PEPA Workbench: : Una Herramienta para Evaluación de Prestaciones basada en Álgebras de Procesos*

Fernando L. Pelayo M. Carmen Ruiz Diego Cazorla

Escuela Politécnica Superior de Albacete
Universidad de Castilla-La Mancha
Campus Universitario s/n
02071, Albacete. España
{fpelayo, carmenr, dcazorla}@info-ab.uclm.es

Resumen En este artículo se presenta una técnica para la evaluación de prestaciones así como una herramienta que implementa dicha técnica. PEPA (Performance Evaluation Process Algebra) es un lenguaje algebraico que permite construir modelos de sistemas que capturan información cuantitativa. PEPA Workbench es una herramienta que, a partir de la especificación PEPA del modelo de un sistema, permite derivar ciertas informaciones acerca de las prestaciones del sistema, tales como porcentaje de utilización de recursos, tiempos de respuesta y rendimientos.

1 Introducción

El término *modelo* puede utilizarse con el significado de "aquello" que representa un sistema y que nos permite hacer predicciones acerca del comportamiento del mismo. Cómo de acertadas son esas predicciones dependerá del nivel de abstracción que presente el modelo frente al sistema real. En algunos casos, sólo necesitaremos una ligera aproximación al comportamiento del sistema, por lo que un alto nivel de abstracción será suficiente. Cuando los resultados que esperamos del estudio del modelo son cruciales, tendremos que hacer una esfuerzo mayor en la especificación del mismo, de manera que el modelo se aproximará al sistema real tanto como sea necesario y posible.

En cualquier caso, un modelo debe verse como una abstracción del sistema. Debemos descartar todos los detalles no relevantes y centrarnos en aquellos aspectos que son esenciales para el comportamiento del sistema.

Hay varias razones que favorecen el estudio sobre modelos frente a la experimentación directa sobre el sistema real. En el mejor de los casos, la experimentación directa puede ser perjudicial para el funcionamiento normal del sistema (p. ej. un administrador de sistemas experimentando sobre un sistema en producción), y en el peor, puede suponer un peligro (p. ej. experimentar sobre un reactor

* Este trabajo ha sido parcialmente financiado con el subproyecto CICYT Evaluación de Rendimientos de Sistemas Distribuidos, TIC2000–0701–C02–02

nuclear real). La experimentación directa es cara (en términos de recursos del sistema) y consume tiempo. Además, es difícil usar los resultados de una experimentación directa para extrapolar resultados a otros escenarios de operación. Por ejemplo, experimentar con una configuración de una red de comunicaciones es probable que no nos de mucha información sobre el comportamiento de dicha red cuando se use una configuración distinta. Esto nos conduce a una de las razones principales para usar modelos en lugar de la experimentación directa: es habitual que el sistema que queremos estudiar no exista.

La idea de que el diseño de un sistema debería estar sujeto a un cierto análisis que asegure que el sistema se comportará *correctamente* es, hoy día, ampliamente aceptada. Los modelos son herramientas comúnmente usadas para este propósito. Por ejemplo el álgebra de procesos CCS [5] puede utilizarse para comprobar que una implementación satisfará las condiciones de comportamiento dadas en la especificación. Este tipo de modelo reciben el nombre de *modelo cualitativo* porque abstrae todos los valores numéricos y se centra en aspectos funcionales. En contraste, otras veces estamos interesados en asegurar que el sistema satisfará ciertas restricciones acerca del comportamiento temporal o la utilización eficiente de los recursos. Estos modelos reciben el nombre de *modelos cuantitativos* y en ellos se basan las técnicas de evaluación de prestaciones.

La evaluación de prestaciones consiste en la *descripción* (o *especificación*), *análisis* y *optimización* del comportamiento dinámico de los sistemas. Para ello, partimos de un modelo cuantitativo del sistema (descripción), a partir del cuál pueden derivarse, utilizando determinadas técnicas, ciertos valores que son críticos para el problema que queremos resolver (análisis). Evidentemente en la especificación del modelo sólo tendremos en cuenta aquellos aspectos cuantitativos que tiene relevancia para el problema, y haremos abstracción del resto. La comparación de los valores obtenidos en el análisis del modelo con los valores esperados nos indicará la bondad del modelo y, por tanto, del sistema. En el caso de que los valores obtenidos no cumplan las expectativas puede actuarse sobre alguno de los parámetros del modelo y repetir las fases de descripción y análisis hasta obtener los resultados esperados (optimización).

La evaluación de prestaciones es útil, entre otras, en las siguientes áreas de estudio: evaluación de diseños alternativos, comparación de sistemas, determinación de los parámetros de funcionamiento óptimo, búsqueda de cuellos de botella, caracterización de la carga del sistema, predicción de cargas futuras. Por ejemplo, considérese el siguiente escenario: una compañía que fabrica coches está diseñando y construyendo un sistema CAD que permite a treinta diseñadores acceder simultáneamente desde estaciones gráficas a una base de datos distribuida. En la fase de diseño deben tomarse decisiones fundamentales acerca de los mecanismos de acceso a la base de datos y de sincronización y comunicación. Por lo tanto, sería deseable que, antes de la implementación, se evaluaran diversas alternativas con el propósito de elegir la más adecuada.

Podemos usar una gran variedad de *medidas* o *índices* para caracterizar el comportamiento de un sistema. Quizás, los más frecuentemente usados son el *tiempo de respuesta* (tiempo que transcurre entre la llegada de una petición y la terminación de esa petición por parte del sistema), el *rendimiento* (número de peticiones satisfechas por unidad de tiempo) y la *utilización* (porcentaje de tiempo en que un recurso está siendo usado). Otras medidas de interés podrían ser el número total de peticiones en un periodo de tiempo, el número medio de peticiones en espera, o el tiempo medio de espera de una petición mientras el sistema está ocupado.

El artículo se estructura de la manera siguiente. En la sección 2 se introduce la teoría relacionada con los procesos Markovianos y los métodos de resolución de dichos procesos cuando se considera el estado de equilibrio. En la sección 3 se define la sintaxis y la semántica operacional del álgebra de procesos estocástica PEPA. En la sección 4 se introduce la herramienta PEPA Workbench, y finalmente en la sección 5 se muestra un caso de estudio.

2 Construcción y Resolución de Procesos Markovianos

2.1 Variables aleatorias

Un modelo estocástico es aquél que tiene un comportamiento aleatorio. Utilizamos dichos modelos porque el mundo exterior que interactúa con los sistemas que queremos estudiar está repleto de comportamientos aleatorios. Además, la naturaleza estocástica de los modelos nos permite considerar comportamientos típicos o medios, en lugar de comportamientos particulares. Nos basaremos en la teoría de probabilidades para construir y resolver nuestros modelos.

La teoría de probabilidades trata con eventos que son impredecibles cuando son tratados de forma individual, pero que muestran cierta regularidad o predictibilidad cuando se tratan en gran número. Estamos familiarizados con estas ideas cuando tiramos una moneda al aire o lanzamos un dado. La no predictibilidad de un solo evento de este tipo se debe a la naturaleza no determinista de la situación. Cuando hablamos de un computador o un sistema de comunicaciones la no predictibilidad de un evento se debe a nuestra incapacidad de medir o especificar las condiciones exactas bajo las cuales se ha producido dicho evento.

Cuando realizamos un experimento generalmente estamos más interesados en ciertos valores generales que pueden obtenerse de él que en cada resultado en particular. Lo que hacemos es construir una correspondencia entre las observaciones que estamos realizando (espacio de muestreo) y el valor que nos interesa. Una función de este tipo, del espacio de muestreo al conjunto de los números reales o naturales, es una *variables aleatoria* continua o discreta, respectivamente. Habitualmente estamos interesados en calcular, para cada valor posible, con que probabilidad la variable aleatoria toma ese valor. Desde el otro punto de vista, el conocimiento de estas probabilidades nos da mucha información acerca del espacio de muestreo modelado por la variable aleatoria.

Cuando la variable aleatoria toma valores discretos (como los números naturales), podemos asociar probabilidades de forma muy intuitiva. Así, la probabilidad de que una variable aleatoria X tome el valor v es igual a la probabilidad de la unión de todos los eventos que conducen a $X = v$.

Supongamos que la variable aleatoria X toma valores $\{x_i \mid i \in I\}$ y para todo $i \in I$ la probabilidad de que $X = x_i$ es igual a p_i. Entonces $\{p_i \mid i \in I\}$ es la *distribución de probabilidad* de X. Esta correspondencia entre x_i y p_i se llama habitualmente *función de masa*. En un experimento de lanzamiento de dado, los variables que toma la variable aleatoria son $\{1, 2, 3, 4, 5, 6\}$ y la distribución de probabilidad es $\{1/6, 1/6, 1/6, 1/6, 1/6, 1/6\}$. Se trata de una distribución *uniforme*.

El *valor esperado* de X se define como

$$E[X] = \sum_{i \in I} x_i \times p_i$$

Informalmente, podemos decir que ese valor es el valor medio que tendrá X. En el ejemplo del dado será $(1 \times 1/6) + \cdots + (6 \times 1/6) = 3.5$.

Para algunos espacios de muestreo, las variables aleatorias con valores discretos no son apropiadas. Por ejemplo, si estamos interesados en conocer el número de líneas telefónicas en uso cada hora entonces una variable aleatoria discreta es suficiente. Sin embargo, si estamos interesados en la duración de cada conexión, la variable puede tomar cualquier valor no negativo, y por la tanto necesitamos una variable aleatoria continua.

En el caso de variables aleatorias continuas no podemos asignar una probabilidad a cada valor. En su lugar, podemos definir la probabilidad de que la variable aleatoria tome un valor menor que un valor dado. Esto se hace mediante una *función de distribución* F. Para una variable aleatoria continua X, la función de distribución se define como

$$F(x) = \Pr(X \leq x)$$

Las funciones de distribución de probabilidad satisfacen ciertas condiciones:

1. $F(x_1) \leq F(x_2)$ si $x_1 \leq x_2$;
2. $F(x) \to 0$ cuando $x \to -\infty$;
3. $F(x) \to 1$ cuando $x \to \infty$;
4. $F(x)$ es continua.

La derivada de la función de distribución es la *función de densidad* $f(x)$

$$f(x) = \frac{dF(x)}{dx}$$

El *valor esperado* de la variable aleatoria continua X se define como

$$E[X] = \int_{-\infty}^{\infty} x f(x) dx$$

La distribución continua de probabilidades más importante desde el punto de vista de la evaluación de prestaciones es la *distribución exponencial (negativa)* que tiene la función de distribución

$$F(x) = 1 - e^{-\mu x} \quad x \geq 0 \quad \text{y función de densidad} \quad f(x) = \mu e^{-\mu x}$$

El valor esperado de una variable aleatoria exponencial con parámetro μ es $1/\mu$. Esta variable aleatoria suele utilizarse para representar el tiempo que transcurre hasta que sucede un determinado evento.

La razón de que la distribución exponencial sea tan usada en evaluación de prestaciones se debe a que presenta algunas propiedades matemáticas muy interesantes:

– La distribución exponencial *no tiene memoria*: si X es una variable aleatoria distribuida exponencialmente, entonces

$$\Pr(X > t + s \mid X > t) = \Pr(X > s) \qquad t > 0, s > 0$$

Si X representa el tiempo de espera hasta que se produzca un evento, esto implica que conocer que han pasado t unidades de tiempo sin que se haya producida el evento no nos da información adicional sobre cuando va a ocurrir el evento; la distribución del tiempo de espera restante es la misma que si no hubiera pasado tiempo.

– Si X_1 y X_2 son dos variables aleatorias distribuidas exponencialmente, de parámetros λ_1 y λ_2 respectivamente, e Y se define como el mínimo de ambas, $Y = \min(X_1, X_2)$, entonces Y también está distribuida exponencialmente, con parámetro $\lambda_1 + \lambda_2$.

2.2 Procesos Estocásticos

Un proceso estocástico es un conjunto de variables aleatorias $\{X(t), t \in T\}$ donde T es el conjunto índice. Cuando consideramos modelos continuos T es el conjunto de los número reales, $\mathbb{R}$. El *espacio de estados* del proceso es el conjunto de todos los posibles valores que las variables aleatorias $X(t)$ pueden tomar. Cada uno de esos valores recibe el nombre de *estado* del proceso.

Por ejemplo, podemos asociar a cada línea telefónica una variable aleatoria que toma los valores 1 o 0 en función de si la línea está ocupada o libre. En cualquier instante las variables aleatorias que representan el sistema serán el conjunto de variables aleatorias que representan cada una de las líneas. El estado del sistema será una n-tupla, donde n es el número de líneas y cada entrada representa el estado de una línea individual. El estado del sistema cambiará cuando se establezca una nueva conexión o termine una ya establecida. Alternativamente, podemos considerar el proceso estocástico caracterizado por una variable aleatoria M, que registra el número de líneas en uso. Un estado de este proceso estocástico se corresponde con varios estados del proceso anterior.

2.3 Procesos Markovianos

Un proceso Markoviano es un proceso estocástico tal que el conjunto de variables aleatorias $\{X(t)\}$ tiene la *propiedad de Markov o de ausencia de memoria*, es decir, dado un valor $X(t)$ con $t \in T$, los valores futuros $X(s)$, $s > t$, no dependen de la historia pasada. Esto quiere decir que una vez que hemos llegado a un estado el comportamiento futuro es el mismo independientemente del camino utilizado para llegar a él.

Un proceso Markoviano está caracterizado por una variable aleatoria $X(t)$, indexada sobre el tiempo t, y un espacio de estados S tal que $X(t) \in S$ para todo t. El comportamiento dinámico el sistema está representado por medio de transiciones entre los estados y el tiempo pasado en cada estado o *tiempo de estancia*. Las transiciones representan eventos del sistema.

Debido a la propiedad de Markov, en cualquier instante de tiempo la distribución de tiempo hasta el próximo cambio de estado es independiente del instante en que se produjo el cambio de estado anterior. En otras palabras, el tiempo de estancia no tiene memoria. Como la única función de distribución de probabilidad que tiene esta propiedad es la exponencial, esto nos indica que el tiempo de estancia en un estado de un proceso Markoviano es una variable aleatoria distribuida exponencialmente.

2.4 Diagramas de transición de estados

Para procesos Markovianos pequeños, la forma más simple de representar el proceso es por medio de un diagrama de transición de estados. En estos diagramas se representa cada estado como un nodo del grafo. Los arcos representan posibles transiciones entre estados del proceso. Los arcos están etiquetados por la *velocidad de transición* entre estados. Como cada transición está gobernada por por una distribución exponencial, la velocidad de la transición será el parámetro de la distribución.

Ejemplo: Considérese una centralita telefónica cuyo estado representaremos simplemente mediante el número de líneas en uso. Si hay 6 líneas, el espacio de estados es $\{0, 1, 2, 3, 4, 5, 6\}$. Cada estado tendrá transiciones a los estados anterior y posterior (excepto los estados 0 y 6 obviamente). Consideramos que la duración media de una conexión es $1/\mu$, luego la velocidad asociada a una transición de liberación de una línea es μ. Por otra parte, el tiempo medio de llegada de nuevas conexiones es $1/\lambda$, luego la velocidad asociada a una transición de ocupación de una línea es λ. El diagrama de estados para este proceso se muestra en la figura 1

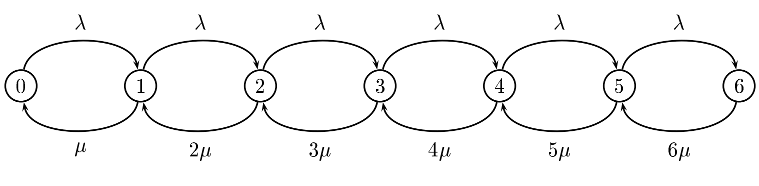


Figura1. Diagrama de transición de estados de un proceso Markoviano

2.5 Matrices generadoras

Un proceso markoviano con n estados puede también caracterizarse por su *matriz generadora infinitesimal* $\mathbf{Q}$ de dimensión $n \times n$. La entrada en la columna j de la fila i $(i \neq j)$, que denotaremos mediante q_{ij}, es la suma de los parámetros que etiquetan los arcos que conectan los nodos i y j en el diagrama de transición de estados. Los elementos de la diagonal se eligen de tal manera que se asegure que las suma de los elementos de cada fila es cero, es decir, $q_{ii} = -\sum_{j \in S, j \neq i} q_{ij}$.

En la figura 2 se muestra la matriz generadora correspondiente al diagrama de estados de la figura 1.

$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & 0 & 0 & 0 \\ \mu & -(\mu+\lambda) & \lambda & 0 & 0 & 0 & 0 \\ 0 & 2\mu & -(2\mu+\lambda) & \lambda & 0 & 0 & 0 \\ 0 & 0 & 3\mu & -(3\mu+\lambda) & \lambda & 0 & 0 \\ 0 & 0 & 0 & 4\mu & -(4\mu+\lambda) & \lambda & 0 \\ 0 & 0 & 0 & 0 & 5\mu & -(5\mu+\lambda) & \lambda \\ 0 & 0 & 0 & 0 & 0 & 6\mu & -6\mu \end{pmatrix}$$

Figura2. Matriz generadora de un proceso Markoviano

2.6 Estado de equilibrio

La evaluación de prestaciones trata, habitualmente, del estudio de los sistemas sobre un periodo largo de tiempo. El tiempo de ejecución del sistema será mucho más largo que el tiempo entre dos eventos individuales del sistema. Desde el punto de vista de la modelización, esto es importante porque puede ocurrir que tengamos que elegir un estado inicial de forma arbitraria. Esto puede tener un gran impacto sobre la simulación, ya que dependiendo del estado inicial elegido tendremos distintos caminos de ejecución sobre el espacio de estados. En los modelos Markovianos evitamos este problema ya que no consideramos caminos individuales sino distribuciones de probabilidad sobre los estados del proceso, que pueden ser deducidas conociendo únicamente las probabilidades instantáneas de transición entre estados.

Cuando ha pasado suficiente tiempo, el modelo (y el sistema) entran en el *estado de equilibrio*. Esto no significa que el sistema permanezca en un estado sin realizar transiciones. Significa que el modelo presenta regularidad y predictibilidad en su comportamiento sobre el espacio de estados. Es decir, la distribución de probabilidad de la variable aleatoria $X(t)$ sobre el espacio de estados S no cambia.

Mediante $\pi_k(t)$ denotamos la probabilidad de que el proceso Markoviano se encuentre en el estado x_k en el instante t, para $x_k \in S$. El equilibrio se alcanza cuando, para todos los estados $x_k \in S$, $\pi_k(t+\tau) = \pi_k(t)$, es decir, el instante en que observamos el modelo no influye en la probabilidad de que se encuentre en un estado determinado. Hablaremos entonces de probabilidades del estado de equilibrio, de forma que mediante π_k denotaremos la probabilidad de que el modelo se encuentre en el estado x_k.

2.7 Ecuaciones de balance global

En el estado de equilibrio, π_i es la proporción de tiempo que el proceso pasa en el estado x_i. Por otra parte, q_{ij} representa la probabilidad instantánea de que el modelo haga una transición desde el estado x_i al estado x_j. Por lo tanto, en un instante, la probabilidad de que ocurra una transición desde x_i hasta x_j será el producto $\pi_i \times q_{ij}$. Este valor recibe el nombre de *flujo de probabilidad.*

Cuando el modelo está en el estado de equilibrio, para que ese equilibrio se mantenga, debemos asumir que el flujo total de probabilidad de entrada al estado es igual al flujo total de probabilidad de salida del estado. Si no fuera así la distribución de probabilidades sobre los estados cambiaría. Para un estado particular x_i se cumple

$$\underbrace{\pi_i \times \sum_{x_j \in S, j \neq i} q_{ij}}_{\text{flujo de salida}} = \underbrace{\sum_{x_j \in S, j \neq i} (\pi_j \times q_{ji})}_{\text{flujo de entrada}}$$

El conjunto de las ecuaciones para todos los estados $x_i \in S$ recibe el nombre de *ecuaciones de balance global.* Teniendo en cuenta que en la matriz generadora se cumplía $q_{ii} = -\sum_{x_j \in S, j \neq i} q_{ij}$, podemos reescribir la ecuación anterior de la forma siguiente

$$\sum_{x_j \in S} \pi_j q_{ji} = 0$$

Y expresando los valores π_i como el vector fila $\boldsymbol{\pi}$, podemos escribir la siguiente ecuación matricial:

$$\boldsymbol{\pi} \mathbf{Q} = 0$$

Los π_i son los valores que queremos encontrar ya que representan la distribución de probabilidad sobre los estados. Si hay n estados, tendremos n ecuaciones. Estas n ecuaciones no son linealmente independientes, por lo que para resolver el sistema necesitamos una ecuación mas que surge de la *condición de normalización* de una distribución de probabilidades:

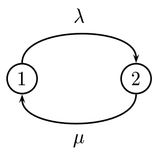
$$\sum_{x_i \in S} \pi_i = 1$$

Con estas $n + 1$ ecuaciones podemos resolver el sistema y encontrar la n incognitas, $\{\pi_i\}$.

El conocimiento de la distribución de probabilidades de los estados del proceso Markoviano en equilibrio nos permite obtener posteriormente diferentes medidas acerca de las prestaciones del sistema, como por ejemplo utilización de recursos, rendimientos, y tiempos de respuesta.

Ejemplo: consideramos un sistema con múltiples CPUs, cada una con su memoria privada, y una memoria común que sólo puede ser accedida en cada instante por un procesador. Las CPUs utilizan su memoria privada un tiempo aleatorio regido por una distribución exponencial de parámetro λ (el tiempo medio que una CPU utiliza su memoria privada entre dos peticiones de acceso a memoria común es $1/\lambda$). La duración del acceso a la memoria común se rige por una distribución exponencial de parámetro μ. Si el sistema sólo tiene un procesador alcanzará dos estados: utiliza memoria privada (1) o utiliza memoria común (2).

El comportamiento del sistema se puede modelar mediante un proceso Markoviano de dos estados cuya diagrama de transiciones de estado y matriz generadora se muestran en la figura 3



$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

Figura3. Proceso Markoviano de dos estados

Resolviendo el sistema

$$\left.\begin{aligned} -\pi_1\lambda + \pi_2\mu = 0 \\ \pi_1\lambda - \pi_2\mu = 0 \\ \pi_1 + \pi_2 = 1 \end{aligned}\right\}$$

obtenemos la distribución de probabilidades en equilibrio $\boldsymbol{\pi} = \left(\dfrac{\mu}{\mu+\lambda}, \dfrac{\lambda}{\mu+\lambda}\right)$, y por tanto la probabilidad de que el procesador esté utilizando su memoria privada es $\mu/(\mu+\lambda)$. En el caso de considerar dos procesadores se obtiene un proceso Markoviano de 5 estados; dando valores a los parámetros podemos conocer, por ejemplo, la probabilidad de que se esté utilizando memoria común por alguno de los procesadores.

3 Performance Evaluation Process Algebra (PEPA)

Las *álgebras de procesos* son lenguajes que permiten la especificación y diseño de sistemas concurrentes, de tal manera que los modelos obtenidos pueden ser estudiados desde un punto de vista cualitativo (p. ej. ausencia de bloqueos).

Las *álgebras de procesos estocásticas* [4] extienden las álgebras de procesos introduciendo una cierta duración a las acciones, duración que está modelada por una variable aleatoria exponencial. La principal ventaja de las álgebras de procesos estocásticas frente a otros formalismos (p. ej. redes de Petri estocásticas) es que incorporan algunas de las buenas propiedades de las álgebras de procesos. La más importante de todas ellas es la *composicionalidad*, es decir, la capacidad de modelar un sistema como una interacción de sus subsistemas.

PEPA (Performance Evaluation Process Algebra) [2] es un álgebra de procesos estocástica. En PEPA, un sistema se describe como un conjunto de *componentes*. Cada componente puede realizar un conjunto de acciones o actividades; una acción a es un par (α, r), dónde α es el *tipo* de la acción y $r \in \mathbb{R}^+$ es el parámetro de la distribución exponencial que gobierna su duración; r puede tomar el símbolo especial $\top$, cuando dicho parámetro no está especificado. Se considera además un tipo de acción *desconocido*, τ, que se utiliza en aquellos casos en el que la identidad de una acción es desconocida o irrelevante.

PEPA utiliza un pequeño conjunto de operadores que permiten construir comportamiento complejos a partir de otros más simples. La sintaxis de PEPA se define como sigue:

$$P ::= (\alpha, r).P \mid P \underset{L}{\bowtie} Q \mid P + Q \mid P/L \mid X \mid A$$

Prefijo: $(\alpha, r).P$. El prefijo es el mecanismo básico que permite construir el comportamiento secuencial de un componente. El componente $(\alpha, r).P$ realiza la actividad (α, r), que tiene el tipo de acción α y una duración dada mediante una distribución exponencial con parámetro r (valor esperado $1/r$). El tiempo para que la actividad se complete estará dado por cierto Δt, obtenido de la distribución exponencial. El componente posteriormente se comporta como el componente P. Si $a = (\alpha, r)$, el componente $(\alpha, r).P$ puede escribirse como $a.P$.

Elección: $P + Q$. El componente $P + Q$ representa un sistema que puede comportarse como P o como Q. $P + Q$ habilita todas las actividades actuales de P y todas las actividades actuales de Q. La primera actividad que se complete permite distinguir (elegir) uno de los componentes de la elección. El otro componente es descartado. La naturaleza continua de las distribuciones de probabilidad asegura que la probabilidad de que P y Q completen una actividad al mismo tiempo es cero. El sistema se comportará posteriormente como P' o Q', donde P' es el componente que resulta de completar una actividad de P , y Q' es el componente que resulta de completar una actividad de Q

Cooperación: $P \underset{L}{\bowtie} Q$. El operador de cooperación es en realidad una familia indexada de operadores, uno por cada posible conjunto L de tipos de acciones. El conjunto L, el *conjunto de cooperación*, define los tipos de acciones sobre los que los componentes P y Q deben sincronizar o cooperar, es decir, determina la interacción entre componentes. El tipo de acción desconocido no puede aparecer en L.

Todas las actividades de P y Q que no aparecen en el conjunto de cooperación L reciben el nombre de *individuales*, y no se ven afectadas por la cooperación. Las actividades que aparecen en L reciben el nombre de *compartidas* y sólo pueden llevarse a cabo cuando están habilitadas en ambas componentes. Por tanto, un componente puede verse bloqueado hasta que el otro componente está preparado para participar. En realidad, de la cooperación de ambos componentes nace una nueva actividad compartida, cuyo tipo es el mismo que el de las dos actividades que sincronizan, y la velocidad es la de la actividad más lenta.

Si una actividad tiene una velocidad no especificada en un componente, se dice que el componente es *pasivo* con respecto a ese tipo de acción. Esto significa que, aunque la cooperación del componente es necesaria para realizar la acción, el componente no contribuye al trabajo realizado.

Ocultamiento: P/L. El componente se comporta como P salvo que los tipos de acciones del conjunto L están *ocultados*. En el lugar del tipo de acción aparece el tipo desconocido τ, que puede ser interpretado como una retraso interno del componente.

La duración de una acción no se ve afectada si es ocultada. Sin embargo, una acción ocultada no puede utilizarse para cooperar con otro componente. En realidad, el tipo de acción τ no es visible para un observador o componente externo.

Variable: X. Si E es un componente que contiene la variable X, entonces $E[P/X]$ representa el componente tal que cada ocurrencia de X se ha reemplazado por el componente P.

Constante: $A \stackrel{def}{=} P$. Una constante es un componente cuyo significado está dado mediante una definición del tipo $A \stackrel{def}{=} P$, que asigna a la constante A el comportamiento del componente P. Permite asignar nombres a componentes (comportamientos).

La semántica operacional del lenguaje se muestra en la tabla 1. En una de las reglas relativas al operador de cooperación aparece el término $r_\alpha(E)$, la velocidad aparente del tipo de acción α en E; este valor se calcula como la suma de todas las acciones de tipo α que están habilitadas en un instante en P.

4 PEPA Workbench

El propósito de PEPA Workbench es verificar si el modelo del sistema ha sido especificado correctamente, y generar el correspondiente proceso Markoviano, obteniendo la matriz generadora. Un ejemplo de esta esquema puede verse en la figura 4.

El modelo de la figura tiene dos componentes independientes que se ejecutan en paralelo sin ninguna cooperación. Cada componente se inicia en el estado P_1 y alcanza dos estados, razón por la que la matriz generadora del proceso Markoviano correspondiente tiene dimensión cuatro.

PEPA Workbench se invoca mediante la orden `pwb`. Las operaciones que se realizan a partir de ese instante son las siguientes:

- Solicita el nombre del fichero que contiene la especificación del sistema. Este fichero tiene la extensión `.pepa`
- Se realiza una análisis sintáctico y semántico de la especificación.

Prefijo $\dfrac{}{(\alpha.r).E \stackrel{(\alpha.r)}{\longrightarrow} E)}$

Cooperación

$$\frac{E \stackrel{(\alpha.r)}{\longrightarrow} E'}{E \underset{L}{\bowtie} F \stackrel{(\alpha.r)}{\longrightarrow} E' \underset{L}{\bowtie} F} \ (\alpha \notin L) \qquad \frac{F \stackrel{(\alpha.r)}{\longrightarrow} F'}{E \underset{L}{\bowtie} F \stackrel{(\alpha.r)}{\longrightarrow} E \underset{L}{\bowtie} F'} \ (\alpha \notin L)$$

$$\frac{E \stackrel{(\alpha.r_1)}{\longrightarrow} E' \quad F \stackrel{(\alpha.r_2)}{\longrightarrow} F'}{E \underset{L}{\bowtie} F \stackrel{(\alpha.R)}{\longrightarrow} E' \underset{L}{\bowtie} F'} \ (\alpha \in L)$$

donde $R = \dfrac{r_1}{r_\alpha(E)} \dfrac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F))$
y $r_\alpha(E)$ es la velocidad aparente de α en E.

Elección

$$\frac{E \stackrel{(\alpha,r)}{\longrightarrow} E'}{E + F \stackrel{(\alpha,r)}{\longrightarrow} E'} \qquad \frac{F \stackrel{(\alpha,r)}{\longrightarrow} F'}{E + F \stackrel{(\alpha,r)}{\longrightarrow} F'}$$

Ocultamiento

$$\frac{E \stackrel{(\alpha,r)}{\longrightarrow} E'}{E/L \stackrel{(\alpha,r)}{\longrightarrow} E'/L} \ (a \notin L) \qquad \frac{E \stackrel{(\alpha,r)}{\longrightarrow} E'}{E/L \stackrel{(\tau,r)}{\longrightarrow} E'/L} \ (a \in L)$$

Constante $\dfrac{E \stackrel{(\alpha,r)}{\longrightarrow} E'}{A \stackrel{(\alpha,r)}{\longrightarrow} E'} \ (A \stackrel{def}{=} E)$

Tabla1. Semántica operacional de PEPA

$$P_1 \stackrel{def}{=} (run, r_1).P_2 \qquad P_2 \stackrel{def}{=} (stop, r_2).P_1$$

$$P_1 \underset{\emptyset}{\bowtie} P_1$$

$$\Downarrow$$

$$\begin{pmatrix} -2r_1 & r_1 & r_1 & 0 \\ r_2 & -r_1 - r_2 & 0 & r_1 \\ r_2 & 0 & -r_1 - r_2 & r_1 \\ 0 & r_2 & r_2 & -2r_2 \end{pmatrix}$$

Figura4. Esquema de PEPA Workbench

– Se obtiene su *grafo de derivación* tomando como base las reglas dadas en la semántica operacional. Este es un grafo en el que los nodos representan términos del lenguaje y los arcos representan las posibles transiciones.

- A partir del grafo de derivación se genera el proceso Markoviano (su matriz generadora). Cada nodo del grafo de derivación se considera un estado del proceso Markoviano, y la velocidad asociada a una transición entre dos estados es la suma de las velocidades que etiquetan los arcos que unen dichos estados.
- Se genera un fichero `.hash` donde se asocia un identificador a cada elemento de la especificación.
- Se genera un fichero `.table` que contiene un listado de las correspondencias entre los estados del proceso Markoviano y los términos sintácticos de la especificación (a través de sus identificadores).
- Se genera un fichero `.maple` que contiene la matriz generadora del proceso Markoviano en un formato compatible con Maple.

En las figuras 5, 6 y 7 se muestra un ejemplo de sesión, así como los ficheros implicados.

```
C:\pepa> pwb
PEPA Workbench Version 0.72 [22-12-1998]
Filename: ejemplo.pepa
Compiling the model
Generating the derivation graph
The model has 4 states
The model has 8 transitions
Writing the hash table file to ejemplo.hash
Exiting PEPA Workbench.
```

Figura5. Ejemplo de una sesión pwb

```
#P1 = (run, r1).P2;          P1->a,              1   a|a
#P2 = (stop, r2).P1;         P2->d,              2   d|a
P1 <> P1                     r1->c,              3   a|d
                             r2->f,              4   d|d
                             run->b,
                             stop->e
```

Figura6. Ficheros ejemplo.pepa, ejemplo.hash y ejemplo.table

Una vez obtenido el fichero `.maple` , utilizando el paquete de álgebra de Maple podemos resolver el sistema y obtener la probabilidad asociada a cada estado alcanzable en el estado de equilibrio. Estos valores se utilizarán posteriormente para derivar otras medidas acerca de las prestaciones del sistema.

PEPA Workbench viene acompañado de un conjunto de herramientas [1,3] que facilitan los cálculos posteriores a la resolución del sistema. Una de ellas es

```
Q[1,2]:=Q[1,2]+r1:
Q[1,1]:=Q[1,1]-r1: # 1--run,r1->2
Q[1,3]:=Q[1,3]+r1:
Q[1,1]:=Q[1,1]-r1: # 1--run,r1->3
Q[2,1]:=Q[2,1]+r2:
Q[2,2]:=Q[2,2]-r2: # 2--stop,r2->1
Q[2,4]:=Q[2,4]+r1:
Q[2,2]:=Q[2,2]-r1: # 2--run,r1->4
Q[4,3]:=Q[4,3]+r2:
Q[4,4]:=Q[4,4]-r2: # 4--stop,r2->3
Q[4,2]:=Q[4,2]+r2:
Q[4,4]:=Q[4,4]-r2: # 4--stop,r2->2
Q[3,4]:=Q[3,4]+r1:
Q[3,3]:=Q[3,3]-r1: # 3--run,r1->4
Q[3,1]:=Q[3,1]+r2:
Q[3,3]:=Q[3,3]-r2: # 3--stop,r2->1
```

Figura7. Fichero ejemplo.maple

PEPA State Finder que permite investigar el espacio de estados generado por PWB y localizar aquellos estados que cumplen una determinada condición.

5 Ejemplo: Red de PCs

5.1 Descripción del sistema

El problema consiste en determinar el tiempo medio de espera de un paquete de datos que debe ser transmitido desde un PC conectado a una token ring. Para evitar conflictos, en este tipo de redes se utiliza un testigo que pasa de un nodo a otro siguiendo una política de turno rotatorio (*round robin*). Un nodo puede trasmitir únicamente si posee el testigo. Consideraremos un escenario en el que una empresa tiene una red con 4 nodos y quiere ampliarlo a 6. El resultado que se quiere conocer a priori es el aumento en el tiempo de espera hasta el envío de un paquete de datos.

Cada PC puede almacenar sólo un paquete en espera, luego cada vez que posea el testigo habrá 0 o 1 paquete en espera. Cada PC genera paquetes para transmitir según una distribución exponencial de parámetro λ. La duración media de la transmisión de un paquete es d (parámetro $\mu = 1/d$), y el tiempo medio que tarda el testigo en pasar de un nodo a otro es m (parámetro $\omega = 1/m$). Si se genera un segundo paquete mientras se transmite el primero, éste deberá esperar hasta la siguiente vez que posea el testigo.

5.2 Modelo del sistema

El primer paso consiste en elegir las variables aleatorias que utilizaremos para caracterizar los estados del sistema. Considerando en primer lugar los nodos, cada PC puede encontrarse en dos estados:

– el PC no tiene ningún paquete que transmitir; o
– el PC tiene un paquete esperando para ser transmitido.

Hacemos corresponder a estas situaciones los valores 0 y 1 respectivamente.

En cuanto al testigo, podemos representar su estado en función del nodo que lo posee. Según esto el espacio de estados del testigo para N PCs sería $\{1, 2, \ldots, N\}$. Para evitar que un PC pueda transmitir dos paquetes seguidos debemos distinguir entre dos posibilidades: que el testigo esté en un nodo antes de la transmisión (n) y que esté es ese mismo nodo después de haber transmitido el paquete de datos ($n+$). En este caso el espacio de estados del testigo para N PCs es $\{1, 2, \ldots, N, 1+, 2+ \ldots, N+\}$.

El estado del sistema completo estará dado por una $(N+1)$–tupla dónde los primeros N valores indican el estado de cada PC y el último el del testigo. El número de estados en el modelo, con N PCs será $2^N \times (2 \times N) = N \times 2^{N+1}$. Para $N = 4$ obtenemos 128 estados, y para $N = 6$ tenemos 768. En la figura 8 se muestra la especificación del sistema para 4 nodos.

```
#PC10 = (arrive,lambda).PC11 + (walkon2,infty).PC10;
#PC11 = (serve1,infty).PC10;

#PC20 = (arrive,lambda).PC21 + (walkon3,infty).PC20;
#PC21 = (serve2,infty).PC20;

#PC30 = (arrive,lambda).PC31 + (walkon4,infty).PC30;
#PC31 = (serve3,infty).PC30;

#PC40 = (arrive,lambda).PC41 + (walkon1,infty).PC40;
#PC41 = (serve4,infty).PC40;

#S1 = (walkon2,omega).S2 + (serve1,mu).(walk2,omega).S2;
#S2 = (walkon3,omega).S3 + (serve2,mu).(walk3,omega).S3;
#S3 = (walkon4,omega).S4 + (serve3,mu).(walk4,omega).S4;
#S4 = (walkon1,omega).S1 + (serve4,mu).(walk1,omega).S1;

(PC10 <> PC20 <> PC30 <> PC40)
    <walkon1,walkon2,walkon3,walkon4, serve1,serve2,serve3,serve4> S1
```

Figura8. Modelo PEPA de una red con 4 PCs

Cada PC tiene inicialmente dos opciones: llega un nuevo paquete a trasmitir (tipo de acción *arrive*) o cede el testigo al siguiente PC (tipos de acción $walkonX$). En caso de que haya llegado un paquete, pasa a un estado en el que espera a tener el testigo para trasmitirlo (tipo de acción $serveX$). Por otra parte, se considera un componente que modela el comportamiento del testigo ($S1$); éste, o bien permitirá al PC que posea el testigo trasmitir o bien pasará el testigo al siguiente nodo. Finalmente, la especificación de todo el sistema consiste en la cooperación de los cuatro PCs y el testigo, con sincronización en todas las acciones.

5.3 Resultados

Como todos los nodos son idénticos, podemos elegir uno de ellos como representante, por ejemplo, PC_1. El tiempo de espera de un paquete es el tiempo medio que PC_1 pasa en el estado PC11 (tiempo que transcurre desde que llega el paquete hasta que finaliza su trasmisión) menos el tiempo medio de trasmisión $(1/\mu)$.

El tiempo medio de permanencia en el estado PC11 puede calcularse como un cociente de los siguientes valores:

– en el numerador, la suma de las probabilidades de todos los estados que tienen un 1 en la primera posición de la tupla (el PC_1 quiere trasmitir)
– en el denominador, la suma de las probabilidades de todos los estados en los que PC_1 quiere trasmitir y además posee el testigo, es decir, los estados de la forma $(1, \ldots, 1)$, multiplicado por la velocidad a la que ocurre la trasmisión, μ.

Para el caso de una red con 4 nodos y con los siguientes valores para los parámetros (en milisegundos)

$$\lambda = 0.01 \qquad d = 10 \qquad \mu = 0.1 \qquad m = 1.0 \qquad \omega = 1.0$$

obtenemos el siguiente resultado:

$$\text{tiempo medio de espera, } W = \frac{0.1333}{0.008666} - \frac{1}{0.1} = 5.3862 \text{ milisegundos}$$

Podemos ampliar la especificación considerando 6 nodos. En ese caso obtenemos el siguiente resultado:

$$\text{tiempo medio de espera, } W = \frac{0.1719}{0.00828} - \frac{1}{0.1} = 10.7678 \text{ milisegundos}$$

Por tanto, el tiempo medio de espera de los paquetes de datos es prácticamente el doble cuando añadimos 2 nuevos PCs a la red.

Referencias

1. G. Clark, S. Gilmore, and J. Hillston. The PEPA performance modelling tools. In *Proceedings of the 7th Annual Workshop on Process Algebra and Performance Modelling*, 1999.
2. J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.
3. J. Hillston and S. Gilmore. PEPA: Performance Evaluation Process Algebra. http://www.dcs.ed.ac.uk/pepa/.
4. J. Hillston and M. Ribaudo. Stochastic process algebras: a new approach to performance modeling. In K. Bagchi and G. Zobrist, editors, *Modeling and Simulation of Advanced Computer Systems*, 1998.
5. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.