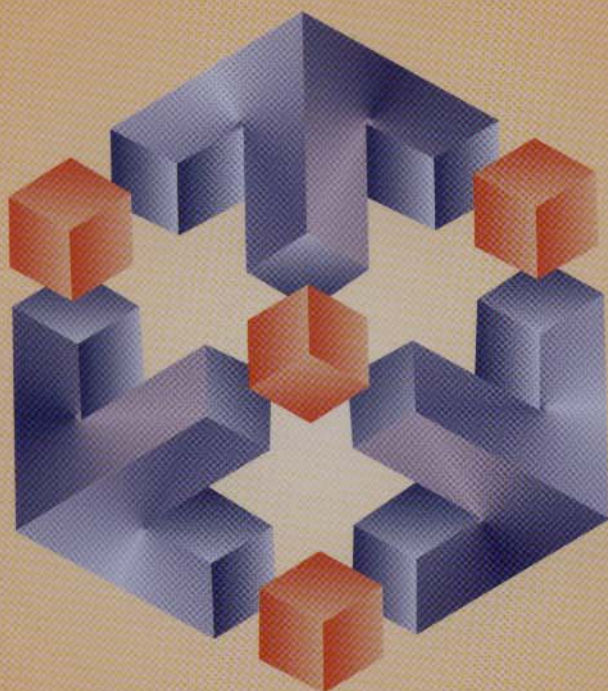


# I JORNADAS DE INFORMÁTICA

Puerto de la Cruz - Tenerife  
17 al 21 de julio de 1995

## ACTAS

Eds. J. M. Troya Linero - C. Rodríguez León



Asociación Española de Informática y Automática  
Universidad de La Laguna  
Universidad de Málaga

# I Jornadas de Informática

*Puerto de la Cruz, 17 - 21 de Julio de 1995*

## Actas

Depósito Legal - MA-777/95

José María Torres López  
Cristina Rodríguez Lallo

## Indice

### Conferencias

- Evolución de las arquitecturas de computadores para altas prestaciones ..... 15  
*Profesor Mateo Valero*  
*CEPBA (Centro Europeo de Paralelismo de Barcelona)*
- A High Level Approach to Real-Time and High-Performance Systems ..... 16  
*P. H. Welch*  
*Professor of Parallel Processing, University of Kent*
- Métodos algebraicos generales para la definición semántica y la verificación de sistemas  
modulares ..... 17  
*Profesor Fernando Orejas*  
*Dpto. de L.S.I. Universitat Politècnica de Catalunya*

### Sesiones

#### Paralelismo I

- Empotrado parcial de cubos y planos sobre toros para solapar cálculos y comunicaciones ... 19  
*A. Suárez, C. N. Ojeda-Guerra*
- Encaminador Asíncrono Wormhole para Redes de Interconexión de Multicomputadores .... 31  
*C. Carrión, F. Vallejo, R. Menéndez, J.A. Gregorio, R. Beivide*
- Implementación de un entorno paralelo orientado al modelo PRAM ..... 41  
*C. León, F. Sande, F. García, C. Rodríguez*
- EDISIS: Un entorno de diseño y Simulación de algoritmos sistólicos y semisistólicos ..... 53  
*F. Fernández, J. Gutiérrez, A. Sánchez*

#### Sistemas de Información y Bases de Datos

- Mecanismos de autorización para los disparadores en los sistemas de bases de datos ..... 63  
*O. Díaz, A. Irastorza, J. Iturrioz*
- Transformación del Modelo Entidad-Relación Extendido (EER) al Modelo Deductivo ..... 75  
*M. A. Dapena, N. Rodríguez*
- Notación para el Comportamiento Reactivo de los Sistemas de Información ..... 85  
*O. Díaz, J. Iturrioz, M. Piattini*

Involucrando al usuario en el desarrollo de sistemas de información .....	101
<i>A. Aguayo, J. Falgueras, A. Guevara, F. Triguero</i>	

CORALFACE/SM: Un interface gráfico para CORAL en arquitectura cliente-servidor usando Smalltalk .....	111
<i>S. Bamonde, R. Catoira, J.M. García-Tizón, V.M. Gullás</i>	

### **Concurrencia**

Construcción de una máquina química abstracta .....	121
<i>C. Herrero, J. Oliver</i>	

Asociatividad en Algebras de Procesos Probabilísticas .....	135
<i>M. Núñez, P. Palao, M. T. Morazán</i>	

Comunicación Monádica de Objetos Funcionales .....	145
<i>J. E. Gallardo, P. Guerrero, B.C. Ruiz</i>	

Corrección de programas y redes de Petri de alto nivel .....	155
<i>F. Tricas, J. Martínez</i>	

Tiempo Lógico en Lenguajes de Tiempo Real .....	165
<i>P.T. Breuer, N. Martínez, A. Marín, L. Sánchez, C. Delgado</i>	

### **Sistemas Informáticos Aplicados I**

Sistema de visión artificial basado en transputers con invariancia del color .....	175
<i>A. Ruiz, I. Llorens, V. Arnau, M. Vicens, J.M. Artigas</i>	

Descripciones declarativas de distinto nivel: aplicación al diseño de invernaderos .....	185
<i>J.R. García, J.F. Bienvenido, J. Sagrado, L.F. Iribarne, A. Becerra, R. Guirado, J.R. Díaz, F. Rodríguez, R.M. Ayala</i>	

Sistema en tiempo real para la representación topográfica de las señales del EEG/EP .....	195
<i>J. J. Merino, J. D. Piñeiro, S. Mañas, J. F. Sigut, L. Moreno</i>	

Una propuesta de invariantes basados en momentos: la transformada de Fourier-Laguerre .....	205
<i>F. Pérez, A. Falcón</i>	

### **Técnicas de Inteligencia Artificial**

Arquitectura para el aprendizaje de clasificadores en un sistema de visión basado en conocimiento. Consideraciones de diseño .....	215
<i>J. Lorenzo, M. Hernández, M. Castrillón</i>	

Una utilización eficiente de búsquedas locales integradas con técnicas globales en un sistema de producción en planta .....	225
<i>J. Arrieta, J.R. Zubizarreta</i>	
Problemas del control óptimo: una aproximación mediante redes neuronales .....	235
<i>A. Hamilton, L. Acosta, N. Marichal, J.A. Méndez, L. Moreno</i>	
Compresión de imágenes mediante redes neuronales con aprendizaje competitivo hebbiano .....	245
<i>M.A. García, F. López</i>	
Sistema basado en el conocimiento para el estudio de señales cerebrales .....	251
<i>J.D. Piñeiro, J.L. Sánchez, S. Mañas, R.M. Aguilar, J.J. Estévez, L. Moreno</i>	
<b><u>Técnicas Algorítmicas</u></b>	
Reflexiones sobre el diseño de Sistemas de Visión Basados en Conocimiento .....	257
<i>J. Cabrera, D. Hernández, A. Falcón, J. Méndez</i>	
Una regla de parada para la Búsqueda con Arranque Múltiple .....	271
<i>J. M. Moreno-Vega, J. A. Moreno</i>	
Una Implementación del Algoritmo de Compresión Lempel-Ziv Welch .....	281
<i>V. G. Ruiz, I. García</i>	
Un algoritmo para medir la seguridad práctica de algunos cifrados en flujo .....	293
<i>P. Caballero, A. Fúster</i>	
Métodos de preselección de píxeles para filtros tipo mediana .....	301
<i>L. García, P. L. Luque, R. Román</i>	
<b><u>Lenguajes y Programación</u></b>	
Síntesis constructiva de Programas Lógicos .....	311
<i>F.J. Galán, M. Toro</i>	
Cremalleras y espirales o cómo co-diseñar mejor .....	323
<i>N. Martínez, A. Marín, S. Deprés, C. Delgado, L. Sánchez</i>	
CASALE I 1.0.B: un lenguaje de simulación discreta basado en objetos .....	333
<i>R. Corchuelo, P. Carrillo</i>	
<b><u>Sistemas Informáticos Aplicados II</u></b>	
Sistema de Interfaz Persona-Computador para Entornos Virtuales .....	343
<i>R. Rodríguez, M. Fernández, M. Pérez, S. Bayarri, F. Martínez</i>	

Control informático global de un sistema ferroviario a escala .....	353
<i>J. M. Vicente, G. Bandera</i>	
Análisis y siseño de tablas de pseudocolor para visualización de imágenes médicas en PC ..	363
<i>V. Arnau, M. Vicens, J.M. Orduña, A. Felipe, A. Palomares</i>	
Programación de un sistema de eculización digital .....	373
<i>J.V. Francés, J. Guerrero, J. Calpe, A. Rosado</i>	

## **Paralelismo II**

Programación de Técnicas Algorítmicas en dos Sistemas Distribuidos: PVM y Transputers .....	381
<i>F. Almeida, F. García, J. Roda, D. Morales, C. Rodríguez</i>	
Implementación de un algoritmo thinning paralelo: El desbalanceo de la carga .....	393
<i>M.D. Gil, J. Roca, I. García</i>	
Paralelización de un algoritmo de validación de protocolos .....	405
<i>F. Rus, P. Merino, M. Díaz</i>	
Programación dinámica paralela para problemas de la mochila 0/1 .....	415
<i>F. Almeida, D. Morales, F. García, C. Rodríguez</i>	

## **Docencia I**

Sobre la enseñanza de las bases de datos: teoría y diseño .....	427
<i>M. Celma, L. Mota, M. A. Pastor, J.C. Casamayor</i>	
Implementación de Tipos Abstractos de Datos sobre Lenguajes Imperativos .....	437
<i>J. M. Molina</i>	
Diseño digital basado en dispositivos lógicos programables: una experiencia docente con ABEL-PLD .....	447
<i>A. Rosado, O. Farpón, M. Bataller, J. Guerrero, J. V. Francés</i>	
Visualización de la ejecución de programas en una CPU segmentada.....	455
<i>A. Suárez, M. Marrero</i>	

## **Docencia II**

Docencia sobre las implicaciones sociales y éticas de la informática .....	465
<i>M. Barceló</i>	
Análisis y valoración de la formación informática en los estudios universitarios .....	475
<i>L. García, L.A. Ureña</i>	

Una Experiencia Docente en Interacción Hombre-Ordenador .....	485
<i>J. Lorés, C. Abarca</i>	
Motivaciones para el trabajo relacionado con empresa del profesorado (las experiencias reales) .....	495
<i>R. Cores, J. Ayude, M. Pérez</i>	
<b><u>Proyectos I</u></b>	
Entorno general de producción automática de prototipos de software orientado a objetos: OOAP (Object-Oriented Assistant Prototyper).....	501
<i>Grupo OASIS</i>	
CCASE - Una herramienta MetaCASE Europea Avanzada Parametrizable .....	515
<i>J.R. Freixanet, J.M. Espejo, J. Canal</i>	
Bases de Datos: Gráficas + Alfanuméricas integradas en la gestión de recursos materiales de la U.P.V.....	531
<i>A. Martínez</i>	
La versión 2 del programa SSD_CIABSI. Un ejemplo real de utilización de medios técnicos en la decisión administrativa .....	545
<i>G. Nistal</i>	
<b><u>Proyectos II</u></b>	
Evaluación de un sistema de ficheros paralelo para máquinas masivamente paralelas.....	555
<i>J. Carretero, F. Pérez, P. de Miguel, F. García, L. Alonso</i>	
PACOTE: Paralelización de un código termohidráulico para la simulación de plantas nucleares .....	567
<i>M. Díaz, L. Llopis, J.L. Pastrana, F. Rus, E. Soler, J.M. Troya</i>	
Proyecto TIMS: Tarjeta Inteligente Multiservicio .....	577
<i>V. Cerverón, V. Cavero, J.V. Pelechano, G. Martín</i>	
DAMOCIA: Diseño asistido mediante ordenador para la construcción de invernaderos automatizados .....	587
<i>J. F. Bienvenido</i>	
<b><u>Proyectos III</u></b>	
EUROWARE, una solución para fomentar la reusabilidad en redes de área extendida .....	597
<i>G. Sánchez</i>	
EVP : Un entorno para la integración de técnicas de descripción formal .....	607
<i>Grupo de Ingeniería del Software. Universidad de Málaga</i>	

I + D en tecnología software. Una experiencia de colaboración Universidad-Empresa ..... 617  
*J.J. Gil, J.C. Yelmo*

Un modelo para comunicaciones multimedia isócronas ..... 627  
*S.P. Labajo, A. Pacheco, E. Rendón, C. Muñoz*

#### **Proyectos IV**

MIX: Un Enfoque Multiagente para Sistemas Híbridos Simbólico-Conexionistas ..... 637  
*J.C. González, J.R. Velasco, C.A. Iglesias, J. Alvarez, A. Escobero*

Proyecto INVAID II : Integración y evaluación de un sistema de D.A.I. basado en visión artificial en áreas urbanas ..... 647  
*J.J. Martínez, R. Ferrís, V. Cavero, J. Martínez, A. Fuertes, G. Martín*

Proyecto ARTIS WA 3 : Integración y evaluación de un sistema de D.A.I. basado en visión artificial en la M-40 (Madrid) ..... 657  
*J.J. Martínez, R. Ferrís, J. Martínez, J. Pelechano, G. Martín*

Descubrimiento de conocimiento para diagnosis de fallos en producción de circuitos integrados ..... 667  
*A. Ribeiro, M.D. del Castillo, L.J. Barrios*

#### **Pósters**

Un planteamiento de las sociedades de objetos como técnica de validación y prototipación de software ..... 675  
*J. Rodeiro, M. Pérez, F.J. Vázquez*

Complejidad y gestión del riesgo en el desarrollo de software orientado a objetos ..... 677  
*E. Barreiro, P. Vázquez*

Planteamientos de seguridad para redes de comunicaciones ..... 679  
*J. Areitio, M. T. Areitio*

Resolución de sistemas dispersos de ecuaciones lineales sobre PVM explotando paralelismo en el método de Gauss-Seidel ..... 681  
*B. Sahelices, A. de Dios, V. Cardeñoso*

Algoritmo para la aproximación de ecuaciones diferenciales con retardos ..... 683  
*C.F. Alastruey, J.R. González de Mendívil*

Computación Universal en Autómatas Celulares ..... 685  
*A. J. Tomeu*

Monitores CRT colorimétricamente controlados ..... 687  
*A. Palomares, A. Lorente, V. Arnau, J.M. Artigas*

Enseñanza Asistida por Ordenador del Lenguaje Ensamblador MC68000 .....	689
<i>F. Quintana, L. Herrera</i>	
La enseñanza de la Ingeniería del Software según el nuevo plan de estudios en la Universidad Jaime I .....	691
<i>C. Campos, O. Coltell</i>	
Técnicas de Multimedia para el autoaprendizaje de conceptos básicos de informática .....	693
<i>O. Coltell, P. J. Sanz, P. Valero, F. Ester</i>	
Multimedia .....	695
<i>A. Rodríguez, J.F. Gálvez, M. Pérez</i>	
El procesado digital de la señal en la titulación de la ingeniería electrónica de la Universitat de Valencia .....	697
<i>J. Calpe, J.F. Guerrero, E. Soria, C. Hernández, J. Espf</i>	
La enseñanza de la electrónica digital en las ingenierías .....	699
<i>V. Arnau, M. Vicens, J.M. Orduña</i>	
Descripción de la red de interconexión SCI (Scalable Coherent Interface) IEEE P1596 .....	701
<i>V. Arnau, V. González, E. Sanchls, J. Ferrer, J.M. López, F. Mora, A. Sebastiá</i>	
MEDIDA: Hacia un modelo de medición .....	703
<i>A. Bernaras, I. Ortiz</i>	
Modelado del Análisis de la Replicación del ADN por Electroforesis Bidimensional en Geles de Agarosa .....	705
<i>A. Rodríguez, E. Viguera, P. Hernández, J.B. Schwartzman, O. Trelles</i>	
Verificación de Conexiones Telefónicas con ESTEREL .....	707
<i>J. Graña, M. Vilares, R. Bernhard</i>	
CASE DFDC. Una herramienta cooperativa de diagramas de flujo de datos .....	709
<i>A. Guevara, A. Aguayo, J. Falgueras, F. Triguero</i>	
Interface QBE para consultas a Bases de Datos Deductivas .....	711
<i>M.A. Pardavila, N. Rodríguez</i>	
Desarrollo de una aplicación para la Diputación Provincial de La Coruña .....	713
<i>J.A. Martínez, A. Rico, J.M. García-Tizón, N. Rodríguez, M. Rodríguez-Losada</i>	
Implementación de un sistema gestor de bases de datos orientadas a objetos .....	715
<i>J. L. Berrocal, G. Rodríguez, J. F. Aldana</i>	

# PARALELIZACIÓN DE UN ALGORITMO DE VALIDACIÓN DE PROTOCOLOS

F. Rus, P. Merino, M. Díaz

Depto. Lenguajes y Ciencias de la Computación  
Fac. Informática-ETSI Telecomunicación  
Universidad de Málaga

## RESUMEN

*SPIN es una herramienta de simulación y validación de protocolos especificados en lenguaje PROMELA, que permite generar un validador específico para cada protocolo en concreto y que está considerado entre los validadores más rápidos actuales [HOLZ91]. El problema que presenta es la imposibilidad de validar un protocolo de mediano o gran tamaño de forma exhaustiva, y por tanto segura, ya que agota la memoria de una máquina intentando almacenar todos los vectores de estados que se van generando. Como respuesta a esta necesidad, la herramienta implementa un método de compactación, de forma que no se almacenan los vectores de estados completos [HOLZ92b].*

*Como una mejora, surge la paralelización de este código, de forma que se aumente la memoria disponible para almacenar los estados analizados, evitando, en la medida de lo posible, el uso de la compactación. Otro de los objetivos es mejorar el algoritmo de búsqueda primero en profundidad.*

*El programa paralelo hará uso de TCP/IP sobre una red de estaciones de trabajo, siendo el usuario el que decidirá el número de procesadores que se utilizarán en cada validación. Se consigue un validador que funciona sobre una máquina virtual de tamaño variable, y que es capaz de analizar protocolos de mayor tamaño.*

## 1 INTRODUCCIÓN

El software para sistemas distribuidos suele ser muy complejo y no puede permitir ningún tipo de error, por lo cual es necesario tener la seguridad de que su funcionamiento es correcto. Para conseguirlo debemos tener unas especificaciones que sean claras, concisas y nunca ambiguas, cosa que sólo podremos conseguir con la utilización de lenguajes formales de especificación [TURN93].

Se suelen usar técnicas de descripción formales (TDFs) para la especificación de los protocolos que se desean validar. Éstas, provienen del trabajo con lenguajes de especificación formal y métodos rigurosos para el desarrollo de sistemas de computación [BUDE87] [EUVI89]. Las TDFs están especialmente indicadas para sistemas concurrentes, distribuidos, en tiempo real, de calidad y seguridad crítica, etc [SICH90].

Los formalismos empleados se pueden clasificar en modelos de transiciones, de programación e híbridos. Como ejemplo de los primeros están las máquinas de estados finitos

(FSM) y las redes de Petri; CCS, CSP y LOTOS pueden servir de ejemplo como modelos de programación. Por último STELLE y SDL como modelos híbridos.

Las herramientas más usadas para el desarrollo de software distribuido pueden ser de simulación, de validación, implementación automática y generación de test. Las primeras permiten analizar el comportamiento del protocolo sin necesidad de realizar una implementación. Las de validación comprueban que determinadas propiedades se cumplen en todos los estados posibles. Mediante el uso de la validación se asegura que la especificación se comporta tal y como pretendíamos especificar. La implementación automática permite generar un programa ejecutable a partir de una especificación. Por último, la generación de test, una vez realizada la implementación, permite tener cierta seguridad (no total) de que la aplicación funciona como se deseaba.

La detección de diversos errores en la especificación del protocolo TCP recogida en el estándar MIL-STD-1778, muestra la necesidad de unas herramientas para una buena implementación. El análisis realizado en [SIDBL85] pone de manifiesto que las implementaciones del protocolo que se realicen a partir de este estándar presentan errores.

La validación de protocolos se entiende como una necesidad si pensamos que es en el momento de la codificación y diseño de un protocolo cuando afloran los fallos de diseño [WEZA78]. Si se valida el diseño de alto nivel se tendrá la certeza de no encontrar un error lógico más adelante, que tendría resultados muy costosos, tanto en capital como en tiempo, en caso de que se haya implementado con ese error, ya que habría que volverlo a diseñar y pasar por las restantes fases hasta una nueva implementación.

Las técnicas de validación más utilizadas se basan en análisis de alcanzabilidad de estados, que consiste en generar y analizar determinados estados del sistema especificado.

Para una validación completa hay que realizar, en principio, un análisis exhaustivo de todos los estados posibles; pero realizar esta tarea de forma manual solo es posible para protocolos pequeños, para el resto será necesario utilizar herramientas computacionales. Esto se debe a que el número de estados a analizar suele ser excesivamente grande conforme aumenta un poco la complejidad del protocolo.

El problema de la explosión de estados se puede paliar en parte usando un método de exploración por el árbol de estados que implemente búsqueda parcial, primero en profundidad, en amplitud, aleatorias, etc. Estos métodos no reducen el número real de estados a explorar en el modelo, aunque sí pueden proporcionar diferentes calidades de análisis en protocolos donde la exploración no puede ser completa.

Otro campo en auge es la aplicación de transformaciones a la especificación original para conseguir un modelo más pequeño del protocolo, al que se aplica un análisis exhaustivo [QUEAZ93]. Es de vital importancia en estos métodos poder garantizar que el protocolo original y el reducido son equivalentes desde el punto de vista de las propiedades a analizar. La mayor parte de las herramientas que lo implementan están basadas en LOTOS [EIJVI89].

Una forma de mejorar el rendimiento de una validación es usar un método de compactación de estados, que no almacena todo el vector de estados, y así poder analizar más estados antes de agotar la memoria disponible.

La solución obtenida con la paralelización de la herramienta SPIN, presenta una serie de ventajas, como son:

- Obtención de un mayor espacio de estados, ya que se crea una máquina virtual con la memoria aportada por la totalidad de todas las máquinas que la componen. De esta forma se consigue validar un protocolo más complejo que de forma secuencial.
- Uso de una búsqueda parcial, con lo que en determinados casos se pueden conseguir mejores resultados que con búsqueda primero en profundidad.

- Se mantiene la posibilidad de usar compactación, que ya contenía la herramienta original, y ahora funciona en paralelo, aumentando de esta forma el tamaño de la tabla hash necesaria, y evitando en gran medida los conflictos.

En el apartado 2 se verá una breve descripción de la herramienta SPIN, donde se tratará su funcionamiento general, así como las opciones que implementa para validación. Seguidamente, en el apartado 3, se describirá el modelo de paralelización utilizado, la estructura de las comunicaciones entre procesos y todos los algoritmos adicionales usados. Para continuar, en el apartado 4, se presenta un análisis de los resultados obtenidos, tanto usando una exploración exhaustiva como un método de compactación, comparándolos con los de la herramienta original. Para finalizar se comentan las conclusiones más relevantes que se han obtenido.

## 2 LENGUAJE PROMELA Y SPIN

SPIN es una herramienta que permite la validación de protocolos, especificados en el lenguaje PROMELA.

Un modelo de validación debe definir las interacciones entre procesos que funcionan en sistemas distribuidos, abstrayendo detalles de bajo nivel, propios de la implementación, centrándose en el diseño de un sistema de reglas para que la interconexión sea consistente.

Los validadores producidos por la herramienta SPIN están entre los más rápidos para búsqueda exhaustiva conocidos hasta la fecha. Estos pueden trabajar de dos formas, haciendo uso de un espacio de estados exhaustivo o usando el método de compactación llamado "bit-state" o supertraza. El primer modo es el ideal, ya que analiza todos los estados posibles, no dejando ninguna posibilidad sin comprobar; el problema que tiene es que sólo se podrán validar protocolos de pequeño y mediano tamaño. El segundo modo almacena un solo bit, en vez del vector de estados completo, haciendo uso de una función hash, que a partir de un vector de estados devuelve un índice de una tabla hash, donde solo se almacena un bit (estado analizado o no). De esta forma el consumo de memoria y tiempo es menor; esto tiene la ventaja de conseguir un espacio de estados mucho menor, con lo cual se pueden validar protocolos de mayor tamaño, pero con la desventaja de que no se puede asegurar que se haya hecho una búsqueda completa [HOLZ91].

Ambos métodos usan un sistema de búsqueda primero en profundidad, que tiene la ventaja de no tener un amplio conjunto de estados explorables en cada momento.

El esquema del programa es el que se muestra en la figura 1, y su funcionamiento es el siguiente:

El inicio lo realiza la rutina *run()*, la cual obtiene memoria y prepara todas las estructuras que el validador usará durante la búsqueda. A continuación se llama al procedimiento *new\_state()*, para realizar todo el trabajo. Éste implementa la búsqueda primero en profundidad de todos los estados ejecutables del modelo; *new\_state()* se llama una sola vez, y no retorna hasta que se hayan analizado todos los estados o se produzca un error. Usa dos estructuras de datos fundamentales, que son el espacio de estados y una matriz de transiciones de gran tamaño, que contiene el modelo de validación PROMELA completo. Cada estado en el modelo produce una entrada en esta matriz, definiendo el efecto de la ejecución.

La matriz de transiciones juega un papel fundamental en la exploración, ya que define las acciones a realizar cuando se ejecuta una transición. Por esto hay que evitar que contenga algún movimiento o transición no estable.

El estado actual del sistema se mantiene en un vector que puede aumentar y disminuir de tamaño dinámicamente. Antes de continuar el análisis de un nuevo estado, se usa una función hash que decide si el estado analizado anteriormente puede ser eliminado, por estar repetido, o debe ser almacenado.

Si se produce un error, el programa llama al procedimiento *uerror()*, para producir una ruta de error que el simulador SPIN puede seguir.

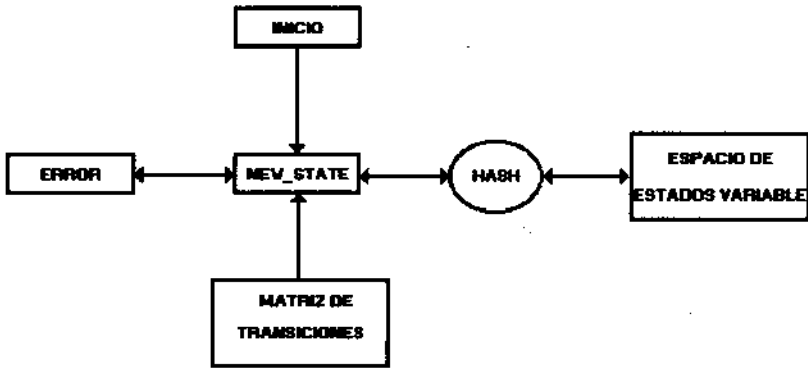


Fig.1 - Esquema de la herramienta SPIN

### 3 PARALELIZACIÓN

El esquema general de la ejecución paralela se muestra en la figura 2, donde se aprecia que hay varios procesos funcionando con un sistema de comunicación punto a punto, de forma que todos los procesos exploran estados y pueden intercambiar información entre ellos.

Existen distintas formas de distribuir el validador, dependiendo de la estructura de comunicaciones que presenten los procesos implicados, como por ejemplo en anillo, punto a punto, etc. El esquema que se ha elegido tiene enlaces punto a punto, usando un protocolo orientado a conexión, consiguiendo de esta forma una mayor velocidad, lo cual es necesario por el volumen de comunicaciones entre procesos.

El proceso principal difiere un poco de los otros, ya que, además de realizar el trabajo de validación y comunicación como hacen todos, es el encargado de la creación del resto de procesos en otras máquinas y de verificar la finalización del programa.

Cada proceso tiene un identificador asignado, empezando por el cero, que pertenece al proceso principal. Cada proceso conoce el número de máquinas que participan en el trabajo, dato que le da el usuario en tiempo de compilación.

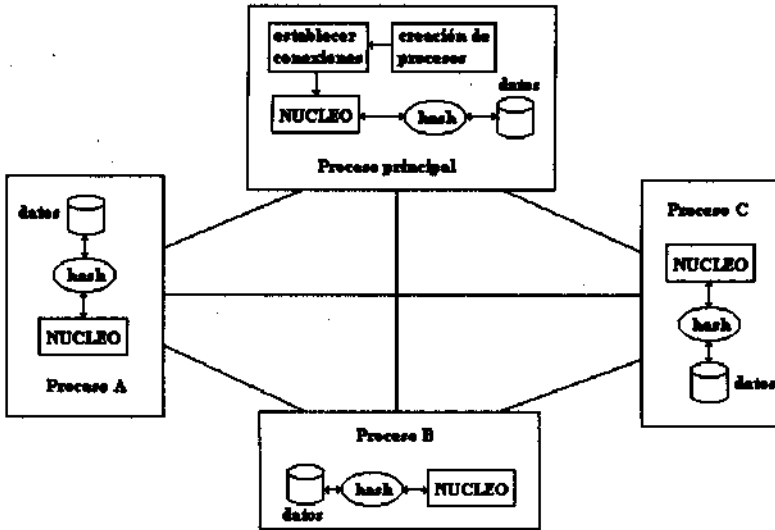


Fig.2 - Esquema de ejecución paralela

El número de procesadores que se usarán para validar el protocolo es variable, y el usuario puede decidir cuántos va a usar, en función de la red disponible, y de las necesidades de memoria.

El funcionamiento del programa paralelo completo se encuentra esquematizado en la figura 3, donde los rectángulos con línea discontinua representan las rutinas importantes desde el punto de vista de las comunicaciones, que son dos, *new\_state()* que se encarga de todo el proceso de la validación, y *hstore()*, que lleva el peso del almacenamiento y comprobación de los estados analizados. Los cilindros con una flecha los nodos de comunicaciones con otros procesos. En la figura 3 se aprecia que hay tres zonas de comunicaciones, los bloques 1, 3 y 6, cada una en una rutina:

- La primera es la encargada de conseguir un vector de estados cuando el proceso en cuestión no tiene ninguno. No se puede entrar en la validación (*new\_state()*) hasta que no se tenga un vector de estados a partir del cual se pueda explorar. Cuando un proceso ha explorado todos los estados posibles a partir de su vector de estados inicial, vuelve a la zona 1 para obtener un nuevo vector inicial.
- Si un proceso, al generar un nuevo estado, comprueba que pertenece a otro procesador, usando el bloque 6, lo envía al que corresponde, que lo comparará y almacenará, si fuera necesario, devolviendo la respuesta al proceso origen.

Una vez que el proceso principal ha creado al resto, todos han de preparar las conexiones para permitir las futuras comunicaciones, de la forma siguiente:

- A fin de crear unas conexiones punto a punto, cada proceso hará una conexión pasiva con los que tengan un número mayor que él, y de forma activa con los que lo tengan menor.

- Cada proceso intenta hacer una conexión activa con los procesos que tengan un número menor que el suyo, comenzando por el cero y acabando con el número propio menos uno.
- Se hacen las conexiones pasivas con los restantes. De este modo se evita que algunos procesos se queden bloqueados esperando la conexión, mientras que el resto ve frustrado su intento de hacerla.

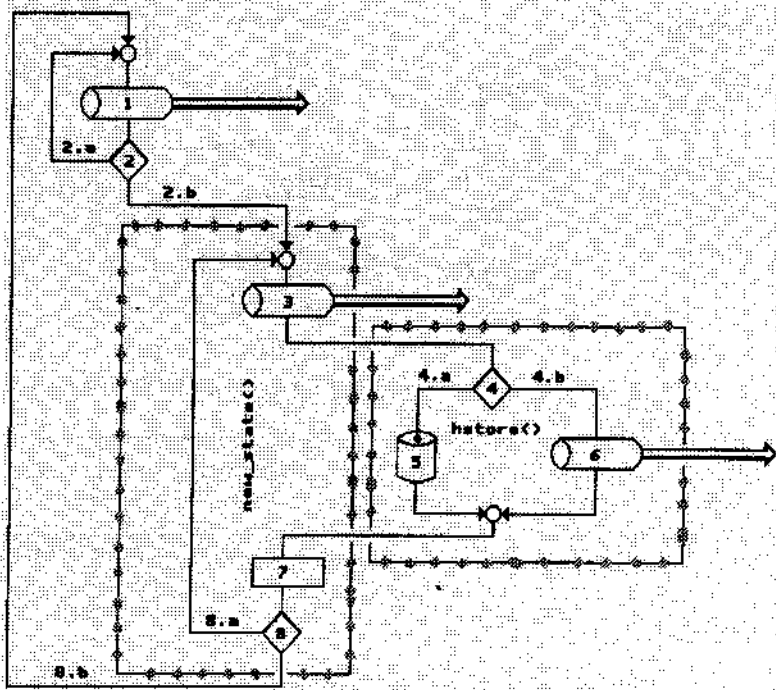


Fig.3 - Algoritmo del programa paralelo

La elección del puerto que debe especificar el proceso que quiere hacer la conexión activa se hace de forma automática e independiente del número de procesos que haya.

El programa tiene dos formas posibles de finalizar, la primera es cuando se ha producido un error, y la segunda cuando la validación se ha completado satisfactoriamente:

- Si se ha producido un error, el proceso que lo ha detectado avisa a los otros de esta circunstancia, los cuales paran su trabajo y ayudan al principal a elaborar la ruta de ejecución, para que pueda ser usada por el simulador de SPIN.
- Si la validación se ha completado satisfactoriamente, todos los procesos estarán ociosos, por lo que el principal decidirá que el programa ha terminado completamente, presentando al usuario los resultados. Para decidir cuando el programa ha terminado, se usa un algoritmo de finalización basado en el de Misra [MISR82] [RAYN88].

El mantenimiento de la información se puede hacer de varias formas, un gran bloque de memoria de datos centralizada, al que acceden todos los procesos para intercambiar sus datos, y la otra posibilidad es tener los datos distribuidos, o sea, cada máquina tiene su proceso y una parte de los datos, con la cual intercambiarán información los otros procesos. Esta última opción tiene como principal inconveniente que hay que mantener los datos coherentes, aumentando el volumen de las comunicaciones.

La primera opción hay que desecharla porque no conseguiríamos un aumento de la memoria en la máquina virtual, no pudiendo analizar más que protocolos pequeños. Se necesita pues aumentar la memoria total, debiendo usar un esquema de memoria distribuida. En vista de lo anterior se ha preferido usar varios procesos, cada cual con una zona de memoria en la máquina correspondiente, manteniendo la coherencia de datos con los otros procesos. La consistencia de datos se mantiene de la siguiente forma:

- La zona de almacenamiento de estados, a la que se accede a través de una función hash, está repartida entre los procesos que participan en la búsqueda. De esta forma se pretende que la distribución de los estados entre los procesos sea equitativa, ya que la función hash debe ser equitativa a la hora de asignar a cada estado su lugar de almacenamiento.
- Cada proceso, antes de almacenar un estado comprueba si el índice que devuelve la función hash pertenece a su rango, en cuyo caso realiza el almacenamiento; en caso contrario ese estado se envía al proceso correspondiente para que éste lo almacene. De esta forma cada estado solo se almacena en una máquina, no teniendo problemas de inconsistencia.

#### 4 ANÁLISIS DE RESULTADOS

Se han realizado pruebas del funcionamiento del programa con varios protocolos; para analizar la ganancia en estados que se produce se ha usado el protocolo de FTP que aparece en [HOLZ91]. La elección de dicho protocolo viene motivada por la necesidad de generar una cantidad de estados tal que permita agotar la memoria de varias máquinas, y así obtener datos comparativos en cuanto a estados generados.

Los datos obtenidos en modo exhaustivo son los siguientes:

	STORED	LINKED	MATCHED	TOTAL
SPIN	71.136	100.328	141.279	312.743
1M	78.362	115.073	161.192	354.627
2M	158.326	215.611	249.850	623.787
3M	199.273	298.907	386.865	805.045
4M	245.862	383.247	507.357	1.136.466

El significado de la tabla anterior es:

- La columna encabezada por STORED nos dice el número de estados explorados por el programa.
- La columna LINKED, cuenta los estados que fueron encontrados en una secuencia atómica.

- Bajo la cabecera MATCHED encontramos el número de estados que fueron analizados y después revisados. Por último aparece la columna TOTAL, que no es más que la suma de cada una de las filas.
- La primera zona de columnas corresponde al programa SPIN original, mientras que las siguientes corresponden al programa paralelo funcionando con una, dos, tres y cuatro máquinas respectivamente.

En teoría, la primera y la segunda filas de la tabla anterior (zonas de columnas en la gráfica) deberían coincidir aproximadamente, ya que se analizarán estados hasta completar la memoria de una máquina, y como las pruebas se realizaron sobre la misma máquina, ha de obtenerse un resultado parecido. Sin embargo hay una diferencia, que se debe achacar a la carga de la máquina en cada momento, o sea, en un momento puede estar ejecutando varios procesos remotos, por petición de los usuarios de otras máquinas.

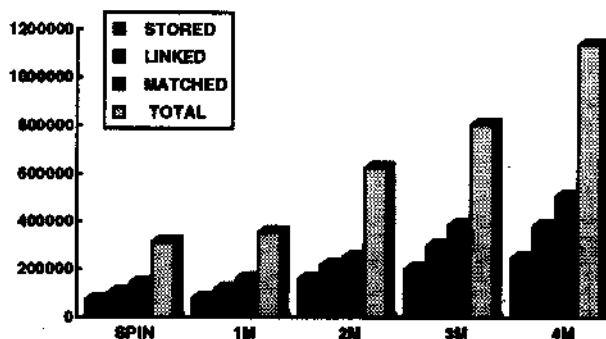


Fig.4 - Estadísticas de estados analizados

La segunda fila no debe ser el doble que la primera, aunque las dos máquinas que ejecutan los procesos tengan la misma cantidad de memoria, debe ser menor que el doble, pero mayor que la primera fila. Esto es así porque durante la ejecución del programa un proceso suele almacenar más estados que el otro, debido a varios factores:

- Uno de los procesos nunca tiene que pedir un vector de estado nuevo, mientras que el otro ha de pedirlo varias veces. Esto hace que el segundo proceso esté algún tiempo parado mientras obtiene los vectores de estado, y el primero agote su memoria antes.
- De los estados que exploran entre los dos hay una mayor parte que debe almacenarse en una de las máquinas. Con esto tenemos que la memoria de una máquina se llenará antes y acabará el programa, debido al equilibrado de carga. Esto debe afectar en poca medida, ya que una función hash debe repartir de forma más o menos equitativa los índices que ofrece.
- Uno de los procesos revisa más estados, por lo que ocupa menos memoria, ya que éstos no tienen que ser almacenados. Por lo tanto, el otro agotará antes la memoria y tendrá más estados almacenados (primera columna de la tabla anterior).

No es importante el porcentaje en sí, ya que éste varía totalmente con el protocolo que se intente validar, pero sí es interesante ver la ganancia de estados que se produce conforme se aumenta el número de procesos.

Los resultados obtenidos en modo supertraza son los siguientes:

	STORED	LINKED	MATCHED
SPIN	2260720	4509366	7016118
1M	1905735	4091138	6482460
2M	3376391	6522690	10483264
3M	4502735	8509441	13906831
4M	5349553	10783824	156195618

El único dato comparable en este caso es el número de estados, ya que los sinónimos debidos a la función hash nos llevarían a resultados engañosos, y muy dependientes del estado de la máquina en el momento de la prueba, porque, si se consigue llenar la memoria del ordenador, el número de conflictos hash será exagerado, y los resultados serán pésimos, tanto con el programa original como con el paralelo. De todas formas, se obtiene un factor de hash mejor con el programa paralelo que con el original, ya que, si el número de máquinas es suficiente, sólo un proceso llega a agotar la memoria y tener un factor parecido al programa original, pero los otros, al haber analizado menos estados, tendrán un factor de hash mejor.

Además habrá una cantidad desproporcionada de estados que se han considerado repetidos, ya que si tienen iguales los índices de hash, se supone que los estados son iguales. Cuantos más estados se vaya analizando, más conflictos habrá, por lo que al final muchos estados serán considerados como revisitados. Esto falsea el total porque en condiciones normales no hay que llenar la memoria. De todas formas, si el protocolo es muy grande, la única manera de analizarlo es usando el programa paralelo, ya que tiene mucho más espacio de memoria.

## 5 CONCLUSIONES

Como se ha visto anteriormente, la paralelización del programa responde a la necesidad clara de poder validar protocolos reales, que con el programa original era difícil si no se disponen de grandes cantidades de memoria principal. Gracias al programa paralelo se consiguen mejores resultados de la validación, haciendo uso de los medios de que habitualmente se dispone: una red de estaciones de trabajo con TCP/IP.

Se ha implementado el programa de dos formas, la primera haciendo que en cada máquina haya un proceso que realiza la exploración y un espacio de almacenamiento de estados, y la segunda haciendo que unas máquinas ejecuten procesos validadores y otras almacenen los estados. Esta segunda opción consume más tiempo, ya que el número de comunicaciones aumenta considerablemente.

Además de obtener una máquina virtual con más memoria disponible, consiguiendo una gran ganancia de estados, se implementa una búsqueda parcial, que suele dar mejores resultados que la exploración primero en profundidad que implementa el programa original.

El programa ha mantenido la opción de compactación que implementaba el original, por si la red no contiene la suficiente memoria para hacer una validación exhaustiva.

Se han añadido 25 rutinas conteniendo 2.400 líneas de código C, al programa original, además de modificar las rutinas de inicio, núcleo y presentación de los resultados que ya contenía el programa original.

Es importante destacar que las ventajas de la paralelización de este programa no sólo las pueden aprovechar los usuarios del lenguaje PROMELA, sino que existe una herramienta que convierte de código SDL a éste [HOPA90].

## 6 BIBLIOGRAFÍA

- [BUDE87] Budkowski, S. y Dembinski, P. "An introduction to Stelle: A specification language for distributed systems". 1987
- [CHU89] Chu, P. M. "Towards automating protocol synthesis and analysis". Ohio State University, Dept. of Computer and Information Science. 1989
- [ELVI89] Van Eijk, P., Vissers, C.A. y Diaz, M. "The formal description Lotos". 1989
- [HOLZ91] Holzmann, G.J. "Design and validation of computer protocols". Prentice Hall Software Series. 1991
- [HOLZ92b] Holzmann, G.J. "Protocol Design: Redefining the State of Art". IEEE Software. Enero 1992
- [HOPA90] Holzmann, G.J. "Validating SDL specification: an experiment". Protocol Specification, Testing and Verification, IX. Elsevier Science. 1990
- [MISR82] Misra, J. y Chandy, K. "Termination detection of diffusing computations in CSP". ACM Toplas 4 (Enero) 1982
- [QUEAZ93] Quemada, J. y Azcorra, A. "Ingeniería de protocolos". XV Escuela de Verano de Informática. La Manga del Mar Menor (Murcia). 1993
- [RAYN88] Raynal, Michel. "Distributed algorithms and protocols". Wiley. 1988
- [SICH90] Sidhu, D.P., Chung, A. y Blumer, T.P. "Experience with formal methods in protocol development". IFIP 1990
- [SIDBL85] Sidhu, D.P. y Blumer, T.P. "Some problems with the specification of the military standard transmission control protocol". SDC Burroughs Report. Noviembre 1985
- [TURN93] Turner, K.J. "Using Formal Description Techniques". Wiley. 1993
- [WEZA78] West, C.H. y Zafiropulo, P. "Automated validation of communications protocols: The CCITT X.21 recommendation". 1978