AMBIENTE PARA LA ESPECIFICACION DE SISTEMAS DE TIEMPO REAL CON REDES DE PETRI

Marcelo Naiouf
Armando De Giusti

Laboratorio de Investigación y Desarrollo en Informática. (L.I.D.I.)
Departamento de Informática. Facultad de Ciencias Exactas.
Universidad Nacional de La Plata. (U.N.L.P.)
50 y 115 1er. Piso, Tel. (021) 21-4633 - (1900) La Plata - Argentina

RESUMEN

Se presenta un ambiente para la especificación y evaluación de sistemas de tiempo real utilizando Redes de Petri extendidas.

Se discuten brevemente diferentes técnicas de especificación de sistemas, y se analizan las ventajas y desventajas del empleo de Redes de Petri extendidas.

Posteriormente se expone la concepción de un ambiente interactivo sobre microcomputadora que permite definir y evaluar sistemas utilizando Redes de Petri a fin de controlar la variable tiempo.

Asimismo, se detalla el empleo de este ambiente en la especificación y desarrollo de un protocolo de comunicaciones, poniendo énfasis en las facilidades propias del ambiente.

Por último se analiza la posibilidad de derivar código ejecutable directamente de la especificación con Petri.

ABSTRACT

An environment for specification and evaluation of real time systems using extended Petri Nets is presented.

Differents system specification technics are briefly discussed, and the advantages and disadvantages of extended Petri Nets' use are analized.

Subsequently, the conception of an interactive environment on microcomputer that permits define and value systems using Petri Nets to control the "time" variable is exposed.

Likewise, the use of this environment for the specification and development of a communication protocol is detailed, emphasizing the facilities of the environment.

Finally, the posibility to derive executable code directly from the specification with Petri nets is analized.

INTRODUCCION

Los sistemas de tiempo real están caracterizados por la urgencia de su requerimiento de performance.

Una especificación es una sentencia precisa de los requerimientos que un producto o sistema debe satisfacer.

La especificación de requerimientos para un sistema(tanto en hardware como en software) es de gran importancia, por el impacto sobre la calidad y utilidad del producto, y sobre la eficiencia y control del proceso de desarrollo y puesta a punto.

La importancia de una buena especificación es aún mayor en los sistemas de ingeniería de tiempo real, los cuales deben ser altamente confiables, capaces de soportar fallas de hardware, problemas de sincronización y errores en el sistema en si mismo. [2] [7] [9].

Entre las diversas técnicas para la especificación se encuentran:

* Algebraica. Consta de dos partes: la sintaxis (las operaciones del sistema son especificadas indicando el número de argumentos, los tipos de los argumentos y el tipo del resultado), y la semántica (las ecuaciones algebraicas que relacionan los valores creados por las operaciones). [3]

Si bien una especificación algebraica es en muchos casos la mejor expresión formal del sistema modelizado y con ella se puede intentar probar corrección, es de difícil aplicación directa en sistemas complejos de tiempo real, por el grado de abstracción y las dificultades de expresión de los fenómenos asociados con el mundo externo al sistema con la rigurosidad asociada al formalismo matemático.

* Gráfica. En este grupo se puede citar a las máquinas de estado y las redes de Petri. Permiten una rápida visión global del sistema especificado. [6]

Los sistemas gráficos interactivos, a partir de los cuales se puede lograr alguna forma de derivación parcial del software de implementación, y al mismo tiempo una prueba heurística de funcionamiento por simulación directa, han tenido un desarrollo creciente en áreas relacionadas con la ingeniería de sistemas de tiempo real por su proximidad con el lenguaje y las formas de expresión conocidas por el especialista (máquinas de estado, tablas de transiciones) y si bien no cubren los detalles de la implementación son muy adecuados para tener un análisis funcional del sistema global.

* Lenguajes. Entre éstos es importante mencionar entre otros lenguajes el CSP de Hoare, que permite especificar concurrencia, paralelismo y un conjunto de elementos útiles para sistemas de tiempo real. [4]

La ventaja relativa de un lenguaje es la cercanía con la forma de expresión habitual del especialista de software, y la posibilidad de obtener a partir de la especificación una derivación del código a implementar. Sin embargo la complejidad de los sistemas de tiempo real ha mantenido un importante "gap" entre los lenguajes de especificación y aquellos de implementación. De todos modos se trata de una línea de intenso trabajo científico actual.

Más allá de la técnica de especificación que se elija, la posibilidad de contar con un ambiente de trabajo dotado de herramientas auxiliares de especificación y dialogo interactivo, permiten avanzar hacia la **derivación** del software del sistema especificado y también se puede obtener una **simulación** automática, la cual revela detalles del comportamiento del sistema especificado.

LAS REDES DE PETRI COMO HERRAMIENTA DE ESPECIFICACION

Las Redes de Petri son una herramienta útil para la especificación y el estudio de sistemas. El análisis de la red permite revelar importante información acerca de la estructura y el comportamiento del sistema modelizado, y esa información puede usarse para evaluar el sistema y sugerir cambios y mejoras. Las Redes de Petri permiten modelizar sistemas con componentes que interactúan concurrentemente.

Una Red de Petri es un multigrafo, bipartito, dirigido, que tiene dos tipos de nodos: **sitios** y **transiciones**.

Los sitios representan las condiciones o estado del sistema y las transiciones los eventos que pueden darse en el mismo.

Los sitios (simbolizados por círculos) y las transiciones (simbolizadas por barras) se vinculan a través de arcos, conformando así una estructura.

En los sitios pueden residir **tokens**. En dicho caso se dice que la red está marcada. Se llama **marca** a una distribución particular de tokens en la red.

La distribución de los tokens gobierna la ejecución de la red, habilitando o deshabilitando transiciones, y permitiendo, en consecuencia, su disparo o no.

Se dice que una transición está habilitada si cada uno de sus sitios de entrada tiene al menos tantos tokens como arcos existen desde el sitio a la transición. Si una transición está habilitada entonces puede dispararse. Los tokens que habilitan a la transición son sus "tokens habilitantes".

Una transición se dispara removiendo todos sus tokens habilitantes desde sus sitios de entrada, y depositándolos en sus sitios de salida (un token por cada arco que va de la transición al sitio).

La ejecución de la red consiste en una secuencia de cambios de estado producida por el disparo de transiciones.

Más precisamente:

Dada una Red de Petri marcada

$$M = \langle S,u \rangle$$

donde

u es una marca inicial y S es la estructura definida por

$$S = \langle P,T,I,O \rangle$$

con

P = {p1,p2,.....,pn} conjunto de sitios (n)=0)
T = {t1,t2,.....,tm} conjunto de transiciones (m)=0)
I : T --->P' función de entrada
O : T --->P' función de salida
P' conjunto de todas las bolsas posibles formadas con elementos de P,

el espacio de estados R(S,u) de la red, tiene por elementos a todas las marcas alcanzables desde dicha marca inicial u. [5]

Las Redes de Petri fueron diseñadas principalmente para **modelización**. Muchos sistemas, especialmente los que tienen componentes independientes, pueden especificarse por una Red de Petri. Estos sistemas pueden ser de muy distintos tipos: hardware, software, sistemas físicos, sociales, etc. En particular, pueden modelizar el flujo de información u otros recursos dentro de un sistema.

En las Redes de Petri un sistema se representa por dos conceptos primitivos: **eventos** y **condiciones**. Los eventos son acciones que tienen lugar en el sistema. La ocurrencia de estos eventos es controlada por el estado del sistema; dicho estado puede ser descripto como un conjunto de condiciones. Una condición es un predicado o descripción lógica del estado del sistema. Como tal, una condición puede ser verdadera o falsa.

Dado que los eventos son acciones, ellos pueden "ocurrir". Para que ocurra un evento, puede ser necesario que se cumplan ciertas condiciones (**precondiciones** del evento). La ocurrencia del evento puede causar que las precondiciones dejen de cumplirse y que otras condiciones (las **postcondiciones**) se vuelvan verdaderas.

Las Redes de Petri tienen un **paralelismo** o **concurrencia** inherentes. En el modelo de Redes de Petri, dos eventos que están habilitados y no interactúan pueden ocurrir independientemente. No hay necesidad de sincronizar eventos (a menos que sea requerido por el sistema subyacente). Cuando la sincronización es necesaria, es fácil de modelizar. Así, las Redes de Petri son ideales para modelizar sistemas de control distribuido con múltiples procesos ejecutándose concurrentemente en el tiempo.

Otra de las características de las Redes de Petri es su naturaleza **asincrónica**: no hay una medida del tiempo o el flujo del tiempo en una Red de Petri, solo interesa definir un orden parcial de la ocurrencia de eventos.

La ejecución de una Red de Petri se ve como una secuencia de eventos discretos. El orden de ocurrencia de los eventos es uno de los posiblemente muchos permitidos por la estructura básica. Esto lleva a un aparente **no determinismo** en la ejecución de una Red de Petri. Si, en algún momento, más de una transición está habilitada, entonces cualquiera de ellas puede ser "la próxima" a dispararse.

Un modelo de Redes de Petri puede usarse para representar y comunicar el diseño de un sistema concurrente.

Sin embargo, la modelización por sí misma es poco útil. Es necesario **analizar** el sistema modelizado. Este análisis puede llevar a una importante comprensión del comportamiento del mismo.

La mayoría de los problemas están relacionados con uno básico: el **problema de alcanzabilidad**. Es decir: dada una Red de Petri S con marca u, y una marca u', u' $\in$ R(S,u) ?

Por ejemplo, el problema de detectar si se puede producir "deadlock" es posible reducirlo a testear si una determinada marca es alcanzable.

Otros problemas importantes tienen que ver con consideraciones de optimización. Si una Red de Petri exhibe un cierto comportamiento (indicado por sus secuencias de disparo de transiciones y su conjunto de alcanzabilidad), es posible cambiarla (optimizarla) sin afectar su comportamiento? Esto puede involucrar el borrado de transiciones "muertas" (las cuales nunca pueden dispararse) y sitios "muertos" (los cuales nunca pueden ser marcados) o quizás la redefinición de algunas transiciones. La idea es ver si dos Redes de Petri diferentes pueden generar la misma secuencia de disparo de transiciones o el mismo conjunto de alcanzabilidad. Esto permitiría modificar la red para incrementar el paralelismo, decrementar el costo de implementación, u otras optimizaciones.

En estos casos, hay que determinar si dos Redes de Petri son **equivalentes** o si una es un subconjunto de la otra.

Las técnicas de análisis más usadas son:

- árbol de alcanzabilidad
- ecuaciones matriciales

Entre las ventajas de las Redes de Petri se puede mencionar la claridad con que se observa el flujo de control global del sistema, además de la ya mencionada concurrencia implícita.

Una desventaja importante de las Redes de Petri es que no permiten especificar tiempos, lo cual es especialmente crítico en sistemas de tiempo real. Además, los tokens no poseen ningún atributo.

Por todo esto son necesarias **extensiones**, como las que permiten tokens coloreados (con atributos) y otras en las cuales se puede agregar tiempos en la especificación de los sistemas. [8]

LA CONCEPCION DE UN AMBIENTE PARA LA ESPECIFICACION DE SISTEMAS DE TIEMPO REAL

Una síntesis de las capacidades del ambiente para especificación que hemos contemplado es la siguiente:

a) **Creación**: permitir ingresar una Red de Petri en forma gráfica o como un conjunto de sitios y transiciones, con la posibilidad de salvarla en disco.

b) **Borrado**: permitir borrar una Red de Petri existente en disco o en memoria.

c) **Modificación**: permitir agregar o borrar sitios y/o transiciones en una Red de Petri existente.

d) **Ejecución**: permitir la ejecución de la red en forma automática o con interacción del usuario.

Se deben utilizar Redes de Petri extendidas que permitan especificar tiempos.

Para poder llevar a cabo estas funciones es necesario contar con una herramienta gráfica interactiva, que permita crear y visualizar la red.

ASPECTOS DE LA IMPLEMENTACION

Se ha elegido una implementación en Turbo Pascal 4.0, ya que éste permite una natural modularización en unidades y un buen manejo gráfico. Otras soluciones como la utilización de Smalltalk [1] se han descartado por los requerimientos de hardware de soporte y las limitaciones respecto del tamaño de red manejable.-

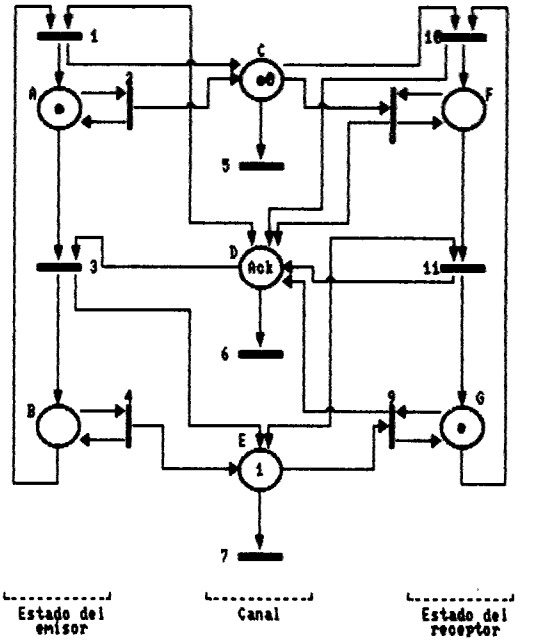
La representación de una red en memoria está dada por listas encadenadas (una para sitios y otra para transiciones) y la información del gráfico almacenada en el segmento de memoria reservado a la pantalla.

Para trasladar una red desde memoria a disco es necesario modificar la representación, la cual está dada por 3 archvos: uno con los sitios, otro con las transiciones y un tercero con las relaciones entre sitios y transiciones.

Los detalles de la representación se encuentran en el Anexo de este artículo.

ANALISIS DE LA UTILIZACION DEL AMBIENTE EN LA ESPECIFICACION DE UN PROTOCOLO

Podemos ejemplificar la utilización del ambiente con la representación con Petri de un protocolo PAR (Positive Acknowledgement with Retransmission) mostrado en la Figura.



1. Emite el 0
2. Timeout
3. Emite el 1
4. Timeout
5. Perdido
6. Perdido
7. Perdido
8. Rechaza el 0
9. Rechaza el 1
10. Procesa el 0
11. Procesa el 1

A. Espera el Aok 0
B. Espera el Aok 1
C. Seq. 0 en la linea
D. Aok en la linea
E. Seq. 1 en la linea
F. Espera el 1
G. Espera el 0

Figura 1 Un modelo de Redes de Petri para un protocolo PAR

A partir del ejemplo es claro que la modelización puede ser mejorada si se cuenta con la posibilidad de indicar el período de "timeout".

Una típica sesión de trabajo con el ambiente incluiría la Creación de la red siguiendo el modelo dado en la figura; una vez creada la red, debería realizarse la Ejecución para detectar posibles errores o mejoras en la especificación, en cuyo caso se utilizaría la función de Modificación.

Una buena alternativa es probar la Ejecución con distintos valores de la variable tiempo para "timeout".

Otra posibilidad es considerar que cada token (en este ejemplo, un paquete) tiene asociado un atributo respecto a si arriba o no arriba. Más aún, se puede pensar que este atributo es la probabilidad de arribo. Con estos elementos podría lograrse una estadística del porcentaje de paquetes que arriban para distintos valores de "timeout".

Obviamente este protocolo tiene una capacidad de buffer igual a 1. Podría intentarse una extensión del sistema físico

llevando dicha capacidad a un número mayor que 1. Una vez modelizado este protocolo podrían intentarse comparaciones con el ejemplo visto anteriormente.

CONCLUSIONES

Se ha presentado un ambiente de especificacion de sistemas de tiempo real,utilizando Redes de Petri extendidas que se ha desarrollado en Turbo Pascal y se remarca la utilidad de este enfoque de especificacion en la especificacion de sistemas de control dedicado de tiempo real.

La linea actual de investigacion es la derivacion directa, a partir de la red de Petri, de codigo ejecutable en lenguaje ADA o MODULA.

ANEXO

a) **Representación de una red en memoria**

(i) Sitios: representados como una lista con nodos del tipo:

```
pun_p    = ^place;
place    = record
nro_p    : byte;      (número de sitio)
nom_p    : string[15]; (nombre del sitio)
cant_tok : word;      (cantidad de tokens en el sitio)
pos_xp   : word;      (coordenada x en la repr. gráfica)
pos_yp   : word;      (coordenada y en la repr. gráfica)
pi_te    : pun;       (puntero inicial a la lista de transiciones de entrada)
pi_ts    : pun;       (puntero inicial a la lista de transiciones de salida)
ps_p     : pun_p      (puntero al siguiente sitio)
end;

pun      = ^elem_lis;
elem_lis =  record
top      : byte;      (indica transición o sitio)
ca       : byte;      (cantidad de arcos)
ps       : pun        (puntero al siguiente elemento)
end;
```

(ii) Transiciones: representadas como una lista con nodos del tipo:

```
pun_t    = ^trans;
trans    = record
nro_t    : byte;          (número de la transición)
nom_t    : string[15];    (nombre de la transición)
t_hab    : integer;       (momento de la habilitación)
t_min    : integer;       (tiempo mínimo de disparo)
t_max    : integer;       (tiempo máximo de disparo)
duración : integer;       (duración del disparo)
pos_xt   : word;          (coordenada x en la repr. gráfica)
pos_yt   : word;          (coordenada y en la repr. gráfica)
pi_pe    : pun;           (puntero inicial a la lista de sitios de entrada)
pi_ps    : pun;           (puntero inicial a la lista de sitios de salida)
ps_t     : pun_t          (puntero a la siguiente transición)
end;
```

(iii) Gráfico: almacenado en el segmento de memoria que corresponde a la pantalla.

b) **Representación de una red en disco**

Una red estará representada en disco por medio de 3 (ó 4) archivos.

(i) Archivo de sitios: con registros del tipo

```
pla_grab = record
nro_p    : byte;      (número del sitio)
nom_p    : string[15]; (nombre del sitio)
cant_tok : word;      (cantidad de tokens en el sitio)
pos_xp   : word;      (coordenada x en la repr. gráfica)
pos_yp   : word       (coordenada y en la la repr. gráfica)
end;
```

(ii) Archivo de transiciones: con registros del tipo

```
tra_grab = record
nro_t    : byte;          (número de la transición)
nom_t    : string[15];    (nombre de la transición)
t_hab    : integer;       (momento de la habilitación)
t_min    : integer;       (tiempo mínimo de disparo)
t_max    : integer;       (tiempo máximo de disparo)
duración : integer;       (duración del disparo)
pos_xt   : word;          (coordenada x en la repr. gráfica)
pos_yt   : word           (coordenada y en la repr. gráfica)
end;
```

(iii) Archivo de enganches: con registros del tipo

```
eng_grab = record
case disc: integer of
       0 : (top : byte;  {nro de tran-
                          sición o sitio}
            ca : byte;); {cantidad de
                          arcos}
       1 : (ast : string[2];){marca de
                          fin de bloque}
end;
```

(iv) Archivo gráfico: contiene la representación gráfica de la red. Este archivo no se genera para todas las redes.

BIBLIOGRAFIA

[1] Amandi, Rossi.
"Utilización del concepto de Programación Orientada a Objetos sobre Redes de Petri". Informe Técnico. UNLP. 1989

[2] Chen, Yeh.
"Formal specification and verification of distributed systems". IEEE Trans. Software Engineering. Nov. 83.

[3] Gehani, McGettrick.
"Software Specification Techiques" Addison Wesley. 1986

[4] Hoare.
"Communicating Sequential Processes". Prentice Hall. 1985

[5] Ludueña, Oyarzabal, Gallard.
"Un sistema de Compilación, Ejecución y Análisis de Redes de Petri Extendidas para Computadoras IBM-PC Compatibles". X14 CLEI y 17a JAIIO. Buenos Aires. 1988

[6] Peterson.
"Petri Net Theory And The Modeling Of Systems". Prentice Hall. 1981

[7] Quirk.(Ed)
"Verification and validation of Real Time Software". Springer Verlag 1985

[8] Yau, Caglayan.
"Distributed Software System Design Representation Using Modified Petri Nets". IEEE Trans. Software Engineering. Vol SE-9. Nro 6. Pp 733-745. 1983

[9] Zave.
"An Operational Approach to Requeriments Specification for Embedded Systems". IEEE Trans. Software Engineering. Vol SE-9 No 3. Mayo 1982

Marcelo Naiouf es docente del Departamento de Informática de la Facultad de Ciencias Exactas de la Univ. Nac. de La Plata (Argentina), donde egresó como Analista de Computación en 1987.-
Actualmente completa los estudios de Licenciatura en Informática y su tema de trabajo en el Laboratorio de Investigación y Desarrollo en Informática (LIDI) se relaciona con Especificación y Diseño de Sistemas de Tiempo Real.-

Armando E. De Giusti es investigador independiente del CONICET en el Departamento de Informática de la Facultad de Ciencias Exactas de la Univ. Nac. de La Plata (Argentina). Es Profesor Titular de Informática. Sus temas de investigación se relacionan con software para sistemas de tiempo real y aplicaciones de automatización industrial y de oficinas.-
De Giusti egresó como Ingeniero Electrónico en la Facultad de Ingeniería de la UNLP y como Calculista Científico en la Facultad de Ciencias Exactas de la UNLP en 1973 y desde entonces se ha dedicado a la docencia e investigación universitaria registrando más de 50 publicaciones científicas. Dirige el Laboratorio de Investigación y Desarrollo en Informática (LIDI).-