, es decir de la que tome un valor menor. En consecuencia, al resolverse la elección se dirá que el retardo ejecutado vendrá definido por la variable aleatoria cuya distribución de probabilidad es el mínimo de las variables aleatorias involucradas en la elección. En [Hil93, Her98] se utiliza este tipo de política en la resolución de elecciones. En la mayoría de los casos en los que se utiliza esta política, se considera que las τ's son urgentes (véase p.ej. [HS95, HHK02]). Por otro lado, cuando se trabaja con elecciones cuantificadas probabilísticamente, o bien

[3]Durante todo este trabajo utilizaremos letras griegas como ξ, ξ_1, ψ, ψ_1, etc. para denotar las variables aleatorias. En el caso de que trabajemos con distribuciones exponenciales exclusivamente, utilizaremos λ, λ', etc. para denotar las variables aleatorias con distribución exponencial de parámetro λ, λ', respectivamente.

cuando las acciones estocásticas tienen pesos asociados, se suele utilizar una política de preselección. Es decir, primeramente se decide qué retardo se va a ejecutar, y después se ejecuta exclusivamente ese retardo (véase p.ej. [BG02b, LN01]).

Propiedades de las álgebras de procesos markovianos

Como ya hemos comentado, estas álgebras de procesos se caracterizan por especificar únicamente retardos aleatorios dados por funciones de distribución de probabilidad exponenciales. La distribución de probabilidad exponencial con parámetro λ está definida mediante la siguiente función:

$$F(t) = \begin{cases} 1 - e^{-\lambda t} & \text{si } t \geq 0 \\ 0 & \text{si } t < 0 \end{cases}$$

Las principales características de estas distribuciones son que la media es igual al inverso del parámetro de la distribución, son cerradas con respecto al mínimo (aunque no con respecto al máximo) y poseen la propiedad de *ausencia o falta de memoria*. Siendo ξ una variable aleatoria distribuida exponencialmente y $t, t' \in \mathbb{R}^+$, decimos que ξ tiene la propiedad de *falta de memoria* si se cumple $\mathrm{P}(\xi \leq t + t' \mid \xi > t) = \mathrm{P}(\xi \leq t')$. Es decir, la probabilidad de que pase un cierto tiempo $t + t'$ sabiendo que ya se han consumido t unidades es equivalente a la probabilidad de que pase el tiempo restante t'.

La principal dificultad a la hora de definir un modelo para procesos estocásticos es la definición del operador de composición paralela. Gracias a la propiedad anterior, en el caso de los modelos markovianos es mucho más sencillo que en el caso general. Veremos a continuación un par de ejemplos generales de cómo se definiría para procesos markovianos. Si consideramos un lenguaje en el que la información estocástica aparece asociada a las acciones (véase p.ej. [Hil93, BG98]), podríamos tener una sintaxis para nuestro lenguaje como la siguiente:

$$P ::= \sum_{i \in I} (a_i, \lambda_i)\,;P_i \mid P \parallel_A P \mid X \mid rec\,X.P$$

Como ya hemos explicado antes, el término $(a, \lambda)\,;P$ denota que la acción a se ofrece después de un retardo dado por una distribución exponencial de parámetro λ, mientras que el término $\sum_{i \in I}(a_i, \lambda_i)\,;P_i$ selecciona la alternativa más rápida. El proceso $P \parallel_A Q$ denota la composición paralela de los procesos P y Q, que deben sincronizar en las acciones que pertenecen a A. El comportamiento del operador de paralelo quedaría definido con las siguientes reglas:

$$\frac{P \xrightarrow{a,\lambda} P',\ a \notin A}{P \parallel_A Q \xrightarrow{a,\lambda} P' \parallel_A Q} \qquad \frac{Q \xrightarrow{a,\lambda} Q',\ a \notin A}{P \parallel_A Q \xrightarrow{a,\lambda} P \parallel_A Q'}$$

$$\frac{P \xrightarrow{a,\lambda} P',\ Q \xrightarrow{a,\mu} Q',\ a \in A}{P \parallel_A Q \xrightarrow{a,\lambda \oplus \mu} P' \parallel_A Q'}$$

La principal diferencia entre los distintos modelos que existen en la literatura es la definición de la función $\oplus : \mathbb{R}^+ \times \mathbb{R}^+ \longrightarrow \mathbb{R}^+$, que nos da el parámetro de la distribución de probabilidad exponencial resultante de la sincronización. Se ha propuesto que esa función sea el producto de las medias, el máximo de las mismas, e incluso en algunos trabajos se prohibe este tipo de sincronización en parte para evitar su definición (como es el caso de EMPA [BG98] que comentaremos después). Sin embargo, ninguna de las soluciones parece completamente adecuada puesto que, en realidad, la distribución resultante debería venir dada por la más lenta de las dos acciones, puesto que la sincronización sólo es posible si ambas acciones están dispuestas. Por lo tanto, se debería considerar el máximo de las dos distribuciones, que no es lo mismo que la distribución exponencial cuyo parámetro es el máximo de las medias de las otras dos. El problema de tomar esta solución consiste en que el máximo de dos distribuciones exponenciales no es una distribución exponencial, por lo que no sería expresable en un modelo markoviano.

En EMPA [BG98] se soluciona el problema del operador de composición paralela definiendo varios tipos distintos de acciones y una versión de paralelo con una serie de restricciones. En este modelo se distinguen tres tipos de acciones.

- Acciones *inmediatas*, denotadas por $(a, \infty_{l,w})$, donde l representa el nivel de prioridad de la acción y w el peso asociado a la misma. Estas acciones se deben ejecutar tan pronto como estén disponibles, es decir, son urgentes.

- Acciones *de tiempo exponencial*, denotadas por (a, λ), donde λ indica el parámetro de la distribución exponencial que define el retardo que sigue la acción.

- Acciones *pasivas*, denotadas por $(a, *)$. Estas acciones pueden esperar tanto tiempo como sea necesario antes de ser ejecutadas. Su duración se fija únicamente por la sincronización con una acción $(a, \infty_{l,w})$ o (a, λ).

La elección entre acciones del mismo tipo se resuelve de distinta forma según sea el tipo de las acciones involucradas. Tenemos elecciones *priorizadas*, *probabilísticas* y *no deterministas*. Cuando tenemos una elección entre acciones inmediatas con distinto nivel de prioridad, ésta se resuelve en favor de la acción más prioritaria. Si ambas acciones tienen el mismo nivel

entonces se resuelve probabilísticamente, teniendo en cuenta los respectivos pesos. Si tenemos dos acciones de tiempo exponencial, la elección se resuelve a favor de la más rápida utilizando la política de carreras. En el caso de dos acciones pasivas nos encontraremos ante una elección no determinista. En el resto de las elecciones, las acciones inmediatas tienen prioridad sobre las de tiempo exponencial y éstas sobre las pasivas. A la hora de definir el operador de composición paralela se impone una restricción a las acciones que van sincronizar: una de las dos acciones debe ser pasiva. De ese modo, la sincronización es inmediata si la otra acción era inmediata, mientras que se ejecutará después de un cierto tiempo dado por una distribución exponencial si la otra acción era de tiempo exponencial.

Cuando se considera un lenguaje donde la información estocástica aparece separada de las acciones, el problema de calcular la distribución asociada a la sincronización de dos acciones ya no surge. En un lenguaje de este tipo la sintaxis podría ser la siguiente:

$$P ::= \sum_{i \in I} a_i \,;\, P_i \mid \sum_{i \in I} \lambda_i \,;\, P_i \mid P \parallel_A P \mid X \mid rec\, X.P$$

En este caso, tendremos dos tipos de transiciones. Por un lado tendremos las transiciones temporales que representarán el paso del tiempo. Por otro lado, aquellas transiciones que denoten la ejecución de acciones habituales. En este caso, el comportamiento del operador de paralelo se definiría de la forma habitual para las acciones visibles, mientras que para las acciones estocásticas se definiría con las siguientes reglas:

$$\frac{P \xrightarrow{\lambda} P'}{P \parallel_A Q \xrightarrow{\lambda} P' \parallel_A Q} \qquad \frac{Q \xrightarrow{\lambda} Q'}{P \parallel_A Q \xrightarrow{\lambda} P \parallel_A Q'}$$

El hecho de que la definición sea tan sencilla es debido a que si consideramos la composición paralela de dos procesos como $\lambda \,;\, P$ y $\lambda' \,;\, Q$, y suponemos que se ejecuta inicialmente el retraso inducido por la variable aleatoria con parámetro λ, la distribución de probabilidad que mide el tiempo que falta para que concluya el retraso inducido por λ' no varía gracias a la propiedad de falta de memoria. Así, se cumple

$$\lambda \,;\, P \parallel_A \lambda' \,;\, Q = \lambda \,;\, (P \parallel_A \lambda' \,;\, Q) + \lambda' \,;\, (\lambda \,;\, P \parallel_A Q)$$

dado que si λ toma el valor t entonces la variable aleatoria que induciría el retardo para que evolucionase el lado derecho del paralelo tendría una distribución que sería igual a la dada por λ' para tomar un cierto valor $t + t'$ condicionada a que sea mayor que t, de modo que $\mathrm{P}(\lambda' \leq t + t' \mid \lambda' > t) = \mathrm{P}(\lambda' \leq t')$.

Un ejemplo de este tipo de lenguaje es IMC, definido en [Her98]. En este trabajo se define un modelo con acciones y transiciones estocásticas exponenciales separadas. La resolución

de las elecciones entre acciones estocásticas se hace mediante una política de carreras y el comportamiento del operador de composición paralela se define de la forma sencilla que hemos indicado antes.

Álgebras de procesos con distribuciones generales

Desgraciadamente, esta sencillez a la hora de definir el operador de paralelo desaparece cuando ampliamos nuestro marco de trabajo incluyendo distribuciones generales. Si consideramos que ξ y ψ son dos variables aleatorias arbitrariamente distribuidas, la siguiente ecuación, válida en el caso markoviano, ya no se cumple:

$$\xi\,;P\parallel_A \psi\,;Q = \xi\,;(P\parallel_A \psi\,;Q) + \psi\,;(\xi\,;P\parallel_A Q)$$

puesto que el paso del tiempo debe quedar reflejado en la componente de la composición que no evoluciona. Así, después del retardo dado por ξ, el tiempo que ha transcurrido debe utilizarse de algún modo para calcular el tiempo residual de ψ, para determinar cuánto tiempo falta antes de que b esté disponible. ¿Quién debería ser esa variable aleatoria nueva? Tenemos básicamente dos opciones:

- Tomar la probabilidad condicionada. En ese caso la nueva variable dependería del tiempo que ha transcurrido y por tanto la semántica operacional generaría sistemas de transiciones infinitos.
- Dividir la acción estocástica en dos acciones, una para iniciar el retardo y otra para acabarlo.

Pasamos a continuación a comentar brevemente algunas de las aproximaciones concretas que se han tomado a la hora de afrontar este problema.

En [DKB98a] se define un sistema de autómatas estocásticos donde existen un conjunto de posiciones que tienen asociados un conjunto de relojes. Cuando se alcanza una cierta posición, a cada reloj asociado se le asigna un valor según la distribución de probabilidad que siga dicho reloj. Cuando todos los relojes que están activados expiran, la acción se ejecuta y el sistema pasa a la siguiente posición. El lenguaje SPADES definido en este trabajo tiene dos operadores nuevos para tratar la información estocástica. En primer lugar, introducen el término $\{\!|C|\!\}P$ que realiza la operación de inicialización de los relojes a los valores dados por las funciones de distribución correspondientes, lo cual se corresponde con el inicio de la acción estocástica. El segundo término que incluyen en el lenguaje es $C \longmapsto P$ que denota el proceso que se comporta como P una vez que los relojes de C han expirado, y que representa

la terminación de la ejecución de la acción estocástica. A la hora de definir el operador de composición paralela, la idea consiste en que las acciones de cada componente que no sincronizan se ejecutan cuando los correspondientes relojes hayan expirado y los relojes asociados a la otra componente no se reinicializan. En el caso de las acciones que sincronizan se ejecuta esta sincronización cuando expiran los relojes asociados a ambas componentes (lo que podría considerarse como el máximo de las variables aleatorias).

En el trabajo definido en [BG02b] las acciones estocásticas aparecen explícitamente en la sintaxis del lenguaje. Estas acciones están separadas de las habituales y aparecen representadas en el lenguaje por su función de distribución de probabilidad. Cada una de las acciones estocásticas tiene asociado un peso, de modo que para resolver las elecciones entre acciones estocásticas se sigue una política de preselección. De ese modo, se elige inicialmente uno de los retardos de forma probabilística y éste es el único que se ejecuta resolviéndose la elección cuando empieza su ejecución. En cuanto a la definición de la semántica operacional, los retardos se dividen en *inicio* del retardo y *terminación* del mismo. Para conocer en cada momento qué retardo de los iniciados es el que finaliza, se establece un mecanismo dinámico de asignación de índices. Este método está ampliamente explicado en el capítulo 6 donde lo hemos adaptado a nuestro lenguaje `VPASPA`.

Semánticas para procesos estocásticos

Como ya hemos comentado, los trabajos en los que se utilizan procesos estocásticos se basan usualmente en semánticas de bisimulación, dejando de lado las semánticas de *testing*. En el caso de los modelos markovianos, todas las semánticas de las que tenemos conocimiento, salvo [BC00], son semánticas de bisimulación. En [BC00], los autores definen una semántica de *testing* para un álgebra de procesos estocásticos markovianos donde un proceso pasa una prueba con una cierta probabilidad que viene dada por la suma del tiempo medio de las acciones estocásticas ejecutadas en las computaciones exitosas.

En el caso de los modelos estocásticos no markovianos, los principales trabajos en el área también se han concentrado en definir semánticas de bisimulación. En el capítulo 5, daremos una detallada explicación de uno de nuestros trabajos donde se presenta una semántica de *testing* [LN01] para un modelo no markoviano.

A la hora de definir las semánticas de bisimulación, en la mayoría de los casos, se estudia por separado el comportamiento de las acciones normales y el de las acciones estocásticas. Habitualmente, en lo referente a las acciones normales se comprueba que su comportamiento sea similar al reflejado en las bisimulaciones clásicas no estocásticas. Es decir, si uno de los

dos procesos puede ejecutar una acción entonces el otro proceso debe ser capaz de imitar esa ejecución con esa misma acción (sin ejecutar ninguna acción más, en el caso de la bisimulación fuerte, y con la posibilidad de ejecutar acciones internas antes y/o después de dicha acción en la bisimulación débil). En cuanto a las acciones estocásticas, ocurre como en las bisimulaciones definidas para procesos probabilísticos: tenemos que tener en cuenta todas las posibles opciones de llegar a una cierta clase de equivalencia ejecutando una única transición estocástica (en el caso de bisimulaciones fuertes) o varias acciones internas y una estocástica (en el caso de la débil). Finalmente, se comprueba que la combinación de todas las distribuciones de probabilidad de las transiciones estocásticas que llevan a esa clase son equivalentes en ambos procesos. En el caso de que la política de carreras sea utilizada para resolver las elecciones, esta combinación consiste en calcular una distribución de probabilidad que refleje el comportamiento de una variable aleatoria distribuida como el mínimo de las distribuciones correspondientes. Sin embargo, en los casos donde las acciones estocásticas tienen pesos o probabilidades asociadas, la definición de la combinación de varias distribuciones de probabilidad es distinta en cada modelo. Por ejemplo, en [BG02b] los autores suman únicamente los pesos de aquellas acciones estocásticas cuya función de distribución sea la misma y, por supuesto, alcancen la misma clase de equivalencia.

1.2. Resumen de la tesis

En esta sección daremos un breve resumen de lo que se incluye en cada uno de los capítulos que forman parte de esta tesis. A pesar de ello, al inicio de cada uno de los capítulos se explicará con mayor detalle los objetivos y los contenidos del mismo.

En el capítulo 2 presentaremos una serie de nociones de bisimulación. Cada una de las definiciones dadas persigue que se cumplan unas determinadas características, identificando cada vez más procesos y por tanto debilitando cada vez más la noción de bisimulación. El objetivo de partida es la definición de una bisimulación que abstraiga (parcialmente) la información sobre la evolución interna de los procesos y que a su vez cumpla que la elección entre varias acciones internas sea asociativa. Llegados a este punto, se incluirá información temporal a los procesos a considerar, de modo que se estudiará la definición de una bisimulación temporal débil basada en las definiciones anteriores, de forma que se siga cumpliendo la asociatividad en las elecciones internas. Las enseñanzas obtenidas de este capítulo se utilizarán a lo largo de este trabajo para definir otras nociones de bisimulación.

En el capítulo 3 se define el primer modelo con información estocástica. Este modelo es la base a partir de la cual iremos trabajando en los siguientes capítulos. Definiremos un

lenguaje, que denominaremos `NMSPA`: *Non-Markovian Stochastic Process Algebra* (álgebra de procesos estocásticos no markovianos). Como ya indica su nombre, consideraremos distribuciones generales en lugar de distribuciones exponenciales. Ya aquí se comprueban las dificultades de la definición de un operador de paralelo para un modelo no markoviano. En esta primera propuesta, basándonos en trabajos dónde se definían extensiones temporales, se define una función que *envejece* el componente del paralelo que no evoluciona. En los siguientes capítulos no continuaremos con esta idea porque la utilización de este tipo de función obliga a considerar sistemas de transiciones con un número infinitos de estados.

El capítulo 4 contiene la definición de un modelo con información estocástica no markoviana, definido a partir de un sistema de transiciones etiquetado, sobre el que se van a definir tres semánticas de bisimulación. Se definirá una bisimulación fuerte, una débil y una que denominaremos bisimulación débil estocástica. Las dos primeras nociones de bisimulación están basadas en [Her98], donde se dan versiones markovianas de estas bisimulaciones, mientras que la última noción de bisimulación es completamente original. En este capítulo se pretende definir una bisimulación que identifique más procesos que el resto de bisimulaciones definidas hasta el momento, es decir, una bisimulación más débil que la débil. Para ello consideramos que no sólo se deben abstraer las transiciones internas sino también algunas transiciones estocásticas que cumplan una serie de características. Obviamente, las transiciones estocásticas no pueden considerarse como transiciones internas, porque tienen asociada una información temporal. Sin embargo, algunas secuencias de transiciones estocásticas se podrían identificar con una única transición. Así pues, nuestra nueva noción representa una *evolución* en el siguiente sentido: dos procesos son bisimilares fuertes si ejecutan las mismas acciones y tienen retardos idénticamente distribuidos independientemente del número de transiciones estocásticas que generan estos retardos. En el capítulo 6 trabajaremos con un álgebra de procesos en lugar de con sistemas etiquetados de transiciones. En ese capítulo daremos las definiciones de las bisimulaciones fuerte y débil cuando aparece el operador de paralelo. Sin embargo, en el capítulo 4 hemos preferido omitirlo debido a que la complejidad de la definición de la semántica operacional podría provocar un aumento innecesario de la complejidad a la hora de estudiar y comprender las semánticas definidas.

En el capítulo 5 estudiaremos una semántica de *testing* para una extensión probabilística del modelo definido en el capítulo anterior. A la hora de definir la semántica de los procesos, la teoría clásica de *testing* utiliza la idea de un *experimentador* que prueba los procesos a través de una serie de experimentos. Dependiendo de los resultados de esas pruebas, el experimentador es capaz de comprobar si dos procesos son o no equivalentes. La noción

de probar un proceso está especificada como la interacción entre el proceso a probar y un conjunto de pruebas. Normalmente, esta interacción se modela como la composición paralela de un proceso y una prueba. Como ya hemos visto en este mismo capítulo, se han definido semánticas de *testing* para extensiones probabilísticas, temporales y probabilístico-temporales. Desgraciadamente, la teoría de *testing* no se ha utilizado mucho en el campo de las álgebras de procesos estocásticos. La definición de una semántica de *testing* (que cumpla buenas propiedades) para este tipo de lenguajes es bastante complicada, debido a que se deben abstraer de alguna manera las secuencias de transiciones estocásticas, y esto no es fácil en general.

En el capítulo 6 combinaremos un álgebra de procesos (no trivial) con un lenguaje funcional (concurrente). En concreto, consideraremos un álgebra de procesos estocásticos con paso de valores que estará basada en los modelos desarrollados en los capítulos anteriores. Sin embargo, en este caso desarrollaremos toda la teoría para un lenguaje más extenso que el visto hasta ahora, pues incluiremos el operador de paralelo y definiremos tanto una bisimulación fuerte como una débil para este lenguaje, basadas ambas en las definiciones del capítulo 4. Puesto que las nociones de bisimulación ya se definieron con detalle en ese capítulo, en este caso nos limitaremos a adaptarlas a nuestro nuevo marco y trataremos con más detalle la definición de la semántica operacional una vez introducido el operador de composición paralela. Para estudiar las propiedades de las especificaciones dadas por el álgebra de procesos, *traduciremos* las mismas a programas funcionales escritos en un lenguaje funcional concurrente. Presentaremos dos ejemplos bastante extensos que muestran cómo las especificaciones pueden traducirse a ese lenguaje y cómo se pueden estudiar propiedades cuantitativas.

Aunque en cada uno de los capítulos iremos presentando un resumen de las principales aportaciones de cada tema, finalizaremos este trabajo con un capítulo en el que mostraremos una visión conjunta de los logros alcanzados, así como de las líneas de trabajo futuro que tenemos intención de explorar.

Capítulo 2

Bisimulaciones globales. Una bisimulación global temporal

Para poder estudiar nociones de bisimulación apropiadas en el marco de los procesos estocásticos, conviene hacer un estudio preliminar sobre las deficiencias de la propia definición en el caso de procesos no estocásticos. En particular, un primer paso a tener en cuenta es la definición de bisimulaciones para procesos temporizados, es decir, procesos donde los *retardos* se especifican de forma determinista. En este capítulo presentaremos varias nociones de *bisimulación global* como un marco para estudiar la semántica de procesos no deterministas. En concreto nos concentraremos en los distintos puntos *débiles* que nos gustaría evitar a la hora de extender la noción clásica de bisimulación débil con información temporal. Así, a través de las bisimulaciones globales que definiremos se conseguirá la equivalencia de diferentes tipos de no determinismo que siempre se han distinguido en el área de la bisimulación débil. En particular, mostraremos como conseguir que se cumpla la asociatividad de las elecciones internas.

Estas bisimulaciones globales se van a definir exactamente igual que la bisimulación débil una vez que las transiciones habituales son sustituidas por nociones adecuadas de transiciones generalizadas. Como consecuencia de las distintas aproximaciones a la bisimulación global, vamos a deducir una noción de *bisimulación global temporal*. Como en el caso no temporal, esta relación será más débil que la bisimulación débil temporal clásica.

Para valorar adecuadamente la definición que daremos de bisimulación global temporal, presentaremos una colección de pequeños ejemplos que ilustrarán cada una de las características de nuestra definición. Finalmente, presentaremos un ejemplo más elaborado para resumir las propiedades de esta noción de bisimulación temporal.

Los resultados que aparecen en este capítulo se encuentran recogidos en [dFLN99a, dFLN99b, dFLN99c].

2.1. Motivación

La semántica operacional es la manera más simple e intuitiva de dar significado a los procesos que se pueden definir a partir de un lenguaje formal. Habitualmente, dicha semántica se define a través de un *sistema etiquetado de transiciones*. Estos sistemas describen las transiciones que cada proceso puede ejecutar y, normalmente, están definidos de forma estructurada mediante la aplicación de reglas de inferencia [Plo81]. A partir del conjunto de transiciones de un proceso podemos obtener sus computaciones, que nos informarán acerca del comportamiento del proceso. Pero este tipo de semánticas no son suficientemente abstractas dado que contienen información completa acerca de la sintaxis de los procesos. Necesitamos por tanto, eliminar parte de la información que aporta la semántica operacional para definir las equivalencias semánticas deseadas. Si tomamos las computaciones de un proceso y eliminamos la información sobre los procesos que etiquetan los estados por los que va pasando, obtenemos la *semántica de trazas*. Esta resulta bastante satisfactoria en tanto y cuando solamente se consideren procesos deterministas. Sin embargo, cuando aparece el no determinismo, las trazas no ofrecen información suficiente para controlar los posibles puntos muertos a los que puede llegar el proceso. El problema proviene del hecho de que estamos considerando computaciones aisladas, puesto que con ello se pierde la información acerca de los puntos donde se toman las decisiones no deterministas. Para mantener esta información, es necesario considerar los árboles de computación de cada proceso, cuyas ramificaciones aportan información sobre los puntos en los que existen elecciones. Gracias a esta semántica, se podrían obtener algunas equivalencias también razonables, pero no triviales, tales como $P + Q \sim Q + P$, donde $+$ representa el operador de elección de CCS. De todos modos, no tendríamos todavía otras equivalencias razonables como, por ejemplo, $P + P \sim P$, debido a que cualquier computación de P aparecería dos veces como una computación de $P + P$. De modo que parece que tomando el árbol de computaciones en lugar del conjunto de computaciones, llegamos justamente al polo opuesto: la equivalencia inducida pasa a ser demasiado fuerte.

La solución habitual a este problema para por introducir una noción de *bisimulación* [Par81, Mil89], que definirá una equivalencia ligeramente más débil que la inducida por el árbol de computaciones. La bisimulación fuerte es una manera bastante adecuada de definir la semántica de procesos cuyas transiciones están etiquetadas con acciones *visibles*. Este tipo

de acciones cumple dos objetivos diferenciados a la hora de definir procesos. Por un lado, se trata de acciones observables desde el exterior, lo que significa que todas las transiciones ejecutadas por el proceso son observables. Por otro lado, cuando se consideran procesos reactivos, las acciones visibles no sólo pueden ser observados, sino que su ejecución será controlada desde el exterior.

No obstante, los procesos que tienen todas sus transiciones visibles no son suficientemente flexibles, por lo que necesitamos algún tipo de transición *interna* que refleje aquellas partes del comportamiento de los procesos que ni es observable, ni resulta controlable desde el exterior. Cuando definimos la semántica operacional de un lenguaje de procesos, normalmente utilizamos transiciones etiquetadas, cuyas etiquetas son acciones visibles, y otras no etiquetadas, o bien etiquetadas con acciones especiales (τ en CCS, i en LOTOS, etc.), que se corresponden con los movimientos internos del proceso. Pero estas acciones internas deben tratarse de manera especial a la hora de capturar el hecho de que no son observables. Por este motivo apareció una nueva noción de bisimulación que es capaz de abstraer parcialmente la evolución interna de los procesos: la *bisimulación débil*. La forma en la que estas transiciones son tratadas en la bisimulación débil permite que si un proceso tiene la posibilidad de ejecutar una transición interna, el proceso que debe simularle pueda elegir entre ejecutar una única acción interna, varias o incluso ninguna. Como consecuencia, utilizando sintaxis de CCS, tenemos que, por ejemplo, se cumple $\tau\,;P \sim P \sim \tau\,;\tau\,;P$. Sin embargo, la bisimulación débil no ignora las transiciones internas cuando aparecen en el ámbito de una elección. Así, tenemos que $P + Q \not\sim P + \tau\,;Q$, lo cual es bastante razonable dado que el segundo proceso puede decidir internamente seguir por la parte derecha de la elección. Desgraciadamente, esta capacidad de *ver* acciones internas se mantiene cuando aparecen elecciones anidadas. De modo que tenemos $\tau\,;(\tau\,;P + \tau\,;Q) + \tau\,;R \not\sim \tau\,;P + \tau\,;(\tau\,;Q + \tau\,;R)$, que en un lenguaje tipo CSP nos llevaría a $(P \oplus Q) \oplus R \not\sim P \oplus (Q \oplus R)$, donde $\oplus$ representa el operador de elección interna.

El hecho de que la elección interna no sea asociativa bajo bisimulación débil es, en nuestra opinión, un serio inconveniente. Erdogmus et al. [EJF96] observaron el mismo problema, aunque ellos han tratado de resolverlo de una manera distinta a la aquí presentada. Consideran que la noción de bisimulación no es errónea, sino que lo que está mal es la semántica operacional a partir de la cual está definida la equivalencia. En consecuencia, sugieren una noción más complicada de sistema de transiciones para definir la semántica operacional de procesos no deterministas. Definiendo así unos *sistemas de transiciones abstractos*, donde las transiciones internas se sustituyen por estados estructurados, formados por varios estados

internos, en cada uno de los cuales se ofrecen varias acciones de manera determinista. Estos sistemas recuerdan bastante a los *árboles de aceptación* [Hen85], aunque estos últimos se utilizaban para un objetivo diferente, en concreto el de definir una semántica denotacional completamente abstracta con respecto a la semántica de *testing*.

Utilizando sistemas de transiciones abstractos, las elecciones no deterministas se capturan de manera estática, y como quiera que el no determinismo es una propiedad estática de los procesos, ésta parece una buena estrategia para afrontar los problemas planteados por las acciones internas. En efecto, no creemos que debiera estar representada con transiciones dinámicas, especialmente cuando esto nos lleva a resultados que no son los deseados, como la no asociatividad del operador de elección interna. Coincidimos con Erdogmus et al. en el hecho de que las transiciones internas no son el mecanismo más adecuado para representar el no determinismo; de hecho, varios autores lo observaron ya mucho antes, aunque no siguieron investigando la cuestión (p.ej. [dNH87]). No obstante, hemos comprobado que los sistemas de transiciones abstractos definidos por Erdogmus et al. no son necesarios para obtener los resultados deseados. Nosotros utilizaremos los sistemas de transiciones originales, manteniendo la semántica operacional de partida, como base para definir nuestra nueva semántica. Sin embargo, no utilizaremos directamente estas transiciones en la definición de bisimulación. Más exactamente, definiremos *bisimulaciones globales* que combinan dos tipos distintos de movimientos: *dinámicos*, que se van a corresponder con la ejecución de acciones visibles, y *estáticos*, que se corresponderán con resoluciones (parciales) de no determinismo. Estos últimos movimientos justifican el nombre de nuestra noción de bisimulación, debido a que las transiciones involucradas no necesitan aparecer al principio de las computaciones, sino que pueden aparecer en cualquier otro sitio.

El principal motivo para desarrollar este trabajo, cuando Erdogmus et al. ya afrontaron el mismo problema, es que aunque ellos capturan la razón por la cual la bisimulación débil es muy fuerte, la solución que presentan no es muy satisfactoria. Está basada en la traducción de los sistemas de transiciones ordinarios a sistemas abstractos en los que las *elecciones mixtas*, como $P + \tau\ ;\ Q$, se traducen a elecciones internas puras, como $\tau\ ;\ P + \tau\ ;\ Q$. En particular, esta traducción no es consistente con la semántica de pruebas (*must*) [Hen88]. Para capturar el significado semántico de esta traducción proponen una bisimulación que ellos denominan *bisimulación coarse*,[1] como la equivalencia necesaria para lograr esta visión abstracta de las transiciones internas. En nuestra opinión, esta aproximación no es correcta, e incluso no hay ninguna razón para considerar esta bisimulación como el representante

[1] Una bisimulación se dice que es *coarser* si identifica más procesos. En consecuencia, Erdogmus et al. definen una bisimulación que identifica más procesos que la noción clásica.

canónico de la noción de bisimulación cuando las transiciones internas son abstraídas correctamente. El único motivo para preferir su noción de bisimulación coarse podría ser que se trata de una definición más estándar que la que vamos a presentar en este capítulo, en el sentido de que para definir su equivalencia únicamente se manejan las transiciones que los procesos pueden ejecutar inicialmente desde el estado en el que se encuentran. Sin embargo, una vez que se considera que los movimientos internos requieren un tratamiento estático, no es razonable restringirse a aquellos movimientos que pueden hacerse al principio de las computaciones. Además, las bisimulaciones globales que presentamos en este capítulo están definidas del mismo modo que la bisimulación débil, una vez que las transiciones ordinarias son reemplazadas por nociones adecuadas de *transiciones generalizadas*. De este modo, para desarrollar nuestro estudio empezaremos a partir de la definición de bisimulación débil y estudiaremos cuales son las diferentes razones por las que podemos estar interesados en ir debilitando esa noción, y veremos cómo podríamos hacerlo. Este *debilitamiento* se producirá sin cambiar el corazón de la definición de bisimulación débil, sino mediante la modificación y adaptación de las correspondientes transiciones generalizadas.

El resto del capítulo está estructurado de la siguiente manera. En la próxima sección presentaremos las diferentes definiciones de bisimulación global que irán apareciendo según vayamos añadiendo propiedades que creemos se deben cumplir. Veremos los formalismos necesarios para describir sistemas de transiciones, daremos la definición de bisimulación global e iremos dando las reglas a partir de las cuales definiremos nuestras transiciones generalizadas. A continuación, en la sección 2.3 mostraremos las principales propiedades de la bisimulación global. En la sección 2.4 definimos una extensión temporal de la noción anterior, la bisimulación global temporal, y mostraremos las ventajas de esta noción con respecto a la noción clásica de bisimulación débil temporal. En la sección 2.5 resumimos las principales ideas que hemos presentado en relación al tratamiento del no determinismo, estudiando las posibles implementaciones de un canal de comunicación defectuoso.

2.2. Bisimulaciones globales

Como se comentó en la introducción, utilizaremos una sintaxis abstracta para modelar nuestros sistemas, en vez de utilizar álgebras de procesos, a fin de que la definición de las reglas sea más sencilla. De modo que definiremos los procesos en términos de *sistemas etiquetados de transiciones*. Utilizando esta sintaxis podremos abstraer las características particulares de cualquier lenguaje en concreto. En otras palabras, en vez de definir una semántica operacional sobre un lenguaje fijo y después estudiar el sistema etiquetado de

transiciones inducido, vamos a considerar directamente un tipo (general) de sistemas etiquetados de transiciones.

Consideraremos un *conjunto de acciones visibles* $\mathtt{Act}$, y una acción especial τ no perteneciente al conjunto $\mathtt{Act}$. Esta acción servirá para representar evoluciones internas de los correspondientes procesos. Finalmente, el conjunto $\mathtt{Act}_\tau$ contiene todas las acciones que se usarán para describir el comportamiento de los procesos, es decir, $\mathtt{Act}_\tau = \mathtt{Act} \cup \{\tau\}$.

Definición 2.1 Definimos un proceso P como la tupla $(S, \longrightarrow, s_0)$, donde S es un *conjunto de estados*, $s_0 \in S$ es el *estado inicial* y $\longrightarrow \subseteq S \times \mathtt{Act}_\tau \times S$ es la *relación de transiciones*. Normalmente escribiremos $s \xrightarrow{\alpha} s'$ en lugar de $(s, \alpha, s') \in \longrightarrow$. □

Habitualmente, identificaremos los nodos de estos sistemas etiquetados de transiciones con los procesos definidos tomando el correspondiente nodo como estado inicial. Formalmente, si $P = (S, \longrightarrow, s_0)$ es un proceso y $s \in S$ denotaremos por P_s al proceso $(S, \longrightarrow, s)$.

Para facilitar la presentación y estudio de las principales características de nuestra semántica consideraremos únicamente sistemas de transiciones en forma de *árbol*, que son aquellos tales que el grafo asociado es un árbol con raíz. Sin embargo, debemos resaltar el hecho de que esta restricción no supone una perdida de generalidad, puesto que desdoblando cualquier sistema de transiciones podemos encontrar un sistema de transiciones equivalente en forma de árbol. Este hecho se debe a que nuestra definición de bisimulación global no utiliza en ningún momento la estructura global subyacente. Además, es aplicable a un sistema arbitrario, finito o infinito. De ese modo, incluso si partimos de un sistema de estados finito con una estructura de red arbitraria, que se convertiría en un árbol infinito una vez desplegado, no encontraremos ningún problema en absoluto. Sin embargo, es cierto que si estamos interesados en la decibilidad de la bisimulación global, debemos mantener la finitud de los sistemas cuando los estudiamos.

A continuación, para poder manejar adecuadamente los procesos, introduciremos cierta notación auxiliar que utilizaremos a lo largo de este capítulo.

Definición 2.2 Sean $P = (S, \longrightarrow, s_0)$ un proceso y $\{s_{a_i} \in S \mid s_0 \xrightarrow{a} s_{a_i}\}$ el conjunto de estados alcanzables mediante la ejecución de la transición $\xrightarrow{a}$ desde s_0, y sea $\{P_{a_i}\}$ el conjunto de los correspondientes procesos inducidos. Del mismo modo, sean $\{P_{\tau_i}\}$ y $\{s_{\tau_i} \in S \mid s_0 \xrightarrow{\tau} s_{\tau_i}\}$ los conjuntos de procesos y de estados, respectivamente, alcanzables después de ejecutar la transición $\xrightarrow{\tau}$ desde s_0. Entonces, para referimos a P utilizaremos la siguiente notación:

- $P = \sum_{a \in \mathtt{Act}} a\,;P_{a_i} + \sum \tau\,;P_{\tau_i}$ si $\{P_{\tau_i}\} \neq \emptyset$
- $P = \sum_{a \in \mathtt{Act}} a\,;P_{a_i}$ si $\{P_{\tau_i}\} = \emptyset$
- $P = stop$ si $\forall a \in \mathtt{Act}: \ \{P_{a_i}\} = \emptyset \ \wedge \ \{P_{\tau_i}\} = \emptyset$

□

Nótese que algunas de las sumas $\sum_{a \in \mathtt{Act}} a\,;P_{a_i}$ pueden ser vacías. Además, escribiremos $P = \sum_{\alpha} \alpha\,;P_{\alpha}$ cuando no necesitemos distinguir entre descendientes de transiciones visibles o internas. Por último, utilizaremos el proceso *stop* para denotar procesos que no pueden ejecutar ninguna acción.

Durante el resto del capítulo utilizaremos, en algunas ocasiones, operadores algebraicos para definir la composición de procesos. Ello será especialmente relevante a la hora de presentar leyes que nuestra equivalencia debe cumplir. En concreto, utilizaremos el operador de elección de CCS, que denotamos por $+$, y un operador de elección interna como el de CSP, que denotamos por $\oplus$. La definición formal de dichos operadores se presenta a continuación.

Definición 2.3 Sean $P_1 = (S_1, \longrightarrow_1, s_1)$ y $P_2 = (S_2, \longrightarrow_2, s_2)$ un par de procesos que verifican $S_1 \cap S_2 = \emptyset$. Definimos la *elección* entre los procesos P_1 y P_2, denotada por $P_1 + P_2$, como el proceso $(S_1 - \{s_1\} \cup S_2 - \{s_2\} \cup \{s_0\}, \longrightarrow, s_0)$, donde $s_0 \notin S_1 \cup S_2$ y la relación de transición se define por medio de

$$s \xrightarrow{\alpha} s' \text{ sii } \begin{array}{l} (s \xrightarrow{\alpha}_1 s' \ \wedge \ s \neq s_1) \ \vee \ (s \xrightarrow{\alpha}_2 s' \ \wedge \ s \neq s_2) \ \vee \\ (s_1 \xrightarrow{\alpha}_1 s' \ \wedge \ s = s_0) \ \vee \ (s_2 \xrightarrow{\alpha}_2 s' \ \wedge \ s = s_0) \end{array}$$

Definimos la *elección interna* entre los procesos P_1 y P_2, denotada por $P_1 \oplus P_2$, como el proceso $(S_1 \cup S_2 \cup \{s_0\}, \longrightarrow, s_0)$, donde $s_0 \notin S_1 \cup S_2$ y la relación de transición partiendo de cualquier $s \in S_1 \cup S_2$ se define como

$$s \xrightarrow{\alpha} s' \text{ sii } (s \xrightarrow{\alpha}_1 s') \ \vee \ (s \xrightarrow{\alpha}_2 s')$$

y además tenemos que $s_0 \xrightarrow{\tau} s_1$ y $s_0 \xrightarrow{\tau} s_2$. □

En la figura 2.1 aparecen ejemplos gráficos que muestran el funcionamiento de estos operadores. Debemos mencionar que, a nivel composicional, nuestro operador de elección interna no funciona exactamente como el correspondiente en CSP. Ello es debido a que en el entorno de una elección externa el operador de elección interna de CSP no *resuelve* la elección, mientras que el nuestro que sí lo hace, al tratarse de prefijar con acciones internas los procesos correspondientes. En todo caso, ello no representa ningún problema dado que en este capítulo

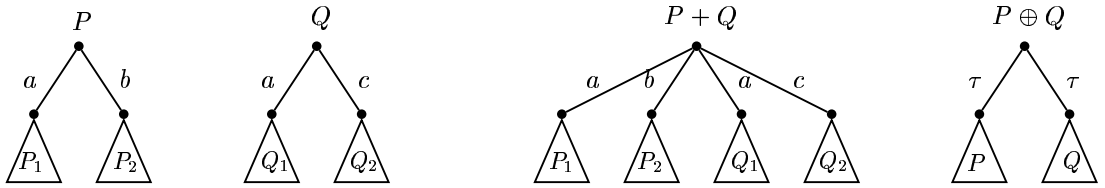


Figura 2.1: Operadores de elección externa e interna.

nuestro objetivo no es el de trabajar en un marco algebraico donde la composicionalidad es fundamental. Adicionalmente, conviene destacar que la definición de nuestro operador de elección debería modificarse si permitiéramos que los procesos presentaran *bucles*. En tal caso, deberíamos mantener, para cada uno de los procesos que formen parte de la elección, tanto sus estados iniciales como las transiciones que parten de éstos.

A continuación, definimos las extensiones habituales de las relaciones de transición para permitir varias ejecuciones de acciones internas seguidas en un único paso.

Definición 2.4 Siendo $P = (S, \longrightarrow, s_0)$ un proceso, definimos las transiciones extendidas $\stackrel{\alpha}{\Longrightarrow}$ con $\alpha \in \mathtt{Act}_\tau$

- Denotamos por $\stackrel{\tau}{\Longrightarrow}$ la clausura reflexiva y transitiva de $\stackrel{\tau}{\longrightarrow}$ dada por $\stackrel{\tau}{\longrightarrow}{}^*$.
- Sean $s, s' \in S$ y $a \in \mathtt{Act}$. Escribimos $s \stackrel{a}{\Longrightarrow} s'$ si existen estados s_1, s_2 tales que

$$s \stackrel{\tau}{\Longrightarrow} s_1 \stackrel{a}{\longrightarrow} s_2 \stackrel{\tau}{\Longrightarrow} s'$$

□

Por último, recordaremos la definición de la noción clásica de bisimulación débil base de nuestra definición de bisimulación global.

Definición 2.5 Una relación de equivalencia $\mathcal{R}$ definida sobre procesos se dice que es una *bisimulación débil* si para cada par de procesos P, Q tales que $P\,\mathcal{R}\,Q$, y para todo $\alpha \in \mathtt{Act}_\tau$ tenemos que

- Si $P \stackrel{\alpha}{\longrightarrow} P'$ entonces existe algún Q' tal que $Q \stackrel{\alpha}{\Longrightarrow} Q'$ y $P'\,\mathcal{R}\,Q'$.

Decimos que dos procesos P y Q son *bisimilares débiles* y lo denotamos por $P \sim_w Q$, si existe una bisimulación débil que contenga el par $\langle P, Q\rangle$. □

Como ya comentamos en el capítulo 1, al tratarse de una relación de equivalencia (en particular, simétrica) el caso simétrico en el que P y Q intercambian sus papeles queda implícitamente capturando en la propia definición.

2.2.1. Definiciones de bisimulaciones globales

En esta sección presentaremos diversas nociones de *bisimulación global*. Es importante observar que cuando hemos utilizado el plural en la sentencia anterior no nos estábamos refiriendo a distintas relaciones a través de las cuales vayamos a definir la noción de bisimulación global, sino a la noción en sí misma. Estas relaciones serán más y más débiles, a medida que tratemos de identificar diferentes tipos de no determinismo. De modo que iremos discutiendo las distintas identificaciones de procesos en las que podríamos estar interesados, cuáles son las propiedades que nos interesan y cuáles son las modificaciones necesarias de las transiciones generalizadas para obtener nociones de bisimulación global que satisfagan esas propiedades. Denotaremos por $\rightsquigarrow$ y $\overset{a}{\rightsquigarrow}$ a las transiciones *estáticas* y *dinámicas generalizadas* que definiremos a lo largo de la sección. Además, consideraremos que $\approx\!>$ representa la clausura reflexiva y transitiva de $\rightsquigarrow$, mientras que $\overset{a}{\approx\!>}$ será igual a $\approx\!> \cdot \overset{a}{\rightsquigarrow} \cdot \approx\!>$.

Definición 2.6 Una relación de equivalencia $\mathcal{R}$ definida entre pares de procesos es una *bisimulación global* si para todo par de procesos P, Q tales que $P\ \mathcal{R}\ Q$ y para cualquier $\alpha \in \mathtt{Act} \cup \{\epsilon\}$ tenemos:

- Si $P \overset{\alpha}{\rightsquigarrow} P'$ existe algún Q' tal que $Q \overset{\alpha}{\approx\!>} Q'$ y $P'\ \mathcal{R}\ Q'$.

Decimos que dos procesos P y Q son *bisimilares globales*, denotado por $P \sim_g Q$, si existe una bisimulación global que contenga el par $\langle P, Q\rangle$. □

Obsérvese que todavía no está definido el significado de $\rightsquigarrow$ y $\overset{a}{\rightsquigarrow}$. Como ya hemos comentado, iremos definiendo *incrementalmente* diferentes variantes de estas nociones de acuerdo con las propiedades que queremos que se satisfagan. Nuestro punto de partida será la bisimulación débil, es decir, partimos de $\sim_g \equiv \sim_w$. De este modo, tenemos que la bisimulación global identificaría a los mismos procesos que la débil, es decir, verifica que la elección es conmutativa tanto para elecciones de acciones visibles como para elecciones entre τ's, la elección entre acciones visibles es asociativa y *stop* es el elemento identidad, y las elecciones de τ's puras, es decir, las elecciones internas, son idempotentes. Así, tenemos las siguientes equivalencias entre procesos:

- $P + stop \sim_g P$
- $P + Q \sim_g Q + P$
- $(P + Q) + R \sim_g P + (Q + R)$
- $P \oplus P \sim_g P$
- $P \oplus Q \sim_g Q \oplus P$

Por el momento, las correspondientes nociones de transiciones generalizadas coinciden con las transiciones ordinarias utilizadas en la definición de bisimulación débil, es decir, $\rightsquigarrow \equiv \xrightarrow{\tau}$ y $\overset{a}{\rightsquigarrow} \equiv \xrightarrow{a}$. Este hecho queda reflejado en la siguiente regla:

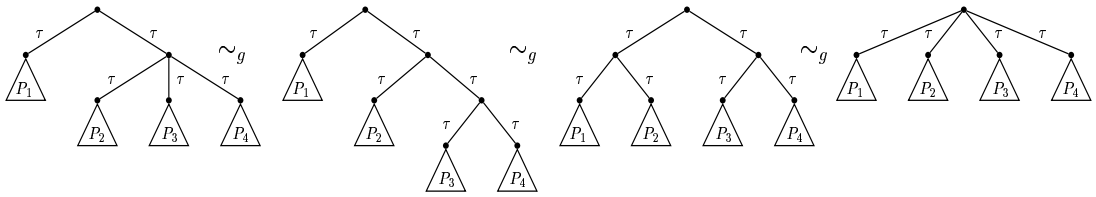


Figura 2.2: Asociatividad de elecciones puras de τ's.

(I) *Preservando la relación de transición de* P:

- Si $P \xrightarrow{\tau} P'$ entonces $P \rightsquigarrow P'$.
- Si $P \xrightarrow{\alpha} P'$ para algún $\alpha \in \mathtt{Act}$ entonces $P \overset{\alpha}{\rightsquigarrow} P'$.

2.2.2. Asociatividad de las elecciones internas

Como ya hemos comentado, una de las principales motivaciones de este trabajo es la de desarrollar una semántica observacional basada en la bisimulación, para la que el operador de elección interna sea asociativo. Nos interesaría que se produjera la situación descrita en la figura 2.2. Para ilustrar cómo debe extenderse la noción de transición generalizada para obtener la deseada asociatividad, estudiaremos el caso en el que tenemos únicamente tres argumentos. Consideremos los siguientes procesos:

$$P = \tau\,;(\tau\,;P_1 + \tau\,;P_2) + \tau\,;P_3 \qquad P' = \tau\,;P_1 + \tau\,;(\tau\,;P_2 + \tau\,;P_3)$$

$$P'' = \tau\,;P_1 + \tau\,;P_2 + \tau\,;P_3$$

Estudiemos qué *capacidades* debemos añadir a la relación de transiciones habitual que para llegar a obtener $P \sim_g P'$ y $P' \sim_g P''$. Para mostrar que P y P' son bisimilares globales, si queremos simular la transición $P \rightsquigarrow \tau\,;P_1 + \tau\,;P_2$, es necesario que P' pueda ejecutar una acción interna τ que no se encuentra en la raíz del proceso. En tal caso, dado que $\tau\,;P_2 + \tau\,;P_3 \xrightarrow{\tau} P_2$, podríamos deducir la transición $P' \rightsquigarrow \tau\,;P_1 + \tau\,;P_2$.

Por otro lado, para conseguir que P'' sea capaz de simular la misma transición de P, P'' debe poder ejecutar simultáneamente dos transiciones internas $P'' \xrightarrow{\tau} P_1$ y $P'' \xrightarrow{\tau} P_2$, sin resolver la elección entre ellas, con lo que P'' evolucionaría a $\tau\,;P_1 + \tau\,;P_2$.

A partir de este caso particular podemos concluir que para conseguir la asociatividad del operador de elección cuando tenemos elecciones entre acciones internas, la definición de

transiciones generalizadas internas se debería extender permitiendo las siguientes transiciones nuevas:

(II) *Ejecución avanzada de un movimiento interno* (apareciendo después de una secuencia de transiciones internas):

$$P_\beta \rightsquigarrow P'_\beta \implies \sum_\alpha \alpha \,;\, P_\alpha + \tau \,;\, P_\beta \rightsquigarrow \sum_\alpha \alpha \,;\, P_\alpha + \tau \,;\, P'_\beta$$

(III) *Ejecución simultánea de varias transiciones internas*:

$$\sum_\alpha \alpha \,;\, P_\alpha + \sum_{i\in I} \tau \,;\, P_i \rightsquigarrow \sum_{i\in I} \tau \,;\, P_i$$

Obsérvese que en ambas reglas algunas de las α's que aparecen en los sumatorios pueden ser iguales a τ. También se dará esta misma situación en algunas de las reglas que aparecerán con posterioridad. Por ejemplo, este es el caso en la regla (IV) donde algunas de las α's pueden ser iguales a a.

2.2.3. Diferentes tipos de no determinismo simétrico

Una vez que tenemos la capacidad de resolver varias elecciones internas al tiempo, fijamos nuestra atención en las distintas formas que existen de generar no determinismo. En particular, se genera una elección interna implícita cuando tenemos una elección externa entre dos procesos que ofrecen algunas acciones visibles comunes. Bajo bisimulación débil este tipo de no determinismo se considera diferente al generado por elecciones internas. Más exactamente, si consideramos dos procesos P_1, P_2 que no son bisimilares débiles, tenemos que el proceso $a \,;\, P_1 + a \,;\, P_2$ no es equivalente a ningún otro proceso que no contenga una elección externa entre dos procesos que ofrezcan la acción a. Es fácil comprobar que este resultado se mantiene cuando consideramos las transiciones generalizadas, tal y como están definidas en este momento, para obtener la asociatividad de las elecciones internas.

Si queremos identificar este tipo de no determinismo con otros similares, necesitamos modificar las nociones de transiciones generalizadas de la manera adecuada. Consideremos los tres procesos siguientes:

$$P_1 = a \,;\, b + a \,;\, c \qquad P_2 = \tau \,;\, a \,;\, b + \tau \,;\, a \,;\, c \qquad P_3 = a \,;\, (\tau \,;\, b + \tau \,;\, c)$$

gráficamente representados en la figura 2.3.

A la hora de definir las correspondientes nociones de bisimulación global tenemos que para que se cumpla $P_1 \sim_g P_2$, el movimiento $P_2 \xrightarrow{\tau} a\,;b$ debe ser simulado eliminando la rama $a\,;c$ de P_1 para obtener $P_1 \rightsquigarrow a\,;b$. Nótese que este *movimiento* es similar al producido mediante la ejecución simultánea de varias transiciones internas. Ello es así porque en ambos casos en el proceso resultante ha disminuido la cantidad de no determinismo. Para poder realizar este tipo de transición generalizada necesitamos la siguiente regla:

(IV) *Reducción de no determinismo* (generado por la oferta simultánea de la misma acción visible):

$$\sum_{\alpha} \alpha\,;P_\alpha + a\,;P_a + \sum_{i\in I} a\,;P_i \rightsquigarrow \sum_{\alpha} \alpha\,;P_\alpha + a\,;P_a$$

Por otro lado, para obtener $P_1 \sim_g P_3$, necesitamos simular la transición $P_3 \overset{a}{\rightsquigarrow} \tau\,;b+\tau\,;c$. En este caso, P_1 debe ejecutar simultáneamente dos acciones a posponiendo la elección entre sus continuaciones, con lo que obtendríamos $P_1 \overset{a}{\rightsquigarrow} \tau\,;b+\tau\,;c$. De modo que en este caso las transiciones que necesitamos generalizar son las visibles. Más exactamente, necesitamos extender la noción de transición visible permitiendo la ejecución simultánea de cualquier conjunto de transiciones $P \xrightarrow{a} P_i$, retrasando la elección entre sus continuaciones P_i. Formalmente, la regla es la siguiente:

(V) *Ejecución simultánea de varios ejemplares de una acción ofrecida*:

$$\forall a \in \mathtt{Act}, \forall I \subseteq I_a \text{ con } I_a = \{i \mid P \overset{a}{\Longrightarrow} P_i\} \implies P = \sum_{\alpha} \alpha\,;P_\alpha + \sum_{i\in I} a\,;P_i \overset{a}{\rightsquigarrow} \sum_{i\in I} \tau\,;P_i$$

Obsérvese que cuando $|I| = 1$ obtenemos $P \overset{a}{\rightsquigarrow} P'$ si $P \xrightarrow{a} P'$. Para reducir el no determinismo generado por elecciones externas transformándolo en elecciones internas podemos elegir entre las dos opciones presentadas previamente, es decir, $P_1 \sim_g P_2$ o bien $P_1 \sim_g P_3$, pero esta elección no es necesariamente exclusiva; tenemos una tercera posibilidad: identificar los tres procesos P_1, P_2 y P_3. Si estamos interesados en esta identificación, está claro que a nivel algebraico bastaría con poner juntas las leyes definen cada uno de los dos casos. Se podría pensar que lo mismo es cierto para las reglas que definen las correspondientes nociones de transición generalizada, debido a que tenemos $P_1 \sim_g P_2$ y $P_1 \sim_g P_3$. Por tanto, vía transitividad, deberíamos concluir $P_2 \sim_g P_3$. Pero, sorprendentemente, esto no es cierto, aunque no es la aplicación de la transitividad lo que falla, sino la hipótesis de que tengamos a la vez $P_1 \sim_g P_2$ y $P_1 \sim_g P_3$. Resulta que las condiciones para tener de forma independiente cada

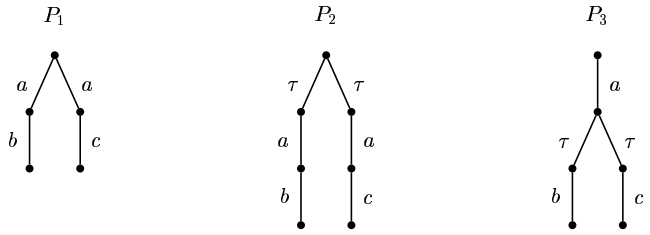


Figura 2.3: Diferentes tipos de no determinismo simétrico.

una de estas equivalencias, bajo la noción adecuada de bisimulación global, no son suficientes para conseguir que se cumplan ambas al tiempo. Esto significa que si intentamos definir una noción de bisimulación global que satisfaga un conjunto de propiedades, a través de una generalización adecuada de las transiciones permitidas, después de cada refinamiento, debemos comprobar que ninguna de las propiedades que antes se satisfacían ha dejado de cumplirse. Este sería el caso en la situación anterior: cuando intentamos obtener $P_1 \sim_g P_3$ después de lograr $P_1 \sim_g P_2$, debemos tener en cuenta que ahora P_1 puede utilizar la nueva transición $P_1 \rightsquigarrow a\,;b$, y entonces, para lograr que P_3 la simule, necesitamos extender nuevamente la definición de transición generalizada. De hecho, P_3 podrá simular esa transición si puede realizar la transición interna $\tau\,;b+\tau\,;c \xrightarrow{\tau} b$ sin ejecutar la acción visible anterior, de modo que obtengamos $P_3 \rightsquigarrow a\,;b$. Es decir, debemos permitir que P_3 ejecute una transición interna en cualquier lugar, y no sólo por debajo de movimientos internos.

De este modo, podemos generalizar la regla (II) que, recordemos, permitía la ejecución avanzada de un movimiento interno después de una secuencia de transiciones internas, obteniendo:

(VI) *Ejecución avanzada de un movimiento interno* (en cualquier lugar):

$$P_\beta \rightsquigarrow P'_\beta \implies \sum_\alpha \alpha\,;P_\alpha + \beta\,;P_\beta \rightsquigarrow \sum_\alpha \alpha\,;P_\alpha + \beta\,;P'_\beta$$

Con el objetivo de evitar la necesidad de aplicar este tipo de *backtracking* después del refinamiento de la noción de transición generalizada, se podría investigar un marco alternativo: la definición de nociones asimétricas de bisimulación global donde los movimientos aplicables para el proceso que *hace el primer movimiento* no son los mismos que los que puede utilizar el proceso que *se está defendiendo* y debe simular el movimiento. Es decir, consideraríamos distintas reglas para atacar y para defender. Sin embargo, esta noción asimétrica de bisimu-

lación es más difícil de entender y además no está claro si el mantenimiento del nombre de bisimulación estaría justificado después de tanto cambio.

Finalmente, se puede comprobar que también es posible considerar el caso en el que tenemos $P_2 \sim_g P_3$ sin tener $P_1 \sim_g P_2$ y $P_1 \sim_g P_3$. En tal caso, las elecciones internas podrían moverse hacia arriba y hacia abajo, pero las elecciones externas entre acciones comunes no podrían reducirse a elecciones internas puras. De modo que para obtener únicamente esta equivalencia necesitaríamos la regla (V), para que P_2 tenga la posibilidad de simular la transición $P_3 \xrightarrow{a} \tau\,;B+\tau\,;c$, y la regla (VI) para que P_3 pueda simular la transición $P_2 \xrightarrow{\tau} a\,;b$. Si eliminamos la regla (IV), obtenemos $P_1 \not\sim_g P_2$. Por otro lado, al disponer de la regla (VI), P_3 puede ejecutar la transición $P_3 \rightsquigarrow a\,;b$, que sin la regla (IV) no es simulable por P_1.

2.2.4. Elecciones mixtas

En esta sección estudiaremos los problemas provocados por las elecciones mixtas, es decir, aquellas que se producen entre acciones internas y acciones externas. En el marco de la bisimulación débil, estas elecciones mixtas no se pueden reducir a ningún otro tipo de no determinismo. Si consideramos que esta situación no es la más idónea, podemos elegir entre dos posibles reducciones. Consideremos el proceso $Q_1 = a\,;A+\tau\,;B$; lo podemos reducir al proceso $Q_2 = (\tau\,;a\,;A)+(\tau\,;B)$ o al proceso $Q_3 = (\tau\,;B)+\tau\,;(a\,;A+B)$ (véase figura 2.4). La primera opción es la seleccionada en [EJF96] pero, como ya hemos comentado, consideramos que esta decisión no es la más razonable. Nosotros somos partidarios de la segunda opción, que se corresponde con la reducción definida bajo semántica de fallos [Hoa85] y bajo la semántica de *testing must* [Hen88]. Esta equivalencia quedaría reflejada en la siguiente regla algebraica, suponiendo que P ofrece acciones visibles:

$$\bullet\ P+\tau\,;Q \sim_g \tau\,;(P+Q)+\tau\,;Q$$

Entonces, si queremos que $Q_1 = a\,;A+\tau\,;B$ y $Q_3 = (\tau\,;B)+\tau\,;(a\,;A+B)$ sean bisimilares globales debemos añadir nuevas reglas para la generación de transiciones generalizadas. Para simular el movimiento $Q_3 \rightsquigarrow a\,;A+B$, el proceso Q_1 debería permitir la *congelación* de su acción interna para obtener $Q_1 \rightsquigarrow a\,;A+B$. Una vez más, necesitamos un tipo de transición distinto a los que han aparecido hasta ahora. Este hecho refleja la independencia de esta propiedad con respecto a las distintas nociones de equivalencia discutidas hasta este momento. Los nuevos tipos de transiciones son los definidos por la siguiente regla:

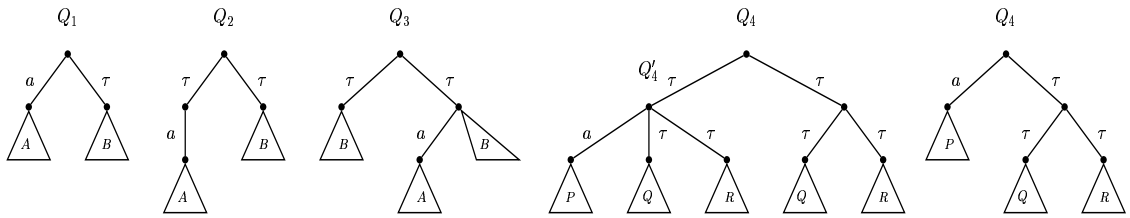


Figura 2.4: Equivalencias con elecciones mixtas.

(VII) *Congelación de transiciones internas*:

$$\sum_{\alpha} \alpha\,;P_\alpha + \tau\,;P' \rightsquigarrow \sum_{\alpha} \alpha\,;P_\alpha + P'$$

Desgraciadamente, una vez alcanzado este punto en nuestro desarrollo, podemos comprobar que ha ocurrido algo extraño, e indeseable, al incluir esta regla. Tenemos que los procesos $Q_4 = \tau\,;(\tau\,;Q + \tau\,;R) + \tau\,;(a\,;P + \tau\,;Q + \tau\,;R)$ y $Q_5 = a\,;P + \tau\,;(\tau\,;Q + \tau\,;R)$ de la figura 2.4 deberían ser bisimilares; sin embargo, vamos a comprobar que esto no es cierto. El proceso Q_4 puede ejecutar la transición estática: $Q_4 \rightsquigarrow Q'_4 = (a\,;P) + (\tau\,;Q) + (\tau\,;R)$ y para simularlo Q_5 debería ejecutar la transición $Q_5 \rightsquigarrow Q_5$. Confiamos en que se cumpla

$$(a\,;P) + (\tau\,;Q) + (\tau\,;R) \sim_g (a\,;P) + \tau\,;(\tau\,;Q + \tau\,;R)$$

pero cuando intentamos comprobar si $Q'_4 = (a\,;P) + (\tau\,;Q) + (\tau\,;R)$ es bisimilar global a $Q_5 = a\,;P + \tau\,;(\tau\,;Q + \tau\,;R)$, tenemos que para simular el movimiento $Q_5 \rightsquigarrow a\,;P + \tau\,;b$, tras aplicar la regla (VI), en el proceso Q'_4 debería estar permitido podar su última rama para obtener $Q'_4 \rightsquigarrow a\,;P + \tau\,;Q$.

Es interesante destacar, una vez más, que el problema aparece cuando un proceso *ataca* ejecutando una transición generalizada que se incluye para incrementar el poder de los procesos para *defenderse* de algún tipo de *ataque demasiado poderoso*. En este caso se trata de la situación que se produce cuando Q'_4 ataca utilizando la regla (VI). Más técnicamente, lo que ocurre es que si cambiamos la definición de transición generalizada no debemos comprobar únicamente que todas las propiedades anteriores se conservan, sino que también la noción de bisimulación global inducida continúa siendo una congruencia. Ello es lo que ocurre exactamente en esta situación: tenemos que las elecciones mixtas no preservarían la bisimulación global, a menos que la noción de transición generalizada no se refine de nuevo. Ello es posible hacerlo por medio de la siguiente regla

(VIII) *Eliminación de ramas* (junto a una acción interna τ):

$$\sum_{\alpha} \alpha\,;P_\alpha + \tau\,;P' + \sum_{\beta} \beta\,;P_\beta \rightsquigarrow \sum_{\alpha} \alpha\,;P_\alpha + \tau\,;P'$$

Se podría pensar que para resolver el problema bastaría con permitir la eliminación de ramas prefijadas por acciones internas. Sin embargo, en tal caso podría llegar a producirse el problema opuesto. Consideremos el proceso $P = a + b + \tau\,;c$. Tenemos que P es débilmente bisimilar a $Q = \tau\,;(a+b+\tau\,;c)+\tau\,;c+a$. Si consideramos el movimiento $Q \rightsquigarrow \tau\,;c+a$, para simularlo P necesita eliminar la acción b que ofrece, lo que está permitido por la última regla introducida para generar transiciones generalizadas. En otro caso, la bisimulación global no sería más débil que la bisimulación débil en contra de nuestras expectativas.

Pero todavía quedan más problemas. En primer lugar, consideremos los siguientes procesos:

$$P = (\tau\,;a) + \tau\,;(a+b+c) \text{ y } Q = (\tau\,;a) + \tau\,;(a+b) + \tau\,;(a+b+c)$$

Tenemos que P no es capaz de simular el movimiento $Q \rightsquigarrow \tau\,;(a+b) + \tau\,;(a+b+c)$; sin embargo, P podría simular el movimiento $Q \rightsquigarrow a+b$ del siguiente modo:

$$P = (\tau\,;a) + \tau\,;(a+b+c) \rightsquigarrow (\tau\,;a) + (a+b+c) \rightsquigarrow a+a+b \rightsquigarrow a+b$$

donde en el primer paso se aplica la regla (VII), en el segundo la regla (VIII) y en el tercero la regla (IV). Además, P también puede simular el movimiento $Q \rightsquigarrow a+b+c$ a través de la ejecución de una transición interna clásica: $P = (\tau\,;a) + \tau\,;(a+b+c) \rightsquigarrow a+b+c$. Como consecuencia, si tomamos $P' = \tau\,;P + \tau\,;P$, tenemos que

$$P' \rightsquigarrow \tau\,;(a+b) + \tau\,;(\tau\,;a + \tau\,;(a+b+c))$$

y entonces sería capaz de simular cualquier paso de Q. De este modo, concluimos que $P' \sim_g Q$ y, por lo tanto, ¡$P \not\sim_g P'$! Ello significa que, bajo la correspondiente noción de bisimulación global, la elección interna no sería ni idempotente, ni asociativa. Afortunadamente, la solución al problema es simple a la vez que natural: basta añadir la siguiente regla para la generación de transiciones internas generalizadas:

(IX) *Replicación de no determinismo*:

$$P \rightsquigarrow \tau\,;P + \tau\,;P$$

Nuestro último problema tiene que ver con nuestra intención de reducir cualquier tipo de no determinismo a otro diferente. Si consideramos los procesos P_1 y P_3 de la figura 2.3, tenemos que P_3 se puede convertir en $P_3' = a;(b+c)$, congelando sus dos transiciones internas. Sin embargo, P_1 no podría simular ese movimiento puesto que no hay ninguna transición generalizada capaz de unir dos transiciones visibles etiquetadas con una a dando lugar a una sola. Así, si queremos mantener esa equivalencia, necesitamos introducir una nueva regla para generar transiciones internas generalizadas.

(X) *Unión de transiciones visibles*:

$$\sum_{\alpha} \alpha\,;P_\alpha + a\,;P_1 + a\,;P_2 \rightsquigarrow \sum_{\alpha} \alpha\,;P_\alpha + a\,;(\tau\,;P_1 + \tau\,;P_2)$$

Debemos prestar atención al hecho de que los últimos problemas que han aparecido, así las últimas reglas añadidas desaparecerían si no considerásemos las elecciones mixtas como un tipo de no determinismo a considerar en el presente marco. Ello es así debido a que no habríamos tenido que introducir la *regla de congelación*, que es la génesis de todos estos problemas. Sin embargo, también es cierto que la idempotencia de la elección interna es una propiedad deseable en cualquier marco, y la introducción de la regla de replicación sería una forma natural de garantizarlo. Como consecuencia se tiene el siguiente resultado.

Lema 2.7 Si bajo la definición de transición generalizada tenemos $P \approx\!> Q$ y $Q \approx\!> P$, entonces bajo la noción inducida de bisimulación global tenemos $P \sim_g Q$.

Parece, sin embargo, que la regla (VIII) podría ser más fuerte de lo necesario. Por ejemplo, como consecuencia de su aplicación obtenemos

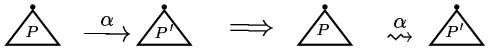
$$P + Q + \tau\,;R \sim_g \tau(\tau\,;R + P) + Q$$

que no es derivable a partir de las propiedades que nos interesan para nuestras nociones de bisimulación global.

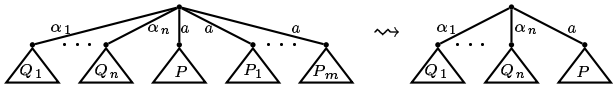
En la siguiente sección nos concentraremos principalmente en la noción de bisimulación que obtenemos juntando todas las reglas introducidas hasta este momento, tanto para transiciones estáticas como para transiciones dinámicas.

2.3. Relación entre bisimulación global y bisimulación débil

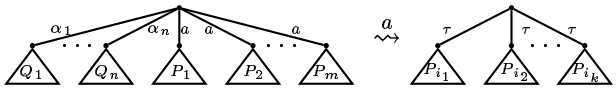
En esta sección consideraremos la noción de bisimulación global inducida por las nociones de transiciones generalizadas definidas por las reglas ilustradas en la figura 2.5. Obsérvese que



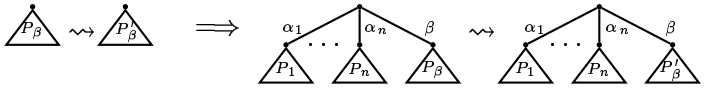
Regla (I) Preservando la relación de transición de P



Regla (IV) Reducción de no determinismo

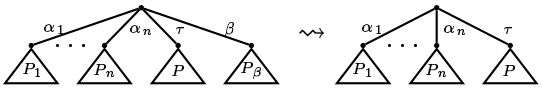


Regla (V) Ejecución simultánea de varios ejemplares de una acción

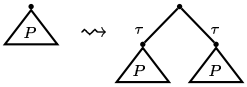


Regla (VI) Ejecución avanzada de un paso

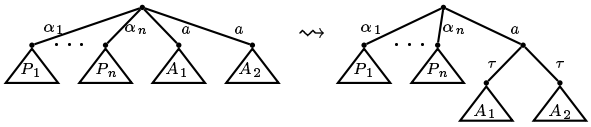
Regla (VII) Congelación de transiciones internas



Regla (VIII) Eliminación de ramas (junto a una τ)



Regla (IX) Replicación de no determinismo



Regla (X) Unión de acciones visibles

Figura 2.5: Transiciones estáticas y dinámicas

no incluimos las reglas que permiten la ejecución avanzada de una acción interna (regla (II)) o de varias acciones internas (regla (III)), puesto que se pueden obtener como casos particulares de las reglas (VI) y (VII), respectivamente.

Desgraciadamente, la técnica más simple para probar que dos procesos son bisimilares, consiste en la mera aplicación de la definición de esta relación, o sea por medio de la construcción de una bisimulación que contenga el par correspondiente, resulta extremadamente complicada cuando se quiere aplicar para probar si dos procesos son bisimilares globales. El motivo es el incremento de la complejidad de las reglas que definen los movimientos permitidos cuando se juega al juego de la bisimulación. Este hecho es especialmente grave para la regla de replicación, debido a que, cuando la aplicamos, el tamaño del sistema de transiciones se incrementa en vez de reducirse.

Por el contrario, si considerásemos el no determinismo de la elecciones mixtas como un tipo de elección primitiva, no tendríamos que introducir la regla de congelación y, por tanto, no sería necesario introducir la regla de replicación, con lo que la aplicación de la técnica habitual para probar la bisimilitud sería mucho más simple.

A continuación probaremos que la bisimulación débil es realmente más débil que la bisimulación global definida según estas reglas. Para ello son necesarias algunas definiciones y resultados auxiliares previos.

Definición 2.8 Sea P un sistema etiquetado de transiciones. Decimos que P es *estable* si $P \stackrel{\tau}{\not\longrightarrow}$, es decir, $\nexists P'$ tal que $P \stackrel{\tau}{\longrightarrow} P'$.

Decimos que P es *determinista en cabeza* si para cualquier P' tal que $P \Longrightarrow P'$ tenemos que $Init^*(P') = Init^*(P)$, donde $Init^*(Q) = \{a \in \mathtt{Act} \mid Q \stackrel{a}{\Longrightarrow}\}$.

Definimos el sistema *congelado en cabeza* de P, denotado por $hf(P)$, como el sistema inducido por la equivalencia $hf(P) \stackrel{a}{\longrightarrow} P''$ sii $P \Longrightarrow P' \stackrel{a}{\longrightarrow} P''$. □

Proposición 2.9 Si P es determinista en cabeza entonces $P \sim_g hf(P)$.
Demostración: Por el lema 2.7 es suficiente que probemos $P \approx\!\!> hf(P)$ y $hf(P) \approx\!\!> P$. Obviamente tenemos $P \approx\!\!> hf(P)$. Para probar la existencia de la otra transición escribamos P en de la forma $P = \sum_i a_i\,;P_i + \sum_j \tau\,;P'_j$. Es fácil comprobar que $hf(P) = \sum_i a_i\,;P_i + \sum_j hf(P'_j)$. Por lo tanto tenemos

$$\begin{aligned} hf(P) &\rightsquigarrow \text{(Regla (IX))}\ \tau\,;hf(P) + \tau\,;hf(P) \rightsquigarrow \text{(Regla (VII))}\ hf(P) + \tau\,;hf(P) \\ &\rightsquigarrow \text{(Regla (VIII))}\ \textstyle\sum_i a_i\,;P_i + \tau\,;hf(P) \rightsquigarrow \text{(Regla (VIII))}\ \sum_i a_i\,;P_i + \tau\,;\sum_j hf(P'_j) \end{aligned}$$

Aplicando la regla (IV), para cada k obtenemos que $\sum_j hf(P'_j) \rightsquigarrow hf(P'_k)$ y aplicando entonces la regla (IX) obtenemos $\sum_j hf(P'_j) \rightsquigarrow \sum_j \tau\,;hf(P'_j)$, para concluir, aplicando la

regla (VII), $hf(P) \approx\!\!> \sum_i a_i\,;P_i + \sum_j \tau\,;hf(P'_j)$. De aquí, aplicando hipótesis de inducción a las componentes de la forma $hf(P'_j)$, se deduce $hf(P) \approx\!\!> \sum_i a_i\,;P_i + \sum_j \tau\,;P'_j = P$ □

Como corolario del resultado anterior tenemos que un proceso cualquiera es equivalente a su proceso congelado en cabeza.

Corolario 2.10 Sean P y P' dos sistemas etiquetados de transiciones. Si P es estable y $P \sim_g P'$ entonces $P' \sim_g hf(P')$.

Definición 2.11 Sean P y Q dos sistemas etiquetados de transiciones. Decimos que P y Q son *idénticamente estables* si ambos son estables o ambos no lo son. □

Esta noción de estabilidad nos permite sustituir procesos equivalentes en el ámbito del operador de elección

Proposición 2.12 Si P, Q son dos sistemas etiquetados de transiciones idénticamente estables tales que $P \sim_g Q$, entonces, para todo sistema de transiciones R tenemos $R+P \sim_g R+Q$.
Demostración: En este caso necesitamos proporcionar una bisimulación global que contenga cualquier par de ese tipo junto con los sistemas que se obtienen de la aplicación en sus componentes de las reglas (VII), (IX) y (X). Necesitamos que P y Q sean idénticamente estables para poder eliminar cualquier rama de R aplicando la regla (VIII). □

Para extender este resultado a la consideración de la expansión de cualquier estado perteneciente a P tenemos el siguiente resultado

Proposición 2.13 Sean P un sistema de transiciones en forma de árbol y L un estado de una hoja del árbol. Sea $P(L \longleftarrow Q)$ el sistema que se obtiene expandiendo el sistema en L a través del sistema Q. Entonces $Q_1 \sim_g Q_2$ implica $P(L \longleftarrow Q_1) \sim_g P(L \longleftarrow Q_2)$.
Demostración: Para probar este resultado es suficiente probar que los conjuntos de pares de la forma $\langle P(L_i \longleftarrow Q_{i,1}), P(L_i \longleftarrow Q_{i,2})\rangle$ forman una bisimulación global, donde $\{L_i\}$ es cualquier conjunto de hojas de P y para cada $i \in I, Q_{i,1} \sim_g Q_{i,2}$. □

Corolario 2.14 Si para cualquier $\alpha \in \Lambda \subseteq \mathtt{Act}_\tau$ tenemos $P_\alpha \sim_g Q_\alpha$ entonces se cumple $\sum_{\alpha\in\Lambda} \alpha\,;P_\alpha \sim_g \sum_{\alpha\in\Lambda} \alpha\,;Q_\alpha$.

Definición 2.15 Dados dos sistemas de transiciones P, Q y $\alpha \in \mathtt{Act}_\tau$, decimos que $\alpha\,;Q$ es una *parte de* P, denotado por $\alpha\,;Q \precsim P$, si existe algún P' tal que $P \stackrel{\alpha}{\Longrightarrow} P'$ y tenemos $P' \sim_w Q$ y $P' \sim_g Q$. □

Proposición 2.16 Sea P un sistema de transiciones, $\{P_i \mid i \in I\}$ un conjunto de sistemas de transiciones y $\alpha_i \in \mathtt{Act}_\tau, i \in I$. Si para cualquier $i \in I$ tenemos $\alpha_i \,;\, Q_i \precsim P$ entonces se cumple $\tau \,;\, P + \sum \alpha_i \,;\, Q_i \sim_w P$ y $\tau \,;\, P + \sum \alpha_i \,;\, Q_i \sim_g P$.

Demostración: El resultado es conocido para la bisimulación débil. Para la bisimulación global tenemos $\tau \,;\, P + \sum \alpha_i \,;\, Q_i \approx\!> P$. Para obtener la transición inversa tenemos que considerar $P \xrightarrow{\tau^k} P_i' \xrightarrow{a} P_i'' \xrightarrow{\tau^l} P_i'''$ tales que $P_i''' \sim_g Q_i$. A partir de aquí deducimos $P \rightsquigarrow P_i'$. Como consecuencia obtenemos

$$\begin{aligned} P \;&\rightsquigarrow \text{ (Regla (IX))} \textstyle\sum_{i\in I\cup\{i_o\}} \tau \,;\, P \rightsquigarrow \text{ (Reglas (V) y (VII))} \; \tau \,;\, P + \textstyle\sum_{i\in I} \tau \,;\, P_i' \\ &\rightsquigarrow \text{ (Regla (VII))} \; \tau \,;\, P + \textstyle\sum_{i\in I} P_i' \rightsquigarrow \text{ (Regla (V))} \; \tau \,;\, P + \textstyle\sum_{i\in I} a \,;\, P_i'' \\ &\rightsquigarrow \text{ (Reglas (V) y (VII))} \; \tau \,;\, P + \textstyle\sum_{i\in I} a \,;\, P_i''' \end{aligned}$$

y, aplicando el corolario 2.14, tenemos $\tau \,;\, P + \sum_{i\in I} a \,;\, P_i''' \sim_g \tau \,;\, P + \sum_{i\in I} a \,;\, Q_i$. Finalmente, podemos concluir $\tau \,;\, P + \sum \alpha_i \,;\, Q_i \sim_g P$. □

Teorema 2.17 Sean P, Q dos sistemas de transiciones. Entonces $P \sim_w Q$ implica $P \sim_g Q$.

Demostración: Razonaremos por inducción sobre el tamaño máximo de P y Q. Si el tamaño máximo de P y Q es 1 entonces $P = Q = \mathit{stop}$. En cualquier otro caso, sea $P = \sum_{i\in I} a_i \,;\, P_i + \sum_{j\in J} \tau \,;\, P_j'$. Si existe $j \in J$ tal que $P_j' \sim_w P$ aplicando la hipótesis de inducción deducimos que $P_j' \sim_g P$. En caso contrario, para cada $j \in J$ estudiamos si existe algún $i \in I$ tal que $a_i \,;\, P_i \precsim P_j'$ o bien algún $j' \in J - \{j\}$ tal que $\tau \,;\, P_{j'}' \precsim P_{j'}'$, y si se diera uno u otro caso eliminaríamos esas ramas aplicando la proposición 2.16.

Finalmente, estudiaremos si existen $i, i' \in I, i \neq i'$ tales que $P_i \sim_w P_i'$ y, en tal caso, aplicando el corolario 2.14 y la regla (III) eliminaríamos uno de ellos.

Una vez aplicadas todas estas simplificaciones, obtenemos una *forma simplificada* de P, que denotaremos por $\mathit{simp}(P)$, que verifica $P \sim_w \mathit{simp}(P), P \sim_g \mathit{simp}(P)$ y el tamaño máximo de P es mayor que el tamaño del proceso simplificado.

Podemos repetir el proceso con Q obteniendo $\mathit{simp}(Q)$ que verifica $Q \sim_w \mathit{simp}(Q)$, $Q \sim_g \mathit{simp}(Q)$ y cuyo tamaño máximo es menor que el del proceso de partida. A partir de este momento podemos suponer, sin perdida de generalidad, que P y Q son sistemas simplificados. Entonces, para obtener

$$P = \sum_{i\in I} a_i \,;\, P_i + \sum_{j\in J} \tau \,;\, P_j' \sim_w \sum_{i\in I'} a_i \,;\, Q_i + \sum_{j\in J'} \tau \,;\, Q_j' = Q$$

dado que P y Q están simplificados, debemos tener $I = I'$ y $|J| = |J'|$ y dos funciones biyectivas $v : I \to I$ y $w : J \to J'$ tales que $P_i \sim_w Q_{v(i)}$ para todo $i \in I$ y $P_j' \sim_w Q_{w(j)}$ para todo $j \in J$. En tal caso podemos aplicar la hipótesis de inducción para concluir que para todo $i \in I$ se cumple $P_i \sim_g Q_{v(i)}$ y para todo $j \in J$ se cumple $P_j' \sim_g Q_{w(j)}$. Finalmente, aplicando el corolario 2.14, obtenemos $P \sim_g Q$. □

2.4. Bisimulación global temporal

En este capítulo extenderemos el estudio de la bisimulación global al marco de los procesos temporales. Es un trabajo interesante por distintos motivos. En primer lugar, a través de esta extensión del modelo queda patente la flexibilidad de nuestras ideas que pueden aplicarse a distintos tipos de álgebras de procesos. Segundo, dado que uno de nuestros objetivos es estudiar y relacionar los distintos tipos de no determinismo, los procesos temporales son un buen banco de pruebas puesto que las transiciones temporales son una fuente de comportamiento no determinista. Una última razón para realizar este estudio se debe a que las diferentes alternativas para definir una semántica a través de relaciones de equivalencia, más débiles que la bisimulación débil, no funcionan como se esperaba en el campo temporal. Por ejemplo, la semántica de *testing* es mucho más fuerte que en el caso no temporal debido a que la urgencia de las acciones internas incrementa el poder discriminatorio de las pruebas [LdFN96]. De hecho, bajo ciertas hipótesis, las adaptaciones de las semánticas de *testing may* y *must* coinciden en el caso de procesos temporizados [LdF99]. Por lo tanto, la definición de semánticas alternativas resulta especialmente interesante en el caso de los procesos temporizados.

2.4.1. Conceptos básicos

Al igual que hicimos a la hora de estudiar los procesos no temporales, en esta sección presentaremos la sintaxis abstracta que vamos a utilizar para modelar sistemas temporales. Para esta extensión, los procesos se definirán en términos de *sistemas etiquetados de transiciones temporales*. Estos sistemas difieren de los utilizados en las secciones anteriores únicamente por el hecho de que las etiquetas de las transiciones también pueden ser unidades de tiempo. Para modelar el tiempo utilizaremos un *conjunto discreto de tiempos* $\mathcal{T}$ al que denominamos a como el *dominio de tiempos*. Más concreto, trabajaremos con $\mathcal{T} = \{\delta, 2\delta, 3\delta, \ldots\}$, donde δ indica la *mínima cantidad* de tiempo que se considera. En consecuencia, el conjunto $\mathcal{T}$ sería isomorfo a $\mathbb{N}$.

En la definición de nuestros sistemas etiquetados de transiciones temporales se tendrán en cuenta algunas de las propiedades que comúnmente se encuentran en los modelos temporales, como la de ser *deterministas en tiempo* (es decir, que un proceso no puede evolucionar a diferentes procesos tras pasar una misma cantidad de tiempo, sin hacer nada más mientras tanto) y *progreso máximo* para las acciones internas (es decir, que un proceso no puede estar parado dejando pasar el tiempo si puede ejecutar una acción interna τ).

Definición 2.18 Definimos un proceso temporal P por medio de una tupla $(S, \longrightarrow, s_0)$, donde S es un *conjunto de estados*, $s_0 \in S$ es el *estado inicial* y $\longrightarrow \subseteq S \times (\mathtt{Act}_\tau \cup \mathcal{T}) \times S$ es la *relación de transición*. Normalmente escribiremos $s \xrightarrow{\alpha} s'$ en lugar de $(s, \alpha, s') \in \longrightarrow$. Impondremos las siguientes restricciones sobre la relación de transiciones:

- **Progreso máximo**: Sea $s \in S$. Si existe $s' \in S$ tal que $s \xrightarrow{\tau} s'$, entonces no existe $s'' \in S$ y $t \in \mathcal{T}$ tales que $s \xrightarrow{t} s''$.
- **Determinista en tiempo**: Sea $s \in S$. Para cada $t \in \mathcal{T}$ existe a lo sumo un estado $s' \in S$ tal que $s \xrightarrow{t} s'$.

□

En la extensión temporal, también identificaremos los nodos de estos sistemas etiquetados de transiciones con los procesos definidos tomando cada uno como estado inicial. Formalmente, si $P = (S, \longrightarrow, s_0)$ es un proceso, y $s \in S$, consideramos el proceso P_s definido como el proceso $(S, \longrightarrow, s)$. Como ya hicimos en el caso de los procesos no temporizados consideraremos únicamente sistemas de transiciones que generan un árbol.

Una vez definida la notación para describir sistemas temporales, explicaremos a través de una serie de ejemplos sencillos el significado de cada una de las construcciones y el poder expresivo del modelo.

En primer lugar, para modelar *time-outs* se puede utilizar el siguiente mecanismo: en el estado inicial ofrecemos las acciones que se pueden ejecutar junto con una transición temporal, cuyo parámetro indica el tiempo durante el cuál dichas acciones están disponibles. Por ejemplo, si queremos especificar un sistema que ofrezca las acciones a y b exactamente en el tiempo 0, y de modo que si ninguna de estas acciones se ejecuta, el proceso entrará en un punto muerto, el correspondiente proceso tendría la forma representada en la figura 2.6 (a).

Los *retardos* se especificarán a través de transiciones temporales. Por ejemplo, si consideramos una variante del ejemplo anterior, en la que las acciones se ofrecen tras transcurrir un determinado tiempo δ, el sistema se representaría como el proceso descrito en la figura 2.6 (b). Para denotar estos retardos utilizaremos la notación $\mathtt{delay}(t)\,;P$ significando que el proceso P empezará a ejecutarse tras t unidades de tiempo. También podemos especificar procesos que contengan distintos retardos. Por ejemplo, una elección entre la acción a retardada 2δ unidades de tiempo y la acción b retardada 4δ unidades de tiempo quedaría descrita por el proceso de la figura 2.6 (c). Obsérvese que la representación anterior es coherente con la hipótesis del paso determinista del tiempo.

Nótese que en nuestra representación consideramos únicamente las transiciones temporales correspondientes al tiempo que transcurra entre dos ofrecimientos de acciones que vienen

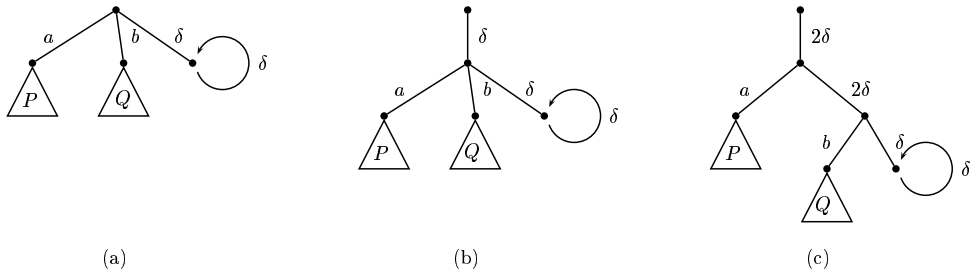


Figura 2.6: Ejemplos de procesos temporales (1/3).

representados por estados. Obviamente, necesitamos permitir a todo proceso con un cierto retardo el *consumo* parcial del mismo, lo que suele expresarse como el *envejecimiento* del proceso. Por ejemplo, el proceso podrá $\texttt{delay}(t)\,;P$ evolucionar a través de una transición temporal con etiqueta de tiempo t' al proceso $\texttt{delay}(t-t')\,;P$ para cualquier $t' < t$. Formalizaremos este concepto más tarde. Otra posibilidad, que simplificaría conceptualmente el modelo, sería considerar que el tiempo que etiqueta las transiciones temporales puede ser únicamente δ. Sin embargo, esta decisión podría influir dramáticamente en el tamaño de los sistemas, dado que tanto el número de transiciones como de estados se incrementaría dramáticamente.

También se pueden especificar sistemas que ofrezcan acciones durante un cierto intervalo de tiempo. Por ejemplo, si el sistema ofrece a en el intervalo $[0..2\delta]$ y b en el intervalo $[\delta..3\delta]$ para representar este sistema utilizaríamos el proceso de la figura 2.7 (a). Nótese que las acciones τ son urgentes, por eso no tiene sentido ofrecer una τ en un intervalo de tiempo. Por otra parte, podemos especificar un proceso que ofrezca una acción durante una cantidad infinita de tiempo. Por ejemplo, el proceso de la figura 2.7 (b) ofrece la acción a en el intervalo $[0..\infty)$, la acción b en $[\delta..\infty)$ y la acción c en $[\delta..2\delta]$. Además, podemos especificar procesos que *bloqueen* el paso del tiempo. Por ejemplo, en la figura 2.8 (a) se muestra un proceso que ofrece las acciones a y b en tiempo 0 y no permite el paso del tiempo.

Finalmente, hay que resaltar que si se usase una sintaxis concreta existirían diversos procesos sintácticos que se comportarían de la misma manera. Por ejemplo, utilizando una sintaxis basada en CCS, los siguientes procesos darían lugar al sistema de transiciones de la figura 2.8 (b):

$$P = (a[0..\delta]\,;P_1) + (\texttt{delay}(\delta)\,;\tau\,;P_2) \qquad Q = (a[0..2\delta]\,;P_1) + (\texttt{delay}(\delta)\,;\tau\,;P_2)$$

$$R = P + (b[2\delta..3\delta]\,;P_3)$$

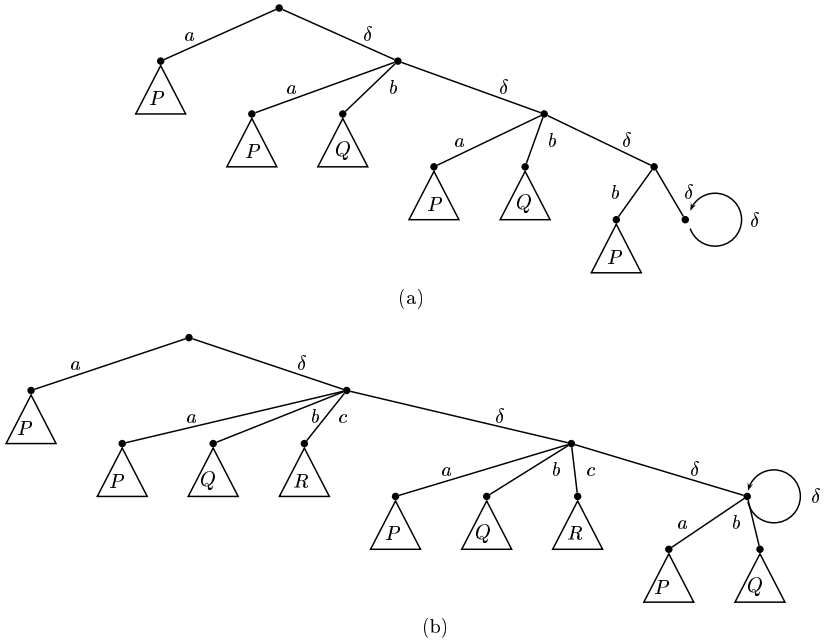


Figura 2.7: Ejemplos de procesos temporales (2/3).

Además, existen procesos con sistemas de transiciones diferentes que sin embargo deberían verse como equivalentes bajo cualquier definición razonable de equivalencia. Por ejemplo, los procesos de la figura 2.8 (c) y (d).

A continuación, para poder manejar adecuadamente los procesos temporales, introduciremos una notación auxiliar análoga a la utilizada en el caso de los procesos no temporizados, pero añadiendo nuevas formas para referirnos a los procesos que pueden evolucionar por medio de transiciones temporales.

Definición 2.19 Sea $P = (S, \longrightarrow, s_0)$ un proceso temporal. Sea $\{s_{a_i} \in S \mid s_0 \xrightarrow{a} s_{a_i}\}$ un conjunto de estados alcanzables tras ejecutar la transición $\xrightarrow{a}$ desde s_0, y sea $\{P_{a_i}\}$ correspondiente el conjunto de los procesos inducidos. Del mismo modo, sean$\{s_{\tau_i} \in S | s_0 \xrightarrow{\tau} s_{\tau_i}\}$ y $\{P_{\tau_i}\}$ el conjunto de estados y de procesos, respectivamente, alcanzables después de la ejecución de la transición $\xrightarrow{\tau}$ desde s_0. Para referirnos a P utilizaremos la siguiente notación:

- $P = \sum_{a \in \mathtt{Act}} a\,;P_{a_i} + \sum \tau\,;P_{\tau_i}$ si $\{P_{\tau_i}\} \neq \emptyset$
- $P = \sum_{a \in \mathtt{Act}} a\,;P_{a_i} + \mathtt{delay}(t)\,;P'$ si $\{P_{\tau_i}\} = \emptyset \wedge \exists P' :\ P \xrightarrow{t} P'$

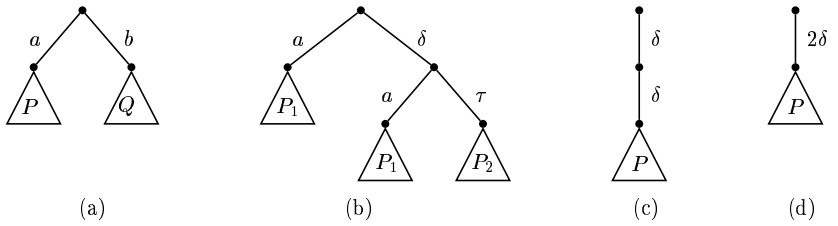


Figura 2.8: Ejemplos de procesos temporales (3/3).

- $P = \sum_{a \in \mathtt{Act}} a\,;P_{a_i}$ si $\{P_{\tau_i}\} = \emptyset \wedge \nexists P' :\ P \xrightarrow{t} P'$
- $P = stop$ si $\forall a \in \mathtt{Act} :\ \{P_{a_i}\} = \emptyset \ \wedge\ \{P_{\tau_i}\} = \emptyset \ \wedge\ s_0 \xrightarrow{\delta} s_0$
- $P = \mathtt{block}$ si $\forall a \in \mathtt{Act} :\ \{P_{a_i}\} = \emptyset$
  $\wedge\ \{P_{\tau_i}\} = \emptyset \wedge\ \nexists t \in \mathcal{T} :\ s_0 \xrightarrow{t} s_0$

□

Al igual que en el caso no temporal, existen sumas $\sum_{a \in \mathtt{Act}} a\,;P_{a_i}$ que pueden ser vacías. Además, utilizaremos la notación $P = \sum_{\alpha} \alpha\,;P_{\alpha}$ en aquellos casos en los que no necesitemos distinguir entre descendientes de transiciones visibles, internas o temporales. Finalmente, utilizaremos $P_1 + P_2 + \cdots + P_n$ para denotar $\sum P_i$.

En la definición previa hemos introducido un concepto nuevo que no aparecía en el caso no temporal: el proceso `block`. Este proceso es similar al proceso *stop* porque ninguno de los dos puede ejecutar acciones. La diferencia entre ellos es que mientras *stop* permite el paso del tiempo, `block` no lo permite. A continuación, añadiremos una extensión más a las ya habituales de las relaciones de transición para permitir varios pasos incluyendo también transiciones temporales.

Definición 2.20 Sea $P = (S, \longrightarrow, s_0)$ un proceso temporal y sean $s, s' \in S$. Escribimos $s \stackrel{t}{\Longrightarrow} s'$ si existen $t_1, t_2, \ldots, t_n \in \mathcal{T}$ tales que $\sum t_i = t$ y existen $s_1, s'_1, s_2, s'_2, \ldots, s_n, s'_n$ tales que

$$s \stackrel{\tau}{\Longrightarrow} s_1 \xrightarrow{t_1} s'_1 \stackrel{\tau}{\Longrightarrow} s_2 \xrightarrow{t_2} s'_2 \cdots \stackrel{\tau}{\Longrightarrow} s_n \xrightarrow{t_n} s'_n \stackrel{\tau}{\Longrightarrow} s'$$

donde $\stackrel{\tau}{\Longrightarrow}$ es la clausura reflexiva y transitiva de $\xrightarrow{\tau}$ (véase la definición 2.4). □

Como ya se ha comentado previamente, necesitaremos generar todas las transiciones temporales que correspondan a aquellas que estén explícitamente representadas en el sistema de transiciones dado. Para ello, extenderemos las relaciones de transición del siguiente modo: dado un proceso $P = \sum_{a \in \mathtt{Act}} a\,;P_{a_i} + \mathtt{delay}(t)\,;P'$, junto con las transiciones que aparezcan

en la relación de transición, consideraremos también que se pueden ejecutar las siguientes transiciones de tiempo:

$$P \xrightarrow{t'} \texttt{delay}(t - t')\,;P' \quad [\text{para } t' < t]$$

Finalizaremos esta sección definiendo la noción clásica de bisimulación temporal débil.

Definición 2.21 Una relación de equivalencia $\mathcal{R}$ sobre procesos se dice que es una *bisimulación temporal débil* si para todo par de procesos P y Q tales que $P\,\mathcal{R}\,Q$, y para toda acción $\alpha \in \texttt{Act}_\tau \cup \mathcal{T}$ tenemos:

- Si $P \xrightarrow{\alpha} P'$ entonces existe algún Q' tal que $Q \stackrel{\alpha}{\Longrightarrow} Q'$ y $P'\,\mathcal{R}\,Q'$.

Decimos que dos procesos P y Q son *bisimilares temporales débiles*, denotado por $P \sim_{wt} Q$, si existe una bisimulación temporal débil que contenga el par $\langle P, Q\rangle$. □

2.4.2. Una bisimulación global temporal

En esta sección definiremos una extensión temporal de las *bisimulaciones globales* explicadas en la sección 2.2. Con esta extensión se pretenden resolver los problemas discutidos en la motivación de este capítulo, pero para los procesos temporales. Nos quedaremos con algunas de las reglas que han aparecido anteriormente, siempre que nos interese lo que conllevan, y añadiremos algunas reglas nuevas de modo que la información temporal también aparezca reflejada en la nueva equivalencia. Denotaremos de nuevo con $\rightsquigarrow$ y $\stackrel{\alpha}{\rightsquigarrow}$, donde $\alpha \in \texttt{Act} \cup \mathcal{T}$, a las transiciones generalizadas *estáticas* y *dinámicas* que definimos a continuación.

Definición 2.22 Sea P un proceso temporal. Escribimos $P \rightsquigarrow P'$ (resp. $P \stackrel{\alpha}{\rightsquigarrow} P'$, con $\alpha \in \texttt{Act} \cup \mathcal{T}$) si la transición correspondiente se puede derivar a partir de las reglas siguientes:

1. *Preservando la relación de transición de P*:
   - Si $P \xrightarrow{\tau} P'$ entonces $P \rightsquigarrow P'$.
   - Si $P \xrightarrow{\alpha} P'$ para algún $\alpha \in \texttt{Act} \cup \mathcal{T}$ entonces $P \stackrel{\alpha}{\rightsquigarrow} P'$.
2. *Ejecución simultánea de varias transiciones internas*:

$$P = \sum_{\alpha} \alpha\,;P_{\alpha_i} + \sum_{i\in I} \tau\,;P_i \rightsquigarrow \sum_{i\in I} \tau\,;P_i$$

3. *Reducción del no determinismo* (generado por varias ofertas simultáneas de la misma acción visible):

$$P = \sum_{\alpha} \alpha\,;P_{\alpha_i} + a\,;P_a + \sum_{i\in I} a\,;P_i \rightsquigarrow \sum_{\alpha} \alpha\,;P_{\alpha_i} + a\,;P_a$$

4. *Ejecución avanzada de un movimiento interno*:

$$P_\beta \rightsquigarrow P'_\beta \implies P = \sum_{\alpha} \alpha\,;P_{\alpha_i} + \beta\,;P_\beta \rightsquigarrow \sum_{\alpha} \alpha\,;P_{\alpha_i} + \beta\,;P'_\beta$$

donde, o bien $\beta \in \mathtt{Act}_\tau$, o bien $\beta = \mathtt{delay}(t)$, para algún $t \in \mathcal{T}$.

5. *Ejecución simultánea de varios ejemplares de una acción ofrecida*:

$$\forall a \in \mathtt{Act}, \forall I \subseteq I_a \text{ con } I_a = \{\!|i \mid P \stackrel{a}{\Longrightarrow} P_i|\!\} \implies P = \sum_{\alpha} \alpha\,;P_\alpha + \sum_{i\in I} a\,;P_i \stackrel{a}{\rightsquigarrow} \sum_{i\in I} \tau\,;P_i$$

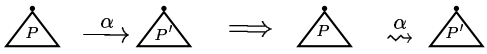
6. *Paso simultáneo de tiempo bajo τ's*:

$$\forall i \in I:\ P_i \stackrel{t}{\approx\!>} P'_i \implies P = \sum_{a\in\mathtt{Act}} a\,;P_{a_i} + \sum_{i\in I} \tau\,;P_i \stackrel{t}{\rightsquigarrow} \sum_{i\in I} \tau\,;P'_i$$

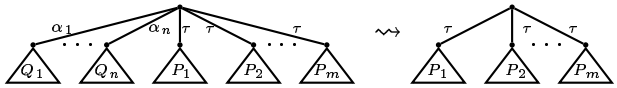
donde $\approx\!>$ representa la clausura reflexiva y transitiva de $\rightsquigarrow$, y tenemos que $P \stackrel{t}{\approx\!>} P'$ si $P \approx\!> \cdot \stackrel{t_1}{\rightsquigarrow} \cdot \approx\!> \stackrel{t_2}{\rightsquigarrow} \cdot \approx\!> \cdots \stackrel{t_n}{\rightsquigarrow} \cdots \approx\!> P'$ con $t = \sum t_i$. Finalmente, para cualquier $a \in \mathtt{Act}$, tenemos que $\stackrel{a}{\approx\!>}$ representa a $\approx\!> \cdot \stackrel{a}{\rightsquigarrow} \cdot \approx\!>$.

□

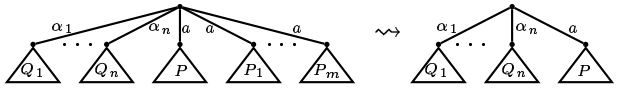
En la figura 2.9 están representadas gráficamente las reglas que hemos presentado en la definición anterior. Comentaremos brevemente el significado intuitivo de cada una de estas reglas en el ámbito de los procesos temporales. En las siguientes secciones veremos las razones por las que cada una de ellas ha sido incluida. La única regla nueva con respecto al caso no temporal es la sexta. Como ya vimos anteriormente, la regla 1 se incluye para mantener las transiciones normales de los procesos, y se corresponde con la regla (I) del caso no temporal. La regla 2 permite a los procesos evolucionar seleccionando un subconjunto de descendientes de transiciones de τ's. Recordemos que, en esta regla, algunas de las α's pueden ser también



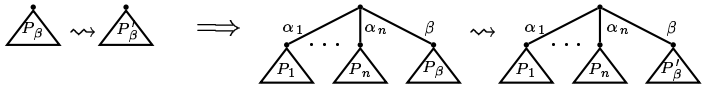
Regla 1 Preservando la relación de transición de P



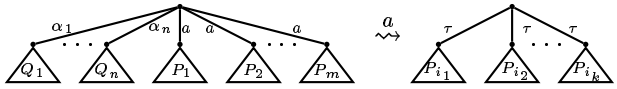
Regla 2 Ejecución simultánea de varias transiciones internas



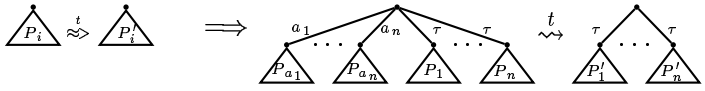
Regla 3 Reducción de no determinismo



Regla 4 Ejecución avanzada de un movimiento interno



Regla 5 Ejecución simultánea de varios ejemplares de a



Regla 6 Paso simultáneo de tiempo bajo τ's

Figura 2.9: Transiciones estáticas y dinámicas para procesos temporales

τ's. Esta regla se corresponde con la regla (III) del caso anterior. La regla 3 permite reducir el no determinismo cortando algunas (pero no todas) de las ramas etiquetadas con una misma acción y se corresponde con la regla (IV) de la versión no temporal. Como en la regla anterior, también algún α puede ser igual a la propia acción a. La regla 4, que se corresponde con la regla (VI) de la sección anterior, indica que el proceso puede evolucionar ejecutando transiciones estáticas no solamente en la raíz, sino también en cualquier otro punto. Se puede observar que, iterando la aplicación de esta regla, se puede ejecutar una transición estática no sólo después de un único paso, sino después de una secuencia de ellos. Además necesitamos dos reglas adicionales para la generación de transiciones dinámicas. La regla 5, que en la sección anterior tenía asignado también ese número, permite la ejecución de más de un único ejemplar de una acción. El proceso alcanzado es una elección no determinista entre las continuaciones de cada una de las acciones ejecutadas, de modo que todos los

estados alcanzados aparecen prefijados por una acción interna. Finalmente, la regla 6 permite envejecer a un proceso por debajo de una τ. De este modo, se retrasa la correspondiente elección interna. Nótese que en este caso se pierden el resto de ramificaciones, puesto que las mismas no han podido evolucionar temporalmente.

Por fin, introduciremos nuestra noción de *bisimulación global temporal* que sigue los mismos patrones que la bisimulación débil, pero reemplazando las transiciones habituales por las transiciones estáticas y dinámicas.

Definición 2.23 Una relación de equivalencia $\mathcal{R}$ definida sobre el conjunto de procesos es una *bisimulación global temporal* si para cada par de procesos P, Q tales que $P\ \mathcal{R}\ Q$, tenemos que para cualquier $\alpha \in \mathtt{Act} \cup \{\epsilon\} \cup \mathcal{T}$:

- Si $P \overset{\alpha}{\rightsquigarrow} P'$ entonces existe Q' tal que $Q \overset{\alpha}{\approx\!>} Q'$ y $P'\ \mathcal{R}\ Q'$.

Decimos que dos procesos P y Q son *bisimilares globales temporales*, y escribimos $P \sim_{gt} Q$, si existe una bisimulación global temporal que contenga el par $\langle P, Q\rangle$. □

La prueba de que la bisimulación débil temporal identifica menos procesos que la bisimulación global temporal es una adaptación de la dada en la sección dedicada al estudio de los procesos sin información temporal. Intuitivamente, el resultado se mantiene debido a que, en particular, las transiciones ordinarias se preservan en la definición de las transiciones estáticas y dinámicas.

Teorema 2.24 Sean P y P' dos procesos. Tenemos que $P \sim_{wt} P'$ implica $P \sim_{gt} P'$.

Pasamos a continuación a mostrar que nuestra noción de equivalencia cumple las propiedades esperadas. En primer lugar veremos como se conserva la asociatividad en el caso en el que se deben tomar *decisiones de tiempo* en diferentes lugares. Como ejemplo, consideraremos los procesos Q y Q' dados en la figura 2.10. A continuación, mostraremos que ambos procesos son equivalentes bajo el contexto de la bisimulación global temporal. Tenemos que cuando Q ejecuta la transición $Q \rightsquigarrow \mathtt{delay}(\delta)\,;(a+b)$, el proceso Q' puede simularlo ejecutando simultáneamente dos acciones internas sin envejecer, aplicando las reglas 2 y 4, obteniendo $Q' \rightsquigarrow \mathtt{delay}(\delta)\ ;(\tau\ ;a+\tau\ ;b)$. Por otro lado, para conseguir que Q simule la transición temporal $Q' \xrightarrow{\delta} \tau\ ;a+\tau\ ;b+\tau\ ;c+\tau\ ;d$, tenemos que Q debe envejecer, pero sin resolver la elección interna que aparece en el estado inicial, con lo que obtenemos $Q \overset{\delta}{\rightsquigarrow} \tau\,;(\tau\,;a+\tau\,;b)+\tau\,;(\tau\,;c+\tau\,;d)$. Esta transición está permitida gracias a la regla 6. En el caso de que Q' decidiese aplicar la regla 2 y 4 obteniendo el proceso $\mathtt{delay}(\delta)\,;(\tau\,;a+\tau\,;c)$. El

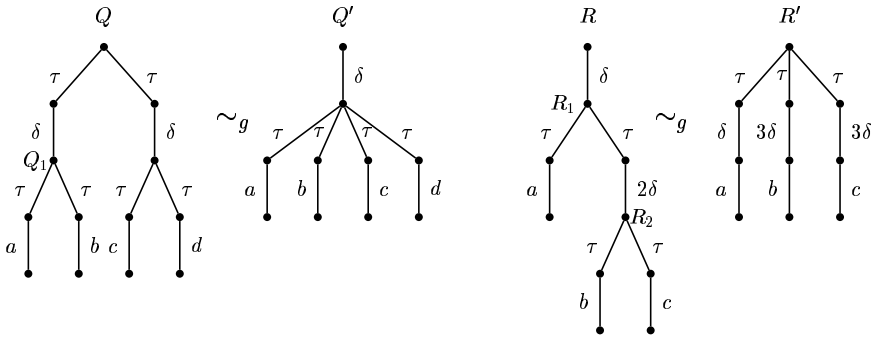


Figura 2.10: Asociatividad en presencia de tiempo.

proceso Q podría simular esta transición aplicando las reglas 1 y 4 de modo que se resolvería la elección entre a y b, a favor de a, y la existente entre c y d, a favor de c, para alcanzar justo el mismo proceso que alcanzó Q' tras su evolución. De modo que Q y Q' son, en efecto, bisimilares globales.

Si consideramos ahora los procesos R y R' de la figura 2.10, tenemos que también son bisimilares globales temporales. Su equivalencia es una consecuencia natural de nuestra teoría, debido a que hemos conseguido abstraer la información sobre *cuándo* se toman las decisiones. Así, tenemos que cuando R ejecuta la transición $R \overset{\delta}{\rightsquigarrow} R_1$, el proceso R' debe envejecer la misma cantidad de tiempo en cada rama. Aplicando la regla 6 obtenemos $R' \overset{\delta}{\rightsquigarrow} R'_1 = \tau\ ;\ a + \tau\ ;\ \mathtt{delay}(2\delta)\ ;\ b + \tau\ ;\ \mathtt{delay}(2\delta)\ ;\ c$. En este punto, R_1 tiene dos opciones: por un lado, si decide ejecutar la transición $R_1 \xrightarrow{\tau} a$, entonces R'_1 puede ejecutar la misma transición para simularlo; por otro lado, si R_1 ejecuta la transición $R_1 \xrightarrow{\tau} \mathtt{delay}(2\delta)\ ;\ (\tau\ ;\ b + \tau\ ;\ c)$, tenemos la misma situación que recogía el ejemplo anterior. Por otro lado, si R' decide evolucionar aplicando la regla 2 alcanzando el estado $\tau\ ;\ \mathtt{delay}(\delta)\ ;\ a + \tau\ ;\ \mathtt{delay}(2\delta)\ ;\ b$, el proceso R podría imitarlo ejecutando una transición interna en R_2 para resolver la elección a favor de b (lo que es posible aplicando las reglas 1 y 4) alcanzando el estado $\mathtt{delay}(\delta)\ ;\ (\tau\ ;\ a + \tau\ ;\ \mathtt{delay}(2\delta)\ ;\ b)$ de modo que los procesos alcanzados son bisimilares globales temporales.

Pasemos a continuación a presentar algunos ejemplos que muestran el comportamiento de nuestra equivalencia en el caso en el que se combinan el no determinismo simétrico y el paso del tiempo. La bisimulación débil temporal ya trata adecuadamente algunas interacciones simples entre elecciones internas generadas por τ's y transiciones de tiempo. Por ejemplo,

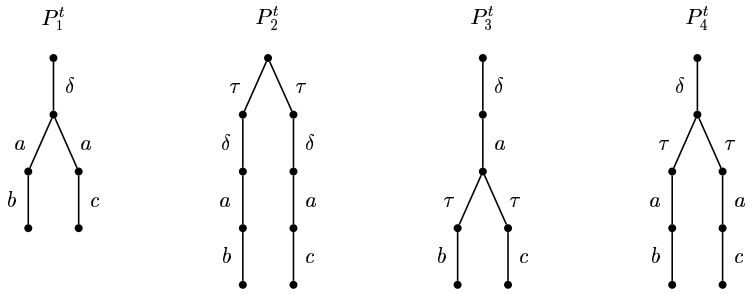


Figura 2.11: Diferentes tipos de no determinismo simétrico en procesos temporales.

tenemos

$$\texttt{delay}(t)\,;P \;\sim_{wt}\; \tau\,;\texttt{delay}(t)\,;P \;\sim_{wt}\; \texttt{delay}(t)\,;\tau\,;P$$

Por el contrario, si mezclamos elecciones no deterministas con tiempo de forma más compleja produce resultados indeseables en el contexto de la bisimulación débil temporal. Por ejemplo, los siguientes procesos:

$$P_1^t = (\texttt{delay}(\delta)\,;a\,;b) + (\texttt{delay}(\delta)\,;a\,;c)$$

$$P_2^t = (\tau\,;\texttt{delay}(\delta)\,;a\,;b) + (\tau\,;\texttt{delay}(\delta)\,;a\,;c)$$

$$P_3^t = \texttt{delay}(\delta)\,;a\,;(\tau\,;b+\tau\,;c)$$

$$P_4^t = \texttt{delay}(\delta)\,;(\tau\,;a\,;b+\tau\,;a\,;c)$$

gráficamente representados en la figura 2.11, no son bisimilares débiles temporales.

Para obtener $P_1^t \sim_{gt} P_2^t$, tenemos que la transición $P_1^t \xrightarrow{\delta} a\,;b+a\,;c$ puede ser simulada por P_2^t aplicando la regla 6, *envejeciendo* sin resolver el no determinismo, para obtener $P_2^t \overset{\delta}{\rightsquigarrow} \tau\,;a\,;b+\tau\,;a\,;c$. Además, para simular la transición $P_2^t \xrightarrow{\tau} \texttt{delay}(\delta)\,;a\,;b$, tenemos que nuestras reglas permiten que P_1^t elimine una de las ramas que aparecen después de las transiciones de tiempo (por medio de las reglas 3 y 4).

Por otro lado, tenemos también que $P_1^t \sim_{gt} P_3^t$ dado que la transición $P_3^t \xrightarrow{\delta} a;(\tau;b+\tau;c)$ puede ser simulada por P_1^t simplemente ejecutando la transición $P_1^t \xrightarrow{\delta} a\,;b+a\,;c$ con lo que obtendríamos los sistemas de transiciones P_1 y P_3 de la figura 2.3. Se mantiene el mismo resultado para P_1^t y P_4^t debido a que cuando uno cualquiera de ellos ejecuta una transición temporal, el otro puede ejecutarla también, de modo que se alcanzan los procesos P_1 y P_2 de la figura 2.3. Además, tenemos que P_2^t y P_3^t son bisimilares globales temporales, puesto que a la hora de simular la transición $P_2^t \xrightarrow{\tau} \texttt{delay}(\delta)\,;a\,;b$, el proceso P_3^t puede ejecutar una transición interna aplicando la regla 4.

Utilizando un razonamiento similar, se puede comprobar que los pares $\langle P_2^t, P_4^t \rangle$ y $\langle P_3^t, P_4^t \rangle$ son bisimilares globales temporales.

2.5. Ejemplo: Un canal de comunicación defectuoso

En esta sección resumimos las principales ideas presentadas en este capítulo a través de un ejemplo simple pero bastante ilustrativo: un canal de comunicación defectuoso. Consideraremos un canal de comunicación en el que cuando se retransmite un mensaje recibido, éste puede o bien mandarse correctamente, o bien mandarse con errores, o bien perderse. Tras dar una especificación más formal del canal de comunicación, presentaremos varios procesos que pueden considerarse implementaciones correctas de esta especificación. Parece lógico desear que todas estas implementaciones sean equivalentes. Sin embargo, como veremos a continuación, bajo la noción de bisimulación débil temporal no lo son, pero en cambio sí lo son bajo la noción de bisimulación global temporal.

2.5.1. El canal de comunicación

Queremos especificar un canal de comunicación defectuoso que cumpla las siguientes propiedades:

1. El canal recibirá eventualmente un mensaje.
2. El canal debe esperar δ unidades de tiempo desde que recibe el mensaje antes de intentar transmitirlo.
3. El canal intentará mandar el mensaje al receptor final. En tal caso se pueden dar tres situaciones:
   - El mensaje se manda de manera correcta al receptor.
   - El mensaje se manda pero el contenido no es correcto.
   - El mensaje se pierde.

   Después de todo esto, el canal se bloquea, es decir, suponemos que trabajamos con un canal de uso único.

2.5.2. La implementación

A partir de la especificación anterior, el implementador tiene varias opciones a la hora de desarrollar una implementación correcta. En la figura 2.12 presentamos algunas posibles

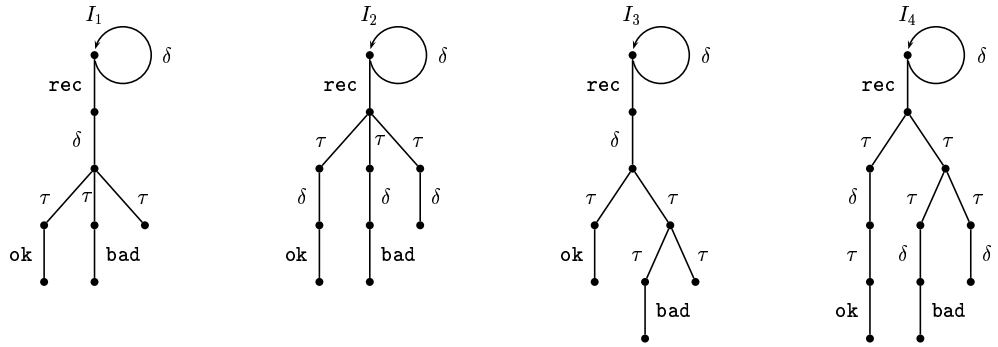


Figura 2.12: Diferentes implementaciones de un canal de comunicación defectuoso.

implementaciones, donde **rec** denota la recepción del mensaje en el canal y las acciones **ok** y **bad** representan, respectivamente, la correcta e incorrecta emisión del lenguaje. Comentemos algunas de las diferencias entre las implementaciones elegidas. En el caso de la implementación I_1 tenemos que el sistema espera hasta que el mensaje llega y, una vez recibido éste, espera el tiempo necesario para poder enviarlo. Pasado ese tiempo decide de manera no determinista si envía correctamente el mensaje, si lo envía mal o si no lo envía. Sin embargo, esta no es la única posibilidad que tiene el implementador. En la implementación I_2, se recibe el mensaje y se decide de manera no determinista sin esperar lo qué hará el sistema con el mensaje (es decir, se toma una decisión acerca de la emisión del mensaje). Después de decidirlo, se espera un tiempo δ independientemente de la decisión tomada y se produce el resultado correspondiente a la decisión tomada. La implementación I_3 es una variante de I_1. En este caso, el implementador distingue inicialmente entre solo dos casos: o bien se emite correctamente el mensaje o bien no se hace tal cosa. En el segundo caso, se produce una nueva elección no determinista entre las dos opciones restantes: o bien se manda incorrectamente el mensaje, o bien se pierde. La implementación I_4 incorpora la misma variante que en el caso anterior, pero partiendo de I_2.

Parece deseable que los cuatro sistemas anteriores sean implementaciones correctas de la especificación dada. Y entonces, ¿deberíamos considerar todas estas implementaciones equivalentes? Bajo la noción de bisimulación débil temporal, la respuesta es negativa. Por ejemplo, I_1 e I_2 no son equivalentes. Si consideramos el estado que alcanza I_1 tras la ejecución de la acción **rec** y pasado un tiempo δ, para simular esto con I_2, debemos comenzar por ejecutar **rec**, para después ejecutar una de las acciones internas tras lo cual permitiríamos el paso del tiempo. Está claro que el estado alcanzado por I_1 no es equivalente a ninguno de los

posibles estados alcanzables desde I_2. Podemos encontrar situaciones parecidas para el resto de los casos. De hecho, para cualquier par $i \neq j$ tenemos $I_i \not\sim_{wt} I_j$. Sin embargo, parece que un observador externo no debiera ser capaz de distinguir entre todas estas implementaciones: todas ellas pueden perder el mensaje o bien mandarlo (correcta o incorrectamente), y todas ellas esperan δ unidades de tiempo entre la recepción del mensaje y su transmisión. La noción de bisimulación global temporal que hemos introducido resuelve este problema, puesto que bajo esta semántica los cuatro procesos sí son equivalentes. El lector puede comprobar fácilmente que combinando los razonamientos realizados en la sección anterior, obtenemos que para todo par i, j se tiene $I_i \sim_{gt} I_j$.

2.6. Conclusiones

En este capítulo hemos presentado diversas nociones de bisimulación global, que hemos utilizado para estudiar la semántica de los procesos no deterministas. Hemos visto varias formas de debilitar la noción de bisimulación débil. A través de las mismas hemos conseguido la equivalencia entre diferentes tipos de no determinismo que se distinguen por la bisimulación débil. En particular, hemos visto como conseguir la asociatividad de la elección interna que, en nuestra opinión, es bastante deseable, pues en la mayoría de los casos el significado natural de estas elecciones es puramente estático. De hecho, podemos concluir que la no asociatividad se produce como consecuencia de la naturaleza dinámica de la noción original de bisimulación junto con un modelado no muy adecuado de las elecciones internas por medio de transiciones internas.

Hemos probado que las distintas nociones de bisimulación global se pueden definir de manera modular refinando las nociones de transiciones generalizadas para conseguir relaciones de equivalencia cada vez más débiles. Desgraciadamente, hemos visto que esta definición acumulativa no se puede aplicar a la ligera, debido a que debemos mantener la congruencia de las reglas, cuyas premisas se van debilitando a medida que la equivalencia semántica se debilita, con lo que estas reglas de congruencia deben comprobarse cada vez que se redefine la equivalencia semántica.

También hemos presentado una extensión temporal de la bisimulación global. Esta nueva semántica se puede considerar como un marco más adecuado para el estudio de la semántica de los procesos temporales no deterministas.

Capítulo 3

NMSPA: Un primer modelo de los procesos estocásticos no markovianos

En este capítulo consideraremos no sólo información temporal, como en el capítulo anterior, sino también información probabilística. En concreto, como indicamos en el capítulo 1, incluiremos este tipo de información a través de variables aleatorias que servirán, o bien para representar el tiempo que debe transcurrir (variables aleatorias degeneradas), o bien probabilidades para resolver elecciones por medio de variables aleatorias, o bien para representar cómo varían las probabilidades de que se ejecuten acciones, según va pasando el tiempo. En este capítulo introducimos un primer modelo: el álgebra de procesos estocásticos no markovianos NMSPA (Non-Markovian Stochastic Process Algebra). Este nuevo lenguaje presenta las características habituales de los modelos estocásticos pero, como novedad, las distribuciones de probabilidad no están obligadas a ser exponenciales. Esta generalización produce un aumento del poder expresivo del lenguaje por varios motivos. Por ejemplo, podremos especificar acciones que se podrán ejecutar con probabilidad 1 cuando haya transcurrido una cierta cantidad finita de tiempo. De este modo, las llamadas acciones *pasivas* se pueden representar de forma natural en nuestro marco, considerando simplemente una variable aleatoria con función de distribución de probabilidad igual a 1, si $t \geq 0$. En los modelos restringidos a distribuciones exponenciales, estas acciones se deben añadir al lenguaje (base) original, como un elemento aparte (véase p.ej. [BG98]). Otras propiedades, como la *urgencia* de la acciones internas, también aparece reflejada en nuestro lenguaje. En otros modelos, por ejemplo [BG98, HHK02], se incluye una noción de acción *inmediata*, como caso particular de acciones urgentes. Por otro lado, en aquellos modelos donde las acciones y los retardos están separados, las acciones internas se consideran normalmente urgentes (p.ej. [HS95, HHK02]).

Nuestra semántica operacional incluye información sobre el tiempo en el que se puede ejecutar la acción por medio de la variable aleatoria que define el mismo. Para definir una semántica de *interleaving* para el operador de paralelo, nos aprovecharemos de las ideas que se utilizan en las álgebras de procesos temporales y que hemos utilizado en el capítulo anterior. El problema principal consistirá en que hemos de reflejar el paso del tiempo pasa para *actualizar* la componente de la composición paralela que no está evolucionando. Para ello *envejeceremos* el proceso correspondiente de acuerdo con el tiempo que ha utilizado la otra componente para ejecutar su transición.

Como ejemplo, especificaremos en nuestro lenguaje el *protocolo del bit alternante*. Éste es un protocolo de comunicación bastante sencillo donde los canales (al igual que en el ejemplo que vimos en la extensión temporal) son defectuosos.

El resto del capítulo está estructurado de la siguiente manera. En la siguiente sección daremos los conceptos básicos que utilizaremos a lo largo de todo el capítulo. En la sección 3.2 presentaremos nuestro lenguaje y su semántica operacional. En la sección 3.3 definiremos una noción de bisimulación fuerte. En la sección 3.4 utilizaremos nuestro lenguaje para especificar el protocolo de bit alternante (ABP). Finalmente, en la sección 3.5 presentaremos nuestras conclusiones.

Este capítulo forma parte de las contribuciones originales de esta tesis y ha dado lugar a la publicación [LN00] en la que se describe el lenguaje `NMSPA`.

3.1. Conceptos básicos

En esta sección introduciremos algunos conceptos auxiliares que precisaremos en la definición del lenguaje `NMSPA`. En primer lugar, recordaremos el concepto de función de distribución de probabilidad visto en el capítulo 1, y definiremos dos funciones para la combinación de variables aleatorias. A partir de este punto, y en el resto de esta tesis, utilizaremos las letras griegas ξ, ψ y ϕ (posiblemente con algún subíndice) para denotar variables aleatorias, las letras latinas para representar acciones visibles y la letra τ para representar la acción interna.

Supondremos que el espacio probabilístico (es decir, el dominio de las variables aleatorias) es el conjunto de los números reales $\mathbb{R}$ y que las variables aleatorias únicamente toman valores positivos en $\mathbb{R}^+$, es decir, dada una variable aleatoria ξ tenemos que $Prob(\xi \leq t) = 0$ para cualquier $t < 0$. El motivo de esta restricción es que las variables aleatorias siempre están asociadas con distribuciones de tiempo.

Definición 3.1 Sea ξ una variable aleatoria. Definimos su *función de distribución de probabilidad*, denotado por F_ξ, como la función $F_\xi : \mathbb{R} \longrightarrow [0,1]$ tal que $F_\xi(x) = Prob(\xi \leq x)$, donde $Prob(\xi \leq x)$ es la probabilidad de que ξ tome valores menores o iguales que x. Denotamos por *unit* ala variable aleatoria definida por $F_{unit}(x) = 1$ para todo $x \geq 0$. Es decir, *unit* es la variable aleatoria correspondiente a la distribución de Dirac concentrada en el 0. Observemos que si se maneja la suma de variables aleatorias, para cualquier variable aleatoria ξ se cumple que $\xi + unit = \xi$. □

Durante el resto del capítulo, principalmente en los ejemplos y cuando no exista posibilidad de confusión, identificaremos las variables aleatorias con sus funciones de distribución de probabilidad.

Durante el resto de esta tesis utilizaremos las siguientes funciones de distribución de probabilidad en nuestros ejemplos:

Distribuciones uniformes. Sean $a, b \in \mathbb{R}^+$ tales que $a < b$. Una variable aleatoria ξ diremos que sigue una distribución uniforme sobre el intervalo $[a, b]$, y en tal caso la denotaremos por $U(a, b)$, si su función de distribución de probabilidad asociada es:

$$F_\xi(x) = \begin{cases} 0 & \text{si } x \leq a \\ \frac{x-a}{b-a} & \text{si } a < x < b \\ 1 & \text{si } x \geq b \end{cases}$$

Estas distribuciones nos permiten mantener la compatibilidad con los intervalos de tiempo que aparecen en algunos modelos de álgebras temporales, en el sentido de que se asigna el mismo *peso* a todos los instantes del intervalo.

Distribución de Dirac. Siendo $t_0 \in \mathbb{R}^+$, diremos que una variable aleatoria ξ sigue una distribución de Dirac (o degenerada) en t_0, y en tal caso la denotaremos por $D(t_0)$, si su función de distribución de probabilidad asociada es:

$$F_\xi(x) = \begin{cases} 0 & \text{si } x < t_0 \\ 1 & \text{si } x \geq t_0 \end{cases}$$

Distribuciones discretas. Sea $P_I = \{(t_i, p_i)\}_{i \in I}$ un conjunto de pares tales que para todo $i \in I$, $t_i \geq 0$, $p_i > 0$, para cualquier $i, j \in I$, si $i \neq j$ entonces $t_i \neq t_j$ y $\sum p_i = 1$. Una variable aleatoria ξ diremos que sigue una distribución discreta con respecto a P_I, y en tal caso la denotaremos por $D(P_I)$, si su función de distribución de probabilidad asociada es:

$$F_\xi(x) = \sum_{i \in I} \{p_i \mid x \geq t_i\}$$

Las distribuciones discretas nos van a permitir expresar acciones *pasivas*.

Distribuciones exponenciales. Sea $0 < \lambda \in \mathbb{R}$. Una variable aleatoria ξ diremos que sigue una distribución exponencial con parámetro λ, y en tal caso la denotaremos por $Exp(\lambda)$ o simplemente con λ, si su función de distribución de probabilidad asociada es:

$$F_\xi(x) = \begin{cases} 1 - e^{-\lambda x} & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}$$

Distribuciones de Poisson. Sea $0 < \lambda \in \mathbb{R}$. Una variable aleatoria ξ diremos que sigue una distribución de Poisson con parámetro λ, y en tal caso la denotaremos por $P(\lambda)$, si toma valores positivos solamente en $\mathbb{N}$, y su función de distribución de probabilidad asociada es:

$$F_\xi(x) = \begin{cases} \sum_{t \in \mathbb{N}, t \leq x} \frac{\lambda^t}{t!} e^{-\lambda t} & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}$$

A continuación, pasamos a definir dos funciones que se aplican a variables aleatorias y que serán utilizadas a la hora de definir la semántica de nuestro lenguaje. La función $\otimes$ tiene como objetivo combinar dos variables aleatorias en el caso de la sincronización. Por otra parte, utilizaremos $\oplus$ para definir la combinación de las variables aleatorias asociadas con las ejecuciones de acción estocástica desde un cierto proceso para alcanzar un cierto conjunto de ellos pertenecientes a una cierta clase. En los siguientes capítulos volveremos a utilizar esta misma función $\oplus$. Obsérvese que estas dos funciones no se corresponden con las definiciones habituales de suma y multiplicación de variables aleatorias.

Definición 3.2 Sean ξ_1 y ξ_2 dos variables aleatorias independientes con distribuciones de probabilidad F_{ξ_1} y F_{ξ_2}, respectivamente. Definimos el *producto combinado* y la *suma combinada* de ξ_1 y ξ_2, denotados por $\xi_1 \otimes \xi_2$ y $\xi_1 \oplus \xi_2$ respectivamente, como las variables aleatorias con función de distribución de probabilidad definidas del siguiente modo:

$$F_{\xi_1 \otimes \xi_2}(x) = F_{\xi_1}(x) \cdot F_{\xi_2}(x)$$
$$F_{\xi_1 \oplus \xi_2}(x) = F_{\xi_1}(x) + F_{\xi_2}(x) - F_{\xi_1 \otimes \xi_2}(x)$$

Estos dos operadores se pueden generalizar a un número arbitrario (finito) de argumentos. Sea $\Psi = \{\xi_i\}_{i \in I}$ un conjunto finito no vacío de variables aleatorias independientes. Definimos el *producto combinado* y la *suma combinada* de las variables de Ψ, denotado por $\otimes\Psi$ y $\oplus\Psi$ respectivamente, como las variables con las siguientes funciones de distribución de probabilidad:

$$F_{\otimes\Psi}(x) = \prod_{i \in I} F_{\xi_i}(x)$$
$$F_{\oplus\Psi}(x) = \sum_{\emptyset \subset \Phi \subseteq \Psi} (-1)^{(|\Phi|+1)} F_{\otimes\Phi}(x)$$

□

Obsérvese que para conjuntos unitarios de variables aleatorias, $\Psi = \{\xi\}$, tenemos que $\oplus\Psi = \otimes\Psi = \xi$. Consideraremos, por conveniencia, $F_{\oplus\varnothing}(x) = 0$ y $F_{\otimes\varnothing}(x) = 0$, para todo $x \in \mathbb{R}$. Nótese que si $\Psi = \{\xi_i\}_{i\in I}$ es un conjunto de variables aleatorias distribuidas exponencialmente con parametros λ_i entonces la variable aleatoria $\oplus\Psi$ se distribuye exponencialmente con parametro $\sum_{i\in I}\lambda_i$. Además, se puede demostrar que $\oplus$ define la variable aleatoria que se distribuye como el mínimo del conjunto de variables aleatorias dadas:

Lema 3.3 Sea $\Psi = \{\xi_1, \xi_2, \ldots, \xi_n\}$ un conjunto no vacío de variables aleatorias independientes, y sea ξ una variable aleatoria distribuida como la variable aleatoria $\text{mín}\{\xi_1, \xi_2, \ldots, \xi_n\}$. Entonces tendremos $F_\xi = F_{\oplus\Psi}$.

3.2. El lenguaje NMSPA

Tras las definiciones preliminares presentaremos la sintaxis de nuestro lenguaje. Consideraremos, como en el capítulo anterior y también en capítulos posteriores, que $\mathtt{Act}$ representa el conjunto de acciones visibles, la acción $\tau \notin \mathtt{Act}$ representa la actividad interna de los procesos, y el conjunto $\mathtt{Act}_\tau$ denota al conjunto de acciones $\mathtt{Act} \cup \{\tau\}$. Por último, consideraremos un conjunto numerable de identificadores Id que se utilizará para definir procesos recursivos.

Definición 3.4 El conjunto de procesos NMSPA se define mediante la siguiente expresión EBNF:

$$P ::= \quad stop \mid X \mid \textstyle\sum_{i\in I}(\alpha_i, \xi_i)\,;P_i \mid P \parallel_A P \mid P/A \mid P[f] \mid recX.P$$

donde $X \in Id$, I es un conjunto finito de índices, $\alpha_i \in \mathtt{Act}_\tau$, $\{\xi_i\}_{\{i\in I\}}$ es un conjunto de variables aleatorias independientes, $A \subseteq \mathtt{Act}$ y $f : \mathtt{Act}_\tau \longrightarrow \mathtt{Act}_\tau$ es una función de renombramiento tal que $f(\tau) = \tau$ y para todo $a \in \mathtt{Act}$ se cumple $f(a) \neq \tau$. □

A partir de ahora, en el resto del capítulo y también de esta tesis, asumiremos que todas las variables aleatorias que aparecen en las definiciones de procesos son independientes. En particular, la misma variable aleatoria no puede aparecer dos veces en la definición de un proceso. Esta restricción no implica que no podamos tener variables aleatorias idénticamente distribuidas, siempre que tengan distintos nombres. Aun así, en ocasiones, por abuso de notación utilizaremos la misma variable aleatoria en transiciones diferentes. Así por medio de dos transiciones etiquetadas con una misma variable aleatoria ξ estaremos indicando en realidad que se trata de dos variables aleatorias independientes ψ_1 y ψ_2, pero que están idénticamente distribuidas.

En nuestro lenguaje denotaremos por *stop* al proceso que no puede ejecutar ninguna acción. Como ya ocurría en el capítulo 2, este proceso deja pasar el tiempo. Naturalmente, cuando en el proceso *stop* pasa el tiempo, obtendremos de nuevo el mismo proceso. La expresión $(a, \xi)\,;P$ denota que se ofrece la acción a después de un retardo determinado por la distribución de probabilidad de la variable aleatoria ξ. De este modo, a se ofrece en tiempo t con probabilidad $P(\xi \leq t)$. Para resolver la elección entre diferentes acciones utilizaremos una *política de carreras*, por lo que el término $\sum_{i\in I}(a_i, \xi_i)\,;P_i$ selecciona la alternativa más rápida. Si $I = \emptyset$, consideramos que $\sum_{i\in I}(a_i, \xi_i)\,;P_i$ se comporta como *stop*. Habitualmente utilizaremos este operador en su forma binaria denotada con $+$. El proceso $P \parallel_A Q$ denota la composición paralela de P y Q imponiendo la sincronización para todas las acciones que pertenezcan al conjunto A. El proceso P/A hace que todas las acciones de P que pertenezcan a A se convertirán en internas. El término $P[f]$ hace que todas las acciones de P se renombren de acuerdo con f. Finalmente, $recX.P$ se utiliza para definir procesos recursivos.

En los siguientes ejemplos ilustramos las principales características de nuestro lenguaje, que ya fueron presentadas en la introducción de este capítulo. En primer lugar veremos cómo representar acciones *pasivas*, continuaremos el ejemplo mostrando cómo representar *elecciones internas probabilísticas*, y por último, presentaremos un ejemplo que ilustra la evolución de un proceso al resolverse las elecciones mediante una *política de carreras*.

Ejemplo 3.5 Veamos cómo representar acciones *pasivas* en nuestro lenguaje. Siendo $a \in$ `Act` y ξ una variable aleatoria distribuida según la función de distribución $D(0)$, el proceso $(a, \xi)\,;P$ estará dispuesto a ejecutar a con probabilidad 1 en tiempo 0, con lo que a se ejecutará tan pronto como el entorno la ofrezca. Consideramos que las acciones visibles no son *urgentes*, es decir, permiten que el tiempo pase en tanto y cuando el entorno no ofrezca la acción. Por el contrario, las acciones τ son urgentes (véase el próximo ejemplo 3.9), de modo que en el proceso anterior, si a fuese τ se ejecutaría en tiempo 0, al no depender del entorno.

A la hora de representar en nuestro lenguaje un tipo de elección (interna) probabilística utilizaremos variables aleatorias con distribución de probabilidad discretas. Consideremos el siguiente proceso:

$$P = (\tau, \xi_1)\,;P_1 + (\tau, \xi_2)\,;P_2$$

donde ξ_1 sigue la función de distribución $D(\{(0,1)\})$ y ξ_2 sigue una distribución discreta $D(\{(0,\frac{1}{2}),\ldots\})$. Debido a la propiedad de urgencia, y como quiera que la τ de la parte izquierda debe ejecutarse en tiempo 0 con probabilidad 1, debemos tomar la decisión en el instante 0. Intuitivamente, la probabilidad de ejecutar la acción τ de la parte izquierda

$$\frac{j\in I \ \wedge \ F_{\xi_j}(t)>0 \ \wedge \ t\leq M_{\mathcal{T}}(\sum_{i\in I}(a_i,\xi_i);P_i,\{\tau\})}{\sum_{i\in I}(a_i,\xi_i);P_i \xrightarrow{a_j,\xi_j,t} P_j}$$

$$\frac{P \xrightarrow{a,\xi,t} P' \ \wedge \ t\leq M_{\mathcal{T}}(Q,\{\tau\}) \ \wedge \ a\notin A}{P\|_A Q \xrightarrow{a,\xi,t} P'\|_A \mathrm{age}(Q,t)} \qquad \frac{Q \xrightarrow{a,\xi,t} Q' \ \wedge \ t\leq M_{\mathcal{T}}(P,\{\tau\}) \ \wedge \ a\notin A}{P\|_A Q \xrightarrow{a,\xi,t} \mathrm{age}(P,t)\|_A Q'}$$

$$\frac{P \xrightarrow{a,\xi_1,t} P' \ \wedge \ Q \xrightarrow{a,\xi_2,t} Q' \ \wedge \ a\in A}{P\|_A Q \xrightarrow{a,\xi_1\otimes\xi_2,t} P'\|_A Q'}$$

$$\frac{P \xrightarrow{a,\xi,t} P' \ \wedge \ t\leq M_{\mathcal{T}}(P,A\cup\{\tau\}) \ \wedge \ a\notin A}{P/A \xrightarrow{a,\xi,t} P'/A} \qquad \frac{P \xrightarrow{a,\xi,t} P' \ \wedge \ t\leq M_{\mathcal{T}}(P,A\cup\{\tau\}) \ \wedge \ a\in A}{P/A \xrightarrow{\tau,\xi,t} P'/A}$$

$$\frac{P \xrightarrow{a,\xi,t} P'}{P[f] \xrightarrow{f(a),\xi,t} P'[f]} \qquad \frac{P\{recX.P/X\} \xrightarrow{a,\xi,t} P'}{recX.P \xrightarrow{a,\xi,t} P'}$$

Figura 3.1: Semántica operacional para NMSPA

debería ser $\frac{1}{1+\frac{1}{2}} = \frac{2}{3}$.

A continuación mostramos cómo evoluciona un proceso siguiendo la política de carreras. Consideremos ahora el proceso $\xi_1 \,;\, a + \xi_2 \,;\, b$ donde ξ_1 y ξ_2 siguen funciones de distribución uniformes en los intervalos $(0,4)$ y $(2,5)$, respectivamente. La política de carreras utilizada para resolver esta elección nos indica que si ξ_1 toma un valor menor que ξ_2, la elección se resolverá en favor de ξ_1, en cuyo caso transcurrido un cierto tiempo se ejecutará la acción a; si ξ_2 tomará un valor menor, pasado un cierto tiempo se ejecutaría la acción b. Supongamos, como caso extremo, que ξ_1 siga la función de distribución $U(0,1)$ y ξ_2 siga $U(7,9)$. En tal caso, el valor máximo de ξ_1 es 1, mientras que el mínimo de ξ_2 es 7, con lo que la elección se resolverá siempre en favor de ξ_1, puesto que el primer proceso siempre va a ser más rápido que el otro. □

Cuando una variable aleatoria siga alguna de las distribuciones de la sección 3.1, indicaremos el hecho con la notación ξ_X, siendo X la distribución en cuestión. Por ejemplo, una variable aleatoria $\xi_{D(t_0)}$ seguiría la distribución de probabilidad $D(t_0)$.

En la figura 3.1 se presenta la semántica operacional del lenguaje NMSPA. Una transición operacional como $P \xrightarrow{a,\xi,t} P'$ indica que P puede ejecutar la acción $a \in \mathtt{Act}_\tau$, pero sólo

después de un retardo definido por la variable aleatoria ξ, que se materializará en el instante t. En este marco no necesitamos etiquetar las transiciones para deferenciarlas unas de otras, dado que suponemos la independencia de todas las variables aleatorias que aparecen en la definición de un proceso. En particular, dados P, a, y t no podríamos derivar nunca dos transiciones diferentes desde P etiquetadas con la misma tupla (a, ξ, t).

En la definición de la semántica operacional utilizamos dos funciones auxiliares: $M_{\mathcal{T}}(P, A)$ y age(P, t). La primera función se utiliza para evitar transiciones *inútiles*. Como dijimos al principio de este capítulo, nuestro modelo obliga a que las acciones internas se ejecuten, como muy tarde, en el primer momento en el que su función de distribución de probabilidad asociada tome el valor 1. Por tanto, necesitamos una función que calcule el tiempo que transcurrirá como máximo hasta que la acción interna *más rápida* alcance la probabilidad 1 de ser ejecutada. A partir de ese momento ninguna acción debería estar ya disponible. En principio necesitamos este tiempo únicamente para las acciones τ, pero en presencia del operador de ocultamiento algunas acciones visibles se convertirán en acciones internas. Necesitamos entonces una función auxiliar que devuelva el conjunto de acciones que un proceso podría ejecutar inicialmente en el caso de que no impusieramos *urgencia*, y que denotaremos por $Ini(P)$,. Para cada una de ellas calcularemos el momento en el que se ejecutarían con probabilidad 1.

Definición 3.6 Para cada proceso $P \in \mathtt{NMSPA}$. Definimos inductivamente la función $Ini(P)$ por medio de

$$Ini(stop) = Ini(X) = \emptyset$$

$$Ini(\textstyle\sum_{i\in I}(a_i, \xi_i)\,;\, P_i) = \{(a_i, t_i) \mid t_i = \min\{t \mid F_{\xi_i}(t) = 1\}\}$$

$$\begin{aligned} Ini(P \parallel_A Q) = \; & \{(a,t) \mid (a,t) \in Ini(P) \;\wedge\; a \notin A\} \cup \{(a,t) \mid (a,t) \in Ini(Q) \;\wedge\; a \notin A\} \cup \\ & \{(a,t) \mid (a,t_1) \in Ini(P) \wedge (a,t_2) \in Ini(Q) \;\wedge\; t = \max(t_1,t_2) \;\wedge\; a \in A\} \end{aligned}$$

$$Ini(P[f]) = \{(a,t) \mid (b,t) \in Ini(P) \;\wedge\; f(b) = a\}$$

$$\begin{aligned} Ini(P/A) = \; & \{(a,t) \mid (a,t) \in Ini(P) \;\wedge\; a \notin A\} \cup \{(\tau,t) \mid (\tau,t) \in Ini(P)\} \cup \\ & \{(\tau,t) \mid (a,t) \in Ini(P) \;\wedge\; a \in A\} \end{aligned}$$

$$Ini(recX.P) = Ini(P)$$

Y siendo $A \subseteq \mathtt{Act}_\tau$, definimos la función $M_{\mathcal{T}} : \mathtt{NMSPA} \times \mathcal{P}(\mathtt{Act}_\tau) \to \mathbb{R}^+ \cup \{\infty\}$ como

$$M_{\mathcal{T}}(P, A) = \min\{t \mid (\alpha, t) \in Ini(P) \;\wedge\; \alpha \in A\}$$

donde suponemos que $\min(\emptyset) = \infty$. □

Nótese que si en el caso del operador paralelo intentáramos definir esta función de forma más sencilla tomando

$$M_{\mathcal{T}}(P \parallel_A Q, B) = \min(M_{\mathcal{T}}(P, B), M_{\mathcal{T}}(Q, B))$$

tendríamos una definición incorrecta, debido a que uno de los procesos podría alcanzar ese tiempo mínimo con una acción $a \in A \cap B$, y en el caso de acciones de sincronización debemos considerar el tiempo asociado con la ejecución de la acción en ambos componentes. Debemos remarcar que esta complejidad añadida aparece porque hemos decidido mantener la urgencia de las transiciones internas. Si desearamos introducir la urgencia en los otros modelos estocásticos conocidos se necesitaría un mecanismo similar al aquí propuesto. Por el contrario, si renunciáramos a mantener la urgencia sería suficiente con tomar $M_{\mathcal{T}}(P, A) = \infty$, para cualesquiera P y A.

La función $\mathrm{age}(P, t)$ representa el paso de t unidades de tiempo en el proceso P. Como ya se comentó anteriormente, la evolución temporal en el entorno de una composición parelela provoca serios problemas, en el momento en el que las distribuciones de probabilidad no estén restringidas a ser exponenciales. Intuitivamente, si tenemos una composición paralela $P \parallel_A Q$ y uno de los procesos evoluciona mediante una acción de *interleaving*, habría transcurrido una cierta cantidad de tiempo, lo que debería quedar reflejado en la otra componente de la composición paralela. En particular, las variables aleatorias asociadas con las acciones ejecutables deberán ser inicialmente reemplazadas por otras nuevas. Una función similar se utiliza en [ABC$^+$94], apareciendo con frecuencia en álgebras de procesos temporales para mantener la urgencia de la acción τ (véase p.ej. [LdFN96]).

Definición 3.7 Sea P un proceso y t un número real no negativo. La función $\mathrm{age}(P, t)$ se define por inducción estructural por medio de

$$\begin{array}{lcl}
\mathrm{age}(\mathit{stop}, t) & = & \mathit{stop} \\
\mathrm{age}(X, t) & = & X \\
\mathrm{age}(\sum_{i \in I}(a_i, \xi_i) \,;\, P_i, t) & = & \sum_{i \in I}(a_i, \xi_i') \,;\, P_i \text{ donde } F_{\xi_i'}(x) = F_{\xi_i}(x + t) \\
\mathrm{age}(P \parallel_A Q, t) & = & \mathrm{age}(P, t) \parallel_A \mathrm{age}(Q, t) \\
\mathrm{age}(P[f], t) & = & \mathrm{age}(P, t)[f] \\
\mathrm{age}(P/A, t) & = & \mathrm{age}(P, t)/A \\
\mathrm{age}(\mathit{rec}X.P, t) & = & \mathrm{age}(P\{\mathit{rec}X.P/X\}, t)
\end{array}$$

□

Cabe destacar que la definición del operador de envejecimiento en la forma aquí indicada ha levantado ciertas críticas en la comunidad *estocástica*. El problema consiste en que en

esta definición se tiene en cuenta el tiempo que ha transcurrido por medio de la *agregación* de la probabilidad asociada con dicho periodo de tiempo. Por ejemplo, supongamos que la probabilidad de que un evento termine antes de transcurrido un tiempo t sea igual a p. Si ha transcurrido ya tiempo t y el evento no ha terminado, entonces consideramos que en tiempo $t+\epsilon$, para ϵ suficientemente pequeño, el evento debería terminar con probabilidad $p+\delta$, donde δ viene dado por la correspondiente función de distribución. Así, toda la probabilidad se acumula al principio de la nueva distribución. Sin embargo, podría parecer razonable que una vez dicho tiempo t ha transcurrido no nos limitáramos a agregar simplemente las correspondientes probabilidades, sino que definiésemos una distribución de probabilidad condicionada al hecho de que *tras tiempo* t el evento no ha terminado. Utilizando esta aproximación alternativa deberíamos sustituir la definición de las variables ξ_i' que aparecen en la definición anterior, de forma que las nuevas funciones de probabilidad fueran iguales a las originales condicionadas al hecho de que el tiempo correspondiente sea mayor que t.

Ejemplo 3.8 Sea $P = (a, \xi_1)\,;P_1 \parallel_\emptyset (b, \xi_2)\,;P_2$. Si tuvieramos una transición de la forma

$$P \xrightarrow{a,\xi_1,t} (P_1 \parallel_\emptyset (b, \xi_2)\,;P_2)$$

el paso del tiempo no quedaría reflejado en la parte derecha del operador. Sin embargo, la variable aleatoria asociada con la acción b no puede continuar siendo ξ_2, pues ha transcurrido un tiempo t. La correspondiente la transición correcta es la dada por

$$P \xrightarrow{a,\xi_1,t} P_1 \parallel_\emptyset \mathrm{age}((b, \xi_2)\,;P_2, t)$$

□

A continuación, daremos un par de comentarios adicionales sobre la semántica operacional a través de algunos ejemplos ilustrativos. Primero, destacaremos el hecho de que la urgencia de las acciones internas se mantiene gracias a la condición adicional impuesta por la función $M_{\mathcal{T}}$, que aparece en la mayoría de las reglas.

Ejemplo 3.9 Consideremos el proceso $P = (\tau, \xi_{U(0,1)})\,;P_1 + (a, \xi_{U(3,4)})\,;P_2$. Veremos que debido a la urgencia de las acciones internas tenemos que el proceso P se comporta operacionalmente como $Q = (\tau, \xi_{U(0,1)})\,;P_1$. Después de una unidad de tiempo, P está dispuesto a ejecutar τ con probabilidad 1, y como se trata de una acción urgente, esta acción se ejecuta como muy tarde en tiempo 1. Mientras que la acción a no estará preparada para su ejecución hasta que no hayan transcurrido, al menos, tres unidades de tiempo. De este modo, aplicando nuestra semántica operacional, tenemos que las transiciones que ambos procesos pueden ejecutar son las dadas por $P, Q \xrightarrow{\tau,\xi_{U(0,1)},t} P_1$, para todo t con $0 < t \leq 1$. □

En cuanto al operador de composición paralela, consideramos que una acción de sincronización se ejecuta después de un tiempo determinado por el *producto combinado* de las correspondientes variables aleatorias. La idea es que la sincronización no tendrá lugar hasta que ambas componentes del paralelo no puedan ejecutar la acción en cuestión. Por tanto, la acción más rápida deberá esperar a la más lenta.

Ejemplo 3.10 Consideremos el proceso $P = (a, \xi_{U(0,1)})\ ;\ P_1 \parallel_{\{a\}} (a, \xi_{U(3,4)})\ ;\ P_2$. Debido a que la acción a debe ejecutarse de forma sincronizada, el proceso P tendrá las mismas transiciones que el proceso $Q = (a, \xi_{U(3,4)})\ ;\ (P_1 \parallel_{\{a\}} P_2)$, pues aunque la aparición en el lado izquierdo de P de la acción a está disponible con probabilidad 1 después de que haya pasado una unidad de tiempo, la aparición de a en el lado derecho no está disponible hasta que no han transcurrido tres unidades de tiempo. De modo que, la parte izquierda de la composición debe esperar hasta que la parte derecha esté también preparada para ejecutar la acción a. □

Como ya se comentó en el capítulo 1, en la literatura se han presentado diferentes propuestas para definir la nueva variable aleatoria asociada a las transiciones generadas por una sincronización. En el caso de los procesos estocásticos restringidos a distribuciones exponenciales, se han considerado el producto, el mínimo, o el máximo de los parámetros de las distribuciones (véase p.ej. [GHR93, ABC+94, Hil96], respectivamente). En [HHK02] no se especifica la función, pudiéndose tomar cualquiera de las anteriores. En [BG98] solamente se puede producir la sincronización cuando una de las acciones involucradas sea pasiva. En el caso de distribuciones generales, [DKB98a] elige la unión del conjunto de relojes respectivos, y por tanto, la acción no se ejecutará hasta que ambos conjuntos de relojes hayan expirado. Esta propuesta es similar, aunque no es exactamente igual a la nuestra en el sentido de que la sincronización no se producirá hasta que el proceso *más lento* no esté dispuesto a ejecutar la correspondiente acción. En [BBG98] la duración de una acción de sincronización se calcula en función de la duración de las acciones correspondientes, pero esta función tampoco se define de forma concreta.

3.3. Semántica de bisimulación para NMSPA

En esta sección definimos una relación de bisimulación fuerte entre procesos de **NMSPA**. Primero debemos dar dos definiciones auxiliares, la primera de las cuales nos permitirá calcular la variable aleatoria acumulada por varias transiciones que nos conducen a un cierto

conjunto de procesos. La segunda será la definición habitual de conjunto cociente inducido por una relación de equivalencia.

Definición 3.11 Sean P un proceso, $a \in \mathtt{Act}_\tau$ y C un conjunto de procesos. Definimos la variable aleatoria asociada a la ejecución de a por parte de P alcanzando un proceso perteneciente a C, denotada por $\xi(P, a, C)$, como

$$\xi(P, a, C) = \oplus\{\psi \mid \exists t, P' \in C : \ P \xrightarrow{a,\psi,t} P'\}$$

Dado un conjunto $\mathcal{C}$ y una relación de equivalencia $\mathcal{R}$ definida sobre él. Definimos el *conjunto cociente de $\mathcal{C}$ inducido por $\mathcal{R}$*, denotado por $\mathcal{C}/\mathcal{R}$, como el conjunto cuyos elementos son las clases de equivalencia inducidas por $\mathcal{R}$, es decir,

$$\mathcal{C}/\mathcal{R} = \{\{y \in X, (x, y) \in \mathcal{R}\} \mid x \in X\}$$

□

La función $\xi(P, a, C)$ combina todas las variables aleatorias asociadas con la misma acción utilizando la función $\oplus$, dada en la definición 3.2. Obsérvese que cada variable aleatoria ψ correspondiente a las distintas transiciones de la forma $\xrightarrow{a,\psi,t}$, con valores diferentes de t se cuentan una única vez.

Definición 3.12 Decimos que una relación binaria $\mathcal{R}$ entre procesos es una *bisimulación fuerte* si para cualquier par de procesos P y Q de NMSPA, tenemos que $P\mathcal{R}Q$ implica:

- $M_{\mathcal{T}}(P, \{\tau\}) = M_{\mathcal{T}}(Q, \{\tau\})$, y
- $\forall a \in \mathtt{Act}_\tau,\ C \in \mathtt{NMSPA}/\mathcal{R} : \xi(P, a, C) \asymp_{[0, M_{\mathcal{T}}(P,\{\tau\})]} \xi(Q, a, C)$.

donde $\xi(P, a, C) \asymp_{[0,t]} \xi(Q, a, C)$ denotan que las variables aleatorias $\xi(P, a, C)$ y $\xi(Q, a, C)$ están idénticamente distribuidas en el intervalo $[0, t]$. Decimos que dos procesos P y Q son *fuertemente bisimilares*, denotado por $P \sim_s Q$, si existe una bisimulación fuerte que contenga el par $\langle P, Q\rangle$. □

Obsérvese que en la definición anterior hemos incluido explícitamente la condición $M_{\mathcal{T}}(P, \{\tau\}) = M_{\mathcal{T}}(Q, \{\tau\})$ para asegurar que ambos procesos tienen su primera acción interna disponible con probabilidad 1 al mismo tiempo.

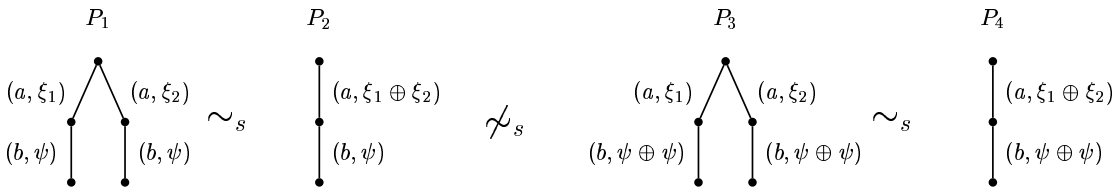


Figura 3.2: Pares de procesos fuertemente bisimilares y otros que no lo son.

Ejemplo 3.13 Siendo ξ_1, ξ_2 y ψ variables aleatorias independientes, consideraremos los siguientes procesos, representados en la figura 3.2:

$$P_1 = ((a,\xi_1)\,;(b,\psi)) + ((a,\xi_2)\,;(b,\psi))$$
$$P_2 = (a,\xi_1 \oplus \xi_2)\,;(b,\psi)$$
$$P_3 = ((a,\xi_1)\,;(b,\psi \oplus \psi)) + ((a,\xi_2)\,;(b,\psi \oplus \psi))$$
$$P_4 = (a,\xi_1 \oplus \xi_2)\,;(b,\psi \oplus \psi)$$

Es fácil comprobar que $P_1 \sim_s P_2$, $P_3 \sim_s P_4$, pero $P_2 \not\sim_s P_4$. Las variables aleatorias con las que ejecutan ambos procesos la acción a están idénticamente distribuidas, pero las distribuciones ψ y $\psi \oplus \psi$ asociadas a b no son iguales puesto que $\psi \oplus \psi$ es más *rápida* que ψ.

Consideremos ahora los procesos

$$Q_1 = (a,\xi)\,;Q + (\tau, \xi_{D(\{(1,1)\})})\,;R$$
$$Q_2 = (a,\psi)\,;Q + (\tau, \xi_{D(\{(1,1)\})})\,;R$$

siendo ξ y ψ dos variables aleatorias igualmente distribuidas en $[0,1]$, es decir, con $\xi \asymp_{[0,1]} \psi$. Entonces, a pesar de que las variables aleatorias asociadas con a en Q_1 y Q_2 pueden no estar igualmente distribuidas para algún $t > 1$, en virtud de la propiedad de urgencia, tenemos que $Q_1 \sim_s Q_2$. □

3.4. Protocolo del bit alternante (ABP)

El *protocolo del bit alternante* (ABP) [BSW69] es una representación simplificada de una clase de protocolos que aseguran la transmisión de un dato a través de un medio defectuoso. Este protocolo se ha especificado anteriormente utilizando multitud de otros modelos de procesos concurrentes. Por ejemplo, aparece en [Mil89] utilizando CCS y utilizando modelos

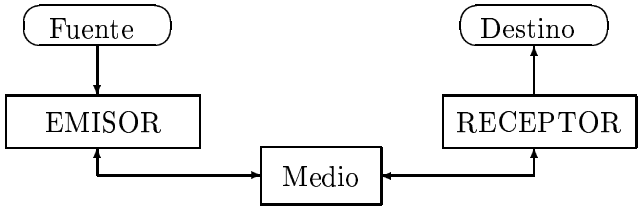


Figura 3.3: Sistema de comunicaciones.

probabilísticos o temporales se puede encontrar en [Chr90b, Sch89, Han91, GN96], para procesos probabilísticos, CSP con tiempo y versiones probabilístico-temporales de CCS y LOTOS, respectivamente.

En esta sección presentaremos nuestra especificación del protocolo del ABP utilizando el lenguaje NMSPA y comentamos las decisiones de diseño tomadas. Por claridad, para definir comportamientos recursivos utilizaremos ecuaciones en lugar del constructor *rec*, pero la traducción a nuestra sintaxis (habitual) es trivial. La descripción informal del protocolo es la siguiente: la *fuente* produce mensajes que el *emisor* debe entregar al *destinatario*; como la transmisión tiene lugar en un *medio* poco fiable, en el que se pueden perder los mensajes; el emisor y el receptor necesitan superar este problema para asegurar que el mensaje llega a su destinatario. El sistema aparece descrito esquemáticamente en la figura 3.3.

Suponemos que la fuente, que nosotros consideraremos que es el entorno, produce mensajes siguiendo una distribución de Poisson con parámetro λ. Por tanto, se podría especificar como $rec\, X.\ (\mathtt{mess}, \xi_{Po(\lambda)}); X$. El emisor esperará hasta recibir un mensaje y cuando lo reciba lo transmitirá al medio, esperando un acuse de recibo del receptor.[1] El emisor espera el acuse de recibo durante tres unidades de tiempo antes de volver a mandar el mensaje, al suponer que éste se ha perdido. Este hecho lo indicamos con una τ distribuida de forma discreta, con probabilidad 1 en tiempo tres. La duplicación de estados y la numeración de acciones para describir los mismos eventos (esto es, enviar un mensaje, recibir un acuse de recibo, etc.) son necesarios para evitar envíos de mensajes duplicados y confusión con los acuses de recibo de

[1] Consideraremos que los mensajes enviados por el emisor, por el receptor y por el medio siguen una distribución de probabilidad uniforme en el intervalo [0,1]. Con este hecho indicamos que los correspondientes sistemas pueden estar ejecutando alguna actividad no observable, pero que van a estar dispuestos para enviar el mensaje en una cantidad finita de tiempo. Realmente, esta decisión de diseño no afecta al comportamiento general del ABP.

mensajes previos. En conclusión, la especificación del emisor es la siguiente:

$$\begin{aligned} S_1 &= (\mathtt{mess}, \xi_{Po(\lambda)}) \,;\, S_2 \\ S_2 &= (\mathtt{smess}_0, \xi_{U(0,1)}) \,;\, S_3 \\ S_3 &= (\mathtt{rack}_0, \xi_{D(\{(0,1)\})}) \,;\, S_1' + (\mathtt{rack}_1, \xi_{D(\{(0,1)\})}) \,;\, S_3 + (\tau, \xi_{D(\{(3,1)\})}) \,;\, S_2 \end{aligned}$$

$$\begin{aligned} S_1' &= (\mathtt{mess}, \xi_{Po(\lambda)}) \,;\, S_2' \\ S_2' &= (\mathtt{smess}_1, \xi_{U(0,1)}) \,;\, S_3' \\ S_3' &= (\mathtt{rack}_1, \xi_{D(\{(0,1)\})}) \,;\, S_1 + (\mathtt{rack}_0, \xi_{D(\{(0,1)\})}) \,;\, S_3' + (\tau, \xi_{D(\{(3,1)\})}) \,;\, S_2' \end{aligned}$$

El receptor espera un mensaje del medio, cuando se recibe el mismo lo enviará a su destino (es decir, al entorno) y enviará un acuse de recibo al emisor a través del medio para indicar que ya ha recibido el mensaje que éste le remitió. Si el mensaje es antiguo, el receptor enviará un acuse de recibo marcado con un número antiguo. Esto ocurre cuando el medio pierde el acuse de recibo del receptor con lo que el emisor no sabe que el mensaje ya ha llegado correctamente y lo vuelve a enviar. Representamos de nuevo la espera paciente del receptor utilizando una acción pasiva. En definitiva, la especificación del receptor es la siguiente:

$$\begin{aligned} R_1 &= (\mathtt{rmess}_0, \xi_{D(\{(0,1)\})}) \,;\, R_2 + (\mathtt{rmess}_1, \xi_{D(\{(0,1)\})}) \,;\, R_3' \\ R_2 &= (\mathtt{deli}, \xi_{U(0,1)}) \,;\, R_3 \\ R_3 &= (\mathtt{sack}_0, \xi_{U(0,1)}) \,;\, R_1' \end{aligned}$$

$$\begin{aligned} R_1' &= (\mathtt{rmess}_1, \xi_{D(\{(0,1)\})}) \,;\, R_2' + (rmess_0, \xi_{D(\{(0,1)\})}) \,;\, R_3 \\ R_2' &= (\mathtt{deli}, \xi_{U(0,1)}) \,;\, R_3' \\ R_3' &= (\mathtt{sack}_1, \xi_{U(0,1)}) \,;\, R_1 \end{aligned}$$

Inicialmente, el medio espera pacientemente un mensaje del emisor o del receptor. Cuando se recibe un mensaje, el medio escoge instantáneamente entre mandarlo al lugar apropiado (o bien el emisor, o bien el receptor, dependiendo de la naturaleza del mensaje) o perderlo. Representaremos esta decisión con $(\tau, \xi_{D(\{(0,p),(1,1-p)\})})$ y $(\tau, \xi_{D(\{(0,1)\})})$ que, como ya explicamos en el ejemplo 3.5, representa que el medio pierde el mensaje con probabilidad $p/(1+p)$ con lo que la probabilidad de que no lo pierda es $1/(1+p)$. Por ejemplo, si queremos especificar un canal defectuoso en el que un 20 % de los mensajes se pierden, entonces elegiríamos

$p = 1/4$. La especificación del medio es la siguiente:

$$\begin{aligned}
M_1 &= (\mathtt{smess}_0, \xi_{D(\{(0,1)\})})\,;M_2 + (\mathtt{rack}_0, \xi_{D(\{(0,1)\})})\,;M_3 \\
&\quad + (\mathtt{smess}_1, \xi_{D(\{(0,1)\})})\,;M_4 + (\mathtt{rack}_1, \xi_{D(\{(0,1)\})})\,;M_5 \\
M_2 &= (\tau, \xi_{D(\{(0,1)\})})\,;(\mathtt{rmess}_0, \xi_{U(0,1)})\,;M_1 + (\tau, \xi_{D(\{(0,p),(1,1-p)\})})\,;M_1 \\
M_3 &= (\tau, \xi_{D(\{(0,1)\})})\,;(\mathtt{sack}_0, \xi_{U(0,1)})\,;M_1 + (\tau, \xi_{D(\{(0,p),(1,1-p)\})})\,;M_1 \\
M_4 &= (\tau, \xi_{D(\{(0,1)\})})\,;(\mathtt{rmess}_1, \xi_{U(0,1)})\,;M_1 + (\tau, \xi_{D(\{(0,p),(1,1-p)\})})\,;M_1 \\
M_5 &= (\tau, \xi_{D(\{(0,1)\})})\,;(\mathtt{sack}_1, \xi_{U(0,1)})\,;M_1 + (\tau, \xi_{D(\{(0,p),(1,1-p)\})})\,;M_1
\end{aligned}$$

El protocolo se describe como la composición paralela de los estados iniciales del emisor, del medio y del receptor. Puesto que no existe comunicación directa entre el emisor y el receptor, el conjunto de acciones en las que tienen que sincronizar ambos es vacío. Las acciones de sincronización entre el emisor y el medio, y entre el receptor y el medio las ocultaremos. Por tanto, obtendríamos

$$\mathtt{ABP} = ((S_1 \parallel_\varnothing R_1) \parallel_A M_1)/A,$$

donde $A = \{\mathtt{smess}_i, \mathtt{sack}_i, \mathtt{rmess}_i, \mathtt{rack}_i \mid i = 0, 1\}$.

3.5. Conclusiones

En este capítulo hemos introducido un primer modelo semántico con información estocástica en el que las distribuciones de probabilidad no están restringidas a ser exponenciales. El lenguaje aquí definido incluye los operadores habituales de las álgebras de procesos estocásticos. Hemos introducido una semántica operacional para el lenguaje, teniendo en cuenta la urgencia de las acciones internas, y a partir de esta semántica operacional hemos definido una noción de bisimulación. Esta equivalencia constituye la base a partir de la cual, describiremos en el capítulo siguiente otras relaciones de bisimulación que identifiquen más procesos. Por último, hemos presentado una especificación del protocolo ABP. Este protocolo es ciertamente muy simple, pero se utiliza de forma habitual para mostrar las principales características de los modelos probabilísticos y/o temporales.

El principal inconveniente que hemos encontrado en este modelo es la dificultad de definir una bisimulación más débil que la dada, es decir, una bisimulación débil que abstraiga en parcial la información interna. Debido a que las acciones tienen tiempos asociados, a la hora de juntar transiciones internas deberíamos unir también sus correspondientes variables aleatorias. En el siguiente capítulo desarrollaremos esta idea de unir las variables aleatorias, de modo que consideraremos el presente lenguaje como un primer paso para un estudio más

detallado de un lenguaje que se irá mejorando paulativamente. Así, en los capítulos siguientes trabajaremos en la definición de lenguajes que se adecúen mejor a nuestras espectativas, y definiremos también otros marcos semánticos, como las semánticas de testing y nociones de bisimulación en las que la actividad interna se abstraiga de alguna forma. Para estas definiciones de bisimulación nos basaremos en las definiciones de bisimulación débil [Mil89], ramificada [GW96] y la bisimulación global [dFLN99b] (vista en el capítulo 2). Como veremos a lo largo de este trabajo, ésta no es en absoluto una tarea sencilla, dado que abstraer la información interna en nuestro marco es algo que dista mucho de ser trivial.

Capítulo 4

Semánticas de bisimulación estocástica

En este capítulo definiremos una noción de bisimulación que capture adecuadamente el comportamiento de los sistemas estocásticos con distribuciones generales. En este caso daremos la definición para sistemas de transiciones con elecciones no deterministas cuyas elecciones se resolverán mediante una política de carreras. La idea principal subyacente consiste en que esta nueva bisimulación identifique diferentes secuencias de variables aleatorias si las *sumas* de las variables aleatorias de cada secuencia están idénticamente distribuidas. Es decir, no identificaremos únicamente secuencias de acciones internas con una de ellas (como se hace habitualmente en las bisimulaciones débiles) sino que también reduciremos (en algunas circunstancias) secuencias de transiciones estocásticas a una única transición. De modo que identificaremos procesos que no se han considerado equivalentes en nociones previas de bisimulación para este tipo de lenguajes. Denominaremos a esta nueva semántica como *bisimulación débil estocástica*. Los resultados de este capítulo aparecen recogidos en [LN02].

Por tanto nuestro propósito consiste en definir una relación de equivalencia que capture por un lado la estructura ramificada de los procesos estocásticos, y, por otro, sea suficientemente abstracta con respecto a las transiciones estocásticas. De modo que ahora pretendemos definir una semántica que identifique más procesos que la bisimulación presentada anteriormente. En este capítulo hemos preferido separar el comportamiento estocástico del funcional debido a que, así como vimos en el capítulo 1, la definición de la semántica es más sencilla. Habitualmente, para identificar más procesos que una bisimulación fuerte, se suele tratar de manera más abstracta la evolución interna de los procesos, dando lugar a una bisimu-

lación débil. Aunque es cierto que la mayoría de las semánticas para procesos estocásticos que han aparecido en la literatura están basadas en la noción de bisimulación,[1] en nuestra opinión, las definiciones previas no abstraen adecuadamente las transiciones estocásticas, pues en realidad las tratan como si fueran esencialmente *visibles*. Por ejemplo, consideremos los procesos que aparecen en la figura 4.1, cuyos estados iniciales son s_1 y s_2. En nuestra opinión, para una elección apropiada de ξ en el segundo proceso s_1 y s_2 deberían ser equivalentes pero, esto normalmente, no se considera así. El motivo es que s_1 puede ejecutar dos transiciones estocásticas consecutivas mientras que s_2 solamente puede ejecutar una. Esta situación representa una notable diferencia con respecto a cómo se trata el tiempo en las álgebras de procesos temporales. Por ejemplo, el modelo temporal definido en el capítulo 2 consideraría que estos dos procesos son equivalentes si ξ_1, ξ_2 y ξ representarán los retardos temporales correspondientes. En general, en cualquier álgebra de procesos temporal que se precie tendremos que un proceso como $\mathrm{delay}(1)\,;\mathrm{delay}(2)\,;Q$ siempre será equivalente al proceso $\mathrm{delay}(3)\,;Q$, cuando $\mathrm{delay}(n)$ indica un retardo de n unidades de tiempo. Y la única diferencia entre los modelos temporales y los estocásticos es que el retardo en los primeros es fijo, con lo que, siguiendo este razonamiento, deberíamos poder identificar s_1 y s_2 para una ξ adecuada, pareciendo en principio una buena candidata para la variable aleatoria ξ la suma de ξ_1 y ξ_2.

En definitiva, en este capítulo propondremos una nueva semántica de bisimulación para procesos estocásticos. Con respecto al tratamiento de las acciones *normales*, nuestra definición sigue la misma aproximación que la definición clásica de bisimulación débil. A continuación, presentaremos algunos ejemplos simples para mostrar los procesos que querríamos identificar, en función de sus comportamientos estocásticos. Consideremos los procesos con estados iniciales s_3, s_4 y s_5 de la figura 4.1; los dos primeros son bisimilares débiles, por lo que nos gustaría identificarlos también bajo nuestra nueva semántica. Por el contrario, a s_5 hasta ahora se le venía considerando *diferente*, debido a que a partir de s_5 se puede ejecutar una transición estocástica, mientras que éste no es el caso ni para s_3 ni para s_4. Pero esta transición nos lleva de nuevo al propio estado s_5, por lo que el significado intuitivo de este proceso es "si el entorno ofrece a, entonces se ejecuta a; si se consume el retardo antes de que se ofrezca a, se ejecuta ξ y el proceso vuelve de nuevo al estado inicial". En consecuencia, no deberíamos ser capaces de distinguir entre s_4 y s_5, pues ambos estados pueden ejecutar

[1] En [BC00] y [LN01] aparecen, que nosotros conozcamos, las únicas excepciones de marcos semánticos no basados en bisimulación. En el primer trabajo se ha definido semánticas de *testing* para un subconjunto de EMPA [BG98], mientras que en el segundo ha sido para un álgebra de procesos estocásticos con distribuciones generales. Véase el capítulo 5 para una presentación detallada de este último trabajo.

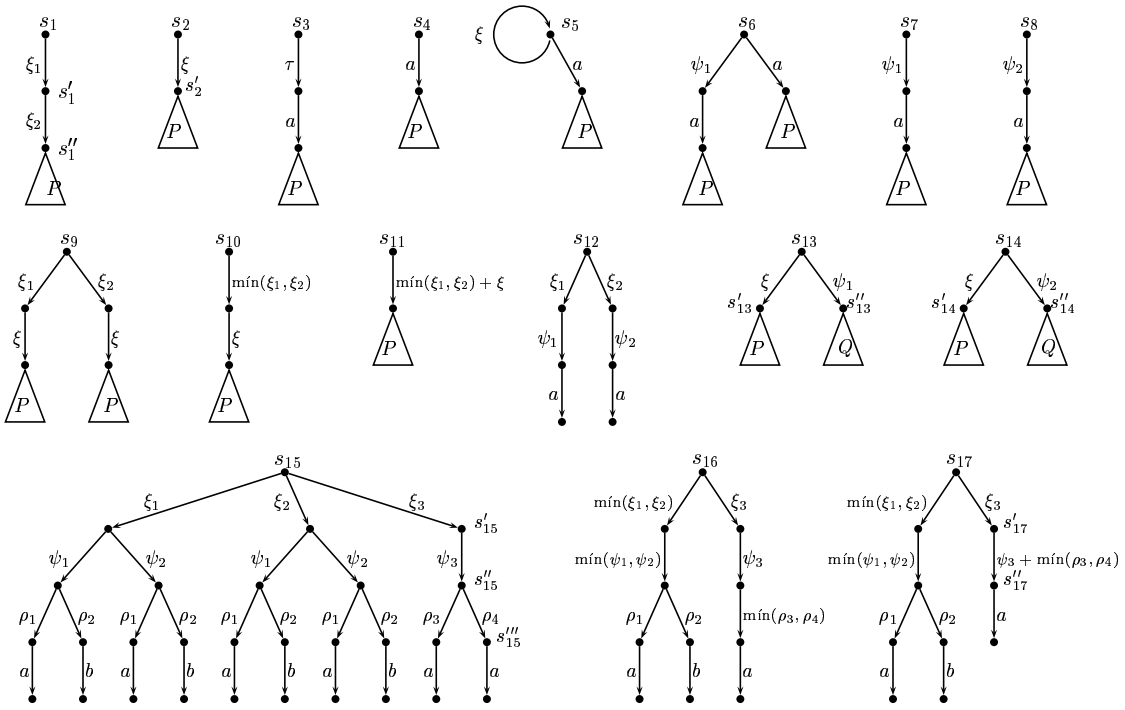


Figura 4.1: Ejemplos de procesos estocásticos.

la acción a tan pronto como la ofrezca el entorno. Utilizando un razonamiento similar, s_6 también debería ser equivalente a los procesos anteriores.

Consideremos ahora los procesos cuyos estados iniciales son s_7 y s_8. Suponiendo que ψ_1 y ψ_2 no están idénticamente distribuidas, la probabilidad de ejecutar a después de transcurrido un cierto tiempo es diferente en ambos procesos, con lo que s_7 y s_8 tampoco no deberían ser equivalentes en nuestra semántica, como no lo eran en ninguno de los marcos definidos hasta la fecha.

Consideremos ahora los procesos con estados iniciales s_9, s_{10} y s_{11}. En las semánticas de bisimulación estocásticas habituales se identifican normalmente los dos primeros procesos, con lo que por supuesto nosotros también lo haremos. La elección en s_9 se resuelve siguiendo la política de carreras, por lo que se elige el retardo *más rápido*, obteniéndose así una variable aleatoria distribuida como el *mínimo* de ξ_1 y ξ_2. Además, siguiendo un razonamiento similar al de los procesos s_1 y s_2, tendremos que s_{11} es equivalente a s_9 y a s_{10}. Este ejemplo representa la clave principal de nuestra semántica: para definirla comprobamos en primer lugar si debemos *unir* algunas transiciones (utilizando el mínimo) por alcanzar tras ellas

estados *equivalentes*. A continuación, comprobamos si se deben sumar las *continuaciones* con la variable aleatoria resultante. Todo ello justificado por el hecho de que si un observador no puede apreciar los estados intermedios donde se toman decisiones temporales, al llevar a estados equivalentes, entonces las transiciones estocásticas deberían considerarse, como comportamiento *interno* de los procesos. Por el contrario, existen ocasiones en las que la ramificación producida por las transiciones estocásticas no se puede abstraer, no pudiéndose unir las correspondientes transiciones estocásticas. Por ejemplo, si consideramos el proceso s_{12} y tenemos en cuenta los comentarios previos sobre s_7 y s_8, no deberíamos unir las dos variables aleatorias iniciales pues tras ellas no se alcanzan estados equivalentes. Por tanto, este proceso no será equivalente a ningún otro, salvo renombramiento trivial de sus variables aleatorias.

La combinación de *mínimos* y *sumas* que hemos comentado anteriormente, provoca que la definición de la semántica deseada no sea tan sencilla de enunciar como nos gustaría. Consideremos los procesos s_{15}, s_{16} y s_{17} que aparecen en la figura 4.1. En nuestra semántica estos tres procesos serán equivalentes. Este ejemplo muestra que sólo se juntan vía el mínimo de las variables aleatorias que llevan a estados equivalentes en este caso ξ_1 y ξ_2. Del mismo modo, podemos unir a continuación ψ_1 y ψ_2 calculando el mínimo de ambas, y también ρ_3 y ρ_4 puesto que nos conducen a estados equivalentes. Después, no existe la posibilidad de juntar las dos transiciones etiquetadas con $\mathrm{mín}(\xi_1, \xi_2)$ y $\mathrm{mín}(\psi_1, \psi_2)$, puesto que esta unión produciría un cambio en la elección estocástica que debe llevarse a cabo en la raíz. En concreto, si no se unen, la elección se resuelve a favor de la variable aleatoria más rápida de entre $\mathrm{mín}(\xi_1, \xi_2)$ y ξ_3. En cambio, si unieramos estas continuaciones la elección nos daría la variable aleatoria más rápida de entre $\mathrm{mín}(\xi_1, \xi_2) + \mathrm{mín}(\psi_1, \psi_2)$ y $\xi_3 + \psi_3$. Por tanto, el resultado de ambas elecciones no sería el mismo. Además, los estados siguientes a la ejecución de ρ_1 y ρ_2 no son equivalentes y, por tanto, no los podemos unir. Finalmente, podemos sumar ψ_3 y $\mathrm{mín}(\rho_3, \rho_4)$ puesto que con ello no se modificará el resultado de ninguna elección.

Para conseguir que la presentación sea lo más sencilla posible, vamos a definir nuestra semántica sobre un sistema de transiciones etiquetado (estocástico). Como ya hemos comentado previamente, diferenciaremos las acciones *ordinarias* de las acciones *estocásticas*. Las variables aleatorias asociadas con las transiciones estocásticas podrán tener cualquier tipo de distribución de probabilidad, al igual que pasaba en el capítulo anterior. [2]

[2]Para nuestros propósitos, necesitamos una clase de distribuciones que sea cerrada bajo mínimo y bajo convolución/suma. Trivialmente, la clase de *todas* las distribuciones es cerrada. A la hora de realizar análisis de rendimiento de sistemas podríamos restringirnos a distribuciones phase-type. Ello es particularmente

En la sección 4.1 presentaremos este tipo de sistema de transiciones etiquetado, en el que inicialmente no habrá información probabilística en las elecciones. Tras ello, en la sección 4.2 definiremos tres nociones distintas de bisimulación: una bisimulación fuerte basada en la del capítulo anterior, una débil donde abstraemos exclusivamente la evolución interna de los procesos y una débil estocástica en la que, además, abstraeremos parcialmente la información estocástica. En la sección 4.3 probaremos las principales propiedades de la nueva noción de bisimulación. Como es habitual, la bisimulación débil será más débil que la bisimulación fuerte, mientras que también probaremos que la nueva bisimulación débil estocástica será más débil que la propia bisimulación débil. Por último, en la sección 4.4 presentaremos las conclusiones de este capítulo.

4.1. Conceptos básicos

En esta sección definiremos nuestro nuevo modelo de procesos estocásticos. A diferencia del modelo considerado en el capítulo anterior las transiciones del sistema que consideraremos podrán estar etiquetadas con acciones y con variables aleatorias. Como ya se comentó en la introducción de este capítulo, hemos decidido separar la información estocástica de las acciones visibles para simplificar así las definiciones siguientes. Los conceptos básicos definidos sobre variables aleatorias se recogieron en el capítulo 3 en las definiciones 3.1 y 3.2. Como ya indicamos en el capítulo anterior, en la definición de las bisimulaciones utilizaremos la *suma combinada*, denotada por $\oplus$, cuando combinemos variables aleatorias asociadas al mismo estado.

A continuación, definiremos nuestra noción de sistema de transiciones etiquetado estocástico. Como en el resto de los capítulos, suponemos que partimos de un conjunto fijo de acciones $\mathtt{Act}$ ($a, a', \ldots$ denotan elementos de $\mathtt{Act}$), que existe una acción especial $\tau \notin \mathtt{Act}$ que representa la actividad interna, y que $\mathtt{Act}_\tau$ representa el conjunto $\mathtt{Act} \cup \{\tau\}$ ($\alpha, \alpha', \ldots$ denotan elementos de $\mathtt{Act}_\tau$). Además denotamos por $\mathcal{V}$ al conjunto de variables aleatorias ($\xi, \xi', \psi, \ldots$ denotan elementos de $\mathcal{V}$). Por último, $\gamma, \gamma', \ldots$ denotarán elementos genéricos de $\mathtt{Act}_\tau \cup \mathcal{V}$.

Definición 4.1 Un *sistema de transiciones etiquetado estocástico* P es un par $(S, \rightarrow)$ donde S es un *conjunto de estados* finito y $\rightarrow : S \times (\mathtt{Act}_\tau \cup \mathcal{V}) \longrightarrow S$ es una *relación de transición* entre estados. □

interesante pues para esta familia de distribuciones existen algoritmos eficientes para calcular algunas medidas de utilidad con los que realizar la evaluación del rendimiento de los sistemas considerados.

Utilizaremos los siguientes convenios: $s \xrightarrow{\gamma} s'$ significa que $(s,\gamma,s') \in \rightarrow$; $s \xrightarrow{\gamma}$ significa que existe algún $s' \in S$ tal que $s \xrightarrow{\gamma} s'$; y escribimos $s \not\xrightarrow{\gamma}$ si no existe un tal $s' \in S$. Decimos que un estado s es *estable* si $s \not\xrightarrow{\tau}$. Denotamos por $\stackrel{\tau}{\Longrightarrow}$ el cierre reflexivo y transitivo de la relación $\xrightarrow{\tau}$; y dado $\gamma \neq \tau$, escribimos $s \stackrel{\gamma}{\Longrightarrow} s'$ si existen s_1, s_2 tales que $s \stackrel{\tau}{\Longrightarrow} s_1 \xrightarrow{\gamma} s_2 \stackrel{\tau}{\Longrightarrow} s'$. Dado un conjunto $A \subseteq \mathtt{Act}_\tau \cup \mathcal{V}$, escribimos $s \xrightarrow{A}$ (resp. $s \stackrel{A}{\Longrightarrow}$) si existe algún $\gamma \in A$ tal que $s \xrightarrow{\gamma}$ (resp. $s \stackrel{\gamma}{\Longrightarrow}$) y escribimos $s \not\xrightarrow{A}$ (resp. $s \not\stackrel{A}{\Longrightarrow}$) si no existe un tal γ.

Intuitivamente, una transición $s \xrightarrow{a} s'$ indica que un proceso puede evolucionar desde el estado s al estado s', ejecutando la acción visible a; mientras que una transición $s \xrightarrow{\tau} s'$ indica que un proceso puede evolucionar internamente de s a s'. Por último, una transición $s \xrightarrow{\xi} s'$ expresa que un proceso puede evolucionar del estado s al estado s' una vez que ha transcurrido un tiempo (aleatorio) definido por ξ. Recordemos que para evitar *efectos laterales* supondremos, como en el resto de capítulos, que todas las variables aleatorias que etiquetan las transiciones son independientes.

En este modelo van a seguir dándose dos propiedades importantes que ya vimos en el capítulo anterior. En primer lugar, para resolver la elección entre acciones estocásticas se utilizará una *política de carreras*. Además, volvemos a considerar que las acciones internas son *urgentes*, es decir, se cumple la propiedad de *progreso máximo*. Antes de definir las distintas semánticas de bisimulación vamos a ver como influyen estas características a la hora de dar significado a los procesos. Los siguientes ejemplos muestran la forma en la que la *política de carreras*, bajo la cual las elecciones se resuelven a favor del retardo más rápido, identifica algunos procesos que en principio podrían no parecer equivalentes.

Ejemplo 4.2 Consideremos los procesos s_{13} y s_{14} de la figura 4.1. Supongamos que ξ es una variable aleatoria distribuida uniformemente en el intervalo $[0,2]$, ψ_1 está distribuida uniformemente en $[1,5]$ y la función de distribución de probabilidad ψ_2 viene dada por la siguiente función:

$$F_{\psi_2}(x) = \begin{cases} 0 & \text{si } x \leq 1 \\ \frac{x-1}{4} & \text{si } 1 < x \leq 4 \\ \frac{x}{2} - \frac{5}{4} & \text{si } 4 < x \leq \frac{9}{2} \\ 1 & \text{si } \frac{9}{2} < x \end{cases}$$

Teniendo en cuenta que ψ_1 y ψ_2 no están idénticamente distribuidas, podríamos pensar que s_{13} y s_{14} no son equivalentes. Sin embargo, como en cada uno de los dos estados se elegirá el retardo más rápido, después de dos unidades de tiempo, se va a elegir ξ con probabilidad 1, pues $Prob(\xi \leq 2) = 1$. Como tenemos que ψ_1 y ψ_2 están idénticamente distribuidas en

el intervalo $[0,4]$, también lo están en el intervalo $[0,2]$, por lo que s_{13} y s_{14} serán de hecho equivalentes. Nótese que aunque F_{ψ_1} y F_{ψ_2} toman valores distintos para tiempos mayores que 4, el retardo dado por ξ se ejecutará con probabilidad 1 cuando han transcurrido dos unidades de tiempo. Por ello, el resto de valores que tomen las correspondientes funciones de distribución serán irrelevantes. □

Ejemplo 4.3 Consideremos de nuevo los procesos s_{13} y s_{14} de la figura 4.1, pero supongamos ahora que ξ está distribuida uniformemente sobre el intervalo $[0, \frac{1}{2}]$ y sean ψ_1 y ψ_2 definidas como en el ejemplo anterior. Tenemos entonces que la probabilidad de que se ejecuten ψ_1 o ψ_2 sería nula. Con lo que en este caso, s_{13} y s_{14} deberían ser equivalentes, siéndolo también al proceso s_2 (también en la figura 4.1). El motivo es que el retardo asociado con ξ se ejecuta con probabilidad 1 antes de que haya transcurrido media unidad de tiempo, y en ese momento ψ_1 y ψ_2 todavía no se pueden ejecutar. □

4.2. Bisimulaciones estocásticas

En cuanto a las bisimulaciones que presentaremos, las dos primeras se corresponden con las nociones típicas de bisimulación fuerte y débil para procesos estocásticos (véase por ejemplo [Her98, BG98] para nociones similares en el marco de las distribuciones markovianas); la tercera noción de bisimulación que definiremos, y que denominaremos *bisimulación débil estocástica*, presenta un refinamiento de las nociones anteriores, lo que nos conduce a una equivalencia más débil. Consideraremos que no sólo podemos abstraer (parcialmente) las acciones internas sino también, en algunos casos, las secuencias de variables aleatorias. Como ya hemos comentado anteriormente, consideramos que las acciones internas son *urgentes*. Por otra parte, consideraremos que un estado que no puede evolucionar, ni interna ni estocásticamente, es un estado que puede *esperar* cualquier cantidad de tiempo antes de ejecutar una acción visible. Partiendo de todas estas consideraciones, introduciremos los siguientes predicados que utilizaremos para poder definir correctamente las semánticas de bisimulación.

Definición 4.4 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico, $s, s' \in S$, $\xi, \psi \in \mathcal{V}$, $t_0 \in \mathbb{R}^+$ y $C \subseteq S$, definimos los siguientes conceptos:

Tiempo de espera máximo. El *tiempo de espera máximo* para un estado s, denotado por $maxW(s)$, viene dado por

$$maxW(s) = \begin{cases} 0 & \text{si } s \xrightarrow{\tau} \\ \text{mín}\{t \mid \exists \xi \in \mathcal{V} : s \xrightarrow{\xi} \wedge F_\xi(t) = 1\} & \text{en caso contrario} \end{cases}$$

Variables aleatorias idénticamente distribuidas. Decimos que ξ y ψ están *idénticamente distribuidas con respecto a* los estados s y s', lo que denotamos por $\xi \asymp_{s,s'} \psi$, si se cumple que $maxW(s) = maxW(s')$, y para cualquier $t \leq maxW(s)$ tenemos $F_\xi(t) = F_\psi(t)$.

Variable aleatoria factible. Decimos que la variable aleatoria ξ es *factible con respecto a* t_0, denotado por $fact_{t_0}(\xi)$, si se tiene que $t_0 > \text{mín}\{t \mid F_\xi(t) > 0\}$.

Conjunto realmente alcanzable. Decimos que s *puede realmente alcanzar* C, denotado por $s \nearrow C$, si existen $s' \in C$ y $\xi \in \mathcal{V}$ tales que $s \xrightarrow{\xi} s'$ y $fact_{maxW(s)}(\xi)$.

□

La función $maxW(s)$ asegurará la propiedad de progreso máximo. Si el estado puede evolucionar internamente, entonces no puede esperar a que transcurra más tiempo, por lo que tendríamos $maxW(s) = 0$. En cambio, si el estado es estable, es decir, no se pueden ejecutar transiciones internas, entonces la función $maxW(s)$ calculará el tiempo máximo que s puede esperar hasta que algún retardo se ejecute con probabilidad 1, donde tomaremos $\text{mín}\, \varnothing = \infty$. Por tanto, si s es estable y no tiene que ejecutar transiciones estocásticas, puede esperar tanto tiempo como sea necesario hasta que una de las transiciones correspondiente a alguna de sus acciones sea requerida por el entorno.

Utilizaremos el predicado $\asymp_{s,s'}$ para comparar variables aleatorias. Como ya hemos mostrado en los ejemplos previos, solamente estamos interesados en comparar las funciones de distribución de probabilidad asociadas a las variables aleatorias en el intervalo de tiempo limitado por el tiempo de espera máximo de los estados correspondientes. Por este motivo, los estados que van a ejecutar las variables aleatorias a comparar deben aparecer como parámetros de este predicado. Por ejemplo, si retomamos el ejemplo 4.2, tenemos que $\psi_1 \asymp_{s_{13},s_{14}} \psi_2$, pues $maxW(s_{13}) = maxW(s_{14}) = 2$. Por este motivo, los estados s_{13} y s_{14} deberían ser equivalentes. Mientras que en el ejemplo 4.3 ni siquiera consideraremos las transiciones etiquetadas con ψ_1 y ψ_2 puesto que no son factibles con respecto al tiempo de espera máximo de sus correspondientes estados.

Hemos definido el predicado $s \nearrow C$ para evitar, en su caso, que debamos calcular más adelante el mínimo de un conjunto vacío de variables aleatorias. Este predicado se cumple cuando s puede alcanzar algún estado del conjunto dado $C \subseteq S$, ejecutando una (y sólo una) transición estocástica.

Antes de presentar nuestra noción de bisimulación fuerte damos una definición auxiliar similar a la utilizada en el capítulo anterior para combinar las variables aleatorias que,

partiendo del mismo estado, llegan a estados equivalentes.

Definición 4.5 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico y $C \subseteq S$. Definimos la *variable aleatoria fuerte* asociada con la ejecución de las transiciones estocásticas desde un estado $s \in S$ alcanzando un estado en C, denotada por $\xi_s(s, C)$, en la forma

$$\xi_s(s, C) = \oplus\{\xi \mid \exists s' \in C \ : \ s \xrightarrow{\xi} s'\}$$

□

La función $\xi_s(s, C)$ combina todas las variables aleatorias que partiendo del estado s llegan a un estado que pertenece al conjunto C utilizando la función $\oplus$. Recordemos que $\oplus$ calcula el mínimo de un conjunto de variables aleatorias.

Definición 4.6 Decimos que una relación de equivalencia $\mathcal{R}$ es una *bisimulación fuerte* sobre un sistema etiquetado de transiciones estocástico $P = (S, \rightarrow)$ si para cada par de estados $s_1, s_2 \in S$ tenemos que $s_1 \mathcal{R} s_2$ implica:

- Para toda $\alpha \in \mathtt{Act}_\tau$ tal que $s_1 \xrightarrow{\alpha} s_1'$ existe algún s_2' tal que $s_2 \xrightarrow{\alpha} s_2'$ y $s_1' \mathcal{R} s_2'$.
- Para cualquier $C \in S/\mathcal{R}$, si $s_1 \nearrow C$ entonces $s_2 \nearrow C$ y $\xi_s(s_1, C) \asymp_{s_1, s_2} \xi_s(s_2, C)$.

Decimos que dos estados $s_1, s_2 \in S$ son *fuertemente bisimilares*, denotado por $s_1 \sim_s s_2$, si existe una bisimulación fuerte que contenga al par $\langle s_1, s_2 \rangle$. □

La primera condición de la definición es la habitual en las definiciones de bisimulación fuerte: cada transición desde un estado debe simularse con una transición ejecutando la misma acción desde el otro. La segunda condición se utiliza para unir varias transiciones estocásticas que llegan a la misma clase de equivalencia. Obsérvese que no es necesario que $\xi_s(s_1, C)$ y $\xi_s(s_2, C)$ estén idénticamente distribuidas para cualquier tiempo; solamente es necesario que tomen los mismos valores para tiempos menores o iguales que el tiempo máximo de espera de ambos estados. Por último, debemos destacar que no incluimos los casos simétricos en los que los estados s_1 y s_2 intercambian los papeles, pues al pedir que $\mathcal{R}$ sea una relación de equivalencia, tenemos que esos casos están implícitamente incluidos, ya que $\mathcal{R}$ debe ser simétrica.

Ejemplo 4.7 Consideremos los estados s_2, s_{13} y s_{14} de la figura 4.1, junto con las variables aleatorias definidas en los ejemplos 4.2 y 4.3. Definimos el conjunto de estados C_1 y C_2 tales que $s_2', s_{13}', s_{14}' \in C_1$ y $s_{13}'', s_{14}'' \in C_2$. Considerando los valores de las variables aleatorias dadas

en el ejemplo 4.2 tenemos que $\xi_s(s_{13}, C_i) \asymp_{s_{13},s_{14}} \xi_s(s_{14}, C_i)$ para $i \in \{1,2\}$ (recuérdese que $maxW(s_{13}) = 2 = maxW(s_{14})$). Por tanto deducimos $s_{13} \sim_s s_{14}$.

Consideremos ahora las variables aleatorias definidas en el ejemplo 4.3. En este caso tenemos que $\xi_s(s_2, C_1) \asymp_{s_2,s_{13}} \xi_s(s_{13}, C_1) \asymp_{s_{13},s_{14}} \xi_s(s_{14}, C_1)$. Obsérvese que no es necesario calcular los valores de $\xi_s(s_j, C_2)$ para $j \in \{2, 13, 14\}$. Por lo tanto, tenemos que la siguiente equivalencia $s_2 \sim_s s_{13} \sim_s s_{14}$. □

A continuación presentaremos la noción de bisimulación débil para procesos estocásticos. Como en el caso de la bisimulación fuerte, impondremos la condición de alcanzabilidad (dada en la definición 4.4) antes de calcular ξ_s. Sin embargo, basándonos en la noción de bisimulación débil probabilística presentada en [BH97], las transiciones correspondientes tendrán que alcanzar el conjunto C^τ, definido por $C^\tau = \{s \mid \exists s' \in C \,:\, s \stackrel{\tau}{\Longrightarrow} s'\}$ en lugar de C.

Definición 4.8 Decimos que una relación de equivalencia $\mathcal{R}$ es una *bisimulación débil* sobre un sistema de transiciones etiquetado estocástico $P = (S, \rightarrow)$ si para cualquier par de estados $s_1, s_2 \in S$ tenemos que $s_1 \mathcal{R} s_2$ implica:

- Para cada $\alpha \in \mathtt{Act}_\tau$ tal que $s_1 \stackrel{\alpha}{\longrightarrow} s_1'$ existe algún s_2' tal que $s_2 \stackrel{\alpha}{\Longrightarrow} s_2'$ y $s_1' \mathcal{R} s_2'$.
- Para cada $C \in S/\mathcal{R}$, si $s_1 \stackrel{\tau}{\Longrightarrow} s_1' \nearrow C^\tau$ entonces existe s_2' tal que $s_2 \stackrel{\tau}{\Longrightarrow} s_2' \nearrow C^\tau$ y $\xi_s(s_1', C^\tau) \asymp_{s_1',s_2'} \xi_s(s_2', C^\tau)$.

Decimos que dos estados $s_1, s_2 \in S$ son *débilmente bisimilares*, denotado por $s_1 \sim_w s_2$, si existe una bisimulación débil que contenga al par $\langle s_1, s_2 \rangle$. □

Como en la definición de bisimulación fuerte, distinguimos entre transiciones estocásticas y transiciones correspondientes a acciones ordinarias. La primera condición de la definición anterior es la habitual para la bisimulación débil. La segunda condición comprueba si las variables aleatorias asociadas con los estados están idénticamente distribuidas para cualquier tiempo, hasta el correspondiente tiempo máximo de espera de los estados. En tal caso, y para abstraer (parcialmente) la transiciones internas, se permite la evolución ejecutando secuencias (posiblemente vacías) de acciones internas, antes y después de las variables aleatorias.

Ejemplo 4.9 Consideremos la figura 4.2. Los procesos s_{21} y s_{22} son equivalentes bajo ambas definiciones de bisimulación. Además, s_{23} es débilmente bisimilar, que no fuerte, a s_{21} y s_{22}. Por otro lado, s_{24} y s_{25} no son equivalentes bajo ninguna de las bisimulaciones definidas en este capítulo. Estos ejemplos muestran que nuestras nociones son extensiones conservativas del marco clásico no estocástico (véase el siguiente lema 4.11).

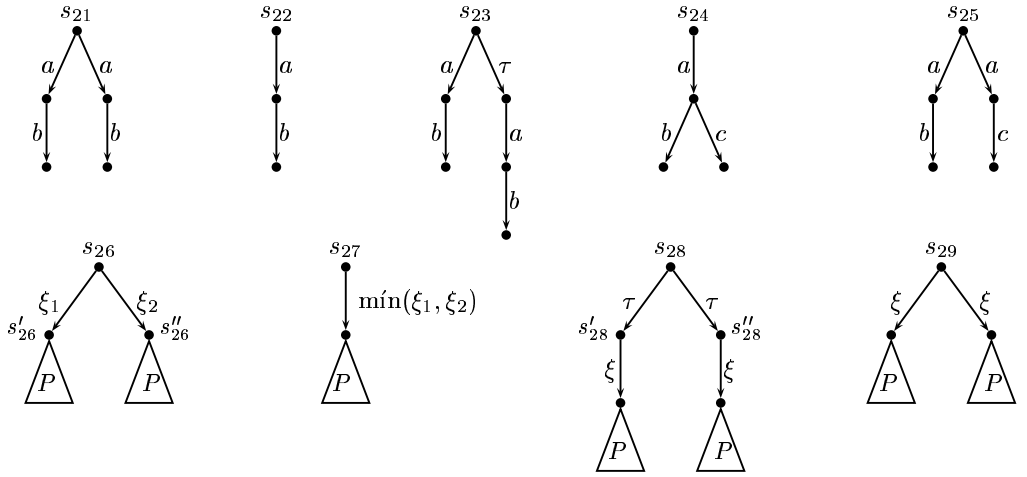


Figura 4.2: Ejemplos de procesos bisimilares fuertes/débiles.

Por otra parte, debemos *unir* las transiciones estocásticas que alcanzan la misma clase de equivalencia. Consideremos los estados s_{26} y s_{27}. Estos dos estados son bisimilares (fuertes y débiles) puesto que las dos transiciones estocásticas de s_{26} llegan a la misma clase de equivalencia. En este sentido, el retardo después del cual P está disponible será el definido por el mínimo de ambas variables aleatorias, es decir, el mismo que en s_{27}. Consideremos ahora el proceso s_2 de la figura 4.1 y el s_{28} de la figura 4.2. Estos procesos no son bisimilares fuertes, pues $s_2 \stackrel{\tau}{\not\longrightarrow}$, mientras que $s_{28} \stackrel{\tau}{\longrightarrow}$. Por el contrario, $s_2 \sim_w s_{28}$. Sin embargo, no identificamos los procesos s_{28} y s_{29}; esto es así, pues si s_{28} evoluciona ejecutando una transición τ, entonces s_{29} no puede simularla. Este resultado es deseable y es habitual en los modelos estocásticos, ya que la identificación de esos dos procesos sólo se justificaría si $\min(\xi, \xi)$ y ξ estuvieran idénticamente distribuidas, lo cual no es cierto. Obsérvese que si reemplazamos una cualquiera de las apariciones de ξ en s_{28} y s_{29} por un retardo cualquiera ψ, sigue cumpliéndose que los dos estados obtenidos, s_{28} y s_{29}, no son equivalentes. □

Las bisimulaciones fuerte y débil cumplen las propiedades habituales. La demostración es una adaptación de la dada en [Mil89], teniendo en cuenta que, en nuestro marco, las relaciones $\sim_s$ y $\sim_w$ también pueden definirse como la unión de todas las relaciones que son bisimulaciones fuertes y débiles, respectivamente.

Lema 4.10 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico. La relación de bisimulación fuerte (débil) definida en P, es decir $\sim_s$ (respectivamente $\sim_w$), es una relación de equivalencia en P. Además, $\sim_s$ (respectivamente $\sim_w$) es una bisimulación fuerte

(respectivamente débil) en P y es la mayor bisimulación fuerte (débil) en P.

Conviene destacar que si consideramos la restricción de estas dos nociones de bisimulación a sistemas etiquetados de transiciones sin evoluciones estocásticas obtenemos las nociones de bisimulación clásicas [Mil89]. Por otra parte, si nos restringimos a sistemas etiquetados de transiciones estocásticos donde las distribuciones son exponenciales, obtenemos las nociones de bisimulación definidas en [Her98]. Desgraciadamente, no es fácil comparar nuestras semánticas con aquellas propuestas anteriormente para marcos con distribuciones generales. Por ejemplo, [BG02b] utiliza una política de preselección, en lugar de una política de carreras, mientras que el mecanismo de los relojes definido en [DKB98a] es demasiado complejo para compararlo con nuestro modelo.

Lema 4.11 Restringidas a sistemas de transiciones etiquetados no estocásticos, las equivalencias $\sim_s$ y $\sim_w$ coinciden con las bisimulaciones fuerte y débil, respectivamente, definidas en [Mil89].
Demostración: La demostración es trivial, puesto que las condiciones impuestas a las acciones visibles e internas en las bisimulaciones fuerte y débil estocásticas son las mismas que las dadas en las definiciones clásicas de la bisimulación fuerte y débil, respectivamente. □

Lema 4.12 Restringidas a sistemas de transiciones etiquetados markovianos, las equivalencias $\sim_s$ y $\sim_w$ coinciden con las bisimulaciones fuerte y débil, respectivamente, definidas en [Her98].
Demostración: La demostración es sencilla teniendo en cuenta que cuando nos restringimos a distribuciones exponenciales tenemos que, para cualquier $s \in S$, el tiempo máximo de espera $maxW(s)$ es igual a 0 ó ∞. Además, para cualquier $C \subseteq S$ tenemos que $s \nearrow C$ puede quedar simplificado a $\exists s' \in C$ tal que $s \xrightarrow{\xi} s'$. □

La necesidad de definir una nueva semántica de bisimulación aparece porque la noción de bisimulación débil definida hasta ahora, presenta un serio problema. Concretamente, esta bisimulación trata las transiciones estocásticas casi como si fueran visibles. Obviamente, las transiciones estocásticas no pueden considerarse como meras transiciones internas puesto que llevan información temporal asociada que las transiciones internas no llevan. Sin embargo, como ya hemos comentado anteriormente, en algunas situaciones se deberían identificar ciertas secuencias de transiciones estocásticas.

A continuación, definiremos nuestra nueva noción de bisimulación, que denominaremos *bisimulación débil estocástica*. En lo que respecta a las acciones visibles e internas, nuestra

nueva relación se comporta como la bisimulación débil. En lo que respecta a las transiciones estocásticas, empezamos por calcular las variables aleatorias asociadas a cada secuencia de transiciones internas y estocásticas que nos conducen a cada clase. Sin embargo, a la hora de comparar estas informaciones no debemos tener en cuenta las clases correspondientes a todos los estados así accesibles, puesto que esto podría llevarnos a resultados no deseados. Por ejemplo, consideremos s_1 y s_2 en la figura 4.1. Pretendemos que s_1 y s_2 estén en la misma clase de equivalencia, y que s_1'' y s_2' estén también en la misma clase. Pero s_1' no pertenece a ninguna de estas dos clases, por lo que para que se tenga en efecto la equivalencia entre s_1 y s_2 no deberíamos tener en cuenta esa clase accesible desde s_1, pero no desde s_2. Del mismo modo, los estados s_{15}' y s_{17}', que aparecen en la misma figura serán equivalentes, a pesar de que existe un estado s_{15}'' alcanzable por s_{15}', pero no por s_{17}', lo que hace que estos dos estados no sean equivalentes débiles.

A continuación, introduciremos una serie de predicados por medio de los cuales definiremos las condiciones que han de cumplir los estados alcanzables tras la ejecución de transiciones internas y/o estocásticas, para que debamos o no unir las transiciones que los alcanzan y/o sus *continuaciones*.

Definición 4.13 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico, $\mathcal{R}$ una relación de equivalencia sobre S y $s \in S$. Definimos los siguientes predicados:

Clase de equivalencia. Denotamos por $[s]_{\mathcal{R}}$ a la *clase de equivalencia* inducida por $\mathcal{R}$ que contiene al estado s, es decir, al conjunto $[s]_{\mathcal{R}} = \{s' \in S \mid s\mathcal{R}s'\}$.

Estabilización de un estado. Decimos que $s' \in S$ es una *estabilización del estado* s con respecto a $\mathcal{R}$, denotado por $Stab_{\mathcal{R}}(s, s')$, si $s' \in [s]_{\mathcal{R}}$ y $s' \stackrel{\tau}{\not\rightarrow}$ y para cualquier $r \in S$ con $s \stackrel{\tau}{\Longrightarrow} r$ se tiene que $r \in [s]_{\mathcal{R}}$.

Estocásticamente no determinista. Decimos que s es *estocásticamente no determinista* con respecto a $\mathcal{R}$, y escribimos $Snd_{\mathcal{R}}(s)$, si existen $s_1, s_2 \in S$ tales que $s \nearrow [s_1]_{\mathcal{R}}$, $s \nearrow [s_2]_{\mathcal{R}}$ y $[s_1]_{\mathcal{R}} \neq [s_2]_{\mathcal{R}}$.

Estado alcanzable determinísticamente. Decimos que s *alcanza determinísticamente* s' ejecutando *transiciones estocásticas* con respecto a $\mathcal{R}$, y escribimos $\mathcal{D}_{\mathcal{R}}(s, s')$, si para cualquier r tal que $s \stackrel{\tau}{\Longrightarrow} r$ tenemos que $[s]_{\mathcal{R}} = [r]_{\mathcal{R}}$ y existe $s'' \in S$ tal que $s \stackrel{\tau}{\Longrightarrow} s'' \nearrow [s']_{\mathcal{R}}$, y para cualquier $r' \in S$ tal que $s'' \nearrow [r']_{\mathcal{R}}$ tenemos $[s']_{\mathcal{R}} = [r']_{\mathcal{R}}$.

Estocásticamente determinista. Decimos que s es *estocásticamente determinista* con respecto a $\mathcal{R}$, denotado por $Sd_{\mathcal{R}}(s)$, si para cualquier estado $r \in S$ con $s \stackrel{\tau}{\Longrightarrow} r$ se tiene

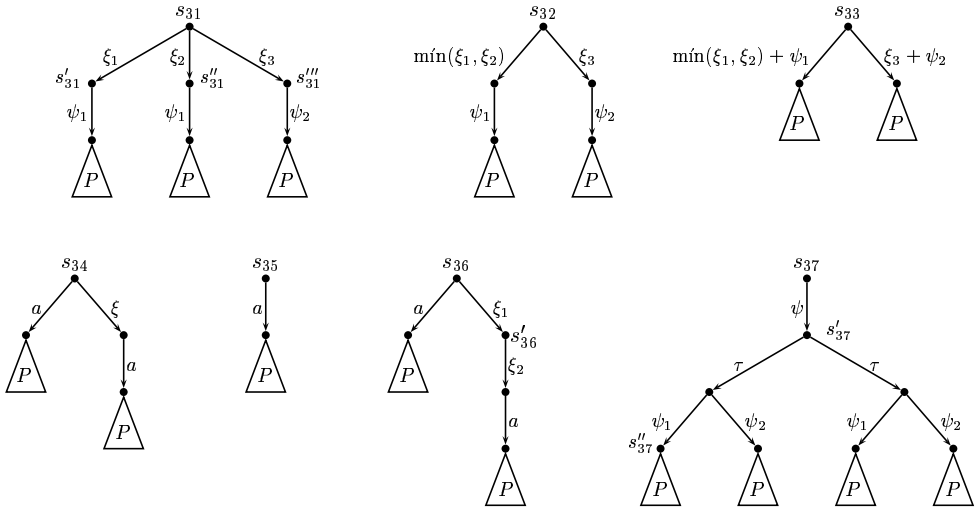


Figura 4.3: Ejemplos de procesos bisimilares débiles estocásticos.

$r \in [s]_{\mathcal{R}}$ y se cumple una de las dos condiciones siguientes:

- $s \stackrel{\mathcal{V}}{\not\Longrightarrow}$,
- $\mathcal{D}_{\mathcal{R}}(s, s')$, y existen estados $s_1, \ldots, s_n$, y variables aleatorias $\xi_1, \ldots, \xi_n$ tales que $s_n \in [s]_{\mathcal{R}}$. □

El predicado $Stab_{\mathcal{R}}(s, s')$ indica que, o bien s es estable o que desde s se puede alcanzar a un estado estable sin salir de su misma clase de equivalencia por medio de una secuencia de transiciones internas. Para calcular las variables aleatorias asociadas que se corresponden con el acceso desde la clase de s a cada clase accesible desde ella, utilizaremos cualquier representante estable s' de la clase de partida.

El predicado $Snd_{\mathcal{R}}(s)$ nos señala el caso en el que no deben unirse las *continuaciones* después de la ejecución de acciones estocásticas. Rechazamos de inmediato el estudio de las continuaciones si los estados alcanzados después de la ejecución de las transiciones estocásticas no pertenecen a la misma clase de equivalencia. Consideremos el estado s_{13} de la figura 4.1, junto con la definición de variables aleatorias que aparece en el ejemplo 4.3. Tenemos que *no* se cumple $Snd_{\mathcal{R}}(s_{13})$, puesto que no se tiene $s_{13} \nearrow [s''_{13}]_{\mathcal{R}}$. Si ambas variables aleatorias fueran factibles y los procesos P y Q no fueran equivalentes en $\mathcal{R}$, entonces se tendría $Snd_{\mathcal{R}}(s_{13})$.

Consideremos ahora s_{31} en la figura 4.3. En este caso, las continuaciones después de la ejecución de sus transiciones estocásticas iniciales, esto es, después de ξ_1, ξ_2 y ξ_3, no son

equivalentes a menos que ψ_1 y ψ_2 estén idénticamente distribuidas. De modo que s'_{31} y s'''_{31} no serán equivalentes. Si suponemos que las variables aleatorias ξ_1, ξ_2 y ξ_3 son tales que $maxW(s_{31}) > \text{mín}\{t \mid F_{\xi_i}(t) > 0\}$ para $i \in \{1,2,3\}$, entonces se considerará todas las transiciones estocásticas, con lo que tenemos $Snd_{\mathcal{R}}(s_{31})$.

Por contra, el predicado $\mathcal{D}_{\mathcal{R}}(s,s')$ representa la situación inversa: si todas las transiciones estocásticas que salen de un estado llegan a estados que pertenecen a la misma clase de equivalencia, entonces podemos unir todas las continuaciones. Por ejemplo, en la figura 4.3 tenemos $\mathcal{D}_{\mathcal{R}}(s'_{37}, s''_{37})$. Por tanto, si $\xi = \psi + (\psi_1 \oplus \psi_2)$ entonces s_{37} debería ser equivalente al estado s_2 que aparece en la figura 4.1.

El predicado $Sd_{\mathcal{R}}(s)$ se cumple cuando estado no puede ejecutar ninguna acción estocástica o bien tras la ejecución de una acción de este tipo llega siempre una misma clase y tras una serie de transiciones internas y estocásticas que salen de s llegan de nuevo a estados que pertenecen a la de partida. Este predicado sirve para saber si nos encontramos en estados cuyas ejecuciones son bucles infinitos de transiciones estocásticas o internas que nos llevan constantemente a la misma clase de equivalencia. Por ejemplo, en la figura 4.1 tenemos $Sd_{\mathcal{R}}(s_5)$.

La siguiente función representa el núcleo de la definición de nuestra noción de bisimulación débil estocástica. El punto clave consiste en calcular la suma de las continuaciones solamente cuando no se modifique la elección en el punto anterior.

Definición 4.14 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico, $\mathcal{R}$ una relación de equivalencia sobre S y $C \in S/\mathcal{R}$. Definimos la *variable aleatoria débil estocástica* asociada con la ejecución de transiciones estocásticas a partir del estado s_0 de S alcanzando un estado perteneciente a C,[3] denotado por $\xi_{wst}(s_0, C, \mathcal{R})$, como

$$\xi_{wst}(s_0,C,\mathcal{R}) = \begin{cases} unit & \text{si } s_0 \in C^\tau \wedge Sd_{\mathcal{R}}(s_0) \\ \xi_s(s,C^\tau) & \text{si } s_0 \notin C^\tau \wedge Stab_{\mathcal{R}}(s_0,s) \wedge \Big(s \xrightarrow{\text{Act}} \vee Snd_{\mathcal{R}}(s) \vee \exists s' \in S : \\ & \quad \big[\mathcal{D}_{\mathcal{R}}(s,s') \wedge \neg Sd_{\mathcal{R}}(s) \wedge \big(s' \xRightarrow{\text{Act}} \vee \nexists s'' \in S : \mathcal{D}_{\mathcal{R}}(s',s'')\big)\big]\Big) \\ \xi_s(s,[s']^\tau_{\mathcal{R}}) + \xi_{wst}(s',C,\mathcal{R}) & \text{si } s_0 \notin C^\tau \wedge Stab_{\mathcal{R}}(s_0,s) \wedge s \xrightarrow{\text{Act}}\!\!\!\!\!\!/\;\; \wedge \mathcal{D}_{\mathcal{R}}(s,s') \wedge \neg Sd_{\mathcal{R}}(s) \wedge \\ & \quad s' \xRightarrow{\text{Act}}\!\!\!\!\!\!/\;\; \wedge \exists s'' \in S : \mathcal{D}_{\mathcal{R}}(s',s'') \end{cases}$$

□

Recordemos que *unit* es la variable aleatoria concentrada en el 0, con lo que su función de distribución toma el valor 1 para cualquier $t \geq 0$. Por su parte, para cada clase A, A^τ

[3] Naturalmente no todas las clases resultarán así accesibles, por lo que si consideramos ξ_{wst} como una función, se trataría de una función partical.

representaba al conjunto $\{s \in S \mid \exists s' \in A : s \stackrel{\tau}{\Longrightarrow} s'\}$.

Estamos en el primer caso de la definición de la función si s_0 pertenece a la clase que estamos interesados, o se puede llegar a esta clase después de la ejecución de varias acciones internas y es estocásticamente determinista. Por ejemplo, para s_4 y s_5 de la figura 4.1 tenemos $\xi_{wst}(s_4, [s_4]_{\mathcal{R}}, \mathcal{R}) = \xi_{wst}(s_5, [s_5]_{\mathcal{R}}, \mathcal{R}) = \mathit{unit}$. Los otros dos casos de la definición corresponden al caso en el que s_0 no pertenece a C^{τ}.

Estamos en el segundo caso si existe un estado estable $s \in S$, alcanzado desde s_0 después de la ejecución de transiciones internas, de manera que, o bien s puede ejecutar una acción visible o bien no es cierto que todas las transiciones estocásticas iniciales factibles y sus continuaciones son todas estocásticas y nos conducen a una misma clase de equivalencia. En tal caso, nos limitamos a considerar las transiciones estocásticas iniciales que parten del estado s. En la figura 4.3 tenemos que $\xi_{wst}(s_{31}, [s'_{31}]_{\mathcal{R}}, \mathcal{R}) = \xi_{wst}(s_{31}, [s''_{31}]_{\mathcal{R}}, \mathcal{R}) = \xi_1 \oplus \xi_2$ y $\xi_{wst}(s_{31}, [s'''_{31}]_{\mathcal{R}}, \mathcal{R}) = \xi_3$.

Estamos en el tercer caso si todas las transiciones estocásticas factibles iniciales llevan a la misma clase de equivalencia y, sus continuaciones llevan a estados equivalentes. En tal caso, no deberíamos parar de *contar* después de las transiciones estocásticas iniciales, por lo que unimos estas transiciones utilizando ξ_s, para luego agregar las variables aleratorias asociadas a las continuaciones. Consideremos los estados s_{15} y s_{17} que aparecen en la figura 4.1. En este caso tenemos $\xi_{wst}(s'_{15}, [s'''_{15}]_{\mathcal{R}}, \mathcal{R}) = \xi_{wst}(s'_{17}, [s''_{17}]_{\mathcal{R}}, \mathcal{R}) = \psi_3 + (\rho_3 \oplus \rho_4)$.

Por último, resulta obvio comprobar que los tres casos son mutuamente excluyentes.

En las definiciones previas de bisimulación hemos impuesto la condición $s \nearrow C$ antes de definir $\xi_s(s, C)$, para asegurar que la función quedaba bien definida. Pero en este caso, esta condición no es suficiente para asegurar que $\xi_{wst}(s, C, \mathcal{R})$ quedar definida. Existen dos razones por las cuales esta función podría estar indefinida: o bien no existe un camino para llegar a C desde s, o existe un camino pero éste es *ilegal*, donde entendemos que un camino $p \equiv s \stackrel{\xi_1}{\Longrightarrow} s_1 \ldots s_{n-1} \stackrel{\xi_n}{\Longrightarrow} s_n$ es *legal* si los estados del camino cumplen las condiciones de cualquiera de las tres cláusulas de la definición de $\xi_{wst}(s, C, \mathcal{R})$. Desgraciadamente la definición formal de camino legal no puede ser otra cosa que el desdoblamiento reiterado de la condición que define el dominio de la función (recursiva) ξ_{wst}. Intuitivamente estos estados son justamente los que en los ejemplos previos decidiamos que no debíamos tener en cuenta por ser meros intermediarios en el proceso, como s'_1 en el caso de s_1 en la figura 4.1. Damos a continuación la defición formal.

Definición 4.15 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico, $C \subseteq S$, $\mathcal{R}$ una relación de equivalencia en S y $s, s_i \in S, \xi_i \in \mathcal{V}$ con $1 \leq i \leq n$. Decimos que

$p \equiv s \stackrel{\xi_1}{\Longrightarrow} s_1 \stackrel{\xi_2}{\Longrightarrow} s_2 \cdots s_{n-1} \stackrel{\xi_n}{\Longrightarrow} s_n$ es un *camino legal para alcanzar* C con respecto a $\mathcal{R}$, denotado por $legal_{\mathcal{R}}(p, C)$, si se cumple alguna de las condiciones siguientes:

1. $s \in C^{\tau}$, $Sd_{\mathcal{R}}(s)$.
2. $s \notin C^{\tau}$, $n = 1$, existe $s_0 \in S$ tal que $Stab_{\mathcal{R}}(s, s_0)$, $fact_{maxW(s_0)}(\xi_1)$, $s_1 \in C^{\tau}$ y se cumple una de las siguientes condiciones:
   - $s_0 \xrightarrow{\texttt{Act}}$
   - $Snd_{\mathcal{R}}(s_0)$
   - $s_1 \stackrel{\texttt{Act}}{\Longrightarrow}$
   - No existe s'' tal que $\mathcal{D}_{\mathcal{R}}(s_1, s'')$
3. $s \notin C^{\tau}$, existe $s_0 \in S$ tal que $Stab_{\mathcal{R}}(s, s_0)$, para cada $0 \leq i \leq n-1$ tenemos $s_i \xrightarrow{\texttt{Act}}\!\!\!\!\!/\;\;$, $\mathcal{D}_{\mathcal{R}}(s_i, s_{i+1})$, $\neg Sd_{\mathcal{R}}(s_i)$ y por último $s_n \in C^{\tau}$ y $Sd_{\mathcal{R}}(s_n)$.

Decimos que C es *estocásticamente alcanzable* desde s con respecto a $\mathcal{R}$, denotado por $s \not\Rightarrow_{\mathcal{R}} C$, si existe un camino p desde s que cumpla $legal_{\mathcal{R}}(p, C)$. □

Debemos remarcar que si existe un camino legal p desde s a C entonces todos los caminos desde s a estados de C cuyas transiciones estocásticas son todas factibles, son también legales, pues la condición para ser legal afecta a todas las transiciones estocásticas que cada estado puede hacer.

Lema 4.16 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico, $C \subseteq S$, $\mathcal{R}$ una relación de equivalencia en S y $s \in S$. Si $s \not\Rightarrow_{\mathcal{R}} C$ entonces la función $\xi_{wst}(s, C, \mathcal{R})$ está bien definida.

Demostración: Si $s \not\Rightarrow_{\mathcal{R}} C$ es porque existen estados $s_1, \ldots, s_n \in S$, variables aleatorias $\xi_1, \ldots, \xi_n \in \mathcal{V}$ y un camino $p \equiv s \stackrel{\xi_1}{\Longrightarrow} s_1 \stackrel{\xi_2}{\Longrightarrow} s_2 \cdots s_{n-1} \stackrel{\xi_n}{\Longrightarrow} s_n$ tales que $legal_{\mathcal{R}}(p, C)$. Vamos a probar que $\xi_{wst}(s, C, \mathcal{R})$ está bien definida por inducción sobre el número de veces que aplicamos la definición. Si p es un camino legal debe cumplir una de las tres condiciones de la definición 4.15. Si cumple alguna de las dos primeras condiciones, nos encontramos en el caso base, puesto que en esos casos se aplica una sola vez la definición. De modo que si p cumple la primera de las condiciones, la función $\xi_{wst}(s, C, \mathcal{R})$ toma el valor *unit*. En cambio, si cumple la segunda condición entonces $\xi_{wst}(s, C, \mathcal{R}) = \xi_w(s_0, C)$ para un cierto s_0 tal que $Stab_{\mathcal{R}}(s, s_0)$. Por último, supongamos que p cumple la tercera condición y el camino tiene longitud $n+1$. En tal caso, aplicamos la definición de $\xi_{wst}(s, C, \mathcal{R})$ $n+1$ veces. Por hipótesis de inducción podemos suponer que la función está definida en los n primeros pasos. Además, en la última aplicación de la definición tenemos $\xi_{wst}(s_n, C, \mathcal{R}) = unit$ puesto que $s_n \in C^{\tau}$ y $Sd_{\mathcal{R}}(s_n)$. □

Una vez definido el predicado de alcanzabilidad, presentamos la noción de bisimulación débil estocástica:

Definición 4.17 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico y sea $\mathcal{R}$ una relación de equivalencia definida sobre S. Decimos que $\mathcal{R}$ es una *bisimulación débil estocástica* en S si para todo par de estados $s_1, s_2 \in S$ con $s_1 \mathcal{R} s_2$ se tiene:

- Para cada $\alpha \in \mathtt{Act}_\tau$ tal que $s_1 \xrightarrow{\alpha} s_1'$ existe algún s_2' tal que $s_2 \stackrel{\alpha}{\Longrightarrow} s_2'$ y $s_1' \mathcal{R} s_2'$.
- Para cada $C \in S/\mathcal{R}$ tal que $s_1 \stackrel{\tau}{\Longrightarrow} s_1' \stackrel{\tau}{\not\rightarrow}$, y $s_1' \not\Rightarrow_{\mathcal{R}} C$ existe algún s_2' tal que $s_2 \stackrel{\tau}{\Longrightarrow} s_2' \not\Rightarrow_{\mathcal{R}} C$ y $\xi_{wst}(s_1', C, \mathcal{R}) \asymp_{s_1', s_2'} \xi_{wst}(s_2', C, \mathcal{R})$.

Decimos que dos estados $s_1, s_2 \in S$ son *débilmente estocásticamente bisimilares*, denotado por $s_1 \sim_{wst} s_2$, si existe una bisimulación débil estocástica que contenga al par (s_1, s_2). De modo que definimos $\sim_{wst}$ como la unión de todas las bisimulaciones débiles estocásticas. □

La diferencia entre $\sim_w$ y $\sim_{wst}$ viene dada por la definición de ξ_{wst} y la nueva condición $(s \not\Rightarrow_{\mathcal{R}} C)$ para asegurar que calculamos $\xi_{wst}(s, C, \mathcal{R})$ sólo si existe un camino legal desde s para alcanzar C.

Ejemplo 4.18 Consideremos los estados s_{31}, s_{32} y s_{33} de la figura 4.3. Supongamos que todas las transiciones estocásticas son factibles con respecto al estado del que provienen. Tenemos que s_{31} y s_{32} son débilmente bisimilares y (por tanto) débilmente estocásticamente bisimilares. Por el contrario, s_{33} no es equivalente a ellos. La clave está en que mientras la elección que aparece en s_{31} y s_{32} se produce entre ξ_3 y el mínimo de ξ_1 y ξ_2, en el caso de s_{33} la elección quedaría *deformada* por ψ_1 y ψ_2.

Consideremos ahora los estados s_{34}, s_{35} y s_{36}. Los dos primeros son débilmente estocásticamente bisimilares (pero no débilmente bisimilares). Sin embargo, s_{34} y s_{36} no son débilmente estocásticamente bisimilares porque s_{34} siempre es capaz de ejecutar a mientras que esto no es cierto para s_{36}. Ello es así puesto que si se ejecuta la transición estocástica $s_{36} \xrightarrow{\xi_1} s_{36}'$ entonces s_{36}' no es capaz de ejecutar la acción a hasta que concluya el retardo inducido por ξ_2. □

4.3. Propiedades de la bisimulación débil estocástica

En primer lugar vamos a probar que esta nueva bisimulación cumple las propiedades habituales de las bisimulaciones:

Lema 4.19 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico. La bisimulación débil estocástica definida sobre P (es decir, la relación $\sim_{wst}$) es una relación de equivalencia sobre P. Además, $\sim_{wst}$ es una bisimulación débil estocástica y es la mayor bisimulación débil estocástica en P.
Demostración: Sea $\{\mathcal{R}_i\}_{i \in I}$ el conjunto de todas las bisimulaciones débiles estocásticas, con I un conjunto numerable de índices. Dicho conjunto es numerable puesto que nuestros sistemas de transiciones están constituidos por un conjunto finito de estados.

Como en la definición de bisimulación se pide explícitamente que la relación sea de equivalencia, tenemos que todas las relaciones serán de equivalencia y la unión de relaciones de equivalencia también lo es.

A continuación, vamos a probar que $\bigcup_{i \in I} \mathcal{R}_i$ es también una bisimulación débil estocástica. Para cualquier par de estados $s_1, s_2 \in S$ tales que $(s_1, s_2) \in \bigcup_{i \in I} \mathcal{R}_i$ debemos comprobar que se cumplen las condiciones de la definición de bisimulación débil estocástica. Como $(s_1, s_2) \in \bigcup_{i \in I} \mathcal{R}_i$, en particular existe un $j \in I$ tal que $(s_1, s_2) \in \mathcal{R}_j$, que a su vez es una bisimulación débil estocástica. Por tanto, s_1 y s_2 cumplen las condiciones de la definición. En conclusión, $\bigcup_{i \in I} \mathcal{R}_i$ es una bisimulación débil estocástica.

Y que es la mayor bisimulación es consecuencia inmediata de su definición. □

A continuación, presentaremos un resultado que relaciona las tres bisimulaciones presentadas en esta sección. Para su demostración necesitamos una serie de resultados auxiliares. El primero de los lemas prueba que si dos estados cualesquiera son débilmente bisimilares o ambos son estocásticamente no deterministas o ninguno de ellos lo es.

Lema 4.20 Sean $P = (S, \rightarrow)$ un sistema etiquetado de transiciones estocástico y s_1, s_2 estados estables tales que $s_1 \sim_w s_2$. Entonces $Snd_{\sim_w}(s_1)$ sii $Snd_{\sim_w}(s_2)$.
Demostración: $Snd_{\sim_w}(s_1)$ si y sólo si existen $s, s' \in S$ tales que $s_1 \nearrow [s]_{\sim_w}$, $s_1 \nearrow [s']_{\sim_w}$ y $[s]_{\sim_w} \neq [s']_{\sim_w}$. Partiendo de $s_1 \sim_w s_2$, tenemos que $s_1 \nearrow [s]_{\sim_w}$ implica $\xi_w(s_2, [s]^{\tau}_{\sim_w}) = \xi_w(s_1, [s]^{\tau}_{\sim_w})$ y, por tanto $s_2 \nearrow [s]_{\sim_w}$ (y de forma simétrica $s_2 \nearrow [s']_{\sim_w}$)). Con lo que concluimos que $Snd_{\sim_w}(s_2)$. □

El siguiente lema prueba que dos procesos débilmente bisimilares alcanzan determinísticamente tras ejecutar transiciones estocásticas estados de las mismas clase de equivalencia respecto de la bisimulación débil.

Lema 4.21 Sean $P = (S, \rightarrow)$ un sistema etiquetado de transiciones estocástico, y s_1, s_2 estados estables tales que $s_1 \sim_w s_2$. Entonces existe s_1' tal que $\mathcal{D}_{\sim_w}(s_1, s_1')$ sii existe s_2' tal que $\mathcal{D}_{\sim_w}(s_2, s_2')$ y $s_1' \sim_w s_2'$.

Demostración: Si existe s'_1 tal que $\mathcal{D}_{\sim_w}(s_1, s'_1)$, entonces para todo t tal que $s_1 \stackrel{\tau}{\Longrightarrow} t$, $[s_1]_{\sim_w} = [t]_{\sim_w}$ y existen además $s'_1, s''_1 \in S$ tales que $s_1 \stackrel{\tau}{\Longrightarrow} s''_1$, $s''_1 \nearrow [s'_1]_{\sim_w}$ y para cualquier $r \in S$ tal que $s''_1 \nearrow [r]_{\sim_w}$, tendremos $[r]_{\sim_w} = [s'_1]_{\sim_w}$. Como $s_1 \sim_w s_2$, para cualquier t tal que $s_2 \stackrel{\tau}{\Longrightarrow} t$ tenemos $[t]_{\sim_w} = [s_1]_{\sim_w} = [s_2]_{\sim_w}$ y existe $s''_2 \in [s''_1]_{\sim_w}$ tal que $s_2 \stackrel{\tau}{\Longrightarrow} s''_2$ y $\xi_w(s''_1, C^\tau) \asymp_{s''_1, s''_2} \xi_w(s''_2, C^\tau)$, para cualquier $C \in S/\sim_w$ tal que $s''_1 \nearrow C^\tau$. Y como quiera que $s''_1 \nearrow [s'_1]_{\sim_w}$ entonces también $s''_2 \nearrow [s'_1]_{\sim_w}$. Por tanto existe $s'_2 \in [s'_1]_{\sim_w}$ tal que $s''_2 \nearrow [s'_2]_{\sim_w}$.

Ahora debemos probar $\mathcal{D}_{\sim_w}(s_2, s'_2)$. Para cualquier $r' \in S$ tal que $s''_2 \nearrow [r']_{\sim_w}$ existe algún r tal que $s''_1 \nearrow [r]_{\sim_w}$ y $[r]_{\sim_w} = [r']_{\sim_w}$. De donde concluimos $[r']_{\sim_w} = [s'_2]_{\sim_w}$, puesto que $[r]_{\sim_w} = [s'_1]_{\sim_w}$ y $[s'_1]_{\sim_w} = [s'_2]_{\sim_w}$. □

En el siguiente lema se indica que si dos procesos son débilmente bisimilares, los procesos estables a partir de los cuales se ejecutan transiciones estocásticas para llegar a una cierta clase son también débilmente bisimilares.

Lema 4.22 Sean $P = (S, \rightarrow)$ un sistema etiquetado de transiciones estocástico, $s_1, s_2 \in S$ tales que $s_1 \sim_w s_2$ y $C \in S/\sim_w$. Si existe $s'_1 \in S$ tal que $s_1 \stackrel{\tau}{\Longrightarrow} s'_1 \stackrel{\tau}{\not\longrightarrow}$ y $s'_1 \nearrow C^\tau$, entonces existe $s'_2 \in S$ tal que $s_2 \stackrel{\tau}{\Longrightarrow} s'_2 \nearrow C^\tau$, $\xi_w(s'_1, C^\tau) \asymp_{s'_1, s'_2} \xi_w(s'_2, C^\tau)$ y $s'_1 \sim_w s'_2$.
Demostración: Como se cumple que $s_1 \sim_w s_2$ existe un $s'_2 \in S$ tal que $s_2 \stackrel{\tau}{\Longrightarrow} s'_2 \nearrow C^\tau$ y $\xi_w(s'_1, C^\tau) \asymp_{s'_1, s'_2} \xi_w(s'_2, C^\tau)$. Demostraremos por reducción al absurdo que se cumple también la última condición del enunciado. Supongamos que todos los estados s'_2 cumplen todas las condiciones anteriores verifican $s'_1 \not\sim_w s'_2$. Esto podría suceder en principio por no cumplirse cualquiera de las dos condiciones en la definición de $\sim_w$. Ahora bien, el segundo caso es imposible pues las restantes cláusulas de la condición del enunciado afirman exactamente que esto es así. En el primer caso, tendríamos alguna acción $a \in \mathtt{Act}$ tal que $s'_1 \stackrel{a}{\Longrightarrow} s''_1$ y o bien $s'_2 \stackrel{a}{\not\Longrightarrow}$, o bien para todo s''_2 tal que $s_2 \stackrel{a}{\Longrightarrow} s''_2$ tenemos que $s''_1 \not\sim_w s''_2$. En cualquiera de las dos situaciones llegamos a la contradicción de que $s_1 \not\sim_w s_2$. □

Por último presentamos el resultado que relaciona las tres nociones de bisimulación definidas hasta el momento:

Teorema 4.23 Sea $P = (S, \rightarrow)$ un sistema de transiciones etiquetado estocástico y sean $s_1, s_2 \in S$. Entonces $s_1 \sim_s s_2$, implica $s_1 \sim_w s_2$. Además, si $s_1 \sim_w s_2$, entonces $s_1 \sim_{wst} s_2$.
Demostración: La demostración del primer resultado es similar a la dada en [Her98]. En cuanto a la inclusión de $\sim_w$ en $\sim_{wst}$, veremos que $\sim_w$ es una bisimulación débil estocástica.

En lo que se refiere a las acciones incluidas en $\mathtt{Act}_\tau$, la definición coincide en ambos casos.

En lo que se refiere a acciones estocásticas, tenemos que si $s_1 \sim_w s_2$, entonces para cada $C \in S/\sim_w$, como se cumple $s_1 \stackrel{\tau}{\Longrightarrow} s'_1 \stackrel{\tau}{\not\longrightarrow}$ y $s'_1 \nearrow C^\tau$ entonces existe un s'_2 tal que $s_2 \stackrel{\tau}{\Longrightarrow}$

$s_2' \nearrow C^\tau$ y $\xi_w(s_1', C^\tau) \asymp_{s_1', s_2'} \xi_w(s_2', C^\tau)$. Aplicando el lema 4.22, podemos tomar s_1' y s_2' tales que $s_1' \sim_w s_2'$. A partir de este punto, tenemos que demostrar que para cualquier $C \in S/\sim_w$ se tiene $\xi_{wst}(s_1', C, \sim_w) \asymp_{s_1', s_2'} \xi_{wst}(s_2', C, \sim_w)$ si $s_1' \not\Uparrow_{\sim_w} C$ y $s_2' \not\Uparrow_{\sim_w} C$. Lo demostraremos por inducción sobre el número de veces que hemos de desdoblar la definición recursiva de ξ_{wst} para obtener un valor explícito de $\xi_{wst}(s_1', C, \sim_w)$.

El *caso base* se corresponde con el primer y segundo casos de la definición de ξ_{wst}, en los cuales se utiliza una única vez la definición de ξ_{wst}. El *paso inductivo* se corresponde con el tercer caso que es donde encontramos la llamada recursiva. Sin embargo, en lugar de distinguir explícitamente entre caso base y caso inductivo, vamos a ir considerando los casos en los que se encuentran los estados s_1, s_2 y sus continuaciones, para recorrer de este modo todas las posibilidades.

Si $s_1' \in C^\tau$ entonces $s_2' \in C^\tau$, pues ambos estados son equivalentes. Entonces tenemos dos posibilidades. Si tenemos $Sd_{\sim_w}(s_1')$ y $Sd_{\sim_w}(s_2')$, entonces ambos estados cumplen $\xi_{wst}(s_i, C, \sim_w) = unit$. Segunda posibilidad: ni $Sd_{\sim_w}(s_1')$ ni $Sd_{\sim_w}(s_2')$ se cumplen. En tal caso no se calcula $\xi_{wst}(s_i', C, \sim_w)$ ya que $s_i' \not\Uparrow_{\sim_w} C$ no se cumple.

Consideremos que $s_1', s_2' \notin C^\tau$ y que existen $s_1'', s_2'' \in S$ tales que $Stab_{\sim_w}(s_1', s_1'')$ y $Stab_{\sim_w}(s_2', s_2'')$. Entonces $s_2' \nearrow C$ implica $s_2' \stackrel{\tau}{\not\longrightarrow}$. También tenemos que $s_1' \stackrel{\tau}{\not\longrightarrow}$, por lo que $s_1'' = s_1'$ y $s_2'' = s_2'$. Ahora tenemos las siguientes posibilidades:

- Si $s_1' \stackrel{\texttt{Act}}{\Longrightarrow}$, entonces tenemos trivialmente que $s_2' \stackrel{\texttt{Act}}{\Longrightarrow}$. De modo que ambos estados cumplen una de las condiciones del segundo caso de la definición de ξ_{wst}. Por tanto, $\xi_{wst}(s_1', C, \sim_w) \asymp_{s_1', s_1'} \xi_w(s_2', C) \asymp_{s_1', s_2'} \xi_w(s_2', C) \asymp_{s_2', s_2'} \xi_{wst}(s_2', C, \sim_w)$.

- Si $Snd(s_1')$, entonces $Snd(s_2')$ (aplicando el lema 4.20). Como en el caso anterior, ambos estados se encuentran en el segundo caso de la definición de ξ_{wst}. Y por tanto tenemos $\xi_{wst}(s_1', C, \sim_w) \asymp_{s_1', s_1'} \xi_w(s_2', C) \asymp_{s_1', s_2'} \xi_w(s_2', C) \asymp_{s_2', s_2'} \xi_{wst}(s_2', C, \sim_w)$.

- Si existe r_1 tal que $\mathcal{D}_{\sim_w}(s_1', r_1)$ entonces, aplicando el lema 4.21, tenemos que existe r_2 tal que $\mathcal{D}_{\sim_w}(s_2', r_2)$ y $r_1 \sim_w r_2$. Por lo que debemos comprobar qué ocurre con las continuaciones. Existen dos posibilidades:

  - Si $r_1 \stackrel{\texttt{Act}}{\Longrightarrow}$ entonces $r_2 \stackrel{\texttt{Act}}{\Longrightarrow}$. En este caso, ambos estados se encuentran de nuevo en el segundo caso.

  - Si no existe $r_1' \in S$ tal que $\mathcal{D}_{\sim_w}(r_1, r_1')$, entonces ello sucederá por una de las tres causas siguientes:

    1. Existe t_1 tal que $r_1 \stackrel{\tau}{\Longrightarrow} t_1$ y $[r_1]_{\sim_w} \neq [t_1]_{\sim_w}$. Aplicando la definición de $\sim_w$,

existe t_2 tal que $r_2 \stackrel{\tau}{\Longrightarrow} t_2$ y $[r_2]_{\sim_w} \neq [t_2]_{\sim_w}$. Por tanto tenemos que no existe ningún $r_2' \in S$ tal que $\mathcal{D}_{\sim_w}(r_2, r_2')$.

2. Para todo t_1 tal que $r_1 \stackrel{\tau}{\Longrightarrow} t_1$, $[r_1]_{\sim_w} = [t_1]_{\sim_w}$, pero no existe ningún $r_1' \in S$ tal que $r_1 \stackrel{\tau}{\Longrightarrow} t_1 \nearrow [r_1']_{\sim_w}$. Entonces para cualquier t_2 tal que $r_2 \stackrel{\tau}{\Longrightarrow} t_2$ tenemos $[r_2]_{\sim_w} = [t_2]_{\sim_w}$. Supongamos que existe r_2' tal que $r_2 \stackrel{\tau}{\Longrightarrow} t_2 \nearrow [r_2']_{\sim_w}$. Ello en realidad no puede suceder porque $r_1 \sim_w r_2$.
3. Para todo t_1 tal que $r_1 \stackrel{\tau}{\Longrightarrow} t_1$ con $[r_1]_{\sim_w} = [t_1]_{\sim_w}$ existen $r_1', t_1 \in S$ tales que $r_1 \stackrel{\tau}{\Longrightarrow} t_1 \nearrow [r_1']_{\sim_w}$. Pero existe también $t_1' \in S$ tal que $t_1 \nearrow [t_1']_{\sim_w}$ y $[r_1']_{\sim_w} \neq [t_1']_{\sim_w}$. En este caso para todo t_2 tal que $r_2 \stackrel{\tau}{\Longrightarrow} t_2$, $[r_2]_{\sim_w} = [t_2]_{\sim_w}$ existen r_2', t_2' tales que $r_2 \stackrel{\tau}{\Longrightarrow} t_2' \nearrow [r_2']_{\sim_w}$, y existe también $t_2'' \in S$ tal que $t_2' \nearrow [t_2'']_{\sim_w}$ y $[r_2']_{\sim_w} \neq [t_2'']_{\sim_w}$.

Por tanto, tenemos que no existe $r_2' \in S$ tal que $\mathcal{D}_{\sim_w}(r_2, r_2')$. En este caso, ambos estados se encuentran en el segundo caso de la definición de ξ_{wst}. Por tanto tenemos $\xi_{wst}(s_1', C, \sim_w) \asymp_{s_1', s_2'} \xi_{wst}(s_2', C, \sim_w)$.

- Si $s_1' \stackrel{\texttt{Act}}{\not\longrightarrow}$ entonces $s_2' \stackrel{\texttt{Act}}{\not\longrightarrow}$ pues $s_1' \sim_w s_2'$. Además, si existe $r_1 \in S$ tal que $\mathcal{D}_{\sim_w}(s_1', r_1)$ entonces, aplicando el lema 4.21, tenemos que también existe $r_2 \in S$ tal que $\mathcal{D}_{\sim_w}(s_2', r_2)$ y $r_1 \sim_w r_2$. Por lo que, si $r_1 \stackrel{\texttt{Act}}{\Longrightarrow}$ entonces $r_2 \stackrel{\texttt{Act}}{\Longrightarrow}$. Por otra parte, si existe $r_1' \in S$ tal que $\mathcal{D}_{\sim_w}(r_1, r_1')$, aplicando el mismo razonamiento que en los casos anteriores, tenemos que existe $r_2' \in S$ tal que $\mathcal{D}_{\sim_w}(r_2, r_2')$. Entonces, ambos estados se encuentran en el tercer caso de la definición de ξ_{wst}, por lo que tenemos $\xi_{wst}(s_1', C, \sim_w) = \xi_w(s_1', [r_1]^{\tau}_{\sim_w}) + \xi_{wst}(r_1, C, \sim_w)$ y $\xi_{wst}(s_2', C, \sim_w) = \xi_w(s_2', [r_2]^{\tau}_{\sim_w}) + \xi_{wst}(r_2, C, \sim_w)$. Pero tenemos $\xi_w(s_1', [r_1]^{\tau}_{\sim_w}) \asymp_{s_1', s_2'} \xi_w(s_2', [r_2]^{\tau}_{\sim_w})$ y por hipótesis de inducción $\xi_{wst}(r_1, C, \sim_w) \asymp_{r_1, r_2} \xi_{wst}(r_2, C, \sim_w)$ ya que $r_1 \sim_w r_2$.

□

Corolario 4.24 $\sim_s \subsetneq \sim_w \subsetneq \sim_{wst}$.

4.4. Conclusiones

En este capítulo hemos estudiado una semántica de bisimulación para una clase de procesos estocásticos con distribuciones generalizadas. Hemos mostrado que nuestra noción de bisimulación débil estocástica resuelve algunos problemas que causaban distinciones poco intuitivas que se daban bajo las nociones previas de bisimulación.

Se ha trabajado con un modelo con elecciones no deterministas en el que la resolución de elecciones entre acciones estocásticas se realizaba por medio de una política de carre-

ras. En este caso, la noción de bisimulación débil estocástica tiene una definición bastante compleja, pese a ser intuitivamente simple. Como ya se comentó a lo largo del capítulo, el principal problema de esta noción es la necesidad de *preocuparse* de las continuaciones antes de ejecutar las transiciones estocásticas iniciales. Ello es debido a que cuando computamos las variables aleatorias que se ejecutan para llegar a una cierta clase, existen estados en los que no queremos parar, pues deseamos que se ejecuten en un solo paso las transiciones estocásticas que parten del estado de partida. Sin embargo, para definir el correspondiente conjunto de estados hemos tenido que definir ξ_{wst} de forma bastante compleja.

Una cuestión interesante que dará lugar a la continuación de nuestra investigación es la extensión de los presentes resultados a lenguajes más complejos. En concreto, nos gustaría encontrar la forma más adecuada de introducir un operador de composición paralela en el marco estudiado en el capítulo. Como ya hemos indicado en la introducción de esta tesis, definir un operador de paralelo en el contexto de las distribuciones generales es bastante más complicado que en el caso de las álgebras de procesos markovianas. Sin embargo, existen ya algunos modelos en los que coexisten la composición paralela y las distribuciones generales. Podríamos tratar de extender nuestro modelo siguendo las ideas presentadas para el lenguaje `NMSPA`. Desgraciadamente, esta solución produciría una definición de la semántica operacional que genera sistemas de transiciones infinitos. Consideramos más interesante extender el lenguaje basándonos en el trabajo [BG02b], donde se utiliza un álgebra de procesos con operador de composición paralela y distribuciones generalizadas. La idea básica de este trabajo es la división de la ejecución de las transiciones estocásticas en dos partes: el *inicio* de la transición y la *terminación* de la misma. En el capítulo 6 utilizaremos las ideas de ese trabajo para definir nociones de bisimulación fuerte y débil para un modelo algo más complejo que el visto en este capítulo. El problema se presenta cuando intentamos definir nuestra semántica de bisimulación débil estocástica en este nuevo marco, puesto que tendríamos que ir juntando todas las acciones estocásticas que se fueran ejecutando para calcular la variable aleatoria asociada a cada ejecución, lo cual parece complejo en exceso.

Capítulo 5

Una semántica de testing para procesos estocásticos

En este capítulo presentamos una semántica de *testing* para procesos estocásticos. La teoría aquí desarrollada trata procesos cuyas distribuciones de probabilidad no están restringidas a ser exponenciales. Para definir esta semántica de *testing* calculamos la probabilidad con la que un proceso pasa una prueba antes de que haya transcurrido una cierta cantidad de tiempo. De ese modo, dos procesos serán equivalentes si pasan con la misma probabilidad una prueba al transcurrir una misma cantidad de tiempo. La idea principal subyacente en la definición de nuestra semántica aparece ya reflejada en las diferentes nociones de bisimulación débil estocástica que estudiamos en el capítulo anterior. Intuitivamente, tendremos que volver a *unir* de forma adecuada las variables aleatorias asociadas con las computaciones que puede ejecutar la composición del proceso y la prueba. La combinación de los valores de las variables aleatorias asociadas a cada computación nos dará la probabilidad deseada. Parte de los resultados de este capítulo aparecen recogidos en [LN01].

5.1. Introducción

El marco sobre el que definimos nuestra semántica de *testing* se corresponde con una extensión probabilística del modelo definido en el capítulo anterior, de modo que las distribuciones de probabilidad no van a estar restringidas a ser exponenciales y los procesos podrán ejecutar tanto acciones ordinarias (acciones visibles o internas) como acciones estocásticas que representarán retardos. Adicionalmente, nuestro operador de elección incluirá un parámetro indicando el peso otorgado a cada uno de sus argumentos, y la selección entre

las distintas alternativas que aparezcan en cada elección se realizará utilizando una *política de preselección*.

En cuanto a las pruebas, supondremos que no pueden ejecutar transiciones internas. Imponemos esta restricción dado que la semántica de *testing* probabilística produce resultados muy extraños cuando las pruebas tienen la posibilidad de ejecutar este tipo de acciones. Por ejemplo, si consideramos un álgebra de procesos probabilísticos con un único operador de elección (probabilístico) y permitimos que las pruebas ejecuten acciones internas, obtendremos que los procesos $\tau;a;stop$ y $a;stop$ no son equivalentes bajo la correspondiente semántica de *testing* probabilística. Este problema se estudia detenidamente en [CDSY99], mostrando la diferencia entre semánticas de *testing* probabilísticas, con y sin acciones internas en las pruebas.

Como es habitual, definiremos la interacción entre procesos y pruebas como la composición paralela de ambos, sincronizando en todas las acciones visibles. Esta composición produce un (multi)conjunto de computaciones a partir de las cuales extraeremos la información adecuada para definir el paso de una prueba. En contraste con el marco clásico, en el que la respuesta tras una prueba consiste simplemente en indicar sí la prueba se pasó con éxito o no, nosotros necesitamos una que se devuelva una mayor cantidad de información. En concreto, al aplicar una prueba a un proceso obtendremos una probabilidad (que es aquella con la que se ha ejecutado la correspondiente computación), junto con una variable aleatoria generada a partir de todas las variables aleatorias que aparecen en la secuencia. Así, nuestra definición de paso de pruebas tendrá en cuenta no sólo las probabilidades asociadas a las computaciones, sino también el *tiempo* que necesitan esas computaciones para finalizar. Por último, multiplicaremos la probabilidad de cada ejecución por la función de distribución que sigue la variable aleatoria asociada a la misma, de modo que para cada instante de tiempo nos dará la probabilidad total. Por ejemplo, $pass_{\leq 2}(P,T) = 0{,}3$ indica que el proceso P pasa la prueba T con probabilidad 0,3 antes de que transcurran 2 unidades de tiempo. Un mecanismo similar se utiliza en [GLNP97]. En el marco de los procesos estocásticos markovianos existe una alternativa, presentada en [BC00], donde se utiliza el tiempo medio de computación. Dado el hecho de que no restringimos el tipo de distribución de probabilidad, la aplicación de esta alternativa en nuestro modelo igualaría procesos que presentan comportamientos estocásticos diferentes. Ello es así, pues variables aleatorias con la misma esperanza matemática pueden tener distribuciones diferentes.

En la figura 5.1 utilizamos la misma representación gráfica que en los capítulos anteriores para introducir una serie de procesos estocásticos. Como hasta ahora, utilizaremos las letras

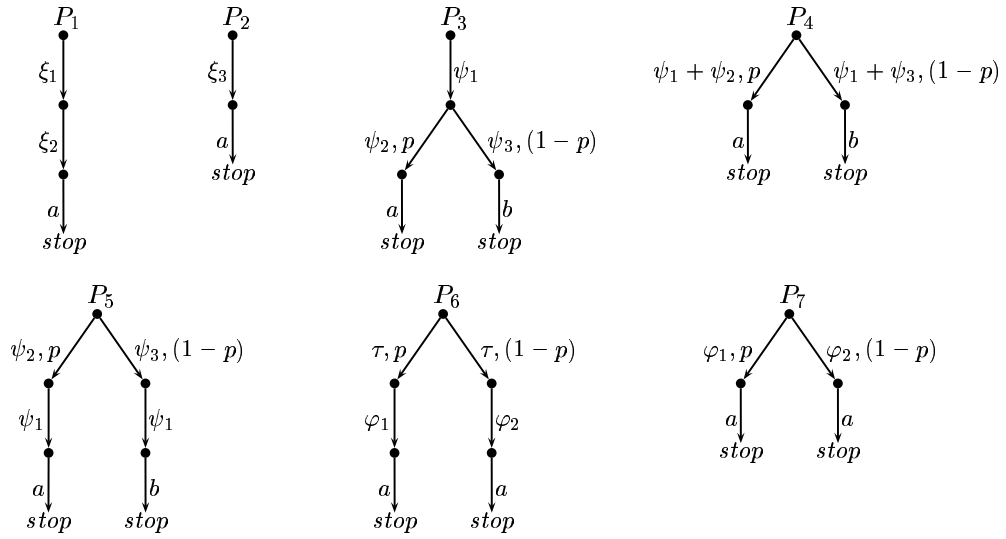


Figura 5.1: Ejemplos de procesos estocásticos.

griegas ξ, φ y ψ para denotar variables aleatorias, las letras latinas para representar acciones visibles, y la letra griega τ para representar la acción interna. Las transiciones estarán etiquetadas con este tipo de acciones, de cualquiera de los tres tipos, junto con una probabilidad (en las representaciones gráficas omitimos esa probabilidad cuando su valor sea 1).

Consideremos los procesos P_1 y P_2. A la hora de distinguirlos o no, la clave estará en la probabilidad de ejecutar a antes de que transcurra un cierto tiempo. Si la variable aleatoria $\xi_1 + \xi_2$ y la variable aleatoria ξ_3 están idénticamente distribuidas, entonces ambos procesos deberían ser equivalentes, siguiendo la línea de razonamiento dado para la definición de bisimulación débil estocástica del capítulo previo.

Consideremos ahora los procesos P_3, P_4 y P_5. Estos procesos serán equivalentes según nuestra semántica de *testing* debido a que en el momento temporal en el que se realiza la elección *probabilística* todos ellos siguen el mismo patrón. Por ejemplo, P_3 sufrirá inicialmente un retardo provocado por la variable aleatoria ψ_1, y a continuación, sufrirá otro de acuerdo con ψ_2, con probabilidad p, o de acuerdo con ψ_3, con probabilidad $1-p$. Si sumamos los retardos correspondientes tenemos que, con probabilidad p, se ejecutará la acción a después del retardo definido por la suma de ψ_1 y ψ_2, y con probabilidad $1-p$ se ejecutará b trás un retardo definido por la suma de ψ_1 y ψ_3. Esta situación se vuelve a repetir en cada uno de los procesos P_4 y P_5.

Por último, los procesos P_6 y P_7 también serán equivalentes bajo nuestra semántica de

testing. La razón por la cual ello es así es que ambas elecciones probabilísticas producen el mismo resultado. Las ideas presentadas con estos ejemplos motivarán nuestra presentación. En un álgebra de procesos estocásticos en la que los retardos están separados de las acciones ordinarias, las acciones estocásticas deben tratarse en parte como acciones *internas* que incluyen cierta información adicional. Obsérvese al respecto que si en este último ejemplo reemplazamos φ_1 y φ_2 por acciones ordinarias b y c, entonces los procesos resultantes no serían equivalentes bajo nuestra semántica de *testing*.

El resto del capítulo está estructurado de la siguiente manera: en la sección 5.2 presentamos nuestro lenguaje y su semántica operacional; en la sección 5.3 presentaremos la semántica de *testing* definiendo el conjunto de pruebas, la interacción entre los procesos y las pruebas, y la noción de paso de una prueba. Después, en la sección 5.4 daremos un conjunto de pruebas esenciales que tienen el mismo poder discriminatorio que toda la familia de pruebas. En la sección 5.5 relacionaremos nuestra semántica con otros modelos de semántica de *testing*, en concreto, consideraremos la noción clásica de *testing*, semánticas de *testing* puramente probabilísticas y una adaptación de [BC00] a nuestro marco. Mostraremos que nuestra noción de *testing* es una extensión conservativa de las dos primeras. Por último, en la sección 5.6 presentaremos nuestras conclusiones sobre el tema.

5.2. Descripción del lenguaje

En esta sección definimos nuestro modelo de procesos estocásticos. Los conceptos básicos sobre variables aleatorias están dados en la definición 3.1. Como en el resto de los capítulos, $\mathtt{Act}$, τ, $\mathtt{Act}_\tau$, $\mathcal{V}$ y Id sirven para denotar el conjunto de acciones visibles, la representante de la evolución interna de los procesos, la unión de $\mathtt{Act}$ y $\{\tau\}$, el conjunto de variables aleatorias y el conjunto de variables de proceso, respectivamente.

Definición 5.1 El conjunto de procesos, que denominaremos $Proc$, viene dado por la siguiente expresión EBNF:

$$P ::= stop \mid X \mid \sum_{i=1}^{n} [p_i]\gamma_i \,;\, P_i \mid rec\, X.P$$

donde $X \in Id$, para cualquier $1 \leq i \leq n$ tenemos que $\gamma_i \in \mathtt{Act}_\tau \cup \mathcal{V}$, $0 < p_i \leq 1$ y $\sum p_i = 1$. □

En la definición anterior, *stop* denota al proceso que no puede ejecutar ninguna acción. En la definición de los procesos omitiremos las apariciones triviales de *stop*, de modo que el

proceso $a;stop$ vendrá representado por a. Hemos incluido un operador de elección probabilístico n-ario similar al definido en [Han91, GSS95]. Sin embargo, utilizaremos azucar sintáctico para las elecciones unarias y binarias.[1] Por ejemplo, $\gamma_1 ; P_1$ denota al proceso $\sum_{i=1}^{1}[1]\gamma_i ; P_i$, mientras que $\gamma_1 ; P_1 \,\Box_p\, \gamma_2 ; P_2$ representa a $\sum_{i=1}^{2}[p_i]\gamma_i ; P_i$, donde $p_1 = p$ y $p_2 = 1 - p$. En este modelo las elecciones no se van a resolver de manera puramente probabilística. Por ejemplo, si consideramos el proceso $a \,\Box_p\, b$ y suponemos que el entorno ofrece únicamente la acción a, el proceso ejecutará a con probabilidad 1, sin tener en cuenta el posible valor de p. En consecuencia, procesos como $a \,\Box_p\, b$ y $\tau ; a \,\Box_p\, \tau ; b$ no serán equivalentes bajo nuestra semántica de *testing*. En cuanto a los términos que aparecen en una elección, $\alpha ; P$ (con $\alpha \in \mathtt{Act}_\tau$) denota un proceso que ejecuta la acción α y pasa a comportarse como P. El término $\xi ; P$ (con $\xi \in \mathcal{V}$) indica que el proceso P sufrirá un retardo de una cierta cantidad de tiempo aleatoria, definida por ξ. En concreto, si $p = P(\xi \leq t)$, P empezará su ejecución con una probabilidad p antes de que se consuman t unidades de tiempo. Por último, $rec\,X.P$ denota la recursión, como de ordinario.

Terminamos la presentación de la sintaxis comentando un par de cuestiones adicionales. En primer lugar, hemos elegido un operador de elección probabilístico n-ario únicamente porque la semántica operacional es más sencilla de definir en este caso. El problema está en que nos gustaría mantener la propiedad de *urgencia*, de modo que cuando un proceso pueda ejecutar una acción interna lo haga sin dejar pasar más tiempo, con lo que las acciones estocásticas no tendrían sentido en tal caso. Esto implica que las probabilidades asociadas anteriormente a esas transiciones estocásticas deben redistribuirse entre las transiciones restantes. En nuestro marco actual de trabajo, necesitamos únicamente una función de normalización simple; si utilizáramos una elección binaria, necesitaríamos una función mucho más complicada (tendría seis casos), que necesitaría dos predicados adicionales, para comprobar la estabilidad y el bloqueo de las componentes de la elección. Dado el hecho de que con nuestra simplificación no perdemos expresividad, hemos preferido mantener una semántica operacional tan simple como sea posible. En segundo lugar, nos gustaría comentar la ausencia del operador de paralelo. Como ya se ha indicado anteriormente, la definición de la semántica operacional de un operador de paralelo es sencilla cuando las distribuciones de probabilidad utilizadas son exponenciales, gracias a la propiedad de *falta de memoria*. Sin embargo, no ocurre lo mismo cuando manejamos distribuciones arbitrarias. Existen distintas propuestas que tratan satisfactoriamente el operador de paralelo. Entre ellas, destaca [BG02b] que presenta un lenguaje muy similar al nuestro, que aunque no tiene una relación

[1] Utilizaremos $\Box$ para denotar el operador de elección binaria, puesto que $+$ denota la suma de variables aleatorias.

de probabilidad entre acciones ordinarias, contiene *pesos* asociados a las acciones estocásticas que resuelven la elección entre este tipo de acciones. Sin embargo, la semántica de *testing* que presentamos aquí ya es suficientemente complicada como para asumir las dificultades adicionales que acarrearía el operador paralelo.

A la hora de definir la semántica operacional de los procesos, las transiciones estarán etiquetadas con una acción perteneciente a $\mathtt{Act}_\tau$ o a $\mathcal{V}$. Estas transiciones contarán con una etiqueta adicional: que define su probabilidad. De este modo, $P \xrightarrow{\gamma}_p P'$ indica que existe una transición desde P a P' etiquetada con una acción $\gamma \in \mathtt{Act}_\tau \cup \mathcal{V}$, que se ejecutará con probabilidad p.

Uno de los problemas clásicos cuando se consideran semánticas operacionales de procesos con elecciones probabilísiticas, consiste en tener en cuenta las diferentes apariciones de una misma transición. Por ejemplo, consideremos el proceso $P = a \Box_{\frac{1}{2}} a$. Si no tenemos cuidado, a la hora de definir el conjunto de transiciones obtendríamos $P \xrightarrow{a}_{\frac{1}{2}} \mathit{stop}$ una única vez, pese a que en realidad ocurre de maneras diferentes. Una propuesta para resolver este problema es la de añadir índices a las transiciones (p.ej. [GSS95]). También podemos incrementar el número de reglas (p.ej. [LS91]), definir una función de transición de probabilidad (p.ej. [SS00]), sumar las probabilidades asociadas con la misma acción (p.ej. [YL92]), o considerar que si una transición se puede derivar de distintas formas, cada derivación genera un ejemplar distinto (p.ej. [NdFL95]). En este último caso, se consideran multiconjuntos de transiciones en lugar de conjuntos. En este trabajo hemos optado por esta última opción, y así utilizaremos $\{\!|$ y $|\!\}$ como delimitadores de los multiconjuntos.

En la definición de la semántica operacional (véase figura 5.2) utilizamos una función auxiliar $\mathcal{N}_1(P)$. Esta función calcula la *probabilidad total* de que el proceso P ejecute alguna acción. Se trata por tanto de una función de normalización que se encarga, entre otras cosas, de mantener la propiedad de *urgencia*. De manera que $\mathcal{N}_1(P)$ devuelve el valor 1 si P no puede ejecutar inmediatamente τ's; en caso contrario, $\mathcal{N}_1(P)$ es igual a la suma de las probabilidades asociadas con acciones pertenecientes a $\mathtt{Act}_\tau$. La primera regla de la semántica operacional indica que si $\gamma \in \mathtt{Act}_\tau$ es una de las acciones que se ofrecen en una elección, esta acción se puede ejecutar, *normalizando* la probabilidad asociada con γ. La segunda regla se utiliza para las transiciones estocásticas. La condición lateral asegura que no se permiten transiciones estocásticas si el proceso puede ejecutar inmediatamente alguna τ. En cuanto a esta condición, debemos remarcar que $\mathcal{N}_1$ toma el valor 1 solamente en dos casos: o bien no existe j tal que $\tau = \gamma_j$, o bien para cualquier i tenemos $\gamma_i \in \mathtt{Act}_\tau$. En el segundo caso, la primera condición de la segunda regla no se cumple. La tercera regla

$$\frac{\gamma_i \in \mathtt{Act}_\tau}{\sum_{i=1}^{n}[p_i]\gamma_i;P_i \xrightarrow{\gamma_i}_{\frac{p_i}{\mathcal{N}_1(\sum_{i=1}^{n}[p_i]\gamma_i)}} P_i} \qquad \frac{\gamma_i \in \mathcal{V} \ \wedge \ \mathcal{N}_1(\sum_{i=1}^{n}[p_i]\gamma_i)=1}{\sum_{i=1}^{n}[p_i]\gamma_i;P_i \xrightarrow{\gamma_i}_{p_i} P_i} \qquad \frac{P[recX.P/X] \xrightarrow{\gamma}_p P'}{recX.P \xrightarrow{\gamma}_p P'}$$

$$\mathcal{N}_1\left(\sum_{i=1}^{n}[p_i]\gamma_i\right) = \begin{cases} 1 & \text{si } \tau \notin \{\gamma_i \mid 1 \leq i \leq n\} \\ \sum\{\!|p_i \mid \gamma_i \in \mathtt{Act}_\tau|\!\} & \text{en caso contrario} \end{cases}$$

Figura 5.2: Semántica operacional.

es estándar en lenguajes tipo CCS, como medio para definir la semántica de los procesos recursivos con respecto al significado de las transiciones.

Utilizaremos los siguientes convenios denotacionales para referirnos a las transiciones de un proceso P:

- Una transición $P \xrightarrow{\alpha}_p P'$ indica que, con probabilidad p, un proceso P puede evolucionar a P' ejecutando la acción α (siempre con $p > 0$).
- Una transición $P \xrightarrow{\xi}_p P'$ indica que, con probabilidad p, un proceso P puede evolucionar a P' tras un retardo cuya duración estará definida por la variable aleatoria ξ.
- $P \xrightarrow{\gamma}$ denota la existencia de un procesos $P' \in \mathit{Proc}$ y una probabilidad $p \in (0,1]$ tales que $P \xrightarrow{\gamma}_p P'$.
- Escribimos $P \not\xrightarrow{\gamma}$ si no existen $P' \in \mathit{Proc}$ y $p \in (0,1]$ tales que $P \xrightarrow{\gamma}_p P'$.
- Dado un conjunto $A \subseteq \mathtt{Act}_\tau \cup \mathcal{V}$, escribimos $P \xrightarrow{A}$ si existe $\gamma \in A$ tal que $P \xrightarrow{\gamma}$.
- Escribimos $P \not\xrightarrow{A}$ si no existe $\gamma \in A$ tal que $P \xrightarrow{\gamma}$.

5.3. Semántica de testing estocástico

En esta sección presentamos nuestra semántica de *testing* estocástico. Como es habitual, está basada en la interacción entre los procesos a probar y las pruebas. En primer lugar, definiremos nuestro conjunto de pruebas, considerando un conjunto de identificadores de pruebas $Id_{\mathcal{T}}$.

Definición 5.2 El *conjunto de pruebas*, que denotamos por $\mathcal{T}$, es el conjunto de pruebas definidas por la siguiente expresión EBNF:

$$T ::= stop \mid \sum_{i=1}^{n} [p_i]\alpha_i \,;\, T_i \mid rec\, X.T$$

donde $X \in Id_{\mathcal{T}}$, para cada $1 \leq i \leq n$ tenemos que $\alpha_i \in \mathtt{Act} \cup \{\omega\}$, $0 < p_i \leq 1$ y $\sum p_i = 1$. □

A la hora de presentar elecciones unarias y binarias. Utilizaremos el mismo azucar sintáctico que el utilizado para los procesos. Hemos añadido una nueva acción ω que indica la terminación exitosa de un procedimiento de prueba. Como hemos comentado en la introducción de este capítulo, no permitimos que las pruebas realicen acciones internas, por lo que una prueba únicamente puede ejecutar acciones que pertenezcan a $\mathtt{Act}$, o la acción especial ω. Explicaremos el significado de estas pruebas siguiendo la analogía de caja negra descrita en [Mil81], donde se considera que los procesos son cajas negras con botones. La prueba $a\,;T$ corresponde a presionar el botón a y si este botón se hunde, entonces continuamos el experimento con la prueba T. Considerando pruebas *no probabilísticas*, una prueba como $a\,;T \,\Box\, b\,;T'$ se puede explicar como la presión de dos botones de manera simultánea. En nuestro modelo, la prueba $a\,;T \,\Box_p\, b\,;T'$ se podría considerar como la presión de dos botones a la vez pero con *distinta* intensidad.

El comportamiento operacional de las pruebas es el mismo que para los procesos, considerando la acción ω como una acción *ordinaria* más. Debemos remarcar que, en este caso, el valor de la función $\mathcal{N}_1$, utilizada en la figura 5.2 como factor de normalización, será siempre 1. La interacción entre un proceso P y una prueba T está modelada mediante la composición paralela de los mismos, denotada por $P \parallel T$. Las reglas que describen cómo interactuan los procesos y las pruebas vienen dadas en la figura 5.3.

A la hora de definir la composición entre procesos y pruebas debemos llegar a un compromiso entre el marco clásico y el marco probabilístico. En el primero, si una prueba puede ejecutar una acción ω entonces el procedimiento de prueba termina, y con éxito. En nuestro caso mantendremos esta interpretación en casos como el de la prueba $a\,;T_1 \,\Box_p\, \omega\,;T_2$, que se comportará exactamente como la prueba ω. Por otro lado, si consideramos el marco probabilístico presentado en [NdFL95], un procedimiento de prueba finaliza (con probabilidad 1) solamente si el proceso probado está en un estado estable. Consideramos entonces que las acciones visibles se pueden ejecutar solamente si la prueba no puede ejecutar la acción ω (primera regla de la figura 5.3); si el proceso puede ejecutar o bien una τ o una acción estocástica entonces la interacción también las ejecutará (segunda y tercera reglas, respectivamente); y, por último, si la prueba puede ejecutar ω, entonces la interacción entre la prueba y el proceso

$$\frac{P \xrightarrow{a}_p P',\ T \xrightarrow{a}_q T',\ T \xrightarrow{\omega} \not\rightarrow}{P \| T \xrightarrow[\mathcal{N}_2(P\|T)]{a\ p\cdot q} P' \| T'} \qquad \frac{P \xrightarrow{\tau}_p P'}{P \| T \xrightarrow[\mathcal{N}_2(P\|T)]{\tau\ p} P' \| T}$$

$$\frac{P \xrightarrow{\xi}_p P'}{P \| T \xrightarrow[\mathcal{N}_2(P\|T)]{\xi\ p} P' \| T} \qquad \frac{T \xrightarrow{\omega},\ P \not\xrightarrow{\mathcal{V}\cup\{\tau\}}_1}{P \| T \xrightarrow{\omega}_{1-instab(P)} stop}$$

$$instab(P) = \sum\{\!\{p \mid \exists \gamma \in \mathcal{V} \cup \{\tau\}, P' \in Proc :\ P \xrightarrow{\gamma}_p P'\}\!\}$$

$$\mathcal{N}_2(P \parallel T) = \begin{cases} 1 & \text{si } T \xrightarrow{\omega} \\ \sum\{\!\{p \cdot q \mid \exists a, P', T' : P \xrightarrow{a}_p P' \wedge T \xrightarrow{a}_q T'\}\!\} + instab(P) & \text{si } T \xrightarrow{\omega} \not\rightarrow \end{cases}$$

Figura 5.3: Interacción entre procesos y pruebas.

también lo harán. Para evitar computaciones inútiles *cortaremos* el procedimiento de prueba tras la ejecución de la acción ω. Esta transición se ejecutará con probabilidad 1 menos la medida de *inestabilidad* del proceso en prueba. La condición lateral de esta regla asegura que no se genera la transición que tendría probabilidad 0. Como es habitual, contamos con una función de normalización, $\mathcal{N}_2(P \| T)$, que calcula la suma de las probabilidades asociadas con las transiciones que están etiquetadas con acciones que pertenecen a $\mathtt{Act}_\tau \cup \mathcal{V}$ que pueden ser ejecutadas por $P \parallel T$.

En la siguiente definición presentamos la noción de computación *exitosa*. Definiremos también algunos conceptos sobre las computaciones exitosas que utilizaremos al dar la noción de paso de una prueba.

Definición 5.3 Sea $P \in Proc$ y $T \in \mathcal{T}$. Una *computación* C del par (P, T) es una secuencia de transiciones finita o infinita

$$C = P \parallel T \xrightarrow{\gamma_1}_{p_1} P_1 \parallel T_1 \xrightarrow{\gamma_2}_{p_2} P_2 \parallel T_2 \xrightarrow{\gamma_3}_{p_3} \cdots \xrightarrow{\gamma_n}_{p_n} P_n \parallel T_n \cdots$$

Diremos que $P \parallel T$ es el *estado inicial* de C y que C es una computación desde $P \parallel T$.

Una computación C es *exitosa* si $P_n \parallel T_n \xrightarrow{\omega}_p stop$ para algún $n \geq 0$ y $p > 0$. En tal caso, diremos que $length(C) = n$. Denotamos por $Success(P \parallel T)$ al multiconjunto de computaciones exitosas desde $P \parallel T$.

Siendo $C \in Success(P \parallel T)$, definimos la *traza* de C, denotada por $trace(C)$, como

$$trace(C) = \begin{cases} \epsilon & \text{si } C = P \parallel T \xrightarrow{\omega}_p stop \\ \langle\gamma\rangle \circ trace(C') & \text{si } C = P \parallel T \xrightarrow{\gamma}_p C' \ \wedge \ \gamma \in \mathtt{Act} \\ trace(C') & \text{si } C = P \parallel T \xrightarrow{\gamma}_p C' \ \wedge \ \gamma \notin \mathtt{Act} \end{cases}$$

Y definimos la *variable aleatoria asociada con* C, denotada por $random(C)$, como:

$$random(C) = \begin{cases} unit & \text{si } C = P \parallel T \xrightarrow{\omega}_p stop \\ random(C') & \text{si } C = P \parallel T \xrightarrow{\gamma}_p C' \ \wedge \ \gamma \in \mathtt{Act}_\tau \\ \gamma + random(C') & \text{si } C = P \parallel T \xrightarrow{\gamma}_p C' \ \wedge \ \gamma \in \mathcal{V} \end{cases}$$

donde *unit*, sigue siendo la variable aleatoria con $F_{unit}(x) = 1$ para todo $x \geq 0$, y $+$ denota la suma de dos variables aleatorias.

Finalmente, definimos la *probabilidad* de C, denotada por $Prob(C)$, como

$$Prob(C) = \begin{cases} p & \text{si } C = P \parallel T \xrightarrow{\omega}_p \\ p \cdot Prob(C') & \text{si } C = P \parallel T \xrightarrow{\gamma}_p C' \end{cases}$$

□

En primer lugar, debemos resaltar que tendremos en realidad un multiconjunto de computaciones ya que pueden existir distintas transiciones para ejecutar una misma acción con una misma probabilidad. En particular, $Success(P \parallel T)$ también es un multiconjunto. La función $trace(C)$ calcula la secuencia de acciones visibles que aparecen en C. Utilizamos la variable aleatoria $random(C)$ para calcular el tiempo que se necesita para ejecutar C; es decir, para cualquier tiempo t, tenemos que $\mathrm{P}(random(C) \leq t)$ nos da la probabilidad de ejecutar C antes de que transcurra un tiempo t. Finalmente, $Prob(C)$ acumula las probabilidades de todas las decisiones tomadas para generar esta computación.

Ejemplo 5.4 Consideremos las computaciones que aparecen descritas en la figura 5.4, donde el símbolo ◎ representa la terminación con éxito del procedimiento de evaluación. Consideremos $P_1 = (\xi_1 \,; a) \,\square_{\frac{1}{3}}\, (a \,; b)$ y $T_1 = (a \,; \omega) \,\square_{\frac{1}{4}}\, (a \,; b)$. El primer gráfico de la figura 5.4 describe el multiconjunto de computaciones que tienen como estado de partida $P_1 \parallel T_1$.

Consideremos el proceso recursivo $P_2 = rec\, X.(\xi_2 \,;\, P_2 \,\square_{\frac{1}{3}}\, a)$ y las pruebas $T_2 = a \,;\, \omega$ y $T_3 = \omega$. En este caso, los multiconjuntos de computaciones desde $P_2 \parallel T_2$ y $P_2 \parallel T_3$ son ambos infinitos, estando descritos en el segundo y tercer grafo de la figura 5.4. □

En nuestro modelo, un proceso P pasa una prueba T antes de un cierto tiempo t con probabilidad p, si esta última probabilidad es igual a la suma de todas las probabilidades

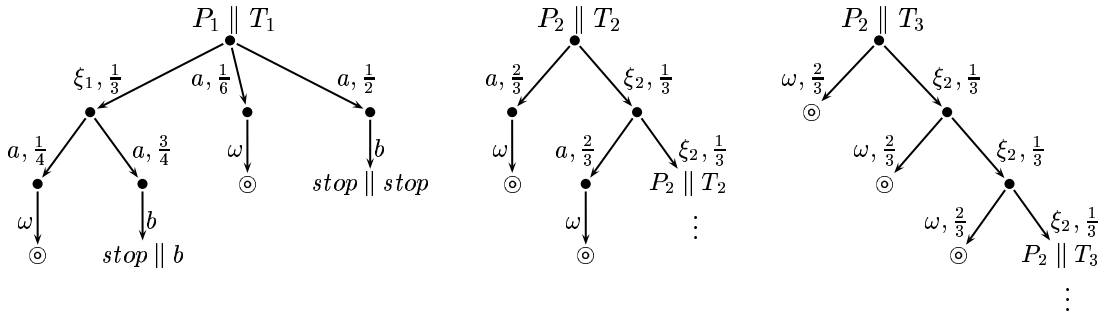


Figura 5.4: Ejemplos de computaciones.

asociadas con computaciones exitosas desde $P \parallel T$ que requieren un tiempo menor o igual que t para concluir.

Definición 5.5 Sean P un proceso, T una prueba, $p \in (0,1]$ y $t \in \mathbb{R}^+$. Decimos que P *pasa* T *antes de un tiempo* t *con probabilidad* p, lo que denotamos por $pass_{\leq t}(P,T) = p$, si

$$\sum_{C \,\in\, Success(P \parallel T)} \mathrm{P}(random(C) \leq t) \cdot Prob(C) = p$$

□

En el siguiente resultado presentamos una definición alternativa de la noción anterior. Su demostración se sigue directamente del hecho de que las computaciones exitosas tienen longitud finita.

Lema 5.6 Siendo P un proceso, T una prueba y $t \in \mathbb{R}^+$, tenemos que

$$pass_{\leq t}(P,T) = \lim_{n \,\to\, \infty} \sum_{\substack{C \,\in\, Success(P \parallel T) \\ length(C) \,<\, n}} \mathrm{P}(random(C) \leq t) \cdot Prob(C)$$

Definición 5.7 *(Equivalencia en testing)* Sean P, Q procesos y $\mathcal{T}' \subseteq \mathcal{T}$ una familia de pruebas. Decimos que P es *equivalente en testing estocástico* a Q con respecto a $\mathcal{T}'$, lo que denotamos por $P \sim_{\mathcal{T}'} Q$, si para cualquier $T \in \mathcal{T}'$ y cualquier $t \in \mathbb{R}^+$ se tiene $pass_{\leq t}(P,T) = pass_{\leq t}(Q,T)$. Si consideramos la familia completa de pruebas $\mathcal{T}$, escribiremos $P \sim_{wst} Q$ en lugar de $P \sim_{\mathcal{T}} Q$, y diremos que P y Q son equivalentes bajo *testing* estocástico. □

En el siguiente ejemplo presentamos algunos procesos que no son equivalentes y algunas pruebas que los distinguen.

Ejemplo 5.8 Consideremos los siguientes procesos: $Q_1 = \tau\,;a\,\Box_p\,\tau\,;b$, $Q_2 = a\,\Box_p\,b$ y $Q_3 = \tau\,;a\,\Box_p\,b$. Como en el caso probabilístico, en nuestro marco los tres procesos tampoco son equivalentes dos a dos. Si consideramos la prueba $T = a\,;\omega$, tenemos que para cualquier $t \in \mathbb{R}^+$, $pass_{\leq t}(Q_1, T) = p$, mientras que $pass_{\leq t}(Q_2, T) = pass_{\leq t}(Q_3, T) = 1$. Además, la prueba $T' = b\,;\omega$ muestra que Q_2 y Q_3 no son equivalentes bajo *testing* estocásticos debido a que para todo $t \in \mathbb{R}^+$ tenemos $pass_{\leq t}(Q_2, T') = 1$, mientras que $pass_{\leq t}(Q_3, T') = 1 - p$.

Consideremos ahora los procesos $R_1 = \xi\,;a$ y $R_2 = \psi\,;a$. Si ξ y ψ no están idénticamente distribuidas, entonces existe un tiempo $t_1 \in \mathbb{R}^+$ tal que $\mathrm{P}(\xi \leq t_1) \neq \mathrm{P}(\psi \leq t_1)$, por lo que $pass_{\leq t_1}(P, \omega) \neq pass_{\leq t_1}(Q, \omega)$.

Sean ahora $R_3 = \xi\,;a$ y $R_4 = a\,;\xi$, y supongamos que ξ no sigue la misma distribución de probabilidad que *unit*. Esto implica que existe $t_1 \in \mathbb{R}^+$ tal que $\mathrm{P}(\xi \leq t_1) = p < 1$. Entonces los dos procesos se pueden distinguir mediante la prueba ω, pues por un lado tenemos $pass_{\leq t_1}(R_3, \omega) = p$, mientras que $pass_{\leq t_1}(R_4, \omega) = 1$.

Consideremos los procesos y las pruebas del ejemplo 5.4. Una vez que hemos calculado el correspondiente conjunto de computaciones, podemos calcular la probabilidad de pasar cada una de las pruebas. Para ello basta con sumar las probabilidades asociadas con las computaciones exitosas. De modo que, que para cada $t \in \mathbb{R}^+$ se tiene $pass_{\leq t}(P_1, T_1) = \frac{1}{12} \cdot \mathrm{P}(\xi_1 \leq t) + \frac{1}{6}$, mientras que

$$pass_{\leq t}(P_2, T_2) = pass_{\leq t}(P_2, T_3) = \sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i \cdot \frac{2}{3} \cdot \mathrm{P}(i \cdot \xi_2 \leq t)$$

donde $n \cdot \xi$ denota la suma de n variables aleatorias independientes distribuidas como ξ. □

5.4. Conjunto de pruebas esenciales

En esta sección mostraremos que el conjunto de pruebas utilizado para distinguir procesos se puede reducir. Concretamente, presentaremos una familia de pruebas que tiene el mismo poder discriminatorio que la familia completa de pruebas $\mathcal{T}$. En primer lugar, podemos restringirnos a pruebas finitas, dejando de lado las pruebas no recursivas. La demostración se realiza utilizando una extensión de la técnica presentada en [GN99], donde se presenta un resultado similar para un álgebra de procesos probabilísticos.

Lema 5.9 Sea $\mathcal{T}_f \subseteq \mathcal{T}$ el conjunto de pruebas sin apariciones del operador de recursión, y P, Q procesos. Entonces $P \sim_{\mathcal{T}_f} Q$ sii $P \sim_{wst} Q$.

Demostración: La implicación de derecha a izquierda es obvia. La demostración de izquierda a derecha la realizaremos por reducción al absurdo. Supongamos que P y Q pasan cualquier

prueba finita con la misma probabilidad antes de un cierto tiempo dado, pero que existe una prueba recursiva T tal que $pass_{\leq t}(P,T) = p \neq q = pass_{\leq t}(Q,T)$ para un $t \in \mathbb{R}^+$. Entonces,

$$\begin{aligned} p \;=\; & \lim_{n\to\infty} \sum_{\substack{C \in Success(P \parallel T) \\ length(C) < n}} \mathrm{P}(random(C) \leq t) \cdot Prob(C) \;\neq \\ & \lim_{n\to\infty} \sum_{\substack{C \in Success(Q \parallel T) \\ length(C) < n}} \mathrm{P}(random(C) \leq t) \cdot Prob(C) \;=\; q \end{aligned}$$

Ahora, dados un proceso R y una prueba T, utilizaremos la notación

$$C_{R\|T}(i) = \sum_{\substack{C \in Success(R \parallel T) \\ length(C) < i}} \mathrm{P}(random(C) \leq t) \cdot Prob(C)$$

Considerando la definición matemática de límite, tenemos que para cualquier $\epsilon > 0$ existen N_p y N_q tales que para cualquier $n > N_p$ se cumple $\mid C_{P\|T}(n) - p \mid \; < \epsilon$ y para cualquier $m > N_q$ se cumple $\mid C_{Q\|T}(m) - q \mid \; < \epsilon$. Tomemos $\epsilon = \frac{\mid p-q \mid}{2}$. Entonces, existen N_p y N_q tales que $\forall n > N = \max\{N_p, N_q\}$ se cumple $\mid C_{P\|T}(n) - p \mid \; < \epsilon$ y $\mid C_{Q\|T}(n) - q \mid \; < \epsilon$. Por tanto, $C_{P\|T}(n) \neq C_{Q\|T}(n)$, es decir, existe un N tal que para todo $n \geq N$ se tiene:

$$\sum_{\substack{C \in Success(P \parallel T) \\ length(C) < n}} \mathrm{P}(random(C) \leq t) \cdot Prob(C) \neq \sum_{\substack{C \in Success(Q \parallel T) \\ length(C) < n}} \mathrm{P}(random(C) \leq t) \cdot Prob(C)$$

Consideremos la prueba finita T' obtenida como resultado de desdoblar N veces cada aparición del operador de recursión en T y después reemplazar las restantes por *stop*. Tenemos que T y T' pueden ejecutar las mismas secuencias de acciones si la longitud de la secuencia es menor o igual que N. Por ello, $P \parallel T'$ y $Q \parallel T'$ pueden ejecutar las mismas computaciones de longitud menor o igual que N que $P \parallel T$ y $Q \parallel T$, respectivamente. En consecuencia, las probabilidades con las que P y Q pasan T' antes de un cierto tiempo son distintas, de modo que $P \not\sim_{\mathcal{T}_f} Q$. □

A continuación, demostraremos que el conjunto de pruebas se puede reducir aún más. En la siguiente definición describimos el que será nuestro conjunto de pruebas *esenciales*.

Definición 5.10 El conjunto de *pruebas esenciales*, denotado por $\mathcal{T}_e$, viene dado por la siguiente expresión BNF:

$$T ::= \sum_{i=1}^{n} [p_i](a_i \,;\, T_i) \mid \omega \qquad \text{donde } T_i = \begin{cases} T & \text{si } i = n \\ stop & \text{en caso contrario} \end{cases}$$

donde $\{a_1, \ldots, a_n\} \subseteq \mathtt{Act}$, para todo $1 \leq i \leq n$ se tiene $0 < p_i \leq 1$ y $\sum p_i = 1$. □

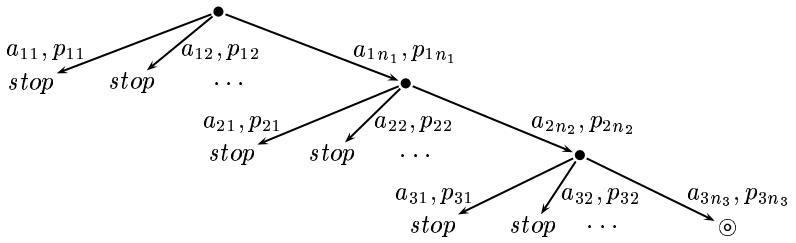


Figura 5.5: Estructuras de las pruebas esenciales de tamaño 3.

Una prueba esencial o bien es la prueba ω, o bien está formada por una elección probabilística generalizada entre un conjunto de acciones visibles. En el caso de la elección entre acciones visibles todas las *continuaciones* son iguales a *stop* excepto una, que será de nuevo una prueba esencial. En la figura 5.5 está representada una prueba esencial de tamaño tres. Debemos resaltar que en [NdFL95, CDSY99] aparecen conjuntos de pruebas esenciales similares. Para probar que no se pierde poder discriminatorio considerando este conjunto de pruebas seguiremos el mismo patrón que en la demostración dada en [CDSY99]. La idea subyacente es que cada prueba finita T tiene un número finito de ejecuciones exitosas. Cuando un proceso P interactua con T cada computación exitosa se corresponde con una ejecución de la prueba, que concluye ejecutando ω, donde las probabilidades se calculan teniendo en cuenta las probabilidades utilizadas por el proceso y la prueba para ejecutar la computación añadiendo todas las probabilidades de P de ejecutar esas ejecuciones exitosas. Considerando las ejecuciones exitosas de T, podemos obtener un conjunto de pruebas esenciales, una por cada ejecución, de manera que la probabilidad de que P pase T pueda definirse a partir de las probabilidades con los que el proceso pase las correspondientes pruebas esenciales. Estas ejecuciones exitosas se representarán no como secuencias de acciones, sino como secuencias de pares (A, a) tales que A es un estado probabilístico y $a \in \mathtt{Act}$. Los estados probabilísticos serán conjuntos de pares (a, p) que indican las acciones (junto con sus probabilidades) que un proceso o una prueba pueden ejecutar en un punto concreto de la secuencia. La suma de todas las probabilidades de los pares pertenecientes a un estado probabilístico será menor que 1 siempre que el proceso pueda ejecutar alguna acción estocástica.

Definición 5.11 Sea $A \subseteq \mathtt{Act} \times (0, 1]$. Decimos que A es un *estado probabilístico* si o bien $A = \emptyset$, o $0 < \sum\{|p \mid (a, p) \in A|\} \leq 1$ y no existen $0 < p, q \leq 1$ y $a \in \mathtt{Act}$ tales que $(a, p), (a, q) \in A$ y $p \neq q$. El *conjunto de acciones* de A, denotado por $ac(A)$, viene definido como el conjunto $ac(A) = \{a \mid \exists p \in (0, 1] \ : \ (a, p) \in A\}$. Definimos la probabilidad de una

acción a perteneciente a A, denotada por $\pi(a, A)$, como p si $(a, p) \in A$ y como 0 en caso contrario.

Siendo $P \in Proc \cup \mathcal{T}$, definimos el *sucesor de* P, y lo denotamos por $S(P)$, como

$$\text{next}(P) = \{(a, p) \mid a \in \texttt{Act} \ \wedge \ p = \sum\{\!|q \mid \exists P' : P \xrightarrow{a}_q P'|\!\} \ \wedge \ p > 0\}$$

□

En la siguiente definición se presenta el conjunto de pruebas esenciales asociadas a una determinada prueba finita al considerar sus ejecuciones exitosas.

Definición 5.12 Sea $T \in \mathcal{T}_f$. Definimos el *conjunto de ejecuciones exitosas* de T, denotado por $SEx(T)$, como el conjunto de ejecuciones de T de la forma

$$r = T \xrightarrow{a_0}_{p_0} T_1 \xrightarrow{a_1}_{p_1} T_2 \cdots T_n \xrightarrow{a_n}_{p_n} T' \xrightarrow{\omega}_{p'}$$

Llamaremos *secuencia de acciones* asociada con la ejecución r de T, denotada por $seq(r)$, a la secuencia:

$$\langle \text{next}(T)a_0, \text{next}(T_1)a_1, \cdots \text{next}(T_n)a_n \rangle$$

donde $\text{next}(T_j)$ es el correspondiente conjunto probabilístico y $a_j \in ac(\text{next}(T_j))$. Para cualquier $r \in SEx(T)$, definimos recursivamente la *prueba esencial asociada a la secuencia* $seq(r) = \langle A_1 a_1, A_2 a_2, \cdots A_n a_n \rangle$, denotada por $T^{seq(r)}$, como $T_n^{seq(r)} = \omega$ si $n = length(r)$, y para todo $1 \leq k < n$:

$$T_k^{seq(r)} = \sum_{b_i \in ac(A_k)} [\pi(b_i, A_k)]\,(b_i\,;T_i) \qquad \text{donde } T_i = \begin{cases} T_{k+1}^{seq(t)} & \text{si } b_i = a_k \\ stop & \text{en otro caso} \end{cases}$$

□

Cuando un proceso interactúa con una prueba cada computación exitosa está relacionada con una ejecución exitosa de dicha prueba. En la siguiente definición describimos esta relación.

Definición 5.13 Sean $P \in Proc$, $T \in \mathcal{T}$ y $r \in SEx(T)$ la ejecución

$$r = T \xrightarrow{a_0}_{p_0} T_1 \xrightarrow{a_1}_{p_1} T_2 \cdots T_n \xrightarrow{a_n}_{p_n} T' \xrightarrow{\omega}_{p'}$$

Definimos el *conjunto de computaciones exitosas relacionadas con* r, denotado por $Comp(r)$, como el subconjunto de computaciones exitosas desde $P \parallel T$ tales que

$$C = P \parallel T \xrightarrow{\gamma_1}_{p_1} P_1 \parallel T_1' \xrightarrow{\gamma_2}_{p_2} P_2 \parallel T_2' \xrightarrow{\gamma_3}_{p_3} \cdots \xrightarrow{\gamma_n}_{p_n} P_n \parallel T_n' \xrightarrow{\omega}_{p}$$

donde $T'_i = \begin{cases} T'_{i-1} & \text{si } \gamma_i \in \mathcal{V} \cup \{\tau\} \\ T_j & \text{si } \gamma_i = a_j \ \wedge \ \forall 1 \leq j' \leq j, \exists 1 \leq i' \leq i \ : \ T'_{i'} = T_{j'}. \end{cases}$ □

Obsérvese que cuando $\gamma_i \in \mathcal{V} \cup \{\tau\}$ el proceso evoluciona, pero la prueba no, de modo que $P_i \parallel T'_i \xrightarrow{\gamma_i}_p P_{i+1} \parallel T'_{i+1}$ con $T'_i = T'_{i+1}$. Sin embargo, cuando γ_i es una acción visible ambos evolucionan al tiempo, obteniendose $P_j \parallel T'_j \xrightarrow{\gamma_i}_p P_{j+1} \parallel T'_{j+1}$, donde $P_j \xrightarrow{\gamma_i}_p P_{j+1}$ y $T'_j = T_i \xrightarrow{\gamma_i}_q T_{i+1} = T'_{j+1}$.

Teorema 5.14 Siendo P, Q un par de procesos, tendremos $P \sim_{\mathcal{T}_e} Q$ sii $P \sim_{wst} Q$.

Demostración: La implicación de derecha a izquierda es obvia. Para ver la implicación de izquierda a derecha mostraremos que la probabilidad con la que un proceso tiene de pasar la prueba es igual a la suma de las probabilidades con los que el proceso pasa el conjunto de pruebas asociadas con las ejecuciones exitosas de la prueba.

Sea $SEx(T)$ el conjunto de ejecuciones exitosas de T. Entonces, para cada $r \in SEx(T)$ tenemos el conjunto de computaciones desde $P \parallel T$ que están relacionadas con r, $Comp(r)$. Considerando $SEx(T) = \{r_1, \cdots, r_m\}$ tenemos que para cualquier el valor que toma $t \in \mathbb{R}^+$, $pass_{\leq t}(P, T)$ viene dado por

$$\sum_{C \in Success(P \| T)} \mathrm{P}(random(C) \leq t) \cdot Prob(C) = \sum_{i=1}^{m} \Big(\sum_{C \in Comp(r_i)} \mathrm{P}(random(C) \leq t) \cdot Prob(C) \Big) =$$

$$= \sum_{r \in SEx(T)} \Big(\sum_{C \in Comp(r)} \mathrm{P}(random(C) \leq t) \cdot Prob(C) \Big) = \sum_{r \in SEx(T)} pass_{\leq t}(P, T^r)$$

Por hipótesis de inducción, dado que la profundidad de T^r es estrictamente menor que la de T, para cada $r \in SEx(T)$ tenemos que $pass_{\leq t}(P, T^r) = pass_{\leq t}(Q, T^r)$, y por tanto, concluimos que $pass_{\leq t}(P, T) = pass_{\leq t}(Q, T)$. □

5.5. Relación con otras nociones de testing

En esta sección compararemos nuestra semántica de *testing* estocástico con otros modelos de *testing*. Concretamente, consideraremos los equivalencias *may* y *must* [dNH84, Hen88], una semántica de *testing* similar a la dada en [CDSY99] y la noción de *testing* (markoviana) de [BC00]. En primer lugar, definiremos un subconjunto del conjunto completo de procesos *Proc* en el que no aparece información estocástica. Por tanto, este nuevo conjunto de procesos, que denotaremos por $Proc_P$, define un álgebra de procesos probabilísticos.

Definición 5.15 El conjunto de *procesos probabilísticos*, denotado por $Proc_P$, viene dado por la expresión EBNF:

$$P ::= stop \mid X \mid \sum_{i=1}^{n} [p_i]\alpha_i\,;P_i \mid rec\,X.P$$

donde $X \in Id_{Proc}$, para cada $1 \leq i \leq n$ tenemos $\alpha_i \in \mathtt{Act}_\tau$, $0 < p_i \leq 1$ y $\sum p_i = 1$. □

A continuación, estudiaremos las nociones de *testing may* y *must* definidas en [dNH84, Hen88] sobre nuestro lenguaje $Proc_P$.

Definición 5.16 Siendo $P \in Proc_P$ y $T \in \mathcal{T}$, escribimos $P\ may\ T$ si $Success(P \parallel T) \neq \varnothing$ y $P\ must\ T$ si para cualquier computación *maximal* (es decir, que no puede extenderse más) C tenemos que $C \in Success(P \parallel T)$.

Siendo $P, Q \in Proc_P$, escribimos $P \approx_{\text{may}} Q$ si para cualquier $T \in \mathcal{T}$ tenemos $P\ may\ T$ sii $Q\ may\ T$, y escribimos $P \approx_{\text{must}} Q$ si para cualquier $T \in \mathcal{T}$ tenemos $P\ must\ T$ sii $Q\ must\ T$. □

Obsérvese que en las definiciones previas no se ha utilizado la información probabilística que contienen ni los procesos ni las pruebas. Podemos redefinir estas nociones de *testing* en nuestro marco como indica el siguiente resultado de demostración sencilla.

Lema 5.17 Sean $P \in Proc_P$ y $T \in \mathcal{T}$. Tenemos que $P\ may\ T$ sii $\exists t \in \mathbb{R}^+$ tal que $pass_{\leq t}(P,T) > 0$. Si P está libre de divergencias entonces $P\ must\ T$ sii $\exists t \in \mathbb{R}^+$ tal que $pass_{\leq t}(P,T) = 1$.

Obsérvese que, en el lema anterior, el valor que tome t es irrelevante. El siguiente resultado, cuya demostración es trivial, afirma que nuestra semántica de *testing* es un refinamiento estricto de las nociones clásicas para procesos libres de divergencias.

Corolario 5.18 Siendo $P, Q \in Proc_P$, tenemos que $P \sim_{wst} Q$ implica $P \approx_{\text{may}} Q$. Además, si P y Q están libres de divergencias también tenemos que $P \sim_{wst} Q$ implica $P \approx_{\text{must}} Q$.

Debemos comentar que si consideramos procesos divergentes el resultado anterior no se cumple en el caso de la semántica must. Por ejemplo, los procesos (no probabilísticos) $rec\,X.(a \,\Box\, \tau\,;X)$ y $a\,;stop$ no son equivalentes bajo la semántica must. Sin embargo, para cualquier $p \in (0,1)$ los procesos probabilísticos $rec\,X.(a \,\Box_p\, \tau\,;X)$ y $a\,;stop$ son equivalentes estocásticos. Por tanto, para un proceso que presente un comportamiento divergente, pasar una prueba con probabilidad 1 no es equivalente a pasarlo en la semántica must. Se puede encontrar una discusión más extensa sobre el tema en [NR99].

La teoría de *testing* probabilístico definida en [CDSY99] calcula la probabilidad de pasar una prueba como la suma de las probabilidades asociadas con todas las computaciones exitosas. En primer lugar, estudian una semántica de *testing* donde las pruebas no tienen la posibilidad de ejecutar acciones internas, para pasar a continuación a estudiar el caso general donde las pruebas pueden contener dichas acciones. En nuestro caso vamos a definir una semántica de *testing* puramente probabilística, siguiendo las líneas de [CDSY99], para pruebas sin acciones internas.

Definición 5.19 Sean $P \in Proc_P$, $T \in \mathcal{T}$ y $p \in (0,1]$. Diremos que P pasa la prueba T con probabilidad p, y lo denotaremos por $P\ pass_p\ T$, si se cumple que $\sum_{C \in Success(P \| T)} Prob(C) = p$.

Dos procesos $P, Q \in Proc_P$ son *equivalentes probabilísticamente en testing*, lo que denotamos por $P \sim_P Q$, si para cualquier prueba T tenemos que $P\ pass_p\ T$ sii $Q\ pass_p\ T$. □

Esta noción de *testing* puede incluirse fácilmente en nuestro marco, como se indica en el siguiente resultado. La demostración es trivial, simplemente teniendo en cuenta que si $P \in Proc_P$ y T es una prueba, entonces para cualquier computación exitosa C desde $P \| T$ tenemos que $random(C) = unit$ y $\mathrm{P}(unit \leq t) = 1$, para cualquier $t \in \mathbb{R}^*$.

Lema 5.20 Siendo $P \in Proc_P$, $T \in \mathcal{T}$ y $p \in (0,1]$, tenemos que $P\ pass_p\ T$ si, y sólo si $\forall t \in \mathbb{R}^+$, $pass_{\leq t}(P,T) = p$. Además, para cualquier $P, Q \in Proc_P$ tenemos que $P \sim_P Q$ si, y sólo si, $P \sim_{wst} Q$.

En [BC00] se presenta una teoría de *testing* Markoviana. Es este modelo las distribuciones de probabilidad están restringidas a ser exponenciales, pero también se resuelven las elecciones con una política de preselección. Puesto que nuestro modelo es diferente del que ellos proponen, no podemos comparar directamente ambas semánticas de *testing*, de modo que para hacerlo adaptaremos su noción de semántica a nuestro marco. La teoría presentada en [BC00] no calcula la suma de variables aleatorias, puesto que el conjunto de las exponenciales no es cerrado bajo la suma, sino que calculan el *tiempo medio* que necesitan las computaciones para ejecutarse. En nuestro caso, podemos considerar la *esperanza* de la variable aleatoria asociada con la ejecución de una computación. Dada nuestra suposición inicial de que todas las variables aleatorias son independientes la esperanza matemática de las variables aleatorias asociada con una computación será igual a la suma de las esperanzas correspondientes a las distribuciones que controlan las transiciones estocásticas ejecutadas a lo largo de la computación.

Definición 5.21 Siendo $P \in Proc$ y $T \in \mathcal{T}$, para cada $t \in \mathbb{R}^+$ definimos la probabilidad con la cual P *pasa* T en *tiempo medio* antes de que un tiempo t haya transcurrido, denotada por $pass_{rate \leq t}(P,T)$, como

$$pass_{rate \leq t}(P,T) = \sum_{C \in Success(P \parallel T)} Prob(C) \cdot \mathrm{P}(\mathrm{E}[random(\mathrm{C})] \leq \mathrm{t})$$

Dos procesos $P, Q \in Proc$ son *equivalentes en tiempo medio*, denotado por $P \sim_{av} Q$, si para cualquier prueba T y cualquier $t \in \mathbb{R}^+$ se cumple que $pass_{rate \leq t}(P,T) = pass_{rate \leq t}(Q,T)$. □

De modo que de manera trivial se cumple el siguiente resultado que relaciona a las nociones de *testing* que hemos visto para estos procesos:

Lema 5.22 Siendo $P, Q \in Proc_P$, tenemos que $P \sim_{wst} Q$ implica $P \sim_{av} Q$.

También es trivial comprobar que la implicación contraria no se cumple. Basta considerar los procesos $P_1 = \xi_1 \,;\, stop$ y $P_2 = \xi_2 \,;\, stop$ con $\mathrm{E}[\xi_1] = \mathrm{E}[\xi_2]$, pero no estándo ξ_1 y ξ_2 idénticamente distribuidas. Entonces tenemos que $P_1 \sim_{av} P_2$, mientras que $P_1 \not\sim_{wst} P_2$.

5.6. Conclusiones

En este capítulo hemos estudiado una semántica de *testing* para una clase de procesos estocásticos con distribuciones generalizadas. La misma ha sido definida siguiendo las ideas presentadas en el capítulo anterior. Una vez dada la definición de paso de una prueba y la equivalencia entre procesos, hemos definido un conjunto de pruebas esenciales cuyo poder discriminatorio es el mismo que el del conjunto total de pruebas. Además, comparamos nuestro trabajo con otras nociones de *testing* definidas en otros marcos, como el probabilístico y el estocástico markoviano.

En cuanto al trabajo que queda por realizar sobre la cuestión, es nuestra intención extender esta noción de *testing* sobre un lenguaje más expresivo que contenga el operador de composición paralela. Como ya indicamos en las conclusiones del capítulo anterior, esta tarea se anuncia bastante complicada, debido a que pretendemos unir una secuencia de variables estocásticas para identificarla con una única variable aleatoria. El problema radica en que en el contexto del operador de paralelo no es nada sencillo calcular en cada momento cuándo empiezan y cuándo acaban de ejecutarse las acciones estocásticas. En el capítulo siguiente definiremos una semántica operacional para procesos estocásticos no markovianos

con operador de paralelo, pero dejamos la definición de una semántica de *testing* para futuros trabajos.

En segundo lugar, queremos definir una caracterización alternativa de esta semántica de *testing*, que nos permita decidir si dos procesos son equivalentes sin necesidad de comprobar la respuesta de estos procesos ante todas las pruebas posibles. En particular, deberíamos definir al respecto a la noción adecuada *conjuntos de aceptación*. Al efecto, pretendemos utilizar extensiones estocásticas de los conjuntos de aceptación definidos en [Hen88], de forma que los conjuntos no estarían formados exclusivamente por acciones visibles, sino que incluirían también las probabilidades de ejecutarlas quedando también reflejada la información estocástica a través de las variables aleatorias que el proceso pueda ejecutar.

Capítulo 6

Un marco integrado para el análisis de procesos estocásticos con comunicación asíncrona

En este capítulo presentamos un marco en el que un álgebra de procesos (no trivial) se combina con un lenguaje funcional (concurrente). En concreto, consideramos un álgebra de procesos estocásticos con paso de valores donde, como ocurría en los lenguajes vistos anteriormente, las distribuciones no están restringidas a ser exponenciales. Para estudiar las propiedades de las especificaciones descritas usando el álgebra de procesos, hemos *traducido* las mismas a programas funcionales escritos en Eden que es un lenguaje funcional muy apropiado para programación concurrente. Por un lado, presenta las características habituales de los lenguajes funcionales modernos, y por otro lado, permite la ejecución de procesos concurrentes. Presentaremos un par de casos de estudio de mediana complejidad mostrando cómo las especificaciones pueden traducirse a Eden y cómo se pueden estudiar propiedades cuantitativas de las primeras a partir de sus traducciones.

El trabajo aquí expuesto aparece recogido en [LNR02, LNR03].

6.1. Introducción

A la hora de estudiar el comportamiento de sistemas, *model checking* [CGP99] representa una técnica muy adecuada para comprobar si dichos sistemas cumplen una determinada propiedad. Sin embargo, las actuales técnicas de model checking no se pueden utilizar para sistemas en los que se incluye información estocástica. Más exáctamente, se puede utilizar

model checking con sistemas estocásticos cuyas distribuciones de probabilidad estén restringidas a ser exponenciales, utilizando técnicas de cadenas de Markov (véase p.ej. [HKMS00]). Desgraciadamente, no podemos aplicar esta misma técnica cuando las distribuciones de probabilidad son *generales* (un primer paso se ha dado en [IHK01] donde se consideran procesos semi-Markovianos). Por lo tanto, para analizar este tipo de sistemas debemos utilizar una aproximación diferente. Una posibilidad, que es la que utilizaremos en este caso, consiste en *simular* el sistema especificado. Intuitivamente, una forma de estudiar las propiedades de una especificación consiste en realizar una *implementación* adecuada y simular su ejecución. En este caso, obtendremos estimaciones *reales* de la ejecución *teórica* del sistema.

En este capítulo vamos a definir un marco integrado que nos va a permitir el estudio de sistemas que contienen información estocástica mediante la simulación de su comportamiento con programas en Eden. Además de los operadores habituales de las álgebras de procesos (elección, paralelo, ocultamiento, etc.) incluimos algunas caraterísticas más complejas para facilitar el diseño de los sistemas reales. Llamaremos a este lenguaje `VPASPA`. En primer lugar, nuestro lenguaje permite al diseñador especificar información temporal dada a través de retardos estocásticos. Por ejemplo, un proceso como $\xi\,;P$ tiene un retardo de t unidades de tiempo con probabilidad $\mathrm{P}(\xi \leq t)$. Debemos destacar que se pueden especificar retardos deterministas (como se presentó en el álgebra de procesos estocásticos del capítulo 2) utilizando la distribución de Dirac y las distribuciones discretas. Como ya se ha comentado anteriormente, la combinación del operador de paralelo y las distribuciones generales complican mucho la definición de los modelos semánticos. Este es el motivo por el cual las álgebras de procesos están basadas normalmente en cadenas semi-Markovianas (existen algunas excepciones como son [BBG98, DKB98b, LN00, BG02b]). Otra de las características principales de nuestro lenguaje es que también permite paso de valores. Habitualmente no se incluye el paso de valores en álgebras de procesos estocásticos y de hecho la única propuesta, que conocemos, que incluye un lenguaje estocástico con esta característica está en [Ber99].

La última característica no estándar que presenta nuestro lenguaje es un mecanismo de comunicación asíncrono (véase p.ej. [dBKP92]). Hemos tomado esta decisión debido a que la comunicación asíncrona aparece en muchas situaciones de la vida real y no se puede codificar fácilmente con los mecanismos usuales de comunicación síncrona que aparecen en la mayoría de las álgebras de procesos. Sin embargo, podríamos incluir también un operador de composición paralela síncrono a nuestro modelo de manera que el diseñador pueda elegir el paradigma de comunicación que desee. Modelaremos la comunicación asíncrona considerando colas asociadas con los canales correspondientes para almacenar los mensajes que se

pasan como valores en la comunicación. Consideremos un proceso como $P = (a!(v)\,;P_1) \parallel P_2$. Permitimos que la parte izquierda de la composición paralela ejecute su acción de salida $a!(v)$ sin sincronizar con la parte derecha. El valor v, que constituye el mensaje enviado, se almacena en una cola asociada con el canal a y el proceso P evoluciona a $P_1 \parallel P_2$. Consideremos ahora un proceso como $Q = (a?(x)\,;Q_1) \parallel Q_2$. La acción de entrada $a?(x)$ puede ejecutarse únicamente si la cola asociada con el canal a no es vacía. En este caso, se coge el valor v' almacenado en la primera posición de la cola, se ejecuta la acción de entrada $a?$ y el proceso Q evoluciona a $Q_1[v'/x] \parallel Q_2$. Es decir, se reemplazan las apariciones libres de la variable x en Q_1 por el valor v'. Si la cola es vacía, entonces el lado izquierdo del proceso Q queda bloqueado hasta que se almacene un valor en dicha cola. Por último, debemos destacar que los posibles usuarios del lenguaje pueden pasar por alto la mayoría de los detalles semánticos subyacentes a nuestro lenguaje. Realmente, un diseñador que utilice `VPASPA` necesita saber el significado intuitivo de los operadores pero no es necesario que posea un conocimiento exahustivo del (quizás demasiado complicado) modelo semántico.

A la hora de buscar un lenguaje de simulación apropiado, se ha encontrado en el paradigma funcional una propuesta realmente adecuada. En primer lugar, debido a la ausencia de efectos laterales, la programación funcional permite razonar ecuacionalmente acerca de los programas. En particular, es posible aplicar transformaciones automáticas. Además, la tarea de verificar, e incluso derivar, programas partiendo de su especificación formal es más sencilla. Entre los diferentes lenguajes funcionales perezosos existentes, actualmente se puede considerar que Haskell [PH99] es el lenguaje estándar. Este lenguaje representa el resultado de un esfuerzo conjunto de la comunidad funcional para definir un lenguaje estándar que pueda ser utilizado tanto para aplicaciones reales como en ámbitos académicos. Existen varios compiladores de Haskell, siendo el más ampliamente utilizado y más eficiente GHC (Glasgow Haskell Compiler) [PHHP92, Pey96]. Este compilador es fácilmente portable a diferentes arquitecturas, un interface COM permite interaccionar con programas escritos en otros lenguajes y además sus fuentes están disponibles de forma gratuita.

Desgraciadamente, Haskell esta basado en una estrategia de evaluación secuencial por lo que no se puede definir una ejecución concurrente de los programas. En los últimos años se han llevado a cabo algunas extensiones de este lenguaje donde se ha incluido la concurrencia (p.ej. [PGF96]) pero normalmente a través de primitivas concurrentes de muy *bajo nivel*. Por el contrario, el lenguaje de programación Eden [BLOP98, BKL+98] es una extensión concurrente de *alto nivel* de Haskell. Este lenguaje es el resultado de un trabajo de investigación conjunto de dos grupos, uno en Alemania y otro en España, y con el apoyo del equipo escocés

que desarrolla el compilador GHC. El principal logro del lenguaje es conseguir concurrencia y paralelismo con un alto nivel de abstracción pero sin perder eficiencia. Además de estas buenas propiedades, Eden incluye un conjunto de utilidades para realizar evaluación del rendimiento (llamado Paradise [HPR00]) que facilita el análisis de los programas, de modo que puede utilizarse para el estudio del comportamiento de los programas en tiempo de ejecución. Recientemente se ha finalizado el primer compilador completo de Eden y a partir de ese momento se han llevado a cabo una gran cantidad de pruebas y casos de estudio sobre él (véase p.ej. [PR01, KLPR01]).

Para hacer una transición suave de la especificación (en el álgebra de procesos) a la implementación (en el lenguaje Eden), vamos a dar un mecanismo para *traducir* de un formalismo al otro.

Con respecto al trabajo relacionado, debemos destacar que se han realizado varias traducciones de álgebras de procesos a otros lenguajes de programación (véase por ejemplo la traducción utilizando LOTOS dada en [MdM88], entre otros). En el área de las álgebras de procesos estocásticos, [D'A99] también ha estudiado la simulación e implementación de un álgebra de procesos estocásticos. Aunque en su trabajo también se utiliza un lenguaje de programación funcional (Haskell), nuestro trabajo se diferencia de éste en bastantes puntos. En su implementación se calcula, para cada punto de la ejecución, el conjunto de todos los posibles estados hacia los que puede evolucionar el sistema. Por tanto, sufre el inconveniente de la explosión de estados. Por el contrario, nuestra propuesta es una implementación realmente concurrente donde los procesos evolucionan independientemente. De hecho, los procesos se ejecutarán en paralelo. Por último, nuestro código generado es un programa normal de modo que, en caso de ser necesario, un usuario con cierto dominio en lenguajes funcionales puede optimizarlo a mano. Finalmente comentaremos que, al menos que nosotros conozcamos, no existen herramientas para analizar especificaciones sobre procesos estocásticos con distribuciones no restringidas a ser exponenciales. En el caso de sistemas donde las distribuciones de probabilidad deben ser exponenciales podemos citar TwoTowers [BCSS98] como una herramienta adecuada para el estudio de propiedades de rendimiento de dichos sistemas.

El resto del capítulo está organizado del siguiente modo. En la sección 6.2 se presentará el álgebra de procesos estocásticos con paso de valores. También daremos una noción de bisimulación para nuestro lenguaje. En la sección 6.3 presentamos las características básicas de Eden, concentrándonos principalmente en aquellas que se van a usar en este capítulo. En particular, se explicará la forma en que se definen los procesos en Eden. En la sección 6.4

presentamos la metodología para la traducción de las especificaciones desarrolladas sobre VPASPA a programas en Eden. El capítulo finalizará con dos ejemplos completos mostrando las características de nuestro método. Presentaremos su especificación, la traducción a Eden y el estudio de algunas propiedades cuantitativas.

6.2. Un álgebra de procesos estocásticos asíncrona con paso de valores

En esta sección se presenta nuestro lenguaje y su semántica operacional. Este modelo está basado en el lenguaje definido en [LN01], descrito en el capítulo 5, pero con importantes modificaciones. Hemos simplificado ligeramente el modelo eliminando las probabilidades asociadas al operador de elección. Por el contrario, hemos añadido algunas características nuevas para aumentar la mayor expresividad del lenguaje, con la correspondiente complicación del modelo semántico asociado. Hemos incluido el paso de valores y un operador de paralelo. Como ya hemos comentado en otras partes de esta tesis, este operador provoca un aumento de la dificultad a la hora de definir la semántica operacional del modelo. Sin embargo, el operador de composición paralela es extemadamente útil para modelar los sistemas en los que estamos interesados. Los procesos pueden sufrir retardos que van a estar representados de nuevo en nuestro lenguaje por variables aleatorias. Recordemos que los conceptos básicos referentes a las variables aleatorias aparecen en las definiciones 3.1 y 3.2. Por último, la comunicación entre procesos será asíncrona, de modo que se distingue entre acciones para transmitir un mensaje (acciones de salida) y acciones para recibirlas (acciones de entrada). A continuación definiremos nuestro alfabeto de acciones que servirán para especificar las correspondientes comunicaciones.

Definición 6.1 Consideramos un conjunto de acciones de *comunicación*, denotado por $\mathtt{Act}$, formado como unión de dos conjuntos $\mathtt{Act} = \mathtt{Input} \cup \mathtt{Output}$, tales que $\mathtt{Input} \cap \mathtt{Output} = \varnothing$. Suponemos que existe una biyección $f : \mathtt{Input} \longrightarrow \mathtt{Output}$, de modo que para cada acción de *entrada* $a? \in \mathtt{Input}$ tenemos que $f(a?)$ representa la acción de *salida* $a! \in \mathtt{Output}$. Si existe una transmisión de un mensaje entre $a?$ y $a!$ diremos que a es el *canal* de comunicación. $\mathcal{C}_{\mathtt{Act}}$ representa el conjunto de canales en $\mathtt{Act}$ ($a, b, \cdots$ denotan elementos de $\mathcal{C}_{\mathtt{Act}}$). Consideraremos un conjunto de valores *Val* para representar los mensajes transmitidos ($v, v', \ldots$ denotan elementos de *Val*) y un conjunto de variables de valores $\mathcal{X}$ ($x, y, \ldots$ denotan elementos de $\mathcal{X}$). Definimos el conjunto de *comunicaciones*, denotado por IO, como el conjunto de acciones de entrada y salida aplicadas a una variable de valores (en el caso de las acciones de entrada)

o a un valor (en el caso de las acciones de salida), es decir,

$$IO = \{c(x) \mid c \in \texttt{Input}, x \in \mathcal{X}\} \cup \{c(v) \mid c \in \texttt{Output}, v \in \mathit{Val}\}$$

Definimos el canal de una acción de comunicación $c(m) \in IO$, y lo denotamos por $ch(c(m))$, como a si $c \in \{a?, a!\}$. □

Consideraremos un conjunto numerable Id de identificadores de procesos. Además, denotaremos por $\mathcal{V}$ al conjunto de variables aleatorias ($\xi, \xi', \psi, \cdots$ denotan elementos del conjunto $\mathcal{V}$). También consideraremos el *conjunto de posiciones*

$$\mathrm{Loc} = \{loc_i \mid loc \in \{l, r\} \ \wedge \ i \in \mathbb{N}^+\}$$

donde l significa izquierda y r derecha. Como ya se indicó en la introducción, utilizaremos colas para el almacenamiento de los mensajes que ya se han transmitido pero todavía no se han recibido. En este caso consideraremos las operaciones habituales de las colas:

- [ ] denota la cola vacía.
- $enqueue(c, \sigma)$ denota que el elemento c se añade a la cola σ.
- Si σ no es vacía, $head(\sigma)$ devuelve el elemento más antiguo de σ; si la cola es vacía, entonces devuelve el valor indefinido $\perp$.
- Si σ no es vacía, $dequeue(\sigma)$ elimina $head(\sigma)$ de σ; si la cola es vacía, entonces devuelve el valor indefinido $\perp$.

A continuación definiremos la sintaxis de nuestra álgebra de procesos.

Definición 6.2 El conjunto de procesos, denotado por `VPASPA`, está definido según la siguiente expresión EBNF:

$$P ::= \mathit{stop} \mid \xi\,;P \mid a?(x)\,;P \mid a!(v)\,;P \mid P + P \mid \texttt{if}\ e\ \texttt{then}\ P\ \texttt{else}\ P \mid P \parallel^{\mu}_{M} P \mid P/A \mid \mathit{rec}\,X.P$$

donde $X \in Id$, $\xi \in \mathcal{V}$, $a?, a! \in \texttt{Act}$, $x \in \mathcal{X}$, $v \in \mathit{Val}$, $A \subseteq \mathcal{C}_{\texttt{Act}}$, e es una expresión tal que $\texttt{Eval}(e) \in \mathit{Bool}$ ($\texttt{Eval}(e)$ representa la evaluación de e), μ es un conjunto de colas definidas sobre Val y $M \subseteq (\mathcal{V} \times \mathbb{N}^+ \times \mathrm{Loc})$. □

Como en los capítulos anteriores, suponemos que todas las variables aleatorias que aparecen en la definición de los procesos son independientes. Por claridad, si dos retardos se representan con la misma variable aleatoria, suponemos que estos retardos tienen asociados realmente dos variables aleatorias independientes idénticamente distribuidas. A continuación, daremos una explicación intuitiva de cada operador de nuestro lenguaje.

- El término *stop* denota un proceso que no puede ejecutar ninguna acción.
- Un proceso $\xi\,;P$ espera una cantidad de tiempo aleatoria que viene determinada por la función de distribución de probabilidad asociada con ξ y después se comporta como el proceso P.
- El proceso $a?(x)\,;P$ espera hasta que recibe un valor v a través del canal a. Entonces, el proceso se comporta como $P[v/x]$, donde $P[v/x]$ denota la sustitución de todas las apariciones libres de x en P por v. Obsérvese que x está ligada en $a?(x)\,;P$.
- El proceso $a!(v)\,;P$ transmite el valor v por el canal a y después se comporta como P.
- El proceso $P+Q$ se comporta o bien como P o bien como Q dependiendo de qué componente ejecute la primera acción. Los retardos son una excepción porque no resuelven las elecciones. Más concretamente, el inicio de un retardo no resuelve la elección. Explicaremos este hecho con más detalle cuando presentemos la semántica operacional.
- El proceso `if` e `then` P `else` Q se comporta como P si la evaluación de la expresión booleana e es `true` y como Q en caso contrario.
- El término $P \parallel_M^{\mu} Q$ puede ejecutar acciones o bien de P o bien de Q. Las acciones de salida se pueden ejecutar inmediatamente mientras que las acciones de entrada deben esperar hasta que se haya recibido al menos un valor a través del canal correspondiente. El significado de los parámetros M y μ asociados a este operador lo explicaremos más adelante.
- El proceso P/A expresa que las acciones que pertenecen a A tienen que ocultarse.
- Finalmente, $recX.P$ denota la definición de un proceso (posiblemente recursivo).

Como veremos en la definición de la semántica operacional, las transiciones estocásticas se ejecutan en dos pasos: inicio y terminación. Para cada variable aleatoria ξ, denotamos por ξ^+ al inicio del retardo mientras que ξ^- denota su terminación. Este mecanismo, introducido en [BG02b], es uno de los mecanismos existentes para tratar distribuciones generales en el contexto de un operador de paralelo. De ese modo, una transición estocástica empieza con ξ^+ pero no finalizará hasta que se ejecute ξ^-. En cuando a los parámetros de la composición paralela, μ almacena todos los mensajes que se van enviando, pero que todavía no se han recibido. Utilizaremos una cola independiente para cada canal. Inicialmente, supondremos que todas las colas de los canales están vacías, es decir, μ contiene una cola vacía para cada $a \in \mathcal{C}_{\texttt{Act}}$. En este caso, decimos que I es el valor inicial del parámetro μ. Debido a la

división en dos partes de las transiciones estocásticas, para evitar efectos *indeseables* con algunos procesos recursivos, añadiremos índices a los retardos. Por ejemplo, en el proceso $rec\,X.(\xi\,;stop)\,\|^{\mu}_{M}\,(a\,;X)$ podría haber varios ejemplares de la acción estocástica de inicio ξ^{+} ejecutándose en paralelo. Para poder asociar cada acción de terminación ξ^{-} con su correspondiente inicio, asignamos un índice que denotará la *posición* en la que ξ^{+} se ha realizado. Está asignación de índices queda almacenada en el conjunto M, y a medida que los retardos vayan finalizando, este conjunto se vaciará. Inicialmente, suponemos que $M = \varnothing$. Debemos remarcar que los índices no se necesitarían si restringiéramos las apariciones del operador de paralelo en el ámbito de las definiciones recursivas. En este caso, podríamos eliminar el parámetro M.

En la definición de la semántica operacional hemos extendido el alfabeto con una acción interna τ. Aunque τ no está incluida explícitamente en la sintaxis del lenguaje, el operador de ocultamiento puede generar acciones internas. Además, consideraremos el conjunto de acciones de inicio $\mathcal{V}^{+} = \{\xi_i^{+} \mid \xi \in \mathcal{V} \ \wedge \ i \in \mathbb{N}^{+}\}$ y el conjunto de acciones de terminación $\mathcal{V}^{-} = \{\xi_i^{-} \mid \xi \in \mathcal{V} \ \wedge \ i \in \mathbb{N}^{+}\}$. Utilizaremos los siguientes convenios: $P \xrightarrow{\omega} P'$ expresa que existe una transición desde P a P' etiquetada con la acción ω; $P \xrightarrow{\omega}$ indica que existe $P' \in \texttt{VPASPA}$ tal que $P \xrightarrow{\omega} P'$; escribimos $P \not\xrightarrow{\omega}$ si no existe $P' \in \texttt{VPASPA}$ tal que $P \xrightarrow{\omega} P'$. Para simplificar la presentación en primer lugar presentamos las reglas operacionales asociadas con la ejecución de acciones por parte de procesos (véase figura 6.1), mientras que en la figura 6.2 daremos las reglas operacionales que sirven para definir el comportamiento estocástico de los procesos.

A continuación, explicaremos las reglas que aparecen en la figura 6.1. Las etiquetas que aparecen en las transiciones operacionales aquí definidas tienen los siguientes tipos: $a \in \mathcal{C}_{\texttt{Act}}$, $\xi \in \mathcal{V}$, $\alpha \in IO \cup \{\tau\}$ y $\omega \in IO \cup \{\tau\} \cup \mathcal{V}^{+} \cup \mathcal{V}^{-}$. Las primeras cuatro reglas son bastante estándar en la definición de semánticas operacionales. Las dos primeras indican que un proceso prefijado por una acción de comunicación tiene la capacidad de ejecutar esa acción. Las dos siguientes definen el comportamiento del operador de elección en la forma usual, es decir, tanto las acciones de comunicación como las acciones internas, puesto que $\alpha \in IO \cup \{\tau\}$, resuelven las elecciones (como veremos en la figura 6.2, esto no es cierto para las acciones estocásticas). Las siguientes seis reglas definen el comportamiento del operador de paralelo. Las dos reglas de la izquierda indican que si una componente de la composición paralela puede ejecutar acciones de salida $a!(v)$ entonces se ejecutará esta acción. El valor v transimitido se almacena en la cola asociada al canal a, es decir μ_a, y el conjunto de colas se modifica adecuadamente utilizando la función *put*. Esta función elimina la antigua

$$\frac{}{a?(x);P \xrightarrow{a?(x)} P} \qquad \frac{}{a!(v);P \xrightarrow{a!(v)} P} \qquad \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$$

$$\frac{P \xrightarrow{a!(v)} P'}{P \|^{\mu}_{M} Q \xrightarrow{a!(v)} P' \|^{\mu'}_{M} Q} \qquad \frac{P \xrightarrow{a?(x)} P',\ \mu_a \neq [\,],\ v = head(\mu_a)}{P \|^{\mu}_{M} Q \xrightarrow{a?(v)} P'[v/x] \|^{\mu''}_{M} Q} \qquad \frac{P \xrightarrow{\tau} P'}{P \|^{\mu}_{M} Q \xrightarrow{\tau} P' \|^{\mu}_{M} Q}$$

$$\frac{Q \xrightarrow{a!(v)} Q'}{P \|^{\mu}_{M} Q \xrightarrow{a!(v)} P \|^{\mu'}_{M} Q'} \qquad \frac{Q \xrightarrow{a?(x)} Q',\ \mu_a \neq [\,],\ v = head(\mu_a)}{P \|^{\mu}_{M} Q \xrightarrow{a?(v)} P \|^{\mu''}_{M} Q'[v/x]} \qquad \frac{Q \xrightarrow{\tau} Q'}{P \|^{\mu}_{M} Q \xrightarrow{\tau} P \|^{\mu}_{M} Q'}$$

$$\frac{P[X/X{:=}P] \xrightarrow{\omega} P'}{X{:=}P \xrightarrow{\omega} P'} \qquad \frac{P \xrightarrow{\omega} P' \ \wedge\ ch(\omega) \notin A}{P/A \xrightarrow{\omega} P'/A} \qquad \frac{P \xrightarrow{\omega} P' \ \wedge\ ch(\omega) \in A}{P/A \xrightarrow{\tau} P'/A}$$

$$\frac{\texttt{Eval}(e),\ P \xrightarrow{\omega} P'}{\texttt{if } e \texttt{ then } P \texttt{ else } Q \xrightarrow{\omega} P'} \qquad \frac{\neg\texttt{Eval}(e),\ Q \xrightarrow{\omega} Q'}{\texttt{if } e \texttt{ then } P \texttt{ else } Q \xrightarrow{\omega} Q'}$$

$$\mu' = put(\mu, \mu_a, v) \qquad \mu'' = get(\mu, \mu_a)$$
$$put(\mu, \mu_a, v) = \mu - \{\mu_a\} \cup \{enqueue(v, \mu_a)\} \qquad get(\mu, \mu_a) = \mu - \{\mu_a\} \cup dequeue(\mu_a)$$

Figura 6.1: Semántica operacional de **VPASPA** (1/2).

cola μ_a, asociada al canal a, y añade la cola $enqueue(v, \mu_a)$. Las reglas de la parte central tratan las acciones de entrada. Si una componente de la composición paralela puede ejecutar una acción de entrada $a?(x)$ y la cola correspondiente al canal a no está vacía, entonces se ejecuta esta acción. En este caso, la cabeza de la cola correspondiente, $head(\mu_a)$, se pasa a la componente que ha ejecutado $a?(x)$ y se reemplazan todas las apariciones libres de x por ese valor. Entonces, el conjunto de colas se modifica utilizando la función *get*. Esta función elimina la cola existente asociada al canal a, es decir μ_a, y añade una nueva en la que el valor más antiguo se ha eliminado, es decir, añade la cola $dequeue(\mu_a)$. Finalmente, las dos reglas de la parte derecha indican que si un proceso de la composición puede evolucionar de forma interna, esta transición también es posible por parte de la composición paralela.

Ejemplo 6.3 Consideremos los procesos $P_1 = a?(x)\,;Q_1$ y $P_2 = a!(v)\,;b?(x)\,;Q_2$. Sea $P = P_1 \|^{I}_{\varnothing} P_2$ la composición paralela de ambos procesos. Tenemos que P sólo puede ejecutar la transición $P \xrightarrow{a!(v)} P_1 \|^{\mu}_{\varnothing} b?(x)\,;Q_2$, donde μ es el conjunto de colas tales que para cualquier $a' \in \mathcal{C}_{\texttt{Act}-\{a\}}$ tenemos $\mu_{a'} = [\,]$ mientras que $\mu_a = [v]$. Consideremos ahora $P' = P_1 \|^{\mu}_{\varnothing} b?(x)\,;Q_2$. La componente derecha de P' está bloqueada hasta que se reciba un

valor a través del canal b. Por lo tanto, en ese momento la única transición disponible es $P' \xrightarrow{a?(v)} Q_1[v/x] \parallel^I_\varnothing b?(x)\,;Q_2$. Nótese que el valor que se había almacenado anteriormente en μ_a se ha consumido. De ese modo, el conjunto de colas vuelve a contener otra vez todas las colas vacías. □

Las siguientes tres reglas son estándar para la definición del comportamiento de los operadores de recursión y ocultamiento. Obsérvese que estas reglas se aplican tanto a acciones de comunicación como a acciones estocásticas, así que no aparecerán repetidas en la figura 6.2. Finalmente, las dos últimas reglas expresan el significado habitual del operador condicional. Como en el caso de las tres reglas anteriores, ω recorre el conjunto de acciones de comunicación y también el de acciones estocásticas.

A continuación presentaremos las reglas operacionales que tratan acciones estocásticas (véase figura 6.2). La definición de esta parte de la semántica operacional está inspirada en [BG02b]. Antes de explicar detalladamente estas reglas vamos a definir todos los predicados y funciones que aparecen en dicha figura. En primer lugar, vamos a necesitar saber en cada momento el conjunto de acciones que el proceso puede ejecutar. Dividiremos las acciones en dos tipos: las estocásticas y las no estocásticas. Así definimos los siguientes conjuntos

Definición 6.4 Definimos el *conjunto de acciones iniciales de un proceso* P, denotado por $\texttt{Init}(P)$, como el conjunto de acciones visibles e internas que un proceso puede ejecutar inmediatamente, es decir,

$$\begin{array}{lcl}
\texttt{Init}(\mathit{stop}) = \texttt{Init}(\xi\,;P) = \varnothing \\
\texttt{Init}(a?(x)\,;P) = \texttt{Init}(a!(v)\,;P) = a \\
\texttt{Init}(P+Q) = \texttt{Init}(P \parallel^\mu_M Q) = \texttt{Init}(P) \cup \texttt{Init}(Q) \\
\texttt{Init}(P/A) = \begin{cases} \texttt{Init}(P) & \text{si } \texttt{Init}(P) \cap A = \varnothing \\ \texttt{Init}(P) - A \cup \{\tau\} & \text{si } \texttt{Init}(P) \cap A \neq \varnothing \end{cases}
\end{array}$$

Definimos el *conjunto de acciones estocásticas iniciales de un proceso* P, denotado por $\texttt{InStoc}(P)$, como el conjunto de acciones estocásticas que puede iniciar el proceso inmediatamente, es decir,

$$\begin{array}{lcl}
\texttt{InStoc}(\mathit{stop}) = \texttt{InStoc}(a?(x)\,;P) = \texttt{InStoc}(a!(v)\,;P) = \varnothing \\
\texttt{InStoc}(\xi\,;P) = \{\xi\} \\
\texttt{InStoc}(P+Q) = \texttt{InStoc}(P \parallel^\mu_M Q) = \texttt{InStoc}(P) \cup \texttt{InStoc}(Q) \\
\texttt{InStoc}(P/A) = \begin{cases} \texttt{InStoc}(P) & \text{si } \texttt{Init}(P) \cap A = \varnothing \\ \varnothing & \text{si } \texttt{Init}(P) \cap A \neq \varnothing \end{cases}
\end{array}$$

□

Por otro lado, nótese que si un proceso puede ejecutar una acción interna entonces no se permiten transiciones estocásticas. Esta propiedad (ya explicada en el capítulo 3) se denomina habitualmente *urgencia* y aparece en la mayoría de las álgebras temporales y estocásticas. Por lo tanto definiremos un predicado auxiliar de *estabilidad*. Intuitivamente, diremos que un proceso P es *estable*, denotado por $stab(P)$, si P no puede ejecutar inmediatamente transiciones internas.

Definición 6.5 Sea $P \in \texttt{VPASPA}$. Definimos por inducción estructural el predicado $stab(P)$ como:

$$stab(stop) = stab(\xi\,;P) = stab(a?(x)\,;P) = stab(a!(v)) = \texttt{true}$$
$$stab(P+Q) = stab(P \parallel_M^{\mu} Q) = stab(P) \ \wedge \ stab(Q)$$
$$stab(P/A) = (Init(P) \cap A = \varnothing)$$

□

Por último, definiremos de nuevo un concepto que apareció en el capítulo 4, el tiempo de espera máximo de un proceso, es decir, el tiempo máximo que puede transcurrir antes de que el proceso tenga que ejecutar una acción interna o una estocástica.

Definición 6.6 Definimos el *tiempo de espera máximo de un proceso* P, denotado por $maxW(P)$, en la siguiente forma

$$maxW(stop) = maxW(a?(x)\,;P) = maxW(a!(v)\,;P) = \infty$$
$$maxW(\xi\,;P) = \text{mín}\{t \mid \mathrm{P}(\xi \leq t) = 1\}$$
$$maxW(P+Q) = maxW(P \parallel_M^{\mu} Q) = \text{mín}(maxW(P), maxW(Q))$$
$$maxW(P/A) = \begin{cases} maxW(P) & \text{si } \texttt{Init}(P) \cap A = \varnothing \\ 0 & \text{si } \texttt{Init}(P) \cap A \neq \varnothing \end{cases}$$

□

En la tabla que aparece en la figura 6.2, las reglas de la izquierda consideran la ejecución de acciones de inicio y las de la derecha de acciones de terminación. Una transición operacional de la forma $\xi\,;P \xrightarrow{\xi_1^+} \xi_1^-\,;P$ indica el inicio de un retardo dado por ξ. De forma similar, $\xi_1^-\,;P \xrightarrow{\xi_1^-} P$ denota la terminación de ese retardo. Consideramos que el inicio de un retardo tiene mayor prioridad que cualquier acción de terminación. La idea consiste en tener en cuenta que la ejecución de una acción de terminación supone el paso de un cierto tiempo, mientras que la ejecución de una acción de inicio no. Así, si considerásemos que una acción

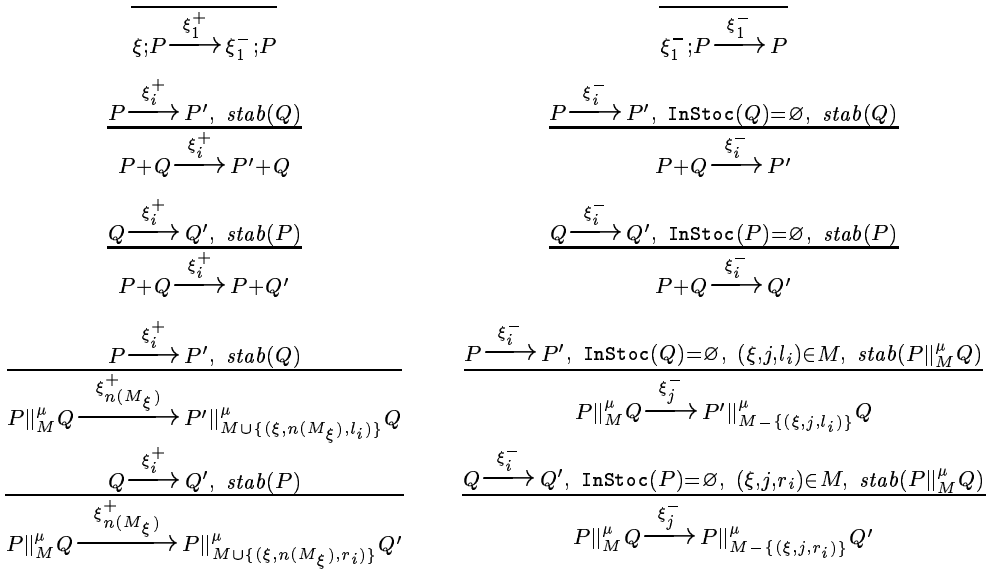


Figura 6.2: Semántica operacional de VPASPA (2/2).

de terminación tuviera prioridad sobre las de inicio, el proceso $P = \xi\,;\alpha_1 \parallel^I_\varnothing \psi\,;\alpha_2$ ejecutaría $P \xrightarrow{\xi_1^+} P' = \xi_1^-\,;\alpha_1 \parallel^I_{\{\xi,1,l_1\}} \xi_2\,;\alpha_2$ y después $P' \xrightarrow{\xi_1^-} \alpha_1 \parallel^I_\varnothing \xi_2\,;\alpha_2$. En este caso, en uno de las dos componentes del paralelo habría transcurrido un tiempo dado por la variable aleatoria ξ, mientras que en la otra componente no habría pasado tiempo. Para evitar este tipo de situaciones, una acción estocástica no puede *completarse* si puede empezar otra acción estocástica. Como ya hemos explicado antes, ponemos índices a las acciones estocásticas para distinguir las distintas apariciones de una misma variable. En el caso del prefijo añadimos la etiqueta 1 (en el contexto del operador de paralelo es necesario un mecanismo más complejo para asignar índices que explicaremos más adelante).

Las siguientes cuatro reglas definen el comportamiento del operador de elección cuando las acciones a ejecutar son estocásticas. En las cuatro hemos considerado que no se puede ejecutar una acción estocástica si alguna componente de la elección puede evolucionar internamente. Las reglas de la parte izquierda indican que una de las componentes puede empezar a ejecutar un retardo. Obsérvese que las acciones de inicio no resuelven las elecciones porque únicamente denotan la posibilidad de retardar un proceso. El retardo finaliza cuando se ejecuta la correspondiente acción de terminación, cuya ejecución viene descrita en las reglas de la parte derecha. Como ya hemos comentado previamente, las acciones de inicio tienen

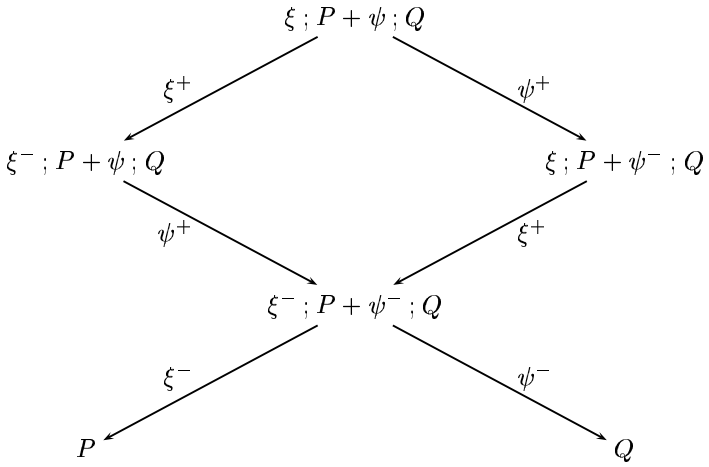


Figura 6.3: Ejemplo de política de carreras.

prioridad sobre las de terminación. El predicado $\texttt{InStoc}(Q) = \varnothing$ indica que el proceso Q no puede iniciar ninguna acción estocástica, así pues P podrá finalizar el retardo. Además, como se muestra en estas reglas, las acciones de terminación resuelven las elecciones. Esta interpretación del operador de elección sigue la llamada *política de carreras*, es decir, se resuelve la elección a favor de la acción más rápida.

Ejemplo 6.7 Consideremos el proceso $P = \xi\,;P_1 + \psi\,;P_2$. Tenemos que P puede ejecutar las transiciones $P \xrightarrow{\xi_1^+} Q_1$ y $P \xrightarrow{\psi_1^+} Q_2$, donde estos procesos son $Q_1 = \xi_1^-\,;P_1 + \psi\,;P_2$ y $Q_2 = \xi\,;P_1 + \psi_1^-\,;P_2$. Entonces, Q_1 puede ejecutar únicamente la transición $Q_1 \xrightarrow{\psi_1^+} \xi_1^-\,;P_1 + \psi_1^-\,;P_2$ mientras que Q_2 puede ejecutar la transición $Q_2 \xrightarrow{\xi_1^+} \xi_1^-\,;P_1 + \psi_1^-\,;P_2$. Es decir, en ambos casos tenemos que después de dos transiciones el proceso P evoluciona al proceso $R = \xi_1^-\,;P_1 + \psi_1^-\,;P_2$. Finalmente, R puede ejecutar una de las dos acciones de terminación. En ambos casos, se resuelve la elección. Es decir, tenemos una de las dos transiciones siguientes $R \xrightarrow{\xi_1^-} P_2$ o $R \xrightarrow{\psi_1^-} P_1$. En cada caso consideramos que P se retardará de acuerdo con la variable aleatoria ξ (ψ, respectivamente). La interpretación intuitiva de estas transiciones es que el proceso P se retardará una cantidad de tiempo que dependerá de qué variable aleatoria tome el menor valor (de acuerdo con sus correspondientes funciones de distribución de probabilidad). En la figura 6.3 aparece gráficamente el comportamiento operacional descrito en este ejemplo.

□

El comportamiento del operador de composición paralela está definido en las siguientes cuatro reglas. Como en el caso anterior, las acciones internas tienen prioridad sobre los retardos y las acciones de inicio tienen prioridad sobre las de terminación. Si cualquiera de los procesos de la composición puede ejecutar una acción de inicio y el proceso es estable, véase las reglas de la parte izquierda, entonces la acción de inicio se ejecuta. Sin embargo, hay que dar un índice a la acción (en la definición 6.8 damos la formulación de la función que determina este nuevo índice). Para dar índices a las variables aleatorias utilizamos nombres dinámicos. Este índice será el mínimo número natural que no está siendo usado en estos momentos para etiquetar otras apariciones de la misma variable aleatoria en el contexto de ese proceso. Cuando la transición estocástica se completa se libera el índice correspondiente. El conjunto de índices asociados con el operador de paralelo, M, almacena los índices que se están utilizando en cada momento y las *posiciones* con respecto al operador de paralelo de las correspondientes variables aleatorias, para conocer qué componente está ejecutando la acción y qué índice tenía asociado fuera del paralelo. Los índices dinámicos que vamos a ir asociando a cada variable aleatoria que se empieza a ejecutar, en el ámbito del paralelo, vendrán dados por la siguiente función que da el menor índice que no está asociado, en ese momento de la ejecución, a esa variable aleatoria en concreto.

Definición 6.8 Sea $\xi \in \mathcal{V}$, $M \subseteq \mathcal{V} \times \mathbb{N}^+ \times \mathrm{Loc}$. Definimos el *mínimo valor no utilizado en M para etiquetar con un índice a ξ*, denotado por $n(M_\xi)$, como

$$n(M_\xi) = \min\left(\{j \in \mathbb{N}^+ \mid \nexists (\xi, j, loc_i) \in M, loc \in \{l, r\}, i \in \mathbb{N}^+ \ \wedge \ 1 \leq j \leq mx\} \cup \{mx + 1\}\right)$$

donde $mx = \max\{j \in \mathbb{N}^+ \mid (\xi, j, loc_i) \in M, loc \in \{l, r\}, i \in \mathbb{N}^+\}$. □

Por ejemplo, si almacenamos en M la tupla $(\xi, n(M_\xi), l_i)$, esto significa que se ha ejecutado ξ^+ en la parte izquierda del operador de paralelo con un índice asociado dado por $n(M_\xi)$ debido a que P ha ejecutado ξ_i^+.

Si P puede ejecutar la acción de terminación ξ_i^-, se cumple $(\xi, j, l_i) \in M$ y además no se pueden ejecutar ni acciones internas ni acciones de inicio, entonces la composición paralela ejecutará la misma acción de terminación pero con el índice j. Además, el índice correspondiente se borra de M por lo que queda liberado. Tenemos una situación similar para Q y (ξ, j, r_i).

A continuación vamos a definir el conjunto de *acciones de entrada* y *acciones de salida* que los procesos pueden ejecutar. Utilizaremos estos conjuntos en la sección 6.4 a la hora de definir el mecanismo de traducción de los procesos descritos en `VPASPA` a programas en Eden.

Definición 6.9 Definimos inductivamente el conjunto de *acciones de entrada* que un proceso P puede ejecutar, denotado por $\texttt{Inputs}(P)$, como:

$$\texttt{Inputs}(\mathit{stop}) = \varnothing$$
$$\texttt{Inputs}(\xi\,;P) = \texttt{Inputs}(P)$$
$$\texttt{Inputs}(a?(x)\,;P) = \texttt{Inputs}(P) \cup \{a\}$$
$$\texttt{Inputs}(a!(v)\,;P) = \texttt{Inputs}(P)$$
$$\texttt{Inputs}(P + Q) = \texttt{Inputs}(P) \cup \texttt{Inputs}(Q)$$
$$\texttt{Inputs}(P \parallel_M^{\mu} Q) = \texttt{Inputs}(P) \cup \texttt{Inputs}(Q)$$
$$\texttt{Inputs}(P/A) = \texttt{Inputs}(P) - A$$
$$\texttt{Inputs}(X := P) = \texttt{Inputs}(P)$$
$$\texttt{Inputs}(\texttt{if } e \texttt{ then } P \texttt{ else } Q) = \texttt{Inputs}(P) \cup \texttt{Inputs}(Q)$$

Denotamos por $\texttt{Outputs}(P)$ al conjunto de *acciones de salida* que P puede ejecutar. Formalmente:

$$\texttt{Outputs}(\mathit{stop}) = \varnothing$$
$$\texttt{Outputs}(\xi\,;P) = \texttt{Outputs}(P)$$
$$\texttt{Outputs}(a?(x)\,;P) = \texttt{Outputs}(P)$$
$$\texttt{Outputs}(a!(v)\,;P) = \texttt{Outputs}(P) \cup \{a\}$$
$$\texttt{Outputs}(P + Q) = \texttt{Outputs}(P) \cup \texttt{Outputs}(Q)$$
$$\texttt{Outputs}(P \parallel_M^{\mu} Q) = \texttt{Outputs}(P) \cup \texttt{Outputs}(Q)$$
$$\texttt{Outputs}(P/A) = \texttt{Outputs}(P) - A$$
$$\texttt{Outputs}(X := P) = \texttt{Outputs}(P)$$
$$\texttt{Outputs}(\texttt{if } e \texttt{ then } P \texttt{ else } Q) = \texttt{Outputs}(P) \cup \texttt{Outputs}(Q)$$

□

Finalmente, nos gustaría remarcar que buscando un equilibrio entre la complejidad del modelo semántico y las capacidades del lenguaje, hemos decidido optar por un lenguaje muy expresivo. Realmente, un diseñador que utilice nuestro lenguaje puede aprovechar las características complejas, simplemente conociendo una interpretación intuitiva del funcionamiento de los operadores, sin necesidad de manejar los detalles semánticos.

6.2.1. Semánticas de bisimulación estocástica para VPASPA

En esta sección definiremos dos nociones de bisimulación (fuerte y débil) para nuestro lenguaje. En primer lugar, utilizaremos la función $\oplus$ presentada en la definición 3.2 para

computar la variable aleatoria distribuida como el mínimo de un conjunto de variables aleatorias. Recordemos brevemente que si ξ_1 y ξ_2 son dos variables aleatorias independientes con funciones de distribución F_{ξ_1} y F_{ξ_2}, la *suma combinada* de ξ_1 y ξ_2 es la variable aleatoria que tiene como función de distribución $F_{\xi_1 \oplus \xi_2}(x) = F_{\xi_1}(x) + F_{\xi_2}(x) - F_{\xi_1}(x) \cdot F_{\xi_2}(x)$. Debido a que la resolución de una elección entre dos variables aleatorias se realiza mediante una política de carreras y también debido a la urgencia de las acciones internas, la definición de bisimulación fuerte no es tan simple como en otro modelos estocásticos. Como ya se vió en el capítulo 4, al tener en cuenta estas características del lenguaje existen procesos que aparentemente no son bisimilares, pero que realmente sí lo son. Para definir adecuadamente esta bisimulación recordaremos unos predicados y una función que dimos en ese capítulo.

Definición 6.10 Sean P, Q dos procesos y ξ, ψ dos variables aleatorias. Decimos que ξ y ψ están *idénticamente distribuidas con respecto a* P y Q, denotado por $\xi \asymp_{P,Q} \psi$, si $maxW(P) = maxW(Q)$ y para cualquier $t \leq maxW(P)$, $\mathrm{P}(\xi \leq t) = \mathrm{P}(\psi \leq t)$.

Sean $\xi \in \mathcal{V}$ una variable aleatoria y $t_0 \in \mathbb{R}^+$. Decimos que ξ *es factible en tiempo* t_0, denotado por $fact_{t_0}(\xi)$, sii $t_0 > \mín\{t \mid \mathrm{P}(\xi \leq t) > 0\}$.

Sean P un proceso y ξ una variable aleatoria. Decimos que un conjunto C de procesos es *realmente alcanzable* por P, denotado por $P \nearrow C$, si existen $P' \in C$ y $\psi \in \mathcal{V}$ tales que $P \xrightarrow{\psi} P'$ y $fact_{maxW(P)}(\psi)$.

Sean $P \in \texttt{VPASPA}$ y $C \subseteq \texttt{VPASPA}$. Definimos la *estimación acumulada de llegar al conjunto* C *desde* P como $\xi_s(P, C) = \oplus\{\xi \mid P \xrightarrow{\xi^+} P' \text{ y } P' \in C\}$. □

Obsérvese que la semántica operacional permite generar todas las transiciones estocásticas sean factibles o no. Por tanto, a la hora de calcular la estimación acumulada de llegar a un cierto conjunto C desde un proceso P se tendrán en cuenta todas las transiciones. Sin embargo, como veremos en el siguiente resultado, la combinación de dos variables aleatorias donde sólo una es factible dará como resultado únicamente la factible.

Lema 6.11 Sean $\xi, \psi \in \mathcal{V}$ variables aleatorias independientes. Sean $t_\xi = \mín\{t \mid F_\xi(t) = 1\}$ y $t_\psi = \mín\{t \mid F_\psi(t) = 1\}$. Si $t_\xi < \mín\{t \mid F_\psi(t) > 0\}$ entonces $\xi \oplus \psi$ y ξ están idénticamente distribuidas.

Demostración: El resultado se demuestra trivialmente aplicando

$$F_{\xi \oplus \psi}(t) = F_\xi(t) + F_\psi(t) - F_\xi(t) \cdot F_\psi(t)$$

□

A continuación, presentamos nuestras nociones de bisimulación.

Definición 6.12 Una relación de equivalencia $\mathcal{R}$ sobre VPASPA es una *bisimulación estocástica fuerte* si y sólo si para cualquier par de procesos $P, Q \in$ VPASPA tenemos que $(P, Q) \in \mathcal{R}$ implica que para cualquier $\alpha \in IO \cup \{\tau\}$ y cualquier $C \in$ VPASPA$/\mathcal{R}$:

- Si $P \xrightarrow{\alpha} P'$ entonces existe Q' tal que $Q \xrightarrow{\alpha} Q'$ y $(P', Q') \in \mathcal{R}$.
- Si $P \nearrow C$ entonces $Q \nearrow C$ y $\xi_s(P, C) \asymp_{P,Q} \xi_s(Q, C)$.

Decimos que dos procesos $P, Q \in$ VPASPA son *bisimilares estocásticos fuertes*, denotado por $P \sim_s Q$, si existe una bisimulación estocástica fuerte que contenga el par $\langle P, Q \rangle$. □

Definición 6.13 Una relación de equivalencia $\mathcal{R}$ sobre VPASPA es una *bisimulación estocástica débil* si y sólo si para cualquier par de procesos $P, Q \in$ VPASPA tenemos que $(P, Q) \in \mathcal{R}$ implica que para cualquier $\alpha \in IO \cup \{\tau\}$ y cualquier $C \in$ VPASPA$/\mathcal{R}$:

- Si $P \xrightarrow{\alpha} P'$ entonces existe Q' tal que $Q \stackrel{\alpha}{\Longrightarrow} Q'$ y $(P', Q') \in \mathcal{R}$.
- Si $P \Longrightarrow P' \nearrow C^\tau$, existe Q' tal que $Q \Longrightarrow Q' \nearrow C^\tau$ y $\xi_s(P', C^\tau) \asymp_{P',Q'} \xi_s(Q', C^\tau)$.

Decimos que dos procesos $P, Q \in$ VPASPA son *bisimilares estocásticos débiles*, denotado por $P \sim_w Q$, si existe una bisimulación estocástica débil que contenga el par $\langle P, Q \rangle$. □

El siguiente resultado es similar a los ya presentados en el capítulo 4.

Lema 6.14 La bisimilitud estocástica fuerte (débil) sobre VPASPA es una relación de equivalencia sobre VPASPA. Es más, $\sim_s$ ($\sim_w$) es la bisimulación estocástica fuerte (débil) más grande sobre VPASPA.

6.3. El lenguaje funcional concurrente Eden

En esta sección damos una descripción somera de las principales características de Eden, centrándonos en las más relacionadas con el presente trabajo.[1] En particular, omitiremos la descripción del mecanismo de tipado. En cuanto al paradigma funcional puro daremos una breve explicación de un mecanismo muy útil para mejorar la comprensión de los programas funcionales: ajuste de patrones. La mayoría de los programas escritos utilizando ajuste de patrones se pueden reescribir utilizando construcciones `if then else`, pero el resultado de los programas no es tan elegante.

[1] Una presentación completa del lenguaje Eden puede verse en [BLOP98].

Un programa funcional consiste en una lista de definiciones de funciones. Cada función puede definirse en términos de varias reglas. Utilizamos ajuste de patrones para especificar cuándo se puede utilizar una regla. Podremos aplicar una regla sólo si los argumentos se *ajustan* al *patrón* correspondiente. Por ejemplo, en el siguiente ejemplo se aplica la primera regla si la segunda entrada se *ajusta* al *patrón* asociado a la lista vacía, mientras que la segunda regla se aplica en caso contrario:

```
map f []     = []
map f (x:xs) = f x : map f xs
```

donde `[]` denota una lista vacía y `(x:xs)` denota una lista cuya cabeza es `x` y cuyo resto es `xs`. Como se muestra más abajo, la función `map` se puede definir también sin utilizar ajuste de patrones. Sin embargo, en general, las versiones con ajuste de patrones suelen ser más cortas y claras.

```
map f xs = if empty xs then []
                       else f x : map f xs
```

A partir de este momento, nos concentraremos en las características concurrentes de Eden. Eden extiende a Haskell con construcciones sintácticas para definir y crear procesos concurrentes. Se distingue entre *abstracciones de procesos* y *concreciones de procesos*. Las primeras se utilizan para definir el comportamiento de los procesos, pero sin crearlos. Las segundas se utilizan para crear concreciones de los procesos. Esta distinción permite la generación de tantas instancias del mismo proceso como sea necesario. Eden incluye una nueva expresión `process x -> e` donde `x` es la entrada (o entradas) y `e` es la salida (o salidas). Las abstracciones de procesos se pueden comparar con funciones, consistiendo la principal diferencia en que las primeras, cuando se crean, se ejecutan en un proceso independiente. Por ejemplo, el siguiente proceso tiene dos entradas y dos salidas. Recibe como entradas un entero y una cadena y produce como salidas una cadena y un entero.

```
p = process (n,s) -> (out1,out2)
 where (out1,out2) = f n s
       f 0 s = ("hola",length s)
       f n s = ("adios",length s)
```

Lo primero que este proceso necesita conocer es si el primer parámetro es cero o no. Por lo tanto, debe esperar hasta que reciba un valor a través del primer canal de entrada. Después, dependiendo del valor, dará como salida o bien la cadena 'hola' y la longitud de la segunda

entrada (si el valor es 0) o la cadena 'adios' y también la longitud de la segunda entrada (en otro caso).

Se logra una *concreción de un proceso* utilizando el operador infijo predefinido `(#)`. Cada vez que se encuentra una ligadura `outputs = p # inputs` se crea un nuevo proceso para evaluar la abstracción `p`. Este nuevo proceso será capaz de recibir valores a través de los canales de entrada asociados a `inputs`, y enviar valores a través de `outputs`. Se detectarán los lectores reales de `outputs` y los productores reales de `inputs` a través de las dependencias de datos de los procesos. Por ejemplo, consideremos la siguiente abstracción del proceso `q`.

```
q = process () -> (c,g)
  where (a,b,c) = p1 # (e,f)
        (d,e)   = p2 # (h,b)
        (f,g,h) = p3 # (a,d)
```

En este caso, `p1` podrá recibir datos de la segunda salida de `p2` y de la primera salida de `p3`. El lector de `c` y `g` dependerá del contexto en el cual se cree el proceso `q`. Además de las abstracciones y concreciones de procesos, Eden proporciona una función predefinida, llamada `merge`, que se puede utilizar para combinar varias listas en una sola. Este mecanismo es muy útil para hacer una selección de una lista de alternativas. Por ejemplo, en el caso de querer seleccionar entre dos alternativas se puede utilizar la siguiente función `merge2`:

```
data Either a b = Left a | Right b
merge2 xs ys = merge [map Left xs, map Right ys]
```

Después de combinar dos listas `xs` e `ys` podemos utilizar ajuste de patrones sobre la lista mezclada para saber cuales de las entradas vienen de la lista de la *izquierda* y cuales de la lista de la *derecha*. Esta función será muy útil para elegir entre dos alternativas posibles:

```
... f (merge2 xs ys) ...
f (Left x : zs ) = ...
f (Right y : zs) = ...
```

Ahora presentaremos un ejemplo en Eden. El siguiente proceso define el comportamiento de un reloj simple (que llamaremos `timer`).

```
timer t = process starts -> f starts
  where f (x:xs) = seq (sleep t) (TimeOut : f xs)
```

Un reloj recibe como entrada una lista de señales. Cuando recibe un `Start`, el proceso `timer` espera `t` segundos. Una vez transcurrido este tiempo, comunica un `TimeOut`, es decir, indica que el tiempo ha acabado, y después empieza otra vez, de manera recursiva, esperando un nuevo `Start`. La función `seq` se utiliza para componer secuencialmente dos acciones y `sleep` es una primitiva de Haskell para retardar un proceso. Obsérvese que Eden no incluye constructores para *enviar* o *recibir* mensajes. En este caso la comunicación es automática, y sólo ocurre por la dependencia de datos de los procesos. Este mecanismo es parecido al de la comunicación en las álgebras de procesos. Téngase también en cuenta que la entrada del proceso `timer` es una lista de mensajes de *inicio*, es decir, la lista `starts`, y la salida también es una lista. De hecho, los procesos en Eden utilizan listas para ambos canales, el de entrada y el de salida. Esto permite el uso del mismo canal para comunicarse varias veces. Por otra parte, considerando los canales como listas podemos simular procesos recursivos (del álgebra de procesos) utilizando un único proceso en Eden. Este proceso en Eden se definirá a través de una función recursiva. En este caso, leer varias veces del mismo canal se corresponde con leer varios elementos de la correspondiente lista de entrada. Por otra parte, enviar mensajes varias veces a través de un canal de salida se corresponde con escribir varios elementos en la lista de salida. Supondremos por tanto que cualquier proceso en Eden se comunica utilizando listas.

Por último, comentaremos las características que incluye Eden para analizar propiedades cuantitativas. Si estamos interesados en medir diferentes aspectos de nuestros programas, en general no es suficiente con ejecutarlos (varias veces). Para conocer el comportamiento *real* de nuestros programas necesitamos algún tipo de realimentación. Podemos obtener esa realimentación utilizando Paradise [HPR00], un *profiler* para Eden basado en GranSim [Loi96]. Utilizando Paradise, los programas en Eden se ejecutan normalmente pero se registran, en un fichero, todos los eventos relevantes ocurridos durante la ejecución: creación de procesos, comunicación de valores, procesos que quedan bloqueados, etc. Una vez acabada la ejecución, se pueden utilizar varias herramientas de visualización para analizar los resultados registrados. También es posible combinar ficheros correspondientes a distintas ejecuciones para obtener estadísticas de propiedades acerca del comportamiento de los procesos. Esto permite comprobar si los resultados actuales encajan realmente con las predicciones teóricas y así obtener estadísticas sin realizar complejos estudios teóricos.

6.4. La traducción de VPASPA a Eden

En esta sección presentamos cómo las especificaciones escritas en el álgebra de procesos se traducen a programas en Eden. Consideraremos que un programa en Eden implementa una especificación en VPASPA si cualquier transición (de entrada, salida o interna) de la especificación puede *simularse* con los canales del programa en Eden, y si cualquier retardo de la especificación queda reflejado en el programa. El resto de la sección está dedicado a presentar nuestra metodología para realizar dicha traducción. Para cualquier proceso $P \in$ VPASPA escribimos $\texttt{translation}(P)$ para denotar el programa en Eden producido como una traducción de P.

Supongamos una especificación escrita en VPASPA donde tenemos un conjunto de ecuaciones como:

$$X_1 := P_1 \qquad X_2 := P_2 \qquad \cdots \qquad X_n := P_n$$

Impondremos la siguiente restricción: cualquier aparición de operadores de pararelo en los procesos P_j serán de la forma: $X_{i_1} \|_{\emptyset}^{\mu_1} (X_{i_2} \|_{\emptyset}^{\mu_2} \cdots (X_{i_{j-1}} \|_{\emptyset}^{\mu_{i_{j-1}}} X_{i_j}) \cdots)$ donde $\{X_{i_1}, \cdots, X_{i_j}\} \subseteq \{X_1, \cdots, X_n\}$. Intuitivamente, consideramos que cualquier aparición del operador de paralelo representa la composición de procesos *reales* (por lo tanto, tienen un *nombre*). Obsérvese que ésta no es una restricción real porque $P \|_{\emptyset}^{\mu} Q$ se puede definir siempre como $X := P, Y := Q$ y entonces $X \|_{\emptyset}^{\mu} Y$. Esta restricción simplifica el mecanismo de traducción pero sin perder expresividad. Un hecho a tener en cuenta es que nuestra sintaxis permite poner índices en los operadores de paralelo con cualquier conjunto de índices, aunque como ya hemos comentado anteriormente, sólo utilizaremos conjuntos no vacíos de índices en la definición de la semántica operacional. Inicialmente, suponemos que todos los conjuntos de índices son vacíos. Debemos observar también que estos índices no aparecen en la traducción a Eden puesto que cada proceso en Eden tratará sus propios retardos y, por tanto, no puede haber confusiones en cuanto a qué proceso queda retardado. Por una razón similar, no se necesita distinguir entre acciones de inicio y acciones de terminación como en la definición de la semántica operacional.

En primer lugar, calculamos el conjunto de canales de entrada y salida de los procesos VPASPA $P_1, \ldots, P_n$. Para cualquier $1 \leq j \leq n$, sean $I_j = \{i_{j1}, \ldots, i_{js_j}\} = \texttt{Inputs}(P_j)$ y $O_j = \{o_{j1}, \ldots, o_{jr_j}\} = \texttt{Outputs}(P_j)$. Generaremos n abstracciones de proceso en Eden. Estas abstracciones de procesos tendrán el mismo número de canales de entrada y salida que los procesos P_i correspondientes:

```
F1 = process (i11,...,i1s1) -> (o11,...,o1r1)
```

```
 where (o11,...,o1r1) = f1 i11 ... i1s1
F2 = process (i21,...,i2s2) -> (o21,...,o2r2)
 where (o21,...,o2r2) = f2 i21 ... i2s2
 .....
Fn = process (in1,...,insn) -> (on1,...,onrn)
 where (on1,...,onrn) = fn in1 ... insn
```

En el caso especial en el que los canales de salida son independientes, podemos utilizar una función diferente para definir cada uno de estos canales. Estas funciones dependerán solamente de las entradas del proceso:

```
F = process (i1,...,is) -> (o1,...,or)
 where o1 = f1 i1 ... is
        ....
       or = fr i1 ... is
```

Dependiendo de los procesos P_j, las funciones `f1,...,fr` se definirán de distinto modo. Si $P_j = \mathit{stop}$ entonces consideramos simplemente que no hay salidas. Más exactamente, se asocian listas vacías a cada uno de los canales de salida (recordemos que en Eden consideramos que tanto los canales de entrada como los de salida son listas). Los tres casos correspondientes a los prefijos son fáciles de traducir. Si $P_j = a!(v)\,;P$ entonces tendremos el proceso:

```
Fj = process (i1,...,is) -> (o1,...,v:oj,...,or)
 where (o1,...,oj,...,or) = translation(P)
```

donde `oj` es el canal de salida correspondiente a $a!$. En este caso, mandaremos el valor `v` por el canal de salida `oj` y después seguiremos con la traducción de P. Si $P_j := a?(x)\,;P$ tenemos:

```
Fj = process (i1,...,ij,...,is) -> (o1,...,or)
 where (o1,...,or)                      = g i1 ... ij ... is
       g i1 ... (Patternsj : vs) ... is = translation(P)
```

donde `ij` es el canal de entrada correspondiente a $a?$. En este caso, definimos una función auxiliar `g`. Esta función se define con ajuste de patrones en la entrada `ij` y toma distintos valores de acuerdo con el valor recibido a través de este canal de entrada. Los distintos patrones tendrán relación con apariciones de constructores `if then else` en P. Si no hay tales apariciones, tendremos un único patrón (es decir, no habrá distinción en P dependiendo

del valor que tome x); en caso contrario, las condiciones booleanas que aparezcan en la definición del constructor `if then else` se tendrán en cuenta para construir los distintos patrones, `Patterns`.

Ejemplo 6.15 Sea $P := a?(x)$; `if` $(x = 1)$ `then` P_1 `else` P_2. La traducción de P viene dada por:

```
FP = process (i1,...,is) -> (o1,...,or)
 where (o1,...,or) = g i1...is
       g (1 : vs) i2...is = translation(P1)
       g (n : vs) i2...is = translation(P2)
```

donde las entradas del procesos son $\texttt{Inputs}(P_1) \cup \texttt{Inputs}(P_2) \cup \{a\} = \{\texttt{i1}, ..., \texttt{is}\}$, las salidas son $\texttt{Outputs}(P_1) \cup \texttt{Outputs}(P_2) = \{\texttt{o1}, ..., \texttt{or}\}$, y suponemos que el canal de entrada correspondiente a $a?$ es `i1`. □

Si $P_j = \xi\,;P$ entonces tenemos un proceso:

```
Fj = process (i1,...,is) -> (o1,...,or)
 where (o1,...,or) = seq (wait xi) (translation(P))
```

Recordemos que `seq` representa el operador de composición secuencial. Además, `(wait xi)` interrumpe la ejecución del proceso durante una cantidad aleatoria de tiempo que depende de la distribución de probabilidad dada por `xi`. Hemos añadido `wait` a Eden puesto que no había un operador de este tipo en el lenguaje original. La implementación de este operador se basa en el construtor primitivo `sleep` y en la generación de retardos aleatorios. Hemos predefinido en Eden las distribuciones más comunes descritas formalmente en el capítulo 3. Por ejemplo, las distribuciones exponenciales se traducen como `wait(expo lambda)` y la distribución uniforme sobre el intervalo (a, b) se traduce como `wait(unif a b)`. En este capítulo utilizaremos, además de las anteriores, únicamente `wait(dirac a)`, `wait(poisson lambda)` y `wait(normal a b)` para las distribuciones de Dirac, Poisson y normal respectivamente. Obviamente, se pueden añadir más distribuciones en Eden a medida que se vayan necesitando. En este caso, queremos remarcar que los lenguajes funcionales (en particular Eden) son muy adecuados para definir diferentes distribuciones y tratar con ellas.

Análogamente al `if then else`, el operador de elección también se traduce utilizando ajuste de patrones. La diferencia está en que en este caso utilizaremos la función `merge2` para decidir qué rama deberíamos elegir. Para elegir entre dos alternativas mezclamos los canales correspondientes. La rama apropiada se selecciona por ajuste de patrones en la derecha o la izquierda.

```
out = f (merge2 alt1 alt2)
  where f (Left  a1 : vs) = ...
        f (Right a2 : vs) = ...
```

Si P_j es la composición paralela de m variables de procesos, es decir, si P_j es el siguente proceso $P_j = Y_1 \parallel_{\emptyset}^{\mu_1} (Y_2 \parallel_{\emptyset}^{\mu_2} \cdots (Y_{m-1} \parallel_{\emptyset}^{\mu_{m-1}} Y_m) \cdots)$, donde $\{Y_1, \ldots, Y_m\} \subseteq \{X_1, \ldots, X_n\}$, generaremos una concreción de proceso para cada una de las variables:

```
Fj = process (i1,...,is) -> (o1,...,or)
 where (out11,...,out1r1) = G1 # (in11,...,in1s1)
       (out21,...,out2r2) = G2 # (in21,...,in2s2)
                          ........
       (outm1,...,outmrm) = Gm # (inm1,...,inmsm)
```

donde para cualquier $1 \leq l \leq m$ tenemos que `Gl` es la abstracción del proceso en Eden correspondiente a Y_l, $\{$`inl1`, ..., `inlsl`$\}$ es el conjunto de canales de `Gl`, y $\{$`outl1`, ..., `outlrl`$\}$ es el conjunto de canales de salida. Obsérvese que, por la definición de las funciones `Inputs` y `Outputs`, tenemos $\{$`i1`, ..., `is`$\} = \{$`input` $\mid \exists\ 1 \leq k \leq m,\ 1 \leq l \leq s_k :$ `input` $=$ `inkl`$\}$ (de igual modo para $\{$`o1`, ..., `or`$\}$). Debemos remarcar que no necesitamos tratar explícitamente con las colas del operador de paralelo puesto que Eden proporciona una cola interna para cada canal de cada concreción de proceso.

Es bastante fácil implementar el operador de ocultamiento. Las únicas acciones visibles de `Fj` serán las declaradas en sus listas de entrada y salida (es decir, `i1`...`is`, `o1`...`or`), mientras que el resto de acciones ejecutadas por `Gi` son ocultas. Así, si debemos ocultar el conjunto de acciones `A`, es suficiente definir para las entradas `Inputs(Fj)` $= (\bigcup$ `Inputs(Pi)`$) -$ `A` y para las salidas `Outputs(Fj)` $= (\bigcup$ `Outputs(Pi)`$) -$ `A`. Obsérvese que cada salida (p.ej. `outij`) de un proceso puede corresponder con una entrada (p.ej. `inkl`) de otro proceso. En ese caso, para permitir la comunicación es suficiente utilizar el mismo nombre en la traducción (p.ej. `a`) para ambos, la entrada y la salida.

Ejemplo 6.16 Consideremos el proceso P dado en el ejemplo 6.15, donde $P_1 = c?(y)\,;b!(1)$ y $P_2 = d!(0)$, y sea $Q := a!(1)\,;\mathit{stop}$. Finalmente, tomemos el proceso $(P \parallel_{\emptyset}^{I} Q)/\{a\}$. La traducción de este último proceso, dada a continuación, es `F` (la traducción de Q está dada por `FQ`).

```
FQ = process () -> a
   where a = [1]
F = process (c) -> (b,d)
```

```
where (b,d) = FP # (a,c)
      a     = FQ # ()
```

□

Por último, comentaremos que se pueden realizar algunas optimizaciones una vez realizada la traducción de `VPASPA` a Eden. En muchas situaciones, una especificación puede implicar varios procesos P_i replicados que se quieren comunicar con un único proceso Q. En este caso, una traducción ingenua creará un canal distinto en Q para cada uno de los procesos P_i. Así, teniendo n procesos P_i, Q debería tener n canales diferentes. Afortunadamente, un lenguaje funcional como Eden puede tratar correctamente estas situaciones ya que podemos utilizar listas de canales en lugar de manejar explícitamente n canales individuales. Presentaremos un ejemplo ilustrativo de esta situación cuando definamos *Line* en la sección 6.5.2. Por otra parte, en algunas situaciones el proceso Q se comporta de igual modo con todos los procesos P_i. Por lo que cada vez que Q mande un valor v a cualquier proceso P_i también lo mandará al resto de los procesos. Si nos encontramos en este caso, nos podemos aprovechar de las facilidades de multidifusión de Eden, que permiten utilizar un único canal de comunicación para comunicarse con varios procesos. En el siguiente ejemplo, el proceso `q` puede emitir mensajes `start` a n procesos `p` diferentes:

```
outps  = [(p i) # starts | i <- [0..n-1]
starts = q # ()
```

Como optimización final, algunas veces podemos necesitar traducir elecciones múltiples como por ejemplo $\sum_{i=1}^{n} signal_i?(x)$. En este caso, podríamos implementarlo mediante la aplicación $n-1$ veces de la traducción de la elección binaria. Sin embargo, podemos conseguir un proceso Eden más legible utilizando la función `merge`, que puede tratar con listas arbitrariamente grandes de alternativas. Esto puede ser especialmente útil cuando un proceso Q puede recibir mensajes de varios procesos P_i, como veremos en la sección 6.5.2.

6.5. Ejemplos

En esta sección presentamos dos especificaciones relativamente complejas que servirán para mostrar que nuestro mecanismo de traducción produce programas en Eden *suficientemente parecidos* a las especificaciones originales. Debemos resaltar que en nuestras especificaciones, por claridad, utilizaremos en algunos casos procesos auxiliares, además, como ya hicimos en el capítulo 3, para describir comportamientos recursivos utilizaremos ecuaciones en lugar del constructor del lenguaje *rec*. De modo que, en algunos casos, algunos de los

procesos auxiliares que aparecen en la especificación están incluidos en un único proceso en Eden. El patrón seguido para la presentación de ambos ejemplos va a ser el mismo. En primer lugar, explicamos informalmente el problema a tratar. A continuación, indicamos los procesos que utilizaremos para especificar dicho problema. Después, daremos la correspondiente traducción para cada especificación. Por último, en la sección 6.5.3 estudiaremos algunas propiedades cuantitativas de los ejemplos.

Como ya hemos indicado anteiormente, utilizaremos algunas distribuciones de probabilidad que ya hemos presentado en el capítulo 3 de esta tesis, tales como la distribución exponencial, denotada por $\xi_{exp(\lambda)}$ en los ejemplos, distribuciones de Poisson, denotadas por $\xi_{Po(\lambda)}$, distribuciones normales, denotadas por $\xi_{N(a,b)}$, variables aleatorias uniformemente distribuidas sobre el intervalo (a, b), denotadas por $\xi_{U(a,b)}$, y distribuciones de Dirac, denotadas por $\xi_{D(c)}$.

6.5.1. El telescopio espacial Hubble

Este ejemplo está basado en la especificación de este sistema descrita en [Her00]. Sin embargo, hemos introducidos algunos cambios para adaptarlo a nuestro marco, en el que están permitidas distribuciones generales. Hemos elegido este ejemplo porque, aunque es relativamente simple, presenta de una manera muy clara las principales características que aparecen en un álgebra de procesos estocásticos, aunque no se aproveche el hecho de tener paso de valores. El telescopio espacial Hubble (HST: Hubble Space Telescope) es un observatorio astronómico lanzado al espacio en 1990 y cuya esperanza de vida operativa son más de veinte años. Este telescopio tiene seis giróscopos como parte significativa del sistema del HST. Estos giróscopos proporcionan información acerca de la posición del telescopio teniendo en cuenta los movimientos de la aeronave. El sistema puede trabajar perfectamente con únicamente tres giróscopos, es decir, se mantiene operativo aunque fallen tres de los seis giróscopos. Sin embargo, si falla un cuarto giróscopo, el HST pasa a un estado de inactividad y en ese punto existe el riesgo de colisión si el resto de giróscopos falla. Una vez que se ha entrado en ese estado de inactividad, la base debe preparar una misión para reparar el telescopio. Durante el tiempo de esta preparación y reparación, existe la posibilidad de accidente. Si el telescopio no sufre ninguna colisión durante esa fase, una vez arreglados los giróscopos, el sistema se reinicia.

En la especificación del sistema consideraremos los siguientes procesos principales: el proceso que describe el comportamiento de los giróscopos *Gyro*, el controlador *Controller*, el estabilizador *Stabilizer*, la base de la operación *Base* y por último el telescopio *HST*.

El proceso *Gyro*. Un giróscopo va a tener dos estados: puede estar funcionando, que denotaremos por *working*, o puede estar averiado, que denotaremos por *broken*. Inicialmente todos los giróscopos están en el estado *working*. Un giróscopo que está funcionando puede romperse después de transcurrido algún tiempo (dado por una distribución exponencial) o puede recibir la señal *restart* que indica el reinicio del sistema. En este último caso, el estado del giróscopo no varía puesto que continuaría activo; en el primer caso, mandaría una señal de fallo *fail*, y el estado cambiaría a *broken*. Un giróscopo estropeado espera una señal de reinicio, es decir, espera la señal *restart* para cambiar de nuevo al estado *working*.

$$
\begin{aligned}
Gyro_i &:= Gyro_{i,working} \\
Gyro_{i,working} &:= \xi_{exp(\lambda)} \,;\, fail!(-) \,;\, Gyro_{i,broken} + restart_i?(-) \,;\, Gyro_{i,working} \\
Gyro_{i,broken} &:= restart_i?(-) \,;\, Gyro_{i,working}
\end{aligned}
$$

Esta especificación se traduce a Eden como se puede ver a continuación. Como en la especificación el giróscopo tiene dos estados, el proceso en Eden también utiliza una variable que indica si el giróscopo está en estado `Broken` o `Working`. Obsérvese que en el estado `Working` utilizamos la función `merge2` para poder elegir entre dos alternativas posibles: fallar o reiniciar el sistema.

```
gyro = process restarts -> fails
  where fails = f Working restarts
        f Working xs = fWork (merge2 [wait (expo lambda)] xs) xs
          where fWork (Left v : vs) xs     = Fail : f Broken xs
                fWork (Right v: vs) (x:xs) = f Working xs
        f Broken (x:xs) = f Working xs
```

El proceso *Controller*. El controlador recibe o bien un mensaje de fallo desde los giróscopos, denotado por *fail* en la especificación, o bien un mensaje de reinicio desde la base, denotado por *restart*. Si recibe cuatro mensajes de fallo genera un mensaje de paso a la inactividad, que denotaremos por *sleep*, después de transcurrida una cantidad de tiempo fija (que estará dada por una distribución de Dirac). Durante este intervalo de tiempo el controlador podría recibir dos mensajes más de fallo. En este caso se enviaría un mensaje de colisión, denotado por *crash*. El controlador empezará con cero mensajes de fallo recibidos y utilizaremos seis procesos auxiliares para describir el comportamiento del controlador en cada momento.

$$
\begin{array}{lcl}
Controller & := & Controller_0 \\
Controller_3 & := & restart_c?(-)\,;Controller_0 + fail?(-)\,;\xi_{D(c)}\,;sleep!(-)\,;Controller_4 \\
Controller_5 & := & restart_c?(-)\,;Controller_0 + fail?(-)\,;crash!(-)\,;stop \\
Controller_k & := & restart_c?(-)\,;Controller_0 + fail?(-)\,;Controller_{k+1} \\
0 \leq k < 5, k \neq 3 & &
\end{array}
$$

En este caso, para cada uno de los estados del controlador debemos elegir entre dos alternativas. De modo que la traducción utiliza funciones auxiliares para cada una de ellas:

```
controller = process (fails,restarts) -> (sleeps,crashs)
  where (sleeps,crashs) = f 0 restarts fails
        f 3 xs ys = f3 (merge2 xs ys) xs ys
          where f3 (Left v : vs) (x:xs) ys = f 0 xs
                f3 (Right v: vs) xs (y:ys) = seq (wait (dirac c)) (Sleep:ys1,ys2)
                  where (ys1,ys2) = f 4 xs
        f 5 xs ys = f5 (merge2 xs ys) xs ys
          where f5 (Left v : vs) (x:xs) ys = f 0 xs
                f5 (Right v: vs) xs (y:ys) = (ys1,Crash:ys2)
                  where (ys1,ys2) = ([],[])
        f k xs ys = fk (merge2 xs ys) xs ys
          where fk (Left v : vs) (x:xs) ys = f 0 xs
                fk (Right v: vs) xs (y:ys) = f (k+1) xs
```

El proceso *Stabilizer*. El estabilizador es un proceso que comunica los mensajes de fallo que mandan los giróscopos al controlador. Debemos recordar que el parámetro I que aparece en el operador de paralelo denota que inicialmente todas las colas asociadas con los canales están vacías.

$$Stabilizer := (Controller \parallel^I_\varnothing Gyro_1 \parallel^I_\varnothing Gyro_2 \parallel^I_\varnothing Gyro_3 \parallel^I_\varnothing Gyro_4 \parallel^I_\varnothing Gyro_5 \parallel^I_\varnothing Gyro_6)/\{fail\}$$

La implementación en Eden conecta las salidas y entradas de los procesos de la forma adecuada. Todos los mensajes de fallo que provienen de los giróscopos se mezclan en una única lista y se comunican al controlador. Además, desde este proceso se mandan los mensajes de reinicio a los giróscopos. Realmente, este proceso es una optimización del código que resulta aplicando nuestro mecanismo de traducción. Ello es así porque no podemos aprovechar el hecho de que no necesitamos nombres diferentes para comunicarnos con todos los giróscopos:

```
stabilizer = process restarts -> (sleeps,crashs)
```

```
  where fails           = merge [gyro # restarts | i <-[1..6]]
        (sleeps,crashs) = controller # (fails,restarts)
```

El proceso $Base$. La base espera o un mensaje de colisión o un mensaje de inactividad. En el primer caso, el proceso para (es decir, la misión ha concluido). Si se recibe un mensaje de paso a inactividad entonces la base manda mensajes para preparar una misión, lanzarla y reparar los giróscopos estropeados. El tiempo que se necesita para repararlos viene dado por una distribución exponencial μ. Si se reparan los giróscopos se genera un mensaje de reinicio que se manda a todo el sistema.

$$\begin{aligned} Base \;:=\; & crash?(-)\,;finish!(-)\,;stop \\ & +sleep?(-)\,;\xi_{exp(\mu)}\,;restart_c!(-)\,;restart_1!(-)\,;restart_2!(-); \\ & \quad restart_3!(-)\,;restart_4!(-)\,;restart_5!(-)\,;restart_6!(-)\,;Base \end{aligned}$$

Como una optimización, el programa en Eden necesita únicamente un mensaje de reinicio que se transmitirá a todos los giróscopos:

```
base = process (sleeps,crashs) -> (restarts,finishs)
  where (restarts,finishs) = f (merge2 crashs sleeps)
        f (Left x : xs)  = (ys1,Finish:ys2)
          where (ys1,ys2) = ([],[])
        f (Right x : xs) = seq (wait (expo mu)) (Restart: ys1,ys2)
          where (ys1,ys2) = f xs
```

El proceso HST. Por último, el HST se corresponde con el sistema completo. En este proceso tenemos la comunicación entre el estabilizador y la base.

$$HST \;:=\; (Base \,\|^{I}_{\varnothing}\, Stabilizer)/(\{restart_l \mid 1 \leq l \leq 6 \;\vee\; l = c\} \cup \{crash, sleep\})$$

La traducción a Eden de este proceso sólo necesita establecer las conexiones apropiadas para la comunicación:

```
hst = finishs
  where (sleeps,crashs)    = stabilizer # restarts
        (restarts,finishs) = base # (sleeps,crashs)
```

6.5.2. CSMA/CD

Nuestro segundo ejemplo es más complejo que el anterior y está dedicado a describir un protocolo real para redes de área local conectadas mediante un único bus de datos. En [Ber99]

aparece otra formalización de este mismo ejemplo, aunque nuestra formalización diverge de la de allí presentada puesto que utilizamos las características adicionales que proporciona `VPASPA`. En concreto no restringimos las distribuciones a que sean exponenciales, incluimos paso de valores y permitimos comunicaciones asíncronas.

El protocolo descrito es el estándar IEEE 802.3 CSMA/CD (véase p.ej. [Tan96]). Dadas n estaciones conectadas mediante un bus, este protocolo funciona de la siguiente manera. En el caso de que una estación quiera transmitir un valor, en primer lugar debe determinar si el canal está libre o si está siendo utilizado por otra estación. En el primer caso, la estación empieza a enviar el mensaje, mientras que en el segundo caso tiene que esperar una cierta cantidad de tiempo (aleatoria) antes de consultar de nuevo la disponibilidad del canal. Nótese que mientras que una estación está transmitiendo un mensaje se puede producir una colisión con un mensaje de otra estación. Si esta incidencia acontece enonces ambas estaciones deben esperar una cantidad de tiempo (aleatoria) antes de intentar mandar sus mensajes otra vez. Afortunadamente, las colisiones no pueden tener lugar en cualquier momento sino que solamente son posibles en un corto periodo de tiempo, justo al principio de la transmisión del mensaje. Este periodo depende de la latencia de la red: después de un tiempo, todas las estaciones detectarán que el canal está ocupado y no intentarán mandar un nuevo mensaje hasta que el canal esté disponible de nuevo.

En la especificación del sistema se consideran los siguientes procesos principales: la *estación* (*Station*), el *sensor* (*Sensor*), el *canal* (*Channel*), la *línea* (*Line*) y, por último, el $CSMA/CD$.

El proceso *Station*. Una estación se puede modelar como la composición paralela de dos procesos: *GenMsg* que es el generador de mensajes y *StationTrans* que está a cargo de transmitir los mensajes creados por *GenMsg*. Debemos resaltar el hecho de que todos los mensajes generados se almacenarán en la cola del operador de paralelo, por lo que no es necesario ningún proceso para almacenar estos mensajes. Así, el proceso *Station* generará mensajes independientemente de la posibilidad de transmitirlo.

En *GenMsg*, el tiempo que pasa entre dos mensajes que se crean sigue una distribución de Poisson con parámetro λ:

$$GenMsg_i := \quad \xi_{Po(\lambda)} \,;\, genMsg_i!(msg) \,;\, GenMsg_i$$

La traducción de este proceso a Eden es la siguiente:

```
genMsg i = process () -> msgs
  where msgs = f lambda
        f lambda = seq (wait (poisson lambda))(generateMsg i : f lambda)
```

Como el proceso *StationTrans*, cuya tarea es enviar los mensajes generados por el proceso anterior, es más complejo que el anterior, lo dividiremos en cuatro etapas. Una vez que se ha recibido un mensaje desde el generador, el proceso *StationTrans* pregunta al sensor si la línea está disponible o no. Esta tarea la realiza mediante la acción *sensorReady*. A continuación espera en el estado *Sense* hasta que recibe una respuesta. En el caso de que no este libre, tiene que dar marcha atrás. En caso contrario, y después de esperar un retardo fijo (dado por una distribución de Dirac), notifica que está empezando una nueva transmisión, entrando en la etapa *Prop*. La transmisión tiene lugar en esta etapa, pero no es seguro que no habrá colisiones hasta que no se recibe un mensaje *tranMsg*, notificando que el periodo *peligroso* ha pasado. En ese caso, notificamos que el mensaje se ha mandado de forma correcta y después de una cantidad de tiempo, dada por una distribución normal, termina la transmisión del mensaje.

Debemos resaltar que la señal de colisión, *signalColl*, se puede recibir en cualquier etapa del proceso. De ese modo, necesitamos tener una elección en todas ellas con la recepción de ese mensaje. La única excepción a esta regla aparece en la primera etapa: como todavía no se ha intentado iniciar una transmisión, ignoramos la señal de colisión. Cuando se produce una colisión, la estación pasa a la etapa *Back* y espera a que transcurra un tiempo dado por una distribución exponencial, y después pregunta otra vez al sensor si la línea está o no libre.

$$
\begin{aligned}
StationTrans_i &:= genMsg_i?(x)\,;\, sensorReady_i!(-)\,;\, StationSense_i(x) \\
&\quad + signalColl?(-)\,;\, StationTrans_i \\
StationSense_i(msg) &:= sense_i?(x)\,;\, \texttt{if } x = idle \\
&\qquad\qquad \texttt{then } \xi_{D(a)}\,;\, startTransMsg_i!(msg)\,;\, StationProp_i(msg) \\
&\qquad\qquad \texttt{else } StationBack_i(msg) \\
&\quad + signalColl?(-)\,;\, StationBack_i(msg) \\
StationProp_i(msg) &:= tranMsg_i?(-)\,;\, send!(msg)\,;\, \xi_{N(b,c)}\,;\, endTransMsg_i!(-)\,;\, StationTrans_i \\
&\quad + signalColl?(-)\,;\, StationBack_i(msg) \\
StationBack_i(msg) &:= \xi_{exp(\mu)}\,;\, sensorReady_i!(-)\,;\, StationSense_i(msg) \\
&\quad + signalColl?(-)\,;\, StationBack_i(msg)
\end{aligned}
$$

La traducción a Eden también considera las cuatro etapas de la definición. Siguiendo la especificación, la traducción de cada una de estas etapas empieza con una elección entre las dos alternativas, utilizando una función auxiliar para implementar qué tiene que hacerse en cada uno de los casos:

```
stationTrans i = process (msgs,senses,colls,trans) -> (sends,readys,starts,ends)
```

```
  where (sends,readys,starts,ends) = f Trans msgs senses colls trans
        f Trans xs ys zs ws = ftrans (merge2 xs zs) xs ys zs ws
          where ftrans (Left msg : vs) (x:xs) ys zs ws = (ys1,Ready:ys2,ys3,ys4)
                  where (ys1,ys2,ys3,ys4) = f (Sense msg) xs ys zs ws
                ftrans (Right v : vs) xs ys (z:zs) ws  = f Trans xs ys zs ws
        f (Sense msg) xs ys zs ws = fSense (merge2 ys zs) xs ys zs ws
          where fSense (Left Idle : vs) xs (y:ys) zs ws
                       = seq (wait (dirac a))(ys1,ys2,Start i:ys3,ys4)
                  where (ys1,ys2,ys3,ys4) = f (Prop msg) xs ys zs ws
                fSense (Left Busy : vs) xs (y:ys) zs ws = f (Back msg) xs ys zs ws
                fSense (Right v : vs)   xs ys (z:zs) ws = f (Back msg) xs ys zs ws
        f (Prop msg) xs ys zs ws = fProp (merge2 ws zs) xs ys zs ws
          where fProp (Left v : vs) xs ys zs (w:ws) = (msg:ys1,ys2,ys3,ys4)
                  where (ys1,ys2,ys3,ys4) = seq (wait (normal b c))
                                                (zs1,zs2,zs3,End:zs4)
                        (zs1,zs2,zs3,zs4) = f Trans xs ys zs ws
                fProp (Right v:vs) xs ys (z:zs) ws = f (Back msg) xs ys zs ws
        f (Back msg) xs ys zs ws = fBack (merge2 [wait (expo mu)] zs) xs ys zs ws
          where fBack (Left v : vs) xs ys zs ws = (ys1,Ready:ys2,ys3,ys4)
                  where (ys1,ys2,ys3,ys4) = f (Sense msg) xs ys zs ws
                fBack (Right v: vs) xs ys (z:zs) ws = f (Back msg) xs ys zs ws
```

Una estación, especificada por el proceso *Station*, es una composición paralela de un generador de mensajes, *GenMsg*, y un proceso transmisor de mensajes, *StationTrans*.

$$Station_i := \quad (GenMsg_i \parallel^I_{\varnothing} StationTrans_i)/\{genMsg_i\}$$

La traducción a Eden conecta la salida de `genMsg` con la entrada apropiada de `stationTrans`:

```
station i = process (senses,colls,trans) -> (sends,readys,starts,ends)
  where msgs = (genMsg i) # ()
        (sends,readys,starts,ends) = (stationTrans i) # (msgs,senses,colls,trans)
```

El proceso *Sensor*. Para cada estación debe existir un sensor para detectar si se está utilizando o no la línea. De ese modo, el proceso está parametrizado por el estado de la línea, que inicialmente es libre y denotamos por *idle*. Un sensor puede recibir un mensaje que indica que la estación correspondiente quiere saber cómo está la línea y también puede recibir un mensaje *set*. En el primer caso, el sensor le mandará a la estación el estado actual de la línea (mediante el valor v) después de transcurrir un retardo fijo dado por una distribución de Dirac. En el segundo caso, el estado del sensor cambiará después de otro retardo fijo.

$$
\begin{array}{lcl}
Sensor_i & := & SensorS_i(Idle) \\
SensorS_i(v) & := & sensorReady_i?(-)\,;\xi_{D(d)}\,;sense_i!(v)\,;SensorS_i(v) \\
 & & +\ set_i?(x)\,;\xi_{D(e)}\,;SensorS_i(x)
\end{array}
$$

La traducción a Eden es la siguiente:

```
sensor = process (readys,sets) -> senses
  where senses = f Idle (merge2 readys sets) readys sets
        f state (Left x :xs) (y:ys) zs = seq (wait (dirac d))
                                             (state : f state xs ys zs)
        f state (Right x:xs) ys (newState:zs) = seq (wait (dirac e))
                                                    (f newState xs ys zs)
```

El proceso *Line*. El proceso *Line* puede comunicarse con cualquier estación del sistema. Cuando no se está usando una línea (estado *empty* en la especificación) puede recibir un mensaje $startTransMsg$ de cualquier estación para iniciar la transmisión. En este caso, y después de un retardo dado por una distribución de Dirac, notificaremos a todos los sensores que la línea está actualmente ocupada. A continuación el proceso se sitúa en la etapa de transmisión del mensaje, *LineTrans*. En ese punto existen dos alternativas dependiendo de si se recibe o no otro mensaje para empezar una nueva transmisión de otra estación. Si el nuevo mensaje no llega antes de un tiempo dado por una distribución de Dirac, la transmisión se realizará con éxito. En caso contrario, la línea entrará en modo colisión después de un retardo fijo. En el caso de que la comunicación sea posible sin colisión, la línea entra en el estado *Success*, y se lo notifica a la estación correspondiente. Una vez realizada esta tarea, espera hasta que termina la transmisión, pasando al estado *Wait*. Finalmente, manda a los sensores la señal para que pasen al estado de libre después de un retardo fijo. En el caso de que la línea entre en la etapa *Collision*, se envía una señal de colisión y los sensores pasan al estado libre.

$$
\begin{array}{lcl}
Line & := & LineEmpty \\
LineEmpty & := & \sum_{i=1}^{n} startTransMsg_i?(-);\xi_{D(g)};set_1!(busy);\cdots;set_n!(busy);LineTrans_i \\
LineTrans_i & := & \xi_{D(h)}\,;LineSuccess_i + \sum_{j=1}^{n} startTransMsg_j?(-)\,;\xi_{D(g)}\,;LineCollision_i \\
LineSuccess_i & := & transMsg_i!(-)\,;LineWait_i \\
LineWait_i & := & endTransMsg_i?(-)\,;\xi_{D(g)}\,;set_1!(idle)\,;\cdots;set_n!(idle)\,;Line \\
LineCollision_i & := & signalColl!(-)\,;set_1!(idle)\,;\cdots;set_n!(idle)\,;Line
\end{array}
$$

Debemos resaltar que una línea se puede comunicar tanto con n procesos $Sensor_i$ diferentes como con n procesos $Station_i$ distintos. En el caso de los sensores, se comporta siempre igual con ellos. Por tanto podemos aprovechar las ventajas que proporcionan las facilidades de multidifusión de Eden: necesitaremos únicamente definir un canal para comunicarse con los sensores y los mensajes se mandarán automáticamente a todos ellos. En el caso de las estaciones podemos utilizar la misma optimización para las señales de colisión, pero no para los otros mensajes. De ese modo, la tradución utilizará listas de canales `startss`, `endss`, y `transs`. Como `startss` y `endss` son listas de canales de entrada, la traducción empieza convirtiéndolos en canales únicos utilizando la función `merge`.

```
line = process (startss,endss) -> (sets,colls,transs)
  where (sets,colls,transs) = f Empty (merge startss) (merge endss)
        f Empty (Start i:is) es = seq (wait (dirac g))(Busy:ys1,ys2,yss3)
            where (ys1,ys2,yss3) = f (Trans i) is es
        f (Trans i) is es = fTrans (merge2 [wait (dirac h)] is) is es
            where fTrans (Left _ : _) is es = f (Success i) is es
                  fTrans (Right _: _) (i2:is) es = seq (wait (dirac g))
                                                       f (Collision i) is es
        f (Success i) is es = (ys1,ys2, insert i Trans yss3)
            where (ys1,ys2,yss3) = f (Wait i) is es
        f (Wait i) is (e:es) = seq (wait (dirac g))(Idle:ys1,ys2,yss3)
            where (ys1,ys2,yss3) = f Empty is es
        f (Collision i) is es = (Idle:ys1,Coll:ys2,yss3)
            where (ys1,ys2,yss3) = f Empty is es
```

El proceso *Channel*. Un canal está definido mediante la composición paralela de una línea y de n sensores.

$$Channel \quad := \quad ((Sensor_1 \parallel^I_\varnothing \ldots \parallel^I_\varnothing Sensor_n) \parallel^I_\varnothing Line)/\{set_i \mid 1 \leq i \leq n\}$$

La traducción a Eden únicamente necesita conectar de manera adecuada los canales correspondientes a los sensores y a la línea. Obsérvese que todos los mensajess `sets` se envían automáticamente a todos los sensores, mientras que cada uno de los sensores recibe sus propios mensajes `readys` de forma personalizada:

```
channel = process (readyss,startss,endss) -> (colls,transs,sensess)
  where sensess = [(sensor i) # (readyss!!i,sets) | i <- [0..n-1]]
        (sets,colls,transs) = line # (startss,endss)
```

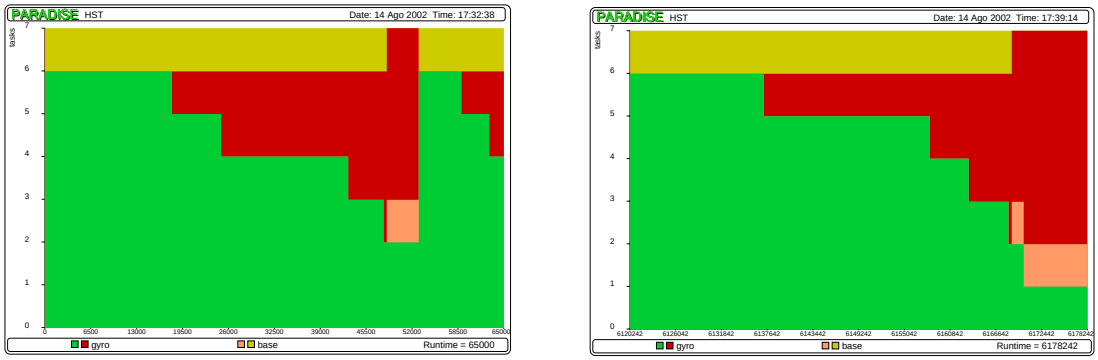


Figura 6.4: Comportamiento del HST: Una primera misión (izquierda) y la colisión final del sistema (derecha).

El proceso $CSMA/CD$. El sistema completo es la composición paralela de n estaciones y un canal.

$$CSMA/CD \quad := \quad ((Station_1 \parallel^I_\varnothing \ldots \parallel^I_\varnothing Station_n) \parallel^I_\varnothing Channel)/A$$

donde $A = \{ startTransMsg_i, transMsg_i, endTransMsg_i, sensorReady_i, sense_i \mid 1 \leq i \leq n \} \cup \{ signalColl \}$.

La traducción del proceso principal crea n estaciones y un canal y conecta sus correspondientes canales de forma adecuada. Utilizamos la función `unzip4` para transformar una lista de tuplas en una tupla de listas. Es necesario utilizar esta función cuando se crea una lista de procesos y cada uno de ellos produce más de una salida:

```
csma_cd = process () -> sendss
  where (sendss,readyss,startss,endss)
            = unzip4 [(station i) # (sensess!!i,colls,transs!!i) | i <- [0..n-1]]
        (colls,transs,sensess) = channel # (readyss,startss,endss)
```

6.5.3. Estudio de las propiedades de rendimiento

En esta sección utilizaremos algunas de las facilidades que proporcionan las herramientas del lenguaje Eden para evaluar el rendimiento de nuestros sistemas mediante las correspondientes traducciones. En primer lugar, con gráficos como los presentados en la figura 6.4, podemos entender mejor el comportamiento de una especificación. En el primer gráfico se aprecia la evolución en el tiempo del estado de los procesos, mostrando si un proceso está

funcionando o bloqueado (es decir, esperando alguna comunicación). La gráfica indica que, inicialmente, hay seis giróscopos trabajando mientras la base está bloqueada, esperando que el controlador pase a estado de inactividad. Después, los giróscopos empiezan a fallar hasta que sólo funcionan dos de ellos. Entonces, la base empieza la preparación de la misión para repararlos. Cuando la misión termina, los seis giróscopos vuelven a funcionar de nuevo y la base queda bloqueada otra vez esperando una nueva misión. El segundo gráfico muestra la colisión del sistema: después de la avería de cuatro giróscopos, la base empieza una nueva misión, pero los otros dos giróscopos fallan antes de poder completar la misión. De ese modo, el sistema entero falla.

No debemos dejar pasar este punto sin resaltar la utilidad que este tipo de gráficas poseen desde el punto de vista didáctico. Al formar a nuevos alumnos, suelen tener problemas para entender a qué puede concluir exactamente una especificación. El uso de gráficas que muestran el comportamiento de los programas facilitan la comprensión de los principios básicos de las especificaciones.

Además de este tipo de gráficas realizadas por Paradise, son interesantes algunas *medidas* cuantitativas de nuestros programas. En el ejemplo del telescopio Hubble, estamos interesados en conocer el tiempo operativo esperado del sistema, así como el número de reparaciones necesarias durante la vida del telescopio. Para conocer la primera información solamente necesitamos almacenar el tiempo que el programa necesita para finalizar. En el segundo caso, debemos indicar a la herramienta que debe contar el número de mensajes `Restart` que se mandan. En la figura 6.5 se muestra la probabilidad de que el sistema colisione después de n años, y la probabilidad de necesitar menos de n misiones antes del colapso. En los dos primeros gráficos suponemos que la reparación del sistema dura un año (en media) mientras que los otros dos gráficos asumimos una duración de nueve meses. En ambos casos suponemos que el tiempo medio transcurrido antes de que un giróscopo falle es de diez años y que el controlador tarda tres días en entrar en modo de inactividad.

Finalmente, también utilizamos Paradise para medir las propiedades del protocolo CSMA/CD. En este caso, los resultados más interesantes consisten en calcular la probabilidad de colisiones. Hemos medido esta probabilidad variando el número de estaciones y también variando la frecuencia con la que se generan los mensajes (para un número fijo de estaciones). Los resultados se muestran en la figura 6.6. Como era de esperar, la probabilidad de colisión se incrementa a medida que aumenta el número de estaciones y también a medida que aumenta la frecuencia de los mensajes.

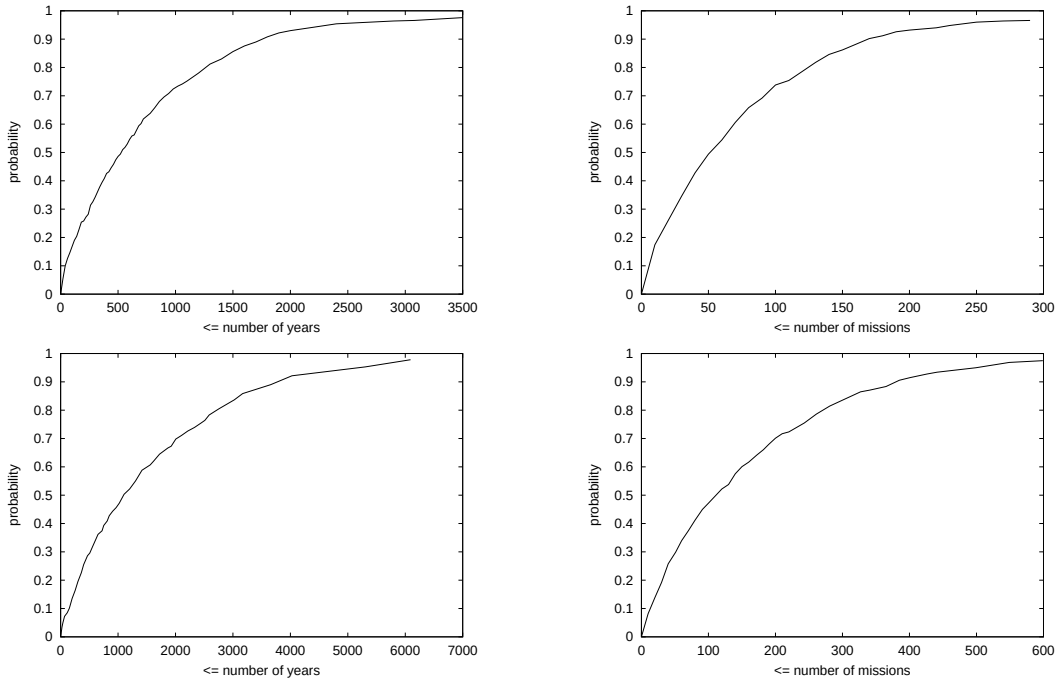


Figura 6.5: Número de años y número de misiones antes de la colisión del sistema (HST).

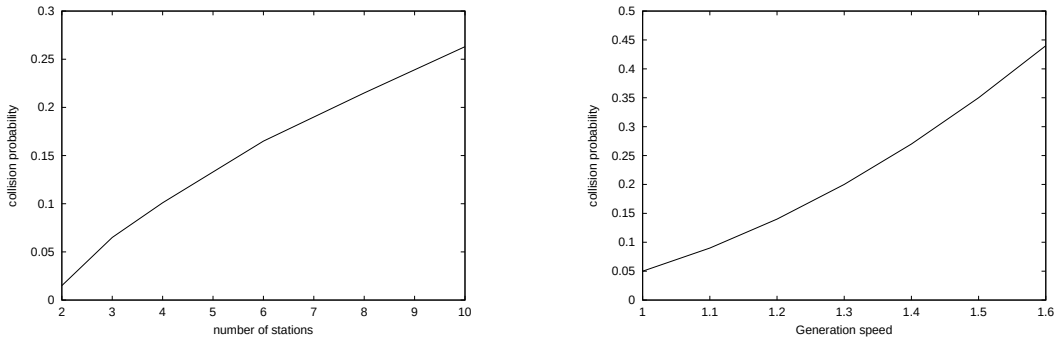


Figura 6.6: Probabilidad de colisión dependiendo del número de estaciones (izquierda) o dependiendo de la velocidad de generación de mensajes (derecha) (CSMA/CD).

6.6. Conclusiones

En este capítulo hemos presentado un marco integrado para el estudio de propiedades cuantitativas de procesos estocásticos. Para ello, hemos *implementado* las especificaciones como programas en Eden y hemos estudiado las propiedades de los programas funcionales resultantes. Para realizar este paso entre el álgebra de procesos y Eden, hemos dado un algoritmo de traducción de las especificaciones a los programas Eden correspondientes. Sin embargo, existen algunas optimizaciones del código que son deseables y que no están recogidas en el algoritmo, siendo por tanto necesario en algunas ocasiones un tratamiento posterior *a mano* de los programas generados por el algoritmo. Finalmente, a través de dos ejemplos, hemos mostrado el funcionamiento de nuestro algoritmo y como se pueden estudiar las correspondientes propiedades cuantitativas.

La principal ventaja de nuestra traducción es que podemos ampliar nuestro conocimiento de la especificación simulando su implementación en Eden. Este hecho es especialmente relevante en el caso de propiedades de evaluación de rendimiento, dado el hecho de que actualmente las técnicas de model checking no se pueden utilizar cuando consideramos distribuciones generalizadas.

Capítulo 7

Conclusiones y trabajo futuro

A lo largo de esta tesis hemos ido definiendo distintos modelos para especificar adecuadamente procesos con información estocástica. Cada uno de los modelos descritos ha tenido como principal objetivo el desarrollo de un estudio sobre las características más importantes de la familia de procesos estocásticos que no tienen restringidas las distribuciones de probabilidad a la clase de distribuciones exponenciales. Pasaremos a continuación a comentar los aspectos más destacados tratados en este trabajo, así como aquellos temas relacionados con los aquí expuestos en los que deseamos profundizar en un futuro cercano.

En un primer momento, se decidió la forma adecuada de definir una nueva semántica de bisimulación que no tuviera los inconvenientes de las semánticas de bisimulación habituales. Con ese objetivo presentamos varias nociones de bisimulación global para procesos sin información cuantitativa. A través de estas nociones se consiguió la equivalencia entre procesos que representaban diferentes tipos de no determinismo y que se distinguían hasta ese momento bajo cualquier noción de bisimulación, aun cuando parecía razonable su identificación.

La principal propiedad que cumplen las nociones definidas en el segundo capítulo de este trabajo es la asociatividad de la elección interna. De hecho, esta propiedad fue la desencadenante de la búsqueda de un nuevo tipo de bisimulación. Una vez satisfecha esta propiedad seguimos buscando otro tipo de propiedades que nos gustaría que cumpliera nuestra relación de equivalencia. Así, para cada propiedad hemos definido una nueva noción de bisimulación donde la única modificación con respecto a cada una de las definiciones anteriores es el tipo de transiciones que los procesos pueden realizar. Con ello, hemos probado que las distintas nociones de bisimulación global se pueden definir de manera modular, refinando las nociones de transición para conseguir relaciones de equivalencia cada vez más débiles. El principal

inconveniente de estas nociones de bisimulación, como vimos en su momento, consiste en la dificultad de extender una de ellas para ampliar las propiedades que satisface. Ello se debe a que la inclusión de nuevas reglas para generar transiciones es un arma de doble filo en el juego de la bisimulación. Por un lado, estas reglas le dan más poder al jugador que se defiende. Por otro lado, también le da más poder al que ataca puesto que tiene más formas de realizar su movimiento al tener la posibilidad de generar transiciones nuevas. Por tanto, que cada vez que se redefina la equivalencia semántica debemos comprobar que se siguen satisfaciendo las propiedades que se cumplían antes de la extensión.

También hemos presentado una extensión temporal de la bisimulación global. Esta nueva semántica se puede considerar como un marco adecuado para el estudio de semánticas de procesos temporales no deterministas. Gracias a esta extensión mostramos que nuestra nueva semántica es fácilmente adaptable para tratar otros tipos de no determinismo como el causado por la información temporal.

Sería interesante estudiar qué nuevas reglas (para generación de transiciones estáticas y dinámicas) se pueden añadir a la noción de bisimulación global temporal presentada aquí para conseguir caracterizar otras semánticas por medio de este tipo de equivalencias. En particular, estamos especialmente interesados en caracterizar tanto las semánticas de *testing* clásicas como la semántica de pruebas amistosa [dFLN97, dFLN98, Lóp99].

Otro trabajo a desarrollar consiste en añadir información probabilística. Es importante tener en cuenta que la noción de bisimulación global cambiaría notablemente ante la presencia de este tipo de información. No obstante, las dificultades para la adaptación de la bisimulación a este marco deben ser mayores puesto que, en general, si consideramos procesos en los que cuantificamos las elecciones no deterministas con probabilidades, entonces las versiones probabilísticas no son necesariamente bisimilares globales (aunque pudieran llegar a serlo, para una distribución adecuada de probabilidades).

A partir de ese primer estudio sobre las nociones de bisimulación se presentó el primer modelo semántico con información estocástica. A la hora de incluir este tipo de información existía una importante cuestión que debíamos contestar: ¿qué tipo de distribuciones de probabilidad ibamos a permitir, únicamente exponenciales o de todo tipo? Como ya se comentó en la introducción de esta tesis doctoral, la principal ventaja de utilizar distribuciones exponenciales consiste en su sencillez a la hora de definir una semántica de *interleaving*. Sin embargo, las distribuciones generales aportan mayor expresividad al lenguaje. De modo que decidimos definir un lenguaje con distribuciones generales. Pese a esta decisión, consideramos un operador de composición paralela, cuya inclusión crea serios problemas a la hora

de estudiar tanto el comportamiento de los procesos, como cualquier equivalencia semántica que queramos definir sobre ellos. Para resolver estos inconvenientes decidimos utilizar una técnica utilizada ya para procesos temporizados. Intuitivamente, las transiciones estocásticas que no evolucionen dentro de la composición paralela deben *envejecer* si otra componente ha realizado una transición estocástica. Esta técnica resuelve el problema pero, desafortunadamente, provoca la generación de sistemas de transiciones con infinitos estados. Por este motivo, en el último capítulo hemos desarrollado otra técnica para el estudio de las transiciones estocásticas en este contexto. En este primer modelo estocástico establecemos las principales características de todos los modelos definidos a partir de ese punto, distribuciones generales, progreso máximo, etc. Para este lenguaje definimos una bisimulación fuerte que constituye la base a partir de la cual, en los siguientes capítulos, se describen relaciones de bisimulación que identifican más procesos.

Junto con el desarrollo de sistemas de transiciones infinitos, otro inconveniente de este modelo es la dificultad de definir una bisimulación débil que abstraiga parcialmente la información interna. Dado que las acciones tienen tiempos asociados, a la hora de unir transiciones internas deberíamos unir también sus variables aleatorias correspondientes. No obstante, este tipo de unión implicaría operar con variables aleatorias de forma que una secuencia de ellas se pudiera corresponder con una única, en el caso de que ésta última tuviera una distribución adecuada. Basándonos en esta idea desarrollamos en esta tesis un estudio sobre distintas semánticas que cumplen esa característica.

Aunque en el modelo inicial las variables aleatorias estaban asociadas al resto de las acciones, decidimos que podrían resultar más sencillas e intuitivas las equivalencias semánticas que pensábamos estudiar en el resto del trabajo si llevábamos a cabo una separación entre retardos y acciones *ordinarias*. En un primer momento, siguiendo los pasos dados en nuestro primer modelo, decidimos estudiar una semántica de bisimulación que identificara algunas secuencias de variables aleatorias con una única transición etiquetada con una variable aleatoria. En realidad, nuestro marco semántico es más complejo y no se limita a unir transiciones consecutivas. Nuestra intención era definir una nueva noción de bisimulación en la que el *branching* producido por las decisiones tomadas por el proceso se mantuviera. Ello es particularmente relevante si consideramos la ramificación producida por transiciones internas. Por otro lado, pretendíamos abstraer secuencias de transiciones estocásticas que se pudieran agrupar sin afectar a esta misma ramificación. A pesar de que el formalismo utilizado para modelar procesos era extremadamente simple, las consideraciones anteriores han llevado a una noción de bisimulación, que llamamos bisimulación débil estocástica, cuya

formulación es extremadamente complicada. Una línea natural de trabajo futuro consiste en considerar un lenguaje más complejo, que incluya en particular un operador de composición paralela, e intentar adaptar nuestra noción de bisimulación al nuevo marco. Este punto está también comentado en las conclusiones del capítulo 6.

A la hora de estudiar una semántica de *testing* decidimos hacerlo sobre una clase de procesos distinta a la empleada en el capítulo anterior. En concreto, las elecciones entre distintas acciones están cuantificadas de forma probabilística. A pesar de esta cuantificación, estas elecciones no se resuelven de forma puramente probabilística, sino que se consideran las posibilidades que el entorno ofrece al proceso. La definición de paso de una prueba y la equivalencia *testing* tienen definiciones muy intuitivas. Cabe destacar que hasta la aparición de nuestro trabajo [LN01], en el cual está basado este capítulo, no se había definido una semántica de este tipo para distribuciones generales por la dificultad existente a la hora de combinar distintas variables aleatorias. No obstante, fuimos capaces de aprovechar las mismas ideas que utilizamos a la hora de definir nuestra noción de bisimulación débil estocástica para combinar variables aleatorias.

El objetivo a corto plazo es extender los resultados que hemos obtenido ya sobre *testing* de procesos estocásticos. En concreto, estamos interesados en definir una caracterización alternativa de nuestra equivalencia de *testing*. Para ello utilizaremos extensiones probabilísticas de los *conjuntos de aceptación* [Hen88] de forma que cada conjunto contenga las acciones que se pueden ejecutar y su probabilidad asociada. A continuación definiremos las transiciones que se forman mediante la unión de aquellas acciones visibles y estocásticas que los procesos pueden ejecutar para alcanzar determininados procesos de aceptación. De este modo, dos procesos serán equivalentes si tras las mismas transiciones llegan a los mismos conjuntos de aceptación.

En el último capítulo se estudia un lenguaje más completo que los anteriores. Sin embargo, para el lenguaje empleado en este último capítulo solamente definimos sobre él dos nociones de bisimulación: una fuerte y una débil. Por tanto, un trabajo que queda por realizar es el estudio de las semánticas de bisimulación y *testing* definidas en esta tesis para lenguajes con operador de composición paralela. Como ya se ha comentado anteriormente, esta tarea no es sencilla puesto que no basta con unir secuencias de acciones estocásticas que se ejecutan secuencialmente, sino también las que se ejecutan en paralelo. Esto provoca que tengamos que combinar variables aleatorias que empiezan a ejecutarse, posiblemente, en distintos instantes de tiempo. Para resolver este problema, intentaremos definir las trazas de las acciones que se van ejecutando en cada momento para poder combinar después las variables aleatorias

correspondientes.

Gracias a la implementación realizada en Eden de procesos especificados en nuestro lenguaje podemos conocer empíricamente algunas propiedades cuantitativas de interés. Para realizar este paso entre el álgebra de procesos y Eden hemos dado un algoritmo de traducción de las especificaciones a los programas Eden correspondientes. Sin embargo, existen algunas optimizaciones del código que son deseables y que no están recogidas en el algoritmo, siendo por tanto necesario, en algunas ocasiones, un tratamiento posterior *a mano* de los programas generados por el algoritmo. La principal ventaja de nuestra traducción es que podemos ampliar nuestro conocimiento de la especificación simulando su implementación en Eden. Este hecho es especialmente relevante en el caso de propiedades de evaluación de rendimiento, dado el hecho de que actualmente las técnicas de model checking no se pueden utilizar cuando consideramos distribuciones generalizadas.

Finalmente, una línea de investigación que no se ha tratado directamente en esta tesis, pero que está ligada en gran medida a la misma, es la aplicación a sistemas de comercio electrónico. En [LNRR02, LNRR03] hemos utilizado álgebras de procesos (en combinación con teoría microeconómica) para especificar formalmente un sistema de trueque electrónico. En dicho sistema, por cada usuario existe un agente que le representa en un mercado local utilizando funciones de utilidad, mientras que otro agente representa a cada mercado en mercados de nivel superior, teniendo en cuenta las funciones de utilidad de todos sus clientes. Nuestras primeras investigaciones nos han llevado a pensar que determinadas decisiones que se toman a la hora de realizar trueques deberían realizarse de forma no determinista. Sin embargo, estas decisiones deberían estar cuantificadas dependiendo de las preferencias de los usuarios. Ahora bien, las preferencias de los usuarios no son estáticas, sino que es bien sabido que pueden variar con el tiempo. Por dicho motivo, es necesario la introducción de información estocástica. De hecho, ya hemos empezado a utilizar modelos simples con información estocástica (similares al descrito en el capítulo 3) con el objetivo de describir el comportamiento de nuestros agentes, si bien nos encontramos aún en una fase bastante preliminar. Así pues, aún será preciso realizar un estudio más profundo en el futuro próximo.

Bibliografía

[AB01] S. Andova and J.C.M. Baeten. Abstraction in probabilistic process algebra. In *TACAS 2001, LNCS 2031*, pages 204–219. Springer, 2001.

[ABC+94] M. Ajmone Marsan, A. Bianco, L. Ciminiera, R. Sisto, and A. Valenzano. A LOTOS extension for the performance analysis of distributed systems. *IEEE/ACM Transactions on Networking*, 2(2):151–165, 1994.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[BA01] M. Bravetti and A. Aldini. Expressing processes with different action durations through probabilities. In *PAPM-PROBMIV 2001, LNCS 2165*, pages 168–183. Springer, 2001.

[BA03] M. Bravetti and A. Aldini. Discrete time generative-reactive probabilistic processes with different advancing speeds. *Theoretical Computer Science*, 290(1):355–406, 2003.

[BB93] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3:142–188, 1993.

[BBG98] M. Bravetti, M. Bernardo, and R. Gorrieri. Towards performance evaluation with general distributions in process algebras. In *CONCUR'98, LNCS 1466*, pages 405–422. Springer, 1998.

[BC00] M. Bernardo and W.R. Cleaveland. A theory of testing for markovian processes. In *CONCUR'2000, LNCS 1877*, pages 305–319. Springer, 2000.

[BCSS98] M. Bernardo, R. Cleaveland, S. Sims, and W. Stewart. TwoTowers: A tool integrating functional and performance analysis of concurrent systems. In *Formal*

Description Techniques for Distributed Systems and Communication Protocols (XI), and Protocol Specification, Testing, and Verification (XVIII), pages 457–467. Kluwer Academic Publishers, 1998.

[BDG94] M. Bernardo, L. Donatiello, and R. Gorrieri. Modelling and analyzing concurrent systems with MPA. In *2nd Workshop on Process Algebra and Performance Modelling*, pages 89–106, 1994.

[Ber99] M. Bernardo. *Theory and Application of Extended Markovian Process Algebra.* PhD thesis, Università di Bologna, 1999.

[BG98] M. Bernardo and R. Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202:1–54, 1998.

[BG99] M. Bravetti and R. Gorrieri. Interactive generalized semi-markov processes. In *7th International Workshop on Process Algebra and Performance Modelling*, pages 83–98, 1999.

[BG02a] M. Bravetti and R. Gorrieri. Deciding and axiomatizing weak ST bisimulation for a process algebra with recursion and action refinement. *ACM Transactions on Computational Logic*, 2002. to appear.

[BG02b] M. Bravetti and R. Gorrieri. The theory of interactive generalized semi-markov processes. *Theoretical Computer Science*, 282(1):5–32, 2002.

[BH97] C. Baier and H. Hermanns. Weak bisimulation for fully probabilistic processes. In *Computer Aided Verification'97, LNCS 1254*, pages 119–130. Springer, 1997.

[BKL+98] S. Breitinger, U. Klusik, R. Loogen, Y. Ortega, and R. Peña. DREAM: the distributed Eden abstract machine. In *Implementation of Functional Languages. LNCS 1467.*, pages 250–269. Springer, 1998.

[BKLL95] E. Brinksma, J.-P. Katoen, R. Langerak, and D. Latella. A stochastic causality-based process algebra. *The Computer Journal*, 38(7):553–565, 1995.

[BLOP98] S. Breitinger, R. Loogen, Y. Ortega, and R. Peña. Eden: Language definition and operational semantics. Technical Report, Bericht 96-10, Philipps-Universität Marburg, Germany, 1998.

[BM89] B. Bloom and A. R. Meyer. A remark on bisimulation between probabilistic processes. In *Logic at Botik'89, LNCS 363*, pages 26–40. Springer, 1989.

[BPS01] J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra.* North Holland, 2001.

[BS01] E. Bandini and R. Segala. Axiomatizations for probabilistic bisimulation. In *28th ICALP, LNCS 2076*, pages 370–381. Springer, 2001.

[BSW69] K.A. Bartlett, R.A. Scantlebury, and P.T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM*, 12(5):260–261, 1969.

[Buc94] P. Buchholz. Markovian process algebra: Composition and equivalence. In *2nd Workshop on Process Algebra and Performance Modelling*, pages 11–30, 1994.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Computer Science 18. Cambridge University Press, 1990.

[CCV+03] D. Cazorla, F. Cuartero, V. Valero, F.L. Pelayo, and J.J. Pardo. Algebraic theory of probabilistic and non-deterministic processes. *Journal of Logic and Algebraic Programming*, 55(1–2):57–103, 2003.

[CCVP01] D. Cazorla, F. Cuartero, V. Valero, and F.L. Pelayo. A process algebra for probabilistic and nondeterministic processes. *Information Processing Letters*, 80:15–23, 2001.

[CdFV97] F. Cuartero, D. de Frutos, and V. Valero. A sound and complete proof system for probabilistic processes. In *4th International AMAST Workshop on Real-Time Systems, Concurrent and Distributed Software, LNCS 1231*, pages 340–352. Springer, 1997.

[CDSY99] R. Cleaveland, Z. Dayar, S.A. Smolka, and S. Yuen. Testing preorders for probabilistic processes. *Information and Computation*, 154(2):93–148, 1999.

[CGP99] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 1999.

[Chr90a] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *CONCUR'90, LNCS 458*, pages 126–140. Springer, 1990.

[Chr90b] I. Christoff. *Testing Equivalences for Probabilistic Processes.* PhD thesis, Department of Computer Systems. Uppsala University, 1990.

[CSZ92] R. Cleaveland, S.A. Smolka, and A.E. Zwarico. Testing preorders for probabilistic processes. In *19th ICALP, LNCS 623*, pages 708–719. Springer, 1992.

[CY88] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *29th IEEE Symposium on Foundations of Computer Science*, pages 338–345. IEEE Computer Society Press, 1988.

[D'A99] P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems.* PhD thesis, Department of Computer Science. University of Twente, 1999.

[dBKP92] F.S. de Boer, J.W. Klop, and C. Palamidessi. Asynchronous communication in process algebra. In *7th IEEE Symposium on Logic In Computer Science (LICS)*, pages 137–147. IEEE Computer Society Press, 1992.

[dFLN97] D. de Frutos-Escrig, L.F. Llana-Díaz, and M.Ñúñez. Friendly testing as a conformance relation. In *Formal Description Techniques for Distributed Systems and Communication Protocols (X), and Protocol Specification, Testing, and Verification (XVII)*, pages 283–298. Chapman & Hall, 1997.

[dFLN98] D. de Frutos-Escrig, L.F. Llana-Díaz, and M.Ñúñez. An invitation to friendly testing. *Journal of Computer Science and Technology*, 13(6):531–545, 1998.

[dFLN99a] D. de Frutos-Escrig, N. López, and M.Ñúñez. Global bisimulations. Technical Report SIP 91/99, Dept. de Sistemas Informáticos y Programación. Universidad Complutense de Madrid, 1999.

[dFLN99b] D. de Frutos-Escrig, N. López, and M.Ñúñez. Global timed bisimulation: An introduction. In *Formal Description Techniques for Distributed Systems and Communication Protocols (XII), and Protocol Specification, Testing, and Verification (XIX)*, pages 401–416. Kluwer Academic Publishers, 1999.

[dFLN99c] D. de Frutos-Escrig, N. López, and M.Ñúñez. An introduction to global bisimulations. In *VII Jornadas de Concurrencia*, pages 113–126, 1999.

[DKB98a] P.R. D'Argenio, J.-P. Katoen, and E. Brinksma. An algebraic approach to the specification of stochastic systems. In *Programming Concepts and Methods*, pages 126–147. Chapman & Hall, 1998.

[DKB98b] P.R. D'Argenio, J.-P. Katoen, and E. Brinksma. General purpose discrete-event simulation using ♠. In *6th International Workshop on Process Algebra and Performance Modelling*, pages 85–102, 1998.

[dNH84] R. deÑicola and M.C.B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[dNH87] R. deÑicola and M. Hennessy. CCS without τ's. In *TAPSOFT'87, LNCS 249*, pages 138–152. Springer, 1987.

[EJF96] H. Erdogmus, R. Johnston, and M. Ferguson. On the operational semantics of nondeterminism and divergence. *Theoretical Computer Science*, 159:271–317, 1996.

[FH82] Y.A. Feldman and D. Harel. A probabilistic dinamic logic. In *14th ACM Symposium on Theory of Computing*, pages 181–195. ACM Press, 1982.

[GHR93] N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. In *16th Int. Symp. on Computer Performance Modelling, Measurement and Evaluation (PERFORMANCE'93), LNCS 729*, pages 121–146. Springer, 1993.

[GJS90] A. Giacalone, C.-C. Jou, and S.A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of Working Conference on Programming Concepts and Methods, IFIP TC 2*. North Holland, 1990.

[GLNP97] C. Gregorio, L. Llana, M.Ñúñez, and P. Palao. Testing semantics for a probabilistic-timed process algebra. In *4th International AMAST Workshop on Real-Time Systems, Concurrent, and Distributed Software, LNCS 1231*, pages 353–367. Springer, 1997.

[GN96] C. Gregorio and M.Ñúñez. Specifying and verifying the Alternating Bit Protocol with Probabilistic-Timed LOTOS. In *COST 247 International Workshop on Applied Formal Methods in System Design*, pages 38–50, 1996.

[GN99] C. Gregorio and M.Ñúñez. Denotational semantics for probabilistic refusal testing. In *PROBMIV'98, Electronic Notes in Theoretical Computer Science 22*. Elsevier, 1999.

[GSS95] R. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative and stratified models of probabilistic processes. *Information and Computation*, 121(1):59–80, 1995.

[GSST90] R. van Glabbeek, S.A. Smolka, B. Steffen, and C.M.N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *5th IEEE Symposium on Logic In Computer Science*, pages 130–141. IEEE Computer Society Press, 1990.

[GW96] R. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[Han91] H. Hansson. *Time and Probability in Formal Design of Distributed Systems.* PhD thesis, Department of Computer Systems. Uppsala University, 1991.

[Hen85] M. Hennessy. Acceptance trees. *Journal of the ACM*, 32(4):896–928, 1985.

[Hen88] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.

[Her98] H. Hermanns. *Interactive Markov Chains.* PhD thesis, Universität Erlangen-Nürnberg, 1998.

[Her00] H. Hermanns. Performance and reliability model checking and model construction. In *5th Int. Workshop on Formal Methods for Industrial Critical Systems*, 2000.

[HHK02] H. Hermanns, U. Herzog, and J.-P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274(1-2):43–87, 2002.

[Hil93] J. Hillston. PEPA: Performance Enhanced Process Algebra. Technical Report CSR-24-93, University of Edinburg, 1993.

[Hil96] J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[HJ89] H. Hansson and B. Jonsson. A framework for reasoning about time and realibility. In *10th IEEE Real-Time Systems Symposium*, pages 102–111. IEEE Computer Society Press, 1989.

[HKMS00] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. Towards model checking stochastic process algebra. In *IFM 2000, LNCS 1945*, pages 420–440. Springer, 2000.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[HPR00] F. Hernández, R. Peña, and F. Rubio. From GranSim to Paradise. In *Trends in Functional Programming*, pages 11–19. Intellect, 2000.

[HR91] M. Hennessy and T. Regan. A process algebra for timed systems. Technical Report CS0591, University of Sussex, 1991.

[HS84] S. Hart and M. Sharir. Probabilistic temporal logics for finite and bounded models. In *16th ACM Symposium on Theory of Computing*, pages 1–13. ACM Press, 1984.

[HS95] P.G. Harrison and B. Strulo. Stochastic process algebra for discrete event simulation. In *Quantitative Methods in Parallel Systems*, pages 18–37. Springer, 1995.

[HS00] P.G. Harrison and B. Strulo. SPADES – a process algebra for discrete event simulation. *Journal of Logic Computation*, 10(1):3–42, 2000.

[IHK01] G.G. Infante López, H. Hermanns, and J.-P. Katoen. Beyond memoryless distributions: Model checking semi-Markov chains. In *PAPM-PROBMIV 2001, LNCS 2165*, pages 57–70. Springer, 2001.

[JP89] C. Jones and G.D. Plotkin. A probabilistic powerdomain of evaluations. In *4th IEEE Symposium on Logic In Computer Science (LICS)*, pages 186–195. IEEE Computer Society Press, 1989.

[KBLL96] J.-P. Katoen, E. Brinksma, D. Latella, and R. Langerak. Stochastic simulation of event structures. In *4th Workshop on Process Algebras and Performance Modelling (PAPM'96)*, pages 21–40. CLUT, 1996.

[KLPR01] U. Klusik, R. Loogen, S. Priebe, and F. Rubio. Implementation skeletons in Eden: Low-effort parallel programming. In *Implementation of Functional Languages, IFL'00, LNCS 2011.* Springer, 2001.

[Koz83] D. Kozen. A probabilistic PDL. In *15th ACM Symposium on Theory of Computing*, pages 291–297. ACM Press, 1983.

[LdF99] L. Llana and D. de Frutos. Relating may and must testing semantics for discrete timed process algebras. In *ASIAN'99, LNCS 1742*, pages 74–86. Springer, 1999.

[LdFN96] L. Llana, D. de Frutos, and M.Ñúñez. Testing semantics for urgent timed algebras. In *3rd AMAST Workshop on Real-Time Systems*, pages 33–45, 1996.

[LL97] L. Léonard and G. Leduc. An introduction to ET-LOTOS for the description of time-sensitive systems. *Computer Networks and ISDN Systems*, 29:271–292, 1997.

[Lla96] L. Llana-Díaz. *Jugando con el Tiempo*. PhD thesis, Universidad Complutense de Madrid, 1996.

[LN00] N. López and M.Ñúñez. `NMSPA`: A non-markovian model for stochastic processes. In *International Workshop on Distributed System Validation and Verification (DSVV'2000)*, pages 33–40, 2000.

[LN01] N. López and M.Ñúñez. A testing theory for generally distributed stochastic processes. In *CONCUR 2001, LNCS 2154*, pages 321–335. Springer, 2001.

[LN02] N. López and M.Ñúñez. Stochastic bisimulation semantics for non-markovian processes, 2002. Submitted for publication.

[LNR02] N. López, M.Ñúñez, and F. Rubio. Stochastic process algebras meet Eden. In *3rd Integrated Formal Methods (IFM), LNCS 2335*, pages 29–48. Springer, 2002.

[LNR03] N. López, M.Ñúñez, and F. Rubio. An integrated framework for the analysis of asynchronous communicating stochastic processes, 2003. Submitted for publication.

[LNRR02] N. López, M.Ñúñez, I. Rodríguez, and F. Rubio. A formal framework for e-barter based on microeconomic theory and process algebras. In *Innovative Internet Computer Systems, LNCS 2346*, pages 217–228. Springer, 2002.

[LNRR03] N. López, M.Ñúñez, I. Rodríguez, and F. Rubio. A multi-agent system for e-barter including transaction and shipping costs. In *Symposium on Applied Computing, SAC 2003*. ACM Press, 2003. 8 pages. In press.

[Loi96] H.W. Loidl. *GranSim User's Guide*. Department of Computing Science, Glasgow University, 1996.

[Lóp99] N. López. Una semántica amistosa para álgebras de procesos concurrentes. Master Thesis. Dept. Sistemas Informáticos y Progra-

mación. Universidad Complutense de Madrid, 1999. Available at `http://dalila.sip.ucm.es/~natalia/publications.html`.

[Low95] G. Lowe. Probabilistic and prioritized models of timed CSP. *Theoretical Computer Science*, 138:315–352, 1995.

[LS89] K. Larsen and A. Skou. Bisimulation through probabilistic testing. In *16th ACM Simposium on Principles of Programming Languages*, pages 344–352. ACM Press, 1989.

[LS91] K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[LT87] N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *6th ACM Symp. on Principles of Distributed Computing*, pages 137–151. ACM Press, 1987.

[MdM88] J.A. Mañas and T. de Miguel. From LOTOS to C. In *FORTE'88*, pages 79–84. North-Holland, 1988.

[Mil81] R. Milner. A modal characterization of observable machine-behaviour. In *6th CAAP, LNCS 112*, pages 25–34. Springer, 1981.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 253:267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[MT90] F. Moller and C. Tofts. A temporal calculus of communicating systems. In *CONCUR'90, LNCS 458*, pages 401–415. Springer, 1990.

[NCS98] K.Ñarayan Kumar, R. Cleaveland, and S.A. Smolka. Infinite probabilistic and nonprobabilistic testing. In *18th Conference on Foundations of Software Technology and Theoretical Computer Science, LNCS 1530*, pages 209–220. Springer, 1998.

[NdF95] M.Ñúñez and D. de Frutos. Testing semantics for probabilistic LOTOS. In *Formal Description Techniques VIII*, pages 365–380. Chapman & Hall, 1995.

[NdFL95] M.Ñúñez, D. de Frutos, and L. Llana. Acceptance trees for probabilistic processes. In *CONCUR'95, LNCS 962*, pages 249–263. Springer, 1995.

[NR99] M.Ñúñez and D. Rupérez. Fair testing through probabilistic testing. In *Formal Description Techniques for Distributed Systems and Communication Protocols (XII), and Protocol Specification, Testing, and Verification (XIX)*, pages 135–150. Kluwer Academic Publishers, 1999.

[NS91] X.Ñicollin and J. Sifakis. An overview and synthesis on timed process algebras. In *Computer Aided Verification'91, LNCS 575*, pages 376–398. Springer, 1991.

[NS94] X.Ñicollin and J. Sifakis. The algebra of timed process, ATP: Theory and application. *Information and Computation*, 114(1):131–178, 1994.

[Núñ96] M.Ñúñez. *Semánticas de Pruebas para Álgebras de Procesos Probabilísticos.* PhD thesis, Universidad Complutense de Madrid, 1996.

[Núñ02] M.Ñúñez. Algebraic theory of probabilistic processes, 2002. To appear in *Journal of Algebraic and Logic Programming*, 67 pages.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *5th G.I. Conference, LNCS 104*, pages 167–183. Springer, 1981.

[PCVC00] F. Pelayo, F. Cuartero, V. Valero, and D. Cazorla. An example of performance evaluation by using the stochastic process algebra ROSA. In *7th Int. Conf. on Real-Time Systems and Applications*, pages 271–278. IEEE Computer Society Press, 2000.

[Pey96] S. L. Peyton Jones. Compiling Haskell by Program Transformation: A Report from the Trenches. In *ESOP'96 — European Symposium on Programming*, volume 1058 of *LNCS*, pages 18–44, Linköping, Sweden, April 22–24, 1996. Springer-Verlag.

[PGF96] S.L. Peyton Jones, A. Gordon, and S. Finne. Concurrent Haskell. In *ACM Symp. on Principles of Prog. Lang. POPL'96*, pages 295–308. ACM Press, 1996.

[PH99] S.L. Peyton Jones and J. Hughes, editors. *Report on the Programming Language Haskell 98.* Available at http://www.haskell.org, 1999.

[PHHP92] S. L. Peyton Jones, C. V. Hall, K. Hammond, and W. Partain. The Glasgow Haskell Compiler: a Technical Overview. Department of Computer Science, University of Glasgow, December 1992.

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

[PLS00] A. Philippou, I. Lee, and O. Sokolsky. Weak bisimulation for probabilistic systems. In *CONCUR 2000, LNCS 1877*, pages 334–349. Springer, 2000.

[PR01] R. Peña and F. Rubio. Parallel Functional Programming at Two Levels of Abstraction. In *Principles and Practice of Declarative Programming (PPDP01)*, pages 187–198. ACM Press, 2001.

[QdFA93] J. Quemada, D. de Frutos, and A. Azcorra. TIC: A TImed Calculus. *Formal Aspects of Computing*, 5:224–252, 1993.

[QdFML94] J. Quemada, D. de Frutos, C. Miguel, and L. Llana. A timed LOTOS extension. In *Theories and Experiences for Real-Time System Development, Amast Series in Computing*, pages 239–263. World Scientific, 1994.

[Rab63] M.O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[RR88] G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[RR99] G.M. Reed and A.W. Roscoe. The timed failures – stability model for CSP. *Theoretical Computer Science*, 211(1–2):85–127, 1999.

[Sch89] Steve Schneider. *Correctness and Communication in Real-Time Systems.* PhD thesis, Oxford University, 1989.

[Sch95] S. Schneider. An operational semantics for timed CSP. *Information and Computation*, 116:193–213, 1995.

[Sei95] K. Seidel. Probabilistic communicating processes. *Theoretical Computer Science*, 152:219–249, 1995.

[SS00] E.W. Stark and S.A. Smolka. A complete axiom system for finite-state probabilistic processes. In *Proof, Language and Interaction: Essays in Honour of Robin Milner.* MIT Press, 2000.

[SVD01] J. Springintveld, F. Vaandrager, and P.R. D'Argenio. Testing timed automata. *Theoretical Computer Science*, 254(1-2):225–257, 2001.

[Tan96] A.S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.

[Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th IEEE Symposium on Foundations of Computer Science*, pages 327–338. IEEE Computer Society Press, 1985.

[WSS94] S.-H. Wu, S.A. Smolka, and E.W. Stark. Composition and behaviors of probabilistic I/O automata. In *CONCUR'94, LNCS 836*, pages 513–528. Springer, 1994.

[WSS97] S.-H. Wu, S.A. Smolka, and E.W. Stark. Composition and behaviors of probabilistic I/O automata. *Theoretical Computer Science*, 176(1-2):1–37, 1997.

[Yi90] W. Yi. Real-time behavior of asynchronous agents. In *CONCUR'90, LNCS 458*, pages 502–520. Springer, 1990.

[Yi91] W. Yi. *A Calculus of Real Time Systems*. PhD thesis, Department of Computer Science. Chalmers University of Technology, 1991.

[YL92] W. Yi and K.G. Larsen. Testing probabilistic and nondeterministic processes. In *Protocol Specification, Testing and Verification XII*, pages 47–61. North Holland, 1992.