La Serie Universitaria de la Fundación Juan March presenta resúmenes, realizados por el propio autor, de algunos estudios e investigaciones llevados a cabo por los becarios de la Fundación y aprobados por los Asesores Secretarios de los distintos Departamentos.

El texto íntegro de las Memorias correspondientes se encuentra en la Biblioteca de la Fundación (Castelló, 77. Madrid-6).

La lista completa de los trabajos aprobados se presenta, en forma de fichas, en los Cuadernos Bibliográficos que publica la Fundación Juan March.

Los trabajos publicados en Serie Universitaria abarcan las siguientes especialidades: Arquitectura y Urbanismo; Artes Plásticas; Biología; Ciencias Agrarias; Ciencias Sociales; Comunicación Social; Derecho; Economía; Filosofía; Física; Geología; Historia; Ingeniería; Literatura y Filología; Matemáticas; Medicina, Farmacia y Veterinaria; Música; Química; Teología. A ellas corresponden los colores de la cubierta.

Edición no venal de 300 ejemplares
que se reparte gratuitamente a investigadores,
Bibliotecas y Centros especializados de toda España.

Fundación Juan March

172 Sobre el estudio y diseño de protocolos de comunicación/Miguel García Hoffmann.

SERIE UNIVERSITARIA

Fundación Juan March

Miguel García Hoffmann

Sobre el estudio y diseño de protocolos de comunicación.

Fundación Juan March
Serie Universitaria
172

Miguel García Hoffmann

Sobre el estudio y diseño de protocolos de comunicación.

Fundación Juan March
Castelló, 77. Teléf. 225 44 55
Madrid - 6

Este trabajo fue realizado con una Beca de la Convocatoria de España, 1979, individual.
Departamento de INGENIERIA
Centro de trabajo: Instituto de Cibernética del C.S.I.C. Barcelona.

Depósito Legal.: M-925-1982
I.S.B.N.: 84-7075-225-1
Impresión: Gráficas Ibérica. Tarragona, 34. Madrid-7

INDICE

Diversos autores han propuesto técnicas para la especifi cación y verificación de protocolos de comunicación. En este trabajo se presentan primeramente las metodologías más significativas en este campo, para exponer a continuación una nueva metodología de especificación, validación y verificación de protocolos de comunicación que se compara favorablemente con las existentes.

Para especificar el funcionamiento de los elementos integrantes de los protocolos se sugiere la utilización de un lenguaje descriptivo de alto nivel, el LDP. Las descripciones de protocolos en LDP son el punto de partida de los algoritmos de validación y verificación que se proponen.

La validación se basa en la construcción de un árbol que expresa todos los posibles caminos de ejecución del protocolo y permite detectar errores en su estructura lógica tales como bloqueos, bloqueos temporales, comportamiento cíclico, etc.

La verificación requiere expresar en forma de una relación de simulación las propiedades que se desea probar. El algoritmo de verificación construye mediante ejecución simbólica un árbol de prueba para cada punto de paro en la relación de simulación. La verificación se reduce así a la prueba de las condiciones de verificación generadas.

1. INTRODUCCION

En los últimos tiempos viene prestándose una atención creciente al estudio de los sistemas informáticos distribuidos.

Los objetivos y características de los diversos sistemas difieren notablemente, existiendo, sin embargo, ciertos aspectos comunes a todos ellos. Uno de los aspectos comunes a todos los sistemas distribuidos es la necesidad de intercambiar información entre los elementos que lo integran.

El intercambio de información entre los componentes de un sistema distribuido se denomina comunicación, entendiéndose por protocolo de comunicación el conjunto de reglas que la rigen. La materialización hardware o software de estas reglas recibe asimismo la denominación de protocolo.

El diseño y estudio de protocolos de comunicación comporta varias etapas entre las que se cuentan la definición del protocolo, su análisis cualitativo y cuantitativo y su realización práctica y documentación.

El primer aspecto, la definición del protocolo, es de suma importancia. La descripción de gran parte de los protocolos existentes se ha hecho en lenguaje escrito. Esta forma de descripción introduce en la especificación del protocolo las ambigüedades propias del lenguaje humano con el consiguiente riesgo de interpretaciones distintas por parte de las personas encargadas de la realización práctica del protocolo. Diversos autores han propuesto métodos formales para la definición de protocolos que, por otra parte, se prestan mejor que el lenguaje escrito al estudio de sus características.

El objeto del estudio cualitativo de protocolos es la validación de su estructura lógica y su verificación. Por validación se entiende la prueba de si en su funcionamiento el protocolo satisface o no ciertas propiedades lógicas como la ausencia de bloqueos, comportamiento cíclico, etc.. La verificación consiste en demostrar que la comunicación se realiza según las pautas que el diseñador intentó imponer en la definición del protocolo. Practicamente todos los protocolos han de realizar dos funciones

básicas denominadas función de control y función de transporte. La verificación de la función de control suele referirse a la inicialización y mantenimiento de la sincronización de las entidades que realizan el protocolo. El comportamiento correcto de la función de transporte se centra fundamentalmente en garantizar que los mensajes llegan a la estación de destino sin ser alterados, duplicados o reordenados.

El entorno en el que está inmerso el protocolo juega también papel importante en el estudio cualitativo.

El modelo ha de poder representar los procesos integrantes del protocolo así como su entorno. Entre los aspectos que con más frecuencia han de modelarse se cuentan las estructuras de datos y de control de los procesos, las decisiones internas que estos toman en función del valor de ciertas variables, la forma de comunicación y sincronización entre los procesos, los medios de comunicación, los sucesos externos, las interfases entre los diversos niveles de las estructuras jerárquicas de protocolos, las temporizaciones,.....

El análisis cuantitativo de protocolos aborda problemas tales como la determinación de la eficiencia de la comunicación, su banda pasante, etc. La mayor parte de los trabajos en este campo se basan en análisis de colas y simulación, no siendo objeto de este estudio.

En este trabajo se revisan las principales técnicas empleadas en la descripción y análisis cualitativo de protocolos para proponer a continuación una nueva metodologia de especificación, validación y verificación de protocolos de comunicación que se compara favorablemente con las existentes.

2. TECNICAS PARA LA DESCRIPCION Y ESTUDIO CUALITATIVO DE PROTOCOLOS.

En este apartado se resaltan las referencias de los trabajos más importantes publicados hasta 1979 sobre la descripción y estudio cualitativo de protocolos de comunicación. En /27/ puede encontrarse un breve estudio comentado de las diversas técnicas en ellos propuestas. Con el fin de facilitar el estudio de estas técnicas se han referenciado asimismo los trabajos en los que encuentran su fundamento.

Un primer grupo de técnicas son aquellas que basan la descripción de las entidades que realizan el protocolo en alguna forma relacionada con las máquinas de estados finitos. Pueden incluirse en él la modelación mediante redes de Petri, particularmente los trabajos de Merlin, /56/,/57/,/58/, Danthine /18/, Bochmann /5/, y Symons /72/; la extensión de la teoría de coloquios realizada por Danthine y Bremer /17/,/18/,/10/,/11/,/19/, los trabajos de Mezzalira y Schreiber sobre modelación de protocolos mediante máquinas secuenciales de estructura variable /59/ /60/, y, por último, los estudios de Bochmann /4/,/71/,/8/ que ha empleado las máquinas de estados finitos para describir los subsistemas de que consta un protocolo.

Un segundo grupo lo forman las técnicas que emplean lenguajes formales y de alto nivel para la especificación y estudio de protocolos de comunicación: los trabajos de Harangozo /40/, /41/ que emplea gramáticas regulares, los de Bochmann /3/,/6/,/7/ y Stenning /69/, que especifican el funcionamiento de protocolos mediante lenguajes de alto nivel y emplean aserciones para demostrar propiedades y, finalmente, los de Gouda /29/, /30/, /31/ que ha ideado un lenguaje gráfico de alto nivel orientado al estudio de protocolos.

En los últimos tres años se han conseguido avances notables en el estudio de protocolos reales (X.25, HDLC,...) gracias al desarrollo de métodos que permiten la automatización parcial del proceso de diseño y verificación. En este apartado se consideran

los trabajos llevados a cabo por el Data Networks Group de IBM, Zurich, /67/, /74/, /75/, /77/, así como los resultados de otro grupo de investigadores del IBM Thomas J.Watson Research Center, que han desarrollado un método para la verificación automática de protocolos descritos en lenguaje de alto nivel /9/. Otros trabajos relacionados con la automatización del estudio y diseño de protocolos pueden encontrarse en /8/, /19/, /27/, /35/, /66/.

Del estudio de los diferentes trabajos presentados puede concluirse, como en /71/, que existen fundamentalmente tres formas de abordar el estudio cualitativo de protocolos. La primera se basa en la modelación del protocolo mediante algún método relacionado con el modelo de estados y en el análisis de la alcanzabilidad. La segunda forma se fundamenta en la descripción algorítmica de los elementos constituyentes del protocolo y en la prueba de propiedades mediante aserciones. La tercera es la sintetización de las dos anteriores.

Los modelos de estados son adecuados para estudiar los aspectos de control. Sin embargo, al intentar representar la estructura de datos de los distintos elementos o modelar medios de comunicación complejos se genera un número de estados excesivo. Dentro de este grupo, algunos autores han propuesto considerar tan solo un número pequeño de estados básicos del sistema, incluyendo en ellos información del contexto. Se consigue así limitar el número de estados totales. Sin embargo, la aplicación de las técnicas de análisis aptas para los modelos de estados encuentran aquí grandes dificultades.

La descripción algorítmica de protocolos permite, en general, representar adecuadamente la estructura de datos de los procesos que forman el protocolo. En el campo de las redes de computadores los métodos de verificación mediante aserciones han permitido verificar la función de transporte de ciertos protocolos. Sin embargo, se encuentran serias dificultades en la verificación de la función de control, estrechamente vinculada al modelo de estados.

Los trabajos más recientes intentan aprovechar las ventajas y eludir las limitaciones de los grupos de métodos anteriores. Las descripciones son algorítmicas, pero es necesario modelar de alguna forma los estados a que puede llegar el protocolo en su funcionamiento. Los aspectos de control se estudian sobre este modelo mediante el análisis de la alcanzabilidad. Para la verificación se incluyen aserciones sobre el estado y la estructura lógica del protocolo. La demostración de las aserciones combina el análisis de la alcanzabilidad con las pruebas lógicas sobre las aserciones que hacen referencia a la estructura de datos. Buena parte del éxito de esto métodos en el estudio de protocolos reales se debe a la incorporación del computador como ayuda a la verificación.

En el resto de este artículo se presenta un método de definición, validación y verificación que por sus características puede considerarse incluído en este último grupo.

Cabe recordar en este punto la existencia dentro de la International Federation for Information Processing (IFIP) de un Grupo de Trabajo, el 6.1, genérivamente denominado Packet-switched Network Interworking, y dentro de él el Grupo de Estudios C Formal Description and Verification, que se ocupa, entre otros temas, del estudio de la descripción, validación y verificación de protocolos de comunicación.

Las publicaciones periodicas de este Grupo asi como sus recopilaciones bibliográficas pueden ser una buena guia para el seguimiento de la rápida evolución que en los últimos años vienen experimentando estas técnicas.

Los trabajos a los que he aludido en este apartado alcanzan hasta el año 1978 y son, fundamentalmente, los señalados en

Dayton. J.C.,"A Bibliography on the Formal Specification and Verification of Computer Network Protocols", Proc. of the I Symp. on Computer Network Protocols, Liege 1978

3. PROPUESTA DE UNA NUEVA METODOLOGIA PARA EL DISEÑO DE PROTOCOLOS.

3.1. EL MODELO

Un protocolo se considera formado por un conjunto de procesos secuenciales, distribuidos geográficamente o no, que interaccionan entre sí. Los procesos son disjuntos, es decir, no tienen acceso a variables comunes. Los distintos procesos que integran un protocolo se ejecutan en procesadores independientes. Dado que los procesadores carecen de memoria común, los procesos se comunican a través de mensajes. La comunicación se realiza síncronamente; cuando un proceso está en condiciones de enviar un mensaje a otro y éste puede recibirlo se realiza la transferencia. Si uno de los procesos no está dispuesto para realizar la transferencia la ejecución del otro se detiene. El hacer esperar a un proceso que desea transmitir un mensaje hasta que el proceso destino está dispuesto a aceptarlo refleja correctamente la forma de comunicación real en la mayor parte de los casos. De todas formas, siempre que sea necesario pueden definirse específicamente procesos que modelen otra forma de comunicación tales como buffers, monitores, colas, etc.

El medio que utilizan los procesos para comunicarse se modela como un proceso más del protocolo. El lenguaje empleado en la descripción de los procesos permite modelar fácilmente las características del medio de comunicación, entre las que destacan la alteración de la información, la pérdida, duplicación y reordenación de los mensajes.

Las temporizaciones se modelan también como procesos. La especificación de las interfases entre los procesos que forman un protocolo y los procesos de otros niveles de la jerarquía se realiza describiendo el comprotamiento de éstos sin considerar los detalles de su funcionamiento que

sean ajenos a la interconexión.

3.2. EL LENGUAJE DE DESCRIPCION DE PROTOCOLOS.

Existen un cierto número de métodos formales para la descripción de procesos paralelos. Algunos de ellos son adaptaciones o extensiones de lenguajes de alto nivel ya existentes /15/, /36/, /37/ mientras que otros han sido específicamente diseñados para describir el paralelismo o estudiar las características de la comunicación /12/, /32/, /33/, /34/, /38/, /43/. Los protocolos de comunicación son un caso particular de procesos paralelos por lo que los métodos propuestos en las referencias indicadas pueden utilizarse para los protocolos. Sin embargo es conveniente disponer de un método orientado a la descripción de protocolos que sirva de base al estudio de su funcionamiento.

A continuación se exponen las características de un lenguaje para la descripción de protocolos, el LDP, inspirado en los lenguajes propuestos por Hoare /43/ y Hansen /38/ para la descripción de procesos paralelos. El lenguaje es una adaptación de las ideas de Hoare a las necesidades peculiares de los protocolos de comunicación. Debe entenderse que el LDP no es un lenguaje ejecutable sino tan solo una herramienta para la especificación formal de protocolos.

La figura 1 recoge parcialmente la descripción BNF de las sentencias del LDP. Una sentencia define la acción a realizar por el procesador que la ejecuta. La composición secuencial de sentencias se indica mediante el signo; y su ejecución se realiza secuencialmente y en orden. La sentencia nula no tiene efecto alguno, su ejecución es siempre correcta y no modifica el estado de ningún proceso. La ejecución de la sentencia de asignación,

SENTENCIAS

<sentencia>::=<sentencia sin etiqueta>|<etiqueta de sentencia>:
<sentencia sin etiqueta>
<etiqueta de sentencia>::=<entero sin signo>
<sentencia sin etiqueta>::=<sentencia simple>|<sentencia estructurada>
<sentencia simple>::=<sentencia vacia>|<sentencia de asignacion>
<sentencia vacia>::=skip
<sentencia estructurada>::=<sentencia alternativa>|<sentencia repetitiva>|
<sentencia paralela>
<lista de sentencias>::=[<declaracion>;|<sentencia>;]<sentencia>

SENTENCIA PARALELA

<sentencia paralela>::=[<proceso>{||<proceso>}]
<proceso>::=<etiqueta de proceso>::<lista de sentencias>
<etiqueta de proceso>::=<identificador con mayusculas>

SENTFNCIA DE ASIGNACION

<sentencia de asignacion>::=<sentencia de asignacion interna>|
<sentencia de asignacion externa>
<sentencia de asignacion interna>::=<variable>:=<expresion>|<variable
simple booleana>:=#
<variable>::=<variable simple>|<variable estructurada>
<variable estructurada>::=<constructor>(<lista de variables>)
<costructor>::=<vacio>|<identificador>
<lista de variables>::=<vacio>|<variable>{,<variable>}
<expresion>::=<expresion simple>|<expresion estructurada>
<expresion estructurada>::=<constructor>(<lista de expresiones>)
<lista de expresiones>::=<vacio>|<expresion>{,<expresion>}
<sentencia de asignacion externa>::=<sentencia de entrada>|<sentencia
de salida>
<sentencia de entrada>::=<variable>:=<etiqueta de proceso origen>
<sentencia de salida>::=<etiqueta de proceso destino>:=<expresion>
<etiqueta de proceso origen>::=<etiqueta de proceso>
<etiqueta de proceso destino>::=<etiqueta de proceso>

SENTENCIA ALTERNATIVA

<sentencia alternativa>::=[<comando con guardia>{[]<comando con guardia>}]
<comando con guardia>::=<guardia>→<lista de sentencias>
<guardia>::=<lista de guardias>|<lista de guardias>;<sentencia de
asignacion externa>|<sentencia de asignacion externa>
<lista de guardias>::=<elemento de guardia>{;<elemento de guardia>}
<elemento de guardia>::=<elemento de guardia incondicional>|<elemento
de guardia condicional>
<elemento de guardia condicional>::=$<elemento de guardia incondicional>
<elemento de guardia incondicional>::=<expresion booleana>|<declaracion>

SENTENCIA REPETITIVA

<sentencia repetitiva>::=*<sentencia alternativa>

Figura 1: Descripción BNF parcial del LDI

si no falla, puede modificar el estado del proceso en que se encuentra, el estado de otro proceso, o ambos.

La sentencia paralela especifica la ejecución concurrente de todos los procesos que la forman. La ejecución de los procesos se inicia simultáneamente y cada uno de ellos continúa ejecutando su lista de sentencias hasta que ésta se acaba. La ejecución de una sentencia paralela termina cuando terminan todos sus procesos constituyentes. La lista de sentencias que forman un proceso va siempre precedida de un identificador que da nombre al proceso. A fin de distinguir el nombre de un proceso de una variable en las sentencias de asignación externa, los identificadores de proceso se escribirán en mayúsculas. El nombre de una lista de sentencias no debe confundirse con el nombre de un proceso.

La sentencia de asignación interna, excepto en el caso de la función #, tiene el sentido usual de la asignación. El valor calculado por la expresión se asigna a la variable que aparece a la izquierda del signo :=. Este valor, así como la variable a que está destinado, puede ser simple o estructurado. El resultado de una expresión simple puede ser simple o estructurado, según sea el carácter de los operandos. Por el contrario, una expresión estructurada evalúa un valor estructurado cuyos elementos son los valores calculados por cada expresión de la lista de expresiones. La asignación no puede llevarse a cabo si alguno de los valores queda indefinido o si éstos no pueden pasar a ocupar la variable que indica la sentencia de asignación.

Una variable simple puede recibir un valor simple o estructurado, siempre que sean del mismo tipo. En cuanto a las variables estructuradas, pueden recibir tan sólo un valor estructurado con el mismo constructor y tal que el número y tipo de sus elementos constituyentes coincida con el número y tipo de los elementos que forman la variable.

El valor estructurado que no tiene componentes sirve para modelar señales, cuyo nombre es el de su constructor.

La "función incorporada" denotada por el signo # produce de forma aleatoria un valor cierto o falso que se asigna a la variable booleana de la izquierda del signo :=. Esta función sirve para modelar sucesos de los que se desconoce la función de distribución de su aparición.

En el modelo empleado los procesos se ejecutan en procesadores independientes que carecen de memoria común. La transferencia de información de un proceso a otro puede contemplarse como la asignación de un determinado valor calculado en el proceso origen a una variable situada en el proceso de destino. La asignación sólo podrá llevarse a cabo cuando los procesos origen y destino estén dispuestos a realizarla y en el supuesto de que la variable y la expresión sean compatibles. Esto quiere decir que la ejecución de una sentencia de asignación externa queda retrasada hasta que el proceso correspondiente esté en condiciones de ejecutar una sentencia de asignación externa compatible. Cuando ambos están dispuestos la asignación se realiza. Concretamente, la ejecución de una sentencia de salida (entrada) de un proceso A hacia (desde) un proceso B se ejecuta simultáneamente con una sentencia de entrada (salida) del proceso B desde (hacia) el proceso A.

La sentencia alternativa es la forma empleada para introducir el no determinismo en el LDP. Su funcionamiento se basa en los comandos con guardia de Dijkstra /23/. La sentencia alternativa especifica la ejecución de una única lista de sentencias cuya guardia evalue a cierto. Su ejecución comienza evaluando las guardias. La evaluación de una guardia se realiza elemento por elemento y de izquierda a derecha. En el caso de que todas las guardias evaluen un valor falso la sentencia alternativa produce un error. Si más de una guardia es cierta se escoge arbi

trariamente para su ejecución una de las listas de sentencias de guardia cierta.

La evaluación de los elementos de la guardia no tiene un efecto de asignación sobre sus elementos constituyentes, excepto si se trata de una sentencia de asignación externa. El símbolo $ precediendo un elemento de guardia pospone su evaluación hasta que el valor sea cierto, posponiendo por tanto la evaluación de la guardia. Obsérvese que esta estructura de control recoge características de la estructura if y de la when, sugeridas por Hansen /38/ y otros.

Como último (o único) elemento de una guardia puede figurar una sentencia de asignación externa. Esta pospone la evaluación de la guardia hasta que pueda ejecutarse conjuntamente con una sentencia de asignación externa del proceso correspondiente. Cuando todos los elementos de la guardia evaluan un valor cierto y la sentencia de entrada/salida puede ejecutarse, la guardia evalua un valor cierto. Si la sentencia alternativa selecciona este comando para ser ejecutado, la entrada/salida se realiza pasando el control a la lista de sentencias del comando. El permitir la presencia de tan solo una sentencia de asignación externa en la guardia se debe a su carácter ejecutable; los procesos individuales son secuenciales y, por tanto, sólo pueden ejecutar una sentencia de entrada o salida en un instante dado.

La sentencia repetitiva ejecuta, mientras sea posible, los comandos con guardia que forman la sentencia alternativa y tan sólo termina, sin efecto alguno, cuando todas las guardias evaluan un valor falso.

En /27/ pueden verse diversos ejemplos de utilización del LDP que ilustran la potencia de este lenguaje en la modelación y descripción de los diversos aspectos que intervienen en el estudio de los protocolos. Se recoge aquí, a

título ilustrativo, una simplificación del protocolo de bit alternante del National Physical Laboratory /1/, /6/. El protocolo es responsable de controlar la transferencia alternada de mensajes entre una estación emisora S y una receptora R sobre un enlace half-duplex M que puede alterar o perder la información. El emisor aguarda la recepción de un mensaje de aceptación notificando la correcta recepción en R del último mensaje transmitido para proceder al envío del mensaje siguiente. Los mensajes transmitidos por cada estación contienen información para detectar errores producidos en la comunicación y un bit de control denominado el bit alternante. Cada estación conserva el valor del bit alternante del último mensaje enviado, alternando su valor en cada nuevo mensaje transmitido ; el cambio de valor del bit alternante indica la aceptación del mensaje. El protocolo detecta los errores de transmisión y, mediante un temporizador, la pérdida de los mensajes. En ambos casos el emisor retransmite el último mensaje. La figura 2 recoge la descripción LDP del protocolo y el significado de las variables de los procesos S y R. Como puede verse el medio se modela como un proceso que puede alterar o perder la información. Para simplificar se ha considerado que la guardia $time toma un valor cierto cuando todos los otros procesos se encuentran bloqueados por la pérdida de un mensaje. Tampoco he considerado la forma de comunicación entre S,R y sus respectivos usuarios, de modo que el empleo de los vectores indata y outdata pueden parecer artificiales.

Las sentencias de S

```
....datas := indata(pt);pt:=pt+1
```

puede sustituirse por

```
datas := US
```

donde US sería el proceso usuario de S, generador de los

datos que han de transmitirse. Pueden hacerse considera ciones análogas sobre R y su vector outdata.

```
[S::SENDER || M::MEDIUM || R::RECEIVER || C::CLOCK]

SENDER
...*[true →
      [¬ea ∧ ack=seq → datas:=indata(pt); pt:=pt+1; seq=¬seq
      [] ea ∨ ack≠seq → skip]; M:=(datas,seq); C:=start();
      [ack:=M → C:=stop(); ea:=# [] timeout():=C → skip]
    ]

MEDIUM
*[(datam,seqm):=S → lossm:=#; [¬lossm → R:=(datam,seqm) [] lossm → skip]
 [expm:=R → lossa:=#; [¬lossa → S:=expm [] lossa → skip]
 ]

RECEIVER
...*[(datar,seqr):=M → em:=#;
      [¬em ∧ seqr≠exp → outdata(ps):=datar; ps:=ps+1; exp:=¬exp
      [] em ∨ seqr=exp → skip]; M:=exp
    ]

CLOCK
*[start():=S → [stop():=S → skip [] $time → S:=timeout()]]
```

SENDER

seq	bit alternante de S
ack	valor del bit alternante recibido por S
datas	datos enviados por S
indata	vector conteniendo los datos que han de transmitirse
pt	puntero de indata
ea	error detectado por S en la recepción

RECEIVER

exp	bit alternante dr R
seqr	valor del bit alternante recibido por R
datar	datos recibidos por S
outdata	vector conteniendo los datos correctamente recibidos
ps	puntero de outdata
em	error detectado por R en la recepción

Figura 2: Descripción LDP del protocolo de bit alternante.

3.3. ALGUNAS FORMALIZACIONES.

Antes de pasar a considerar la validación de protocolos descritos en LDP conviene introducir algunos conceptos y formalizaciones. Este es el objeto del presente apartado.

3.3.1. Estado de un protocolo.

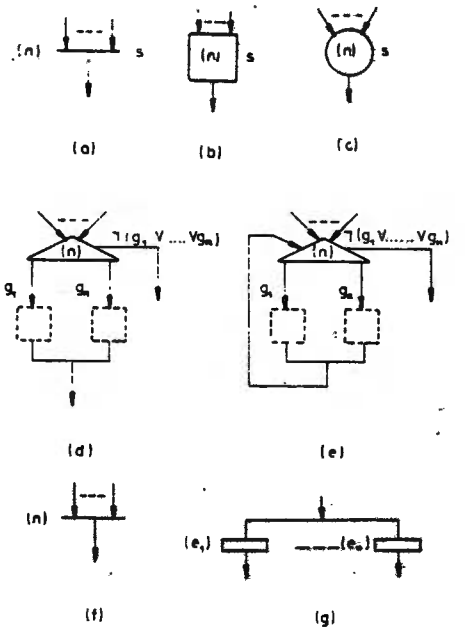
En el método de estudio y diseño de protocolos que se está exponiendo las descripciones son algorítmicas, pero es necesario además caracterizar la noción de estado del protocolo. Se define el estado de un proceso integrante del protocolo como la etiqueta de la sentencia en que se encuentra dicho proceso. En la descripción LDP se colocan etiquetas en los puntos que hayan de considerarse estados. La etiqueta corresponde a un estado cuyas acciones asociadas son las indicadas por las sentencias comprendidas entre la etiqueta considerada y la siguiente etiqueta. El estado de un protocolo queda definido por el estado de sus procesos constituyentes.

3.3.2. El LDP gráfico.

Con el único fin de facilitar la comprensión del funcionamiento de un protocolo y del método de validación que se desarrolla más adelante resulta conveniente la representación gráfica de las descripciones LDP. La figura 3 muestra la representación gráfica de las sentencias del LDP y la figura 4 es la descripción gráfica del protocolo de bit alternante de la figura 2.

3.3.3. Estructura de un protocolo.

Para el estudio del funcionamiento lógico de protocolos descritos en LDP conviene prescindir del detalle de la estructura de datos para centrar la atención en los aspectos de control. Se define la estructura gráfica de un protocolo como el grafo que se obtiene de la descripción LDP prescindiendo de todas las variables y rotulando cada arco con la condición que permita la transición



n: etiqueta de sentencia
s: sentencia
g: guardia
e: etiqueta

Figura 3
Representación gráfica de las sentencias
(a)Sentencia de asignación interna
(b)Sentencia de salida
(c)Sentencia de entrada
(d)Sentencia alternativa
(e)Sentencia repetitiva
(f)Sentencia nula
(g)Sentencia paralela

entre los estados que une. Existen tres tipos de transiciones; libres (p.e. asignación interna), simultáneas con una transición compatible de otro proceso (p.e. asignación externa) y condicionadas (p.e. sentencia alternativa). Junto a los arcos de las transiciones simultáneas

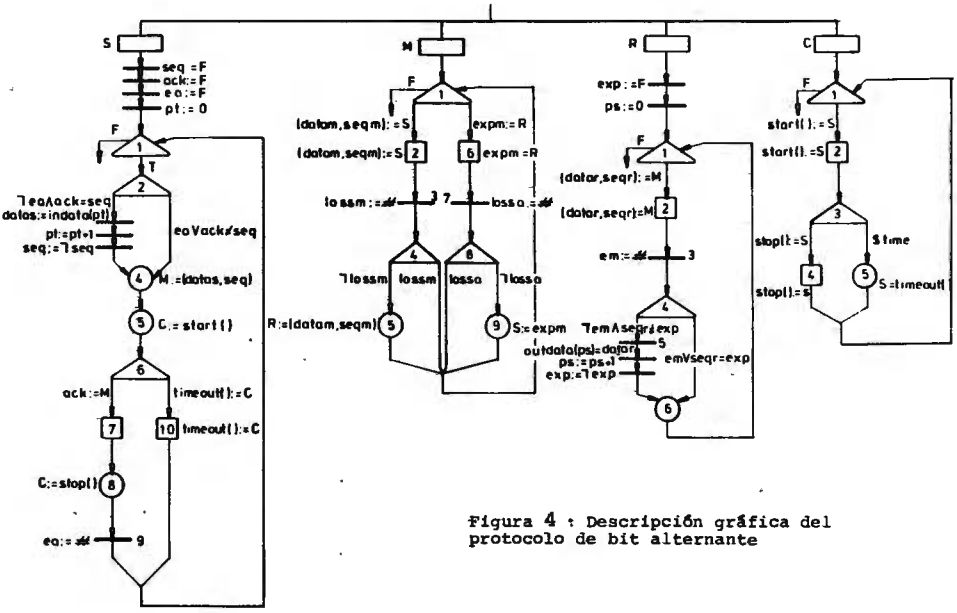


Figura 4 : Descripción gráfica del protocolo de bit alternante

se escribe el nombre del proceso con el que se realiza la transición y la lista de las transiciones entre estados de aquel proceso que puedan realizarse simultáneamente con la transición que se considera. Cuando la ejecución de una transición depende del estado en que se encuentra otro proceso se rotula con los estados del otro proceso en los cuales puede realizarse la transición. La figura 5 es la estructura gráfica del protocolo de bit alternante considerado.

La estructura gráfica de un protocolo conserva las restricciones en la comunicación de la descripción original, siendo más general que esta al suponerse que cualquier elemento de las guardias puede evaluar a cierto. Si se demuestra alguna propiedad de la estructura, tal como la ausencia de bloqueos, puede afirmarse que el protocolo posee esa propiedad.

3.3.4. El tiempo de ejecución.

El estudio de la sincronización de procesos distribuidos involucra ineludiblemente el concepto tiempo. Este problema viene siendo abordado en los últimos años en un intento de formalizar conceptos tales como sistema distribuido, tiempo y espacio en sistemas distribuidos, sucesión de sucesos, etc.... /49/, /51/, /52/. En las descripciones LDP se supone una única referencia absoluta de tiempos lógicos; en la terminología de /52/, un sistema multireferencial perfecto en que la evolución de cada proceso se considera a intervalos de tiempo constantes e iguales para todos. Las transiciones posibles entre estados del sistema quedan perfectamente determinadas conociendo el tiempo lógico de ejecución de las tran

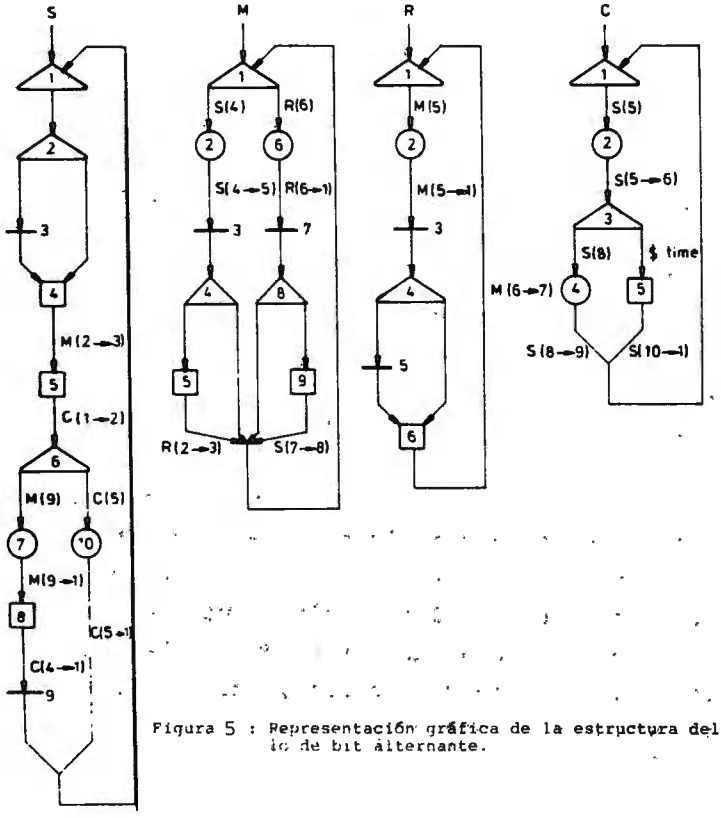


Figura 5 : Representación gráfica de la estructura del protocolo de bit alternante.

siciones individuales de cada proceso.

La selección de la unidad de tiempo lógico y la asignación de éste a las distintas transiciones debe tomar en cuenta las características del sistema real y el grado de precisión que se desea obtener en el estudio. La figura 6 muestra la descripción gráfica temporal del protocolo de bit alternante despreciando los tiempos de ejecución de los procesadores y considerando tan solo los tiempos de transmisión, la temporización y los tiempos de comunicación del proceso fuente y consumidor de mensajes. Un valor siempre adecuado para la unidad de tiempo lógico es el máximo común divisor de los tiempos reales de ejecución.

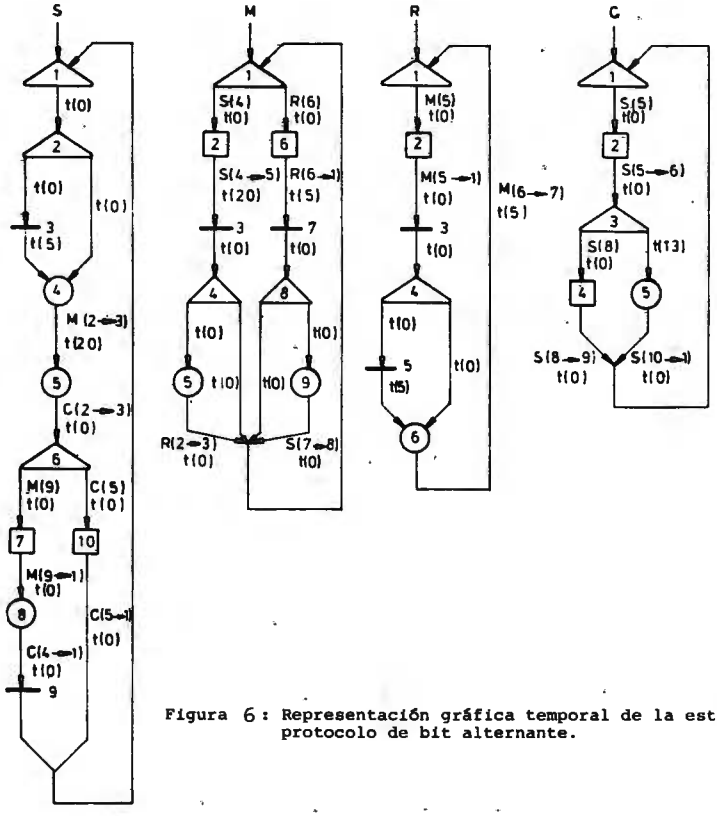


Figura 6: Representación gráfica temporal de la estructura del protocolo de bit alternante.

3.4. VALIDACION DE DESCRIPCIONES LDP.

Se exponen en este apartado los algoritmos de validación y de validación temporal con tiempo de ejecución fijo. Por falta de espacio y por tratarse de una generalización de la validación con tiempo de ejecución fijo no se considera aquí la validación con tiempo de ejecución variable /27/.

3.4.1. Validación no temporal de descripciones LDP.

La validación se basa en el análisis de la alcanzabilidad de la estructura del protocolo. Para buscar el conjunto de estados siguientes de uno dado se introduce formalmente el concepto de perturbación como sigue.

Los n procesos integrantes de un protocolo se numeran en orden creciente. Llamando S_i al conjunto de estados s_i^j en que puede encontrarse el proceso i, el conjunto de estados $\overline{e}_j$ alcanzables por el protocolo es, en general, $E \subset S = S_1 x \ldots x S_n$.

A partir de la estructura gráfica de un protocolo se define para cada proceso i una relación

$R_i \subset SxS$,

denominada relación de perturbación de la forma siguiente:

(a) Para todo par de estados s_i^j, $s_i^k \in S_i$ unidos por una transición sin rotular o rotulada con \$

$$[(x \ldots s_i^j \ldots . x), (x \ldots . s_i^k \ldots . x)] \in R_i$$

en donde una x en la posición p, indica cualquier estado $s_p^q \in S_p$

(b) Para todo par de estados $s_i^j, s_i^k \in S_i$ unidos por una transición simultánea con el proceso l, y para cada $s_l^m \rightarrow s_l^n$

que rotula el arco

$$[\ (x...s_i^j...s_1^m...x),(x...s_i^k...s_1^n...x]\ ] \in R_i$$

(c) Para todo par de estados, s_i^j, $s_i^k \in S_i$ unidos por una transición condicionada al estado de un proceso 1, y para cada (s_1^m) que rotula el arco

$$[\ (x...x_i^j...s_1^m...x),(x...s_i^k...s_1^m...x]\ ] \in R_i$$

La relación R_i permite conocer los estados a que puede llegar un protocolo desde un estado dado por una transición del proceso i. Formalmente se define la perturbación de un estado $\bar{e}_j \in E$ por el proceso i como el conjunto

$$P_i(\bar{e}_j) = \{\ \bar{e}_k \in E \quad | \quad (\bar{e}_j,\bar{e}_k) \in R_i\ \},$$

conjunto que puede ser vacío.

De un modo análogo, se define el conjunto de estados siguientes de un estado $\bar{e}_j \in E$ como el conjunto

$$P(\bar{e}_j) = \{\bar{e}_k \in E \mid \bar{e}_k \in P_1(\bar{e}_j) \cup ..... \cup P_n(\bar{e}_j)\}$$

El conjunto de estados siguientes contiene todos los estados a los que puede llegar el protocolo por la ejecución de una única transición de cualquiera de los procesos que lo forman.

La definición de perturbación y de estado siguiente del apartado anterior permite realizar de un modo simple el análisis de la alcanzabilidad, mediante el cual pueden encontrarse diversos errores de diseño. El análisis se basa en la construcción de un árbol de ejecución. El nudo inicial o raíz contiene el estado inicial; a partir de él se generan todas las posibles secuencias de estados.

Para la construcción del árbol de ejecución se propone el siguiente algoritmo:

(a) Definir una sucesión de conjuntos A(i) de estados del sistema. El conjunto A(0) contiene el estado inicial del protocolo. Hacer i=0.

(b) Tomar un elemento $\bar{e} \in A(i)$ cuyo conjunto de estados siguientes no haya sido aún determinado e ir a (c). De no existir tal elemento mirar si $A(i+1)=\{\phi\}$. Si A(i+1) es vacío la validación ha concluído, en caso contrario hacer i=i+1 y volver al principio de (b).

(c) Determinar el conjunto de estados siguientes $P(\bar{e})$ y añadir a A(i+1) los elementos de $P(\bar{e})$ que no se encuentren ya en algún A(j) para j=1,....i+1. Ir a (b).

Durante la construcción del árbol pueden determinarse ciertos errores de diseño, principalmente:

(a) Bloqueos

Si el conjunto de estados siguientes de uno dado, $P(\bar{e})$, es vacío (apartado (c) del algoritmo), puede afirmarse que $\bar{e}$ es un estado terminal o un bloqueo potencial. Del árbol de ejecución se determina el camino seguido por el protocolo para llegar al estado $\bar{e}$. Del examen de la descripción LDP del protocolo se desprende si el protocolo puede efectivamente recorrer ese camino o no (recuérdese que la estructura del protocolo pierde el sentido de la lógica interna de la descripción original).

(b) Bloqueos temporales

En el apartado (c) del algoritmo se determina si los estados sucesores de uno dado, $P(\bar{e})$, han sido previamente generados. Los estados sucesores que ya existan en el árbol no se consideran más. Si un estado

que ya existe es antecesor de un estado generado, se ha detectado un bucle. De la descripción LDP se determina fácilmente si esta situación es o no correcta.

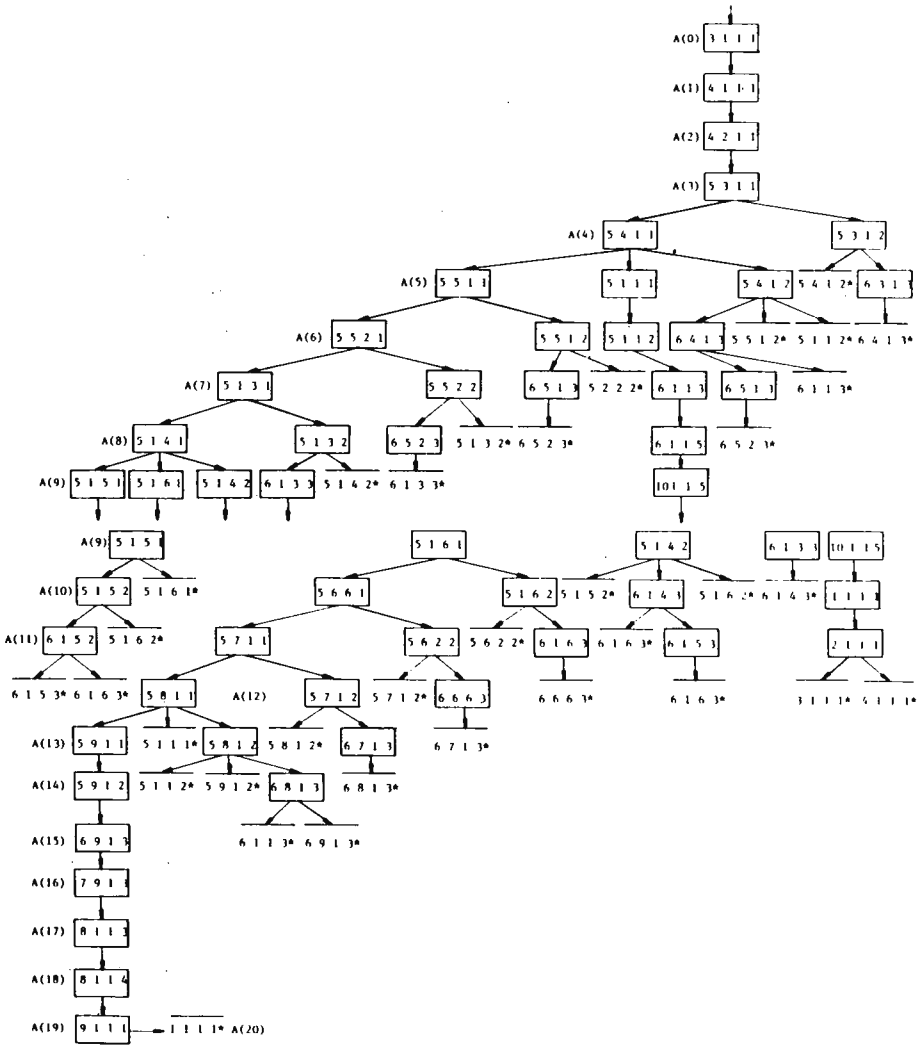


Figura 7: Arbol de ejecución del protocolo de la figura 13.

(c) Comportamiento cíclico

Si el protocolo debe retornar al estado inicial tras un número finito de transiciones, en el análisis de la alcanzabilidad debe aparecer el estado inicial en algún $P(\bar{e})$. Si la validación acaba y no se ha dado esta circunstancia el comportamiento del protocolo es erróneo.

La figura 7 ilustra la validación del protocolo de bit alternante cuya estructura representa la figura 5. Para simplificar se ha supuesto que la guardia $time no se hace cierta mientras sea posible la ejecución de cualquier otra transición. Los asteriscos indican que el estado generado ya existe en el árbol. Del análisis de la alcanzabilidad del protocolo de bit alternante se desprende que su comportamiento es cíclico y carece de bloqueos.

3.4.2. Validación con tiempo de ejecución fijo.

Considerando el tiempo de ejecución de las transiciones la determinación de los sucesores de un estado requiere, además del conocimiento de las características estáticas del protocolo -grafo temporal-, la consideración del camino de ejecución seguido para llegar al estado.

En la validación temporal puede producirse más de una transición en un salto discreto de tiempo lógico. Por ello se relajan las definiciones de perturbación y relación de perturbación como se indica en el párrafo siguiente, donde el símbolo - denota la indeterminación del estado de un proceso. El conjunto de estados del sistema en el que el estado de algún proceso esté indeterminado se representa por $\underline{E}$.

A partir de la descripción gráfica temporal de un pro

tocolo se define para cada proceso una relación T_i denominada de perturbación parcial, de la forma siguiente:

(a) Para cada par de estados $s_i^j, s_i^k \in S_i$ unidos por una transición libre o rotulada con \$.

$$[\ (x_1 \ldots x_i^j \ldots x_n), (- \ldots - s_i^k - \ldots -)\ ] \in T_i$$

donde $x_p = s_p^1 \in S_p$ para algún 1.

(b) Para cada par de estados $s_i^j, s_i^k \in S_i$ unidos por una transición simultánea, y por cada $s_1^m \rightarrow s_1^n \in S_1$ que rotula la transición

$$[(x_1 \ldots s_i^j \ldots s_1^m \ldots x_n), (- \ldots - s_i^k - \ldots - s_1^n - \ldots -)] \in T_i$$

(c) Para cada par de estados s_i^j, $s_i^k \in S_i$ unidos por una transición condicional, y por cada $s_1^m \in S_1$ que la rotula

$$[(x_1 \ldots s_i^j \ldots s_1^m \ldots x_n), (- \ldots - s_i^k - \ldots - s_1^m - \ldots -)] \in T_i$$

Dado un estado $\bar{e} \in E$ del sistema se define para cada proceso la perturbación parcial del estado $\bar{e}$, como el conjunto

$$U_i(\bar{e}) = \{\ (\bar{f}, t_i^{jk}) \in \underline{E} \times N \mid (\bar{e}, \bar{f}) \in T_i\ \}$$

en donde N es el conjunto de números naturales, t_i^{jk} es el tiempo que rotula la transición de s_i^j a s_i^k, elementos i-ésimos de $\bar{e}$ y $\bar{f}$ respectivamente. La perturbación parcial de un estado $\bar{e}$ por un proceso i no es más que la relación de los estados que pueden alcanzar el proceso i y los que interaccionan con él y los tiempos de ejecución de las transiciones.

La validación requiere la construcción de un árbol de ejecución temporal. Los nudos de árbol constan de dos elementos:

(a) El estado $\bar{e}$ del sistema.

(b) Los conjuntos de transiciones parciales posibles de cada proceso y el tiempo que necesiten para realizarse. Estos conjuntos se denominarán perturbaciones parciales dinámicas, y se denotan por V_i.

Obsérvese que la V_i son parte del nudo y no dependen exclusivamente del estado del sistema, $\bar{e}$. Como se verá más adelante su contenido depende del camino seguido para llegar al nudo.

A continuación se presenta el algoritmo utilizado para la generación del árbol de ejecución temporal. El nudo inicial, o raíz, está formado por el estado inicial $\bar{e}_0 \in E$ y las V_i son $V_i = U_i(\bar{e}_0)$ (perturbaciones de $\bar{e}_0$). Para obtener los nudos sucesores de uno dado n, se siguen los pasos indicados a continuación.

(a) Buscar en los V_i^n la transición que requiere un tiempo menor para ejecutarse. Se denota t_{min} al tiempo de ejecución de esta transición.

(b) Buscar los estados sucesores de $\bar{e}$ al cabo del tiempo t_{min}. Para ello

(1) Formar los conjuntos

$$W_i = \{ \bar{f} \mid (\bar{f}, t_{min}) \in V_i^n \}.$$

Cada W_i contiene las transiciones parciales del estado $\bar{e}$ que puede ejecutar el proceso i en un tiempo t_{min}.

(2) Tomando un único elemento de cada W_i formar las transiciones compatibles de los procesos constituyentes que conducen a los nuevos estados $\bar{g}$, sucesores del $\bar{e}$.

(c) Cada estado sucesor $\bar{g}$ constituye un nuevo nudo, cuyas perturbaciones parciales dinámicas V_i^m se forman a partir de las V_i^n del nudo origen en tres pasos

(1) Para todo i tal que $e_i \neq g_i$ hacer $V_i^m = \{\phi\}$.

(2) Para todo i tal que $e_i = g_i$ hacer $V_i^m = V_i^n$ y decrementar todos los tiempos de V_i^m en t_{min}.

(3) Para todo i, añadir a V_i^m los elementos de $U_i(\bar{g})$ que no correspondan a estados siguientes del sistema ya reflejados en algún elemento de V_i^m.

A pesar de la aparente complejidad del algoritmo, su aplicación manual es sencilla. La idea es que de las transiciones que pueden realizarse en un momento dado se seleccionan para su ejecución las que requieren un menor tiempo de ejecución. Los procesos que no han cambiado de estado y que podían ejecutar una transición continúan en esta situación, pero el tiempo que debe transcurrir para que pueda ejecutarse la transición ha disminuido en t_{min}. Por otra parte, en el estado alcanzado pueden admitirse nuevas transiciones, que deben incluirse en la caracterización del nudo.

La figura 8 muestra una simplificación del árbol de ejecución temporal del protocolo de la figura 6. La parte superior de los rectángulos indica el estado del sistema correspondiente al nudo; en la parte inferior se representan los tiempos más pequeños de los V_i. Los V_i vacíos se representan por -, indicando que el proceso correspondiente no puede modificar su estado (puede pensarse que - significa tiempo de ejecución infinito).

El algoritmo de validación adquiere un aspecto similar al de la validación no temporal. Se desglosa en los siguientes pasos:

(a) Definir una sucesión de conjuntos A(i) de nu-

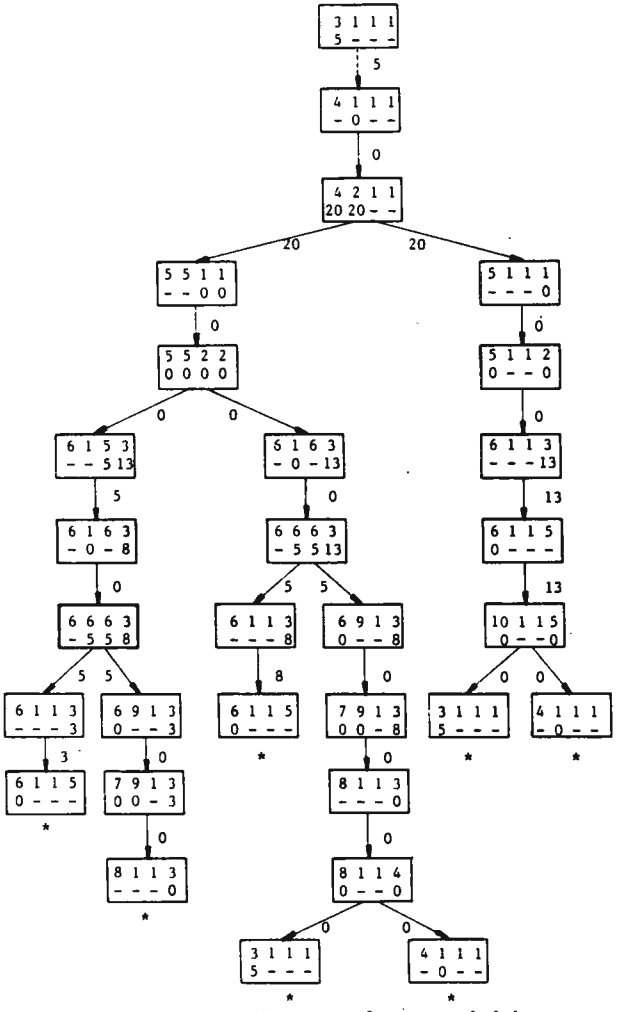


Fig. 8. Arbol de ejecución temporal del protocolo de la figura 13.

dos del árbol. El conjunto A(0) contiene el nudo inicial, $\bar{e}_0$ y $V_i = U_i\ (\bar{e}_0)$. Hacer i=0.

(b) Tomar un nudo de A(i) cuyo conjunto de nudos siguientes no haya sido aún determinado e ir a (c). De no existir tal nudo mirar si A(i+1)=(ϕ). En este caso la validación ha concluído, en caso contrario ha-

cer i:=i+1 y volver al principio de (b).

(c) Determinar los nudos sucesores del nudo considerado y añadir a A(i+1) los que no se encuentran ya en algún A(j) para j=1.....i+1. Ir a (b). Dos nudos n y m son iguales cuando los estados son iguales y $V_i^n = V_i^m$ para todo i.

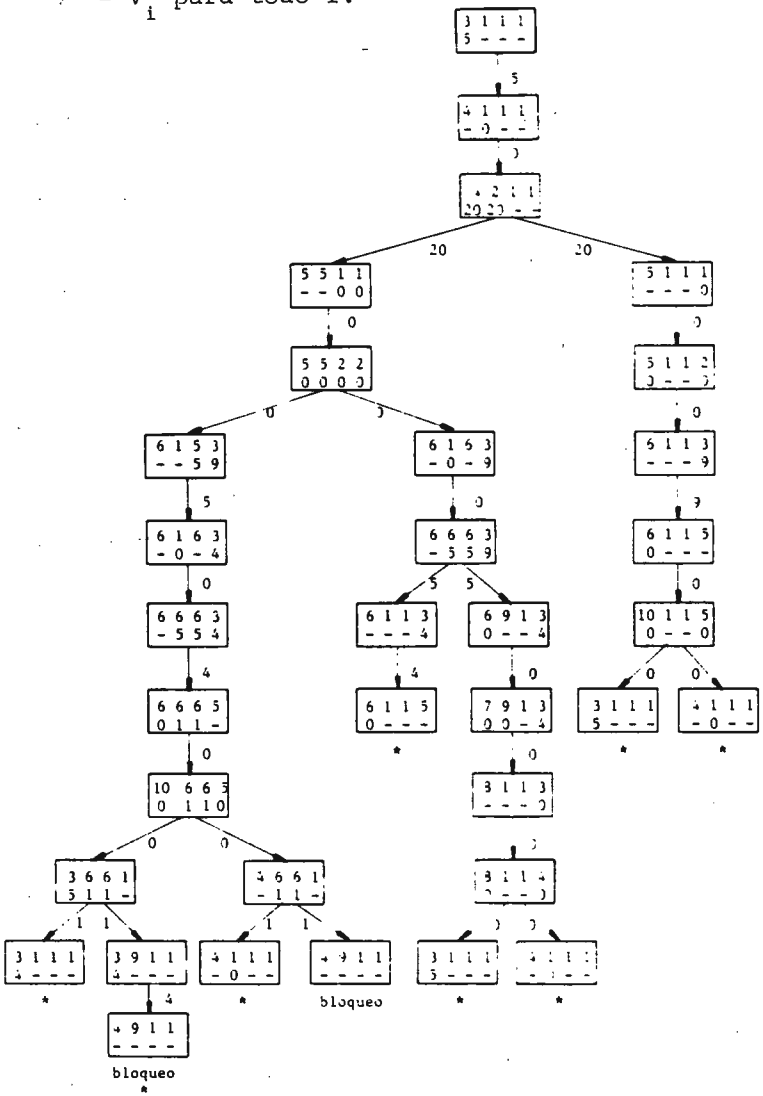


Figura 9 : Arbol de ejecución con temporización t(9).

La figura 8 corresponde a la validación temporal del protocolo de bit alternante que se viene considerando. Es interesante hacer notar la notable reducción en el número de estados (nudos) respecto a la figura 7 que representa la validación no temporal del mismo protocolo. Asimismo, es importante remarcar la simplicidad del proceso de validación a partir de la descripción gráfica temporal LDP de la figura 6. Si en ésta se modifica la temporización, dándole un valor de 9 unidades de tiempo, t(9), se obtiene la (no) validación de la figura 9. En ella se observa claramente que el sistema puede quedar bloqueado en el estado (4911). Este error proviene de la selección inadecuada de la temporización.

3.5. VERIFICACION DE DESCRIPCIONES LDP.

En la prueba estática de corrección de programas se han empleado técnicas diversas /54/. La mayor parte de los esfuerzos se han dirigido hacia los programas secuenciales, basándose los métodos en las ideas de Goldstine y Von Neumann más tarde formalizadas por Naur /61/ y Floyd /26/. El método de las aserciones permite demostrar si un programa es correcto o no respecto a un conjunto de predicados sobre las variables del programa. La verificación requiere el establecimiento de una aserción inicial se cumple al comienzo de la ejecución del programa, deba cumplirse la aserción final al término de la misma. Para facilitar la prueba se añaden aserciones inductivas en puntos intermedios del programa, reduciéndose la verificación de la corrección parcial a la prueba de las denominadas condiciones de verificación. Diversos autores han seguido esta línea proponiendo sistemas automáticos de verificación /22/, /28/, /42/, /46/, /73/.

La extensión del método de las aserciones a la prueba de programas concurrentes encuentra aún mayores dificultades ya que el resultado de la ejecución depende del orden

impredictible en que se entrelazan las acciones de los diversos procesos. Sin embargo, dentro de la programación concurrente existen determinadas aplicaciones cuyas peculiares características hacen más asequible la verificación. Uno de estos casos lo constituyen los protocolos de comunicación, objeto de este trabajo, en los cuales, normalmente, se manejan estructuras de datos simples, como ya se ha indicado. Los trabajos más relevantes en el campo de la verificación automática de protocolos se deben a Bremer y Danthine /10/, /11/, /20/, Brand y Joyner /9/ y Hajek /35/. La metodología de descripción y validación de protocolos propuesta en los puntos anteriores se completa ahora con las técnicas de verificación desarrolladas en este apartado. Sin embargo, dado que el planteamiento teórico y la solución propuesta al problema de la verificación son absolutamente generales, se expondrá el método en el marco más amplio de los procesos concurrentes para señalar posteriormente sus limitaciones y su aplicación al caso particular de los protocolos de comunicación.

3.5.1. Verificación de procesos secuenciales descritos en LDP.

Diversos autores han empleado la ejecución simbólica en la verificación de programas secuenciales /22/, /39/ /47/. La ejecución simbólica es una extensión natural del concepto de ejecución de un programa. Los valores iniciales se convierten en símbolos, las variables en el curso de la ejecución simbólica toman valores simbólicos que serán expresiones compuestas de operadores, constantes y los símbolos iniciales, la verificación requiere la prueba de teoremas en cálculo de predicados de primer orden. En un programa secuencial determinista, fijados los valores de entrada queda determinado el camino de ejecución y, por tanto, el resultado. Las descripciones LDP no son deterministas por lo que los resultados pue-

den ser diferentes para un mismo vector inicial. Por otro lado, si los valores iniciales son símbolos representando todos los valores iniciales posibles, la ejecución simbólica debe permitir determinar todos los posibles caminos de ejecución.

El objeto de la ejecución simbólica de descripciones en LDP es construir el denominado árbol de prueba. Cada nudo del árbol está formado por tres elementos.

(a) Punto de control: es la etiqueta que indica la posición alcanzada en la ejecución simbólica de la descripción.

(b) Vector de estado: está formado por los valores simbólicos de todas las variables de la descripción. Un valor simbólico es una expresión formada por los operadores del lenguaje, constantes y los símbolos iniciales.

(c) Lista de predicados: conteniendo el conjunto de condiciones que los símbolos iniciales han de satisfacer para que el control pueda llegar al nudo.

El nudo inicial o raíz refleja todas las situaciones en que puede encontrarse inicialmente la descripción. Su punto de control es la etiqueta de la primera sentencia, el vector de estado contiene los valores simbólicos iniciales de todas las variables y la lista de predicados las restricciones que deban cumplir los símbolos iniciales.

La ejecución simbólica de una sentencia de asignación del tipo < variable : = expresión > tiene por efecto la asignación a la variable de la expresión simbólica resultante de sustituir las variables de la expresión por sus valores simbólicos. Las sentencias < variable booleana : = # > generan dos sucesores, uno asignando el valor

falso a la variable booleana, el otro asignándole el valor cierto.

En la ejecución simbólica de las sentencias alternativas y repetitivas se evaluan primero las guardias. Para cada guardia que evalue un valor cierto se genera un nudo sucesor. La evaluación de las guardias de cada comando con guardia se realiza como sigue:

(a) Se sustituyen las variables de la guardia por sus valores simbólicos.

(b) Si la lista de predicados implica que la guardia sea falsa, la lista de sentencias del comando con guardia no se ejecutará nunca a partir del nudo que se está considerando. En este caso es innecesario generar un sucesor que considere la ejecución del comando con guardia.

(c) Si la lista de predicados implica un valor cierto de la guardia, se genera un nudo sucesor idéntico al que se está considerando, pero con el punto de control conteniendo la etiqueta de la primera sentencia de la lista de sentencia del comando.

(d) Si la lista de predicados no implica que la guardia sea cierta ni que sea falsa, se crea un sucesor que contemple la posibilidad de que la guardia evalue a cierto. El nuevo nudo tiene el mismo vector de estado que su antecesor; su lista de predicados se obtiene añadiendo la guardia a la lista de predicados de éste y el control se avanza a la lista de sentencias del comando con guardia.

La primera "guardia" que debe considerarse es la que conduce al fin de la ejecución de la sentencia, es decir, la negación de la reunión booleana de las guardias de cada comando con guardia. Si la lista de predicados no im-

plica que esta guardia sea falsa se genera un nudo sucesor añadiendo a la lista de predicados la condición que hace posible este camino. En el caso de la sentencia alternativa este nuevo nudo describe las situaciones en que la descripción LDP es errónea (estas situaciones conducen a la aborción) . Si se trata de una sentencia repetitiva el punto de control es la etiqueta de la siguiente sentencia que ha de ejecutarse simbólicamente y el nuevo nudo no es más que la descripción de las situaciones en que termina la ejecución de la sentencia repetitiva.

Se denominarán puntos de paro a los lugares de las descripciones en los que insertan aserciones. Estas se establecen en forma de una relación de simulación /2/. Cada componente de la relación de simulación consiste en un punto de paro y la aserción correspondiente.

El procedimiento de verificación propuesto consiste en generar para cada punto de paro el árbol de la prueba que se obtiene ejecutando simbólicamente la descripción LDP desde el punto de paro inicial hasta que se encuentra el punto de paro siguiente. En la raíz del árbol el punto de control es la etiqueta del punto de paro inicial, el vector de estado contiene símbolos arbitrarios y la lista de predicados restringe los valores simbólicos iniciales de modo que se cumpla la aserción especificada en la relación de simulación. Las hojas del árbol serán puntos de paro y en ellos ha de probarse que la aserción es cierta.

Se expone a continuación un ejemplo que clarifica el procedimiento a seguir. Se pretende verificar la siguiente descripción LDP del algoritmo de Euclides

```
1:x:=m;  y:=m;  2:*[x>y→3:x:=x-y [] y>x→4:y:=y-x];5:
```

que pretende dejar en la variable x el máximo común divisor de dos enteros positivos m y n . Insertaremos tres aserciones en la descripción, la aserción inicial $(m>0) \wedge (n>0)$, que restringe los valores iniciales de las variables m y n; la aserción final $x = (m,n)$ que expresa el resultado esperado, y la aserción inductiva $(m,n) = (x,y) \wedge (x>0) \wedge (y>0)$, situada al principio del bucle.

La figura 10 muestra el árbol de prueba que comienza en el punto de paro 1 y acaba en el 2. El nudo inicial tiene el punto de control en 1, el vector de estado $(m=\alpha , n=\beta, x=\gamma, y=\delta)$ y la lista de predicados $\alpha>0, \beta>0$. En el punto final ha de probarse que

$$(\alpha>0)\wedge(\beta>0)\supset((\alpha,\beta)=(\alpha,\beta))\wedge(\alpha>0)\wedge(\beta>0),$$

predicado trivialmente cierto.

A partir del punto de paro 2 se construye el árbol de prueba de la figura 10.El punto de control de la raíz es 2, el vector de estado $(m=\alpha, n=\beta, x=\gamma, y=\delta)$ y la lista de predicados $(\alpha,\beta)=(\gamma,\delta)\wedge(\gamma>0)\wedge(\delta>0)$. En el punto de paro final debe probarse el predicado.

$$((\alpha,\beta)=(\gamma,\delta))\wedge(\gamma=\delta)\supset\gamma=(\alpha,\beta)$$

En los puntos 2_1 y 2_2 la prueba toma la forma de los predicados de la figura 10.

Los tres predicados pueden probarse ciertos haciendo uso de los axiomas que definen el máximo común divisor de dos enteros positivos,

$$(a,a) = a$$
$$(a,b) = (b,a)$$
$$(a,b) = (a+b,b)$$

La descripción dada del algoritmo de Euclides es por tanto parcialmente correcta respecto a las aserciones consideradas.

ALGORITMO

1:x:=m;y:=n;2:*[x>y →3:x:=x-y[]y>x →4:y:=y-x];5:

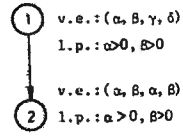
RELACION DE SIMULACION

1; (m>0)∧(n>0)

2; ((m,n)=(x,y))∧(x>0)∧(y>0)

3; x=(m,n)

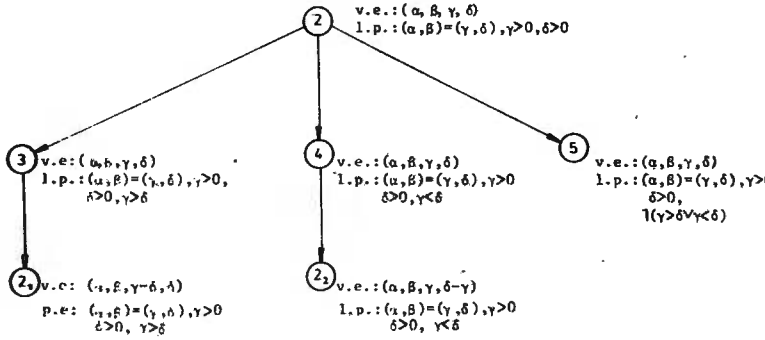
ARBOL DE PRUEBA DESDE 1



PRUEBA EN EL PUNTO DE PARO 2

$(\alpha>0)\wedge(\beta>0)\supset((\alpha,\beta)=(\alpha,\beta))\wedge(\alpha>0)\wedge(\beta>0)$

ARBOL DE PRUEBA DESDE 2



PRUEBA EN LOS PUNTOS DE PARO 5, 2_1, 2_2

En 5 $((\alpha,\beta)=(\gamma,\delta))\wedge(\gamma>0)\wedge(\delta>0)\wedge(\neg(\gamma>\delta\vee\gamma<\delta))\supset\gamma=(\alpha,\beta)$

En 2_1 $((\alpha,\beta)=(\gamma,\delta))\wedge(\gamma>0)\wedge(\delta>0)\wedge(\gamma>\delta)\supset((\alpha,\beta)=(\gamma-\delta,\delta))\wedge(\gamma-\delta>0)\wedge(\delta>0)$

En 2_2 $((\alpha,\beta)=(\gamma,\delta))\wedge(\gamma>0)\wedge(\delta>0)\wedge(\gamma<\delta)\supset((\alpha,\beta)=(\gamma,\delta-\gamma))\wedge(\gamma>0)\wedge(\delta-\gamma>0)$

Figura 10: Verificación de la descripción LDP del algoritmo de Euclides

3.5.2. Verificación de procesos concurrentes descritos en LDP.

El método de verificación de descripciones LDP propuesto en el apartado anterior puede generalizarse de

un modo simple para verificar descripciones de varios procesos concurrentes. En este caso las aserciones no se asocian a puntos concretos de cada proceso sino que hacen referencia a un estado global de la descripción, es decir, las aserciones deben probarse cuando cada uno de los procesos se encuentra en un punto determinado. Los puntos de paro de la relación de simulación hacen referencia, por tanto, a un conjunto de etiquetas, una de cada proceso.

Establecida la relación de simulación, la verificación consiste en construir un árbol de prueba para cada punto de paro que considere todos los posibles caminos de ejecución. En la generación del árbol de prueba puede considerarse, como en la validación , un tiempo variable, fijo o nulo para la ejecución de las distintas transiciones. En lo que sigue se considerará la verificación con tiempo de ejecución de las transiciones fijo. La verificación sin considerar el tiempo de ejecución es un caso particular en el que el tiempo de ejecución de todas las transiciones se considera nulo. La generalización al caso de tiempo de ejecución variable es inmediata.

En la búsqueda de los caminos de ejecución del árbol de prueba se utilizará un algoritmo semejante al propuesto para la validación de protocolos. Allí se construía un árbol, denominado de ejecución temporal, cuyos nudos están formados por el estado del sistema y perturbaciones parciales dinámicas, V_i. El árbol de ejecución temporal determina todos los caminos de ejecución que puede seguir el protocolo, sin tomar en cuenta la estructura de datos. En la generación del árbol de prueba han de observarse las mismas reglas que en la del árbol de ejecución temporal , más las que se derivan de tomar en

cuenta la estructura de datos. A continuación se exponen las características del árbol de prueba y el procedimiento a seguir para generarlo.

Los nudos del árbol de prueba están formados por cuatro elementos

(a) Punto de control: conjunto de etiquetas que caracteriza la posición alcanzada en la ejecución simbólica de la descripción.

(b) Perturbaciones parciales dinámicas: son los conjuntos de transiciones parciales posibles de cada proceso y el tiempo que necesitan para realizarse.

(c) Vector de estado: formado por los valores simbólicos de las variables (expresiones formadas por los operadores del LDP, constantes y símbolos iniciales).

(d) Lista de predicados: conjunto de condiciones sobre los símbolos iniciales para que el control pueda llegar al nudo.

Los nudos sucesores de uno dado, n, se obtienen siguiendo los pasos que a continuación se indican

(a) Buscar en V_i^n la transición que requiere un tiempo menor para ejecutarse, t_{min}.

(b) Buscar los estados sucesores (puntos de control) de $\bar{e}$ al cabo del tiempo t_{min}. Para ello

(1) Formar los conjuntos

$W_i = \{\bar{f} \mid (\bar{f}, t_{min}) \in V_i^n\}$

Los W_i contienen las transiciones parciales del estado $\bar{e}$ que puede ejecutar el proceso i en el tiempo t_{min}.

(2) Tomando un único elemento de cada W_i formar las transiciones compatibles de los procesos cuya ejecución conduciría a los nuevos estados $\bar{g} \in G$, suce

sores del $\bar{e}$. De estas transiciones sólo podrán ejecutarse aquellas que sean consistentes con el vector de estado y la lista de predicados. Los estados a que se llega ejecutando estas transiciones son $\bar{h} \in H \subset G$.

(c) Los nudos sucesores del n quedan caracterizados por

(1) Punto de control: $\bar{h} \in H$

(2) Perturbaciones parciales dinámicas V_i^m, formadas a partir de las V_i^n del nudo, haciendo

(i) Para todo i tal que $e_i \neq h_i$, hacer $V_i^m = \{\phi\}$

(ii) Para todo i tal que $e_i = h_i$, hacer $V_i^m = V_i^n$ y decrementar todos los tiempos de V_i^m en t_{min}.

(iii) Para todo i, añadir a V_i^n los elementos de $U_i(\bar{h})$ que no correspondan a estados siguientes del sistema ya contenidos en algún elemento de V_i^m.

(3) Vector de estado: se obtiene ejecutando simbólicamente los procesos que intervienen en la transición de $\bar{e}$ a $\bar{h}$ considerada.

(4) Lista de predicados: obtenida como en (3).

La ejecución simbólica de una transición conduce al sistema del estado $\bar{e}$ al $\bar{h}$ comporta la ejecución simbólica de cada proceso desde el punto e_i al h_i. Esta se realiza tal y como se ha indicado en el apartado anterior, con la salvedad de las sentencias de asignación externa. La ejecución simbólica de una sentencia de salida en un proceso se realiza "simultáneamente" con la ejecución simbólica de una sentencia de entrada en otro proceso. El efecto de esta ejecución es la asignación de la expresión simbólica de la sentencia de salida a la variable de la entrada.

Sabiendo cómo construir un árbol de prueba el algoritmo de verificación es inmediato. Dado un protocolo descrito en LDP y una relación de simulación que refleje fielmente las propiedades que se desean probar, se genera para cada punto de paro un árbol de prueba. La raíz del árbol se inicializa de forma que

(a) Punto de control: punto de paro inicial ($\bar{e}_0$).

(b) Perturbaciones parciales dinámicas: $V_i = U_i(\bar{e}_0)$.

(c) Vector de estado: valores simbólicos iniciales de todas las variables de la descripción.

(d) Lista de predicados: condiciones que han de cumplir los valores iniciales para que se cumpla la aserción que la relación de simulación asocia al punto de paro $\bar{e}_0$.

La verificación se lleva a cabo mediante el algoritmo siguiente:

(a) Definir una sucesión de conjuntos A(i) de nudos del árbol de prueba. El conjunto A(0) contiene el nudo inicial o raíz. Hacer i = 0.

(b) Tomar un nudo de A(i) cuyo conjunto de nudos sucesores no haya sido aún determinado e ir a (c). De no existir tal nudo mirar si A(i+1) = {ϕ}. En este caso la verificación a partir del punto de paro que se está considerando ha concluído; en caso contrario, hacer i:=i+1 y volver al principio de (b).

(c) Determinar los nudos sucesores del nudo considerado y

  (1) Si algún nudo sucesor es un punto de paro especificado en la relación de simulación, intentar probar la aserción asociada.

  (2) Si algún nudo sucesor se encuentra ya en algún A(j), para j=1,...,i+1, no considerarlo más.

(3) Añadir a A(i+1) el resto de los nudos sucesores.

(4) Ir a (b).

El algoritmo de verificación propuesto permite detectar diversos errores en la descripción LDP de un protocolo, entre ellos:

(a) Si en el apartado (c) el nudo considerado carece de nudos sucesores se ha localizado un estado terminal o un bloqueo.

(b) Si en el apartado (c) punto (1) se prueba que la aserción es falsa, el protocolo no funcionará correctamente respecto a la relación de simulación dada.

(c) Si en el apartado (c) punto (2) el nudo considerado es igual a un antecesor suyo, se ha detectado un bucle.
Del exámen de la descripción se determina si se trata de un bucle normal o si por el contrario es un bloqueo temporal.

A continuación se ilustra el funcionamiento del algoritmo de verificación mediante un ejemplo. La figura 11 es la descripción LDP del protocolo de bit alternante en la que se ha supuesto que el medio puede introducir errores, pero no pérdidas de la información. En esta figura se muestra asimismo, la relación de simulación mediante la cual se expresan las propiedades que se desean verificar. En este ejemplo outdata (a) = indadata (b) significará la igualdad de los elementos de los vectores outdata e indata de índice menor o igual que a y b, respectivamente. La primera componente de la relación de simulación establece la igualdad del elemento 0 de estos dos vectores antes de la inicialización. La segunda componente expresa que cuando el proceso emisor está en situación de tomar un nuevo mensaje de indata para transmitir y el proceso receptor espera la recepción de un mensaje, de-

```
[S::SENDER || R::RECEIVER]

SENDER
1:ea:=ack:=seq:=false;pt:=1;
*[true →
  2:[¬ea∧ack=seq →3:datas:=indata(pt);pt:=pt+1;seq:=¬seq
    [] ea∨ack≠seq →skip];4:R:=(datas,seq);5:ack:=R;6:ea:=#
 ]


RECEIVER
1:exp:=false;ps:=1
*[true →
  2:(datar,seqr):=S;em:=#;
    4:[¬em∧seqr≠exp →5:outdata(ps):=datar;ps:=ps+1;exp:=¬exp
      [] em∨seqr=exp →skip];6:S:=exp
 ]


                RELACION DE SIMULACION
     1-1; outdata(0)=indata(0)
     3-2; seq=exp∧ps=pt∧outdata(ps-1)=indata(pt-1)
```

Figura 11: Descripción LDP del protocolo de bit alternante y relación de simulación para la verificación.

ben cumplirse las condiciones seq=exp, ps=pt y outdata (ps-1) = indata(pt-1).

La figura 12 recoge la descripción gráfica del protocolo a partir de la cual se construyen fácilmente los dos árboles de prueba que requiere la verificación, uno para cada punto de paro.

El primer árbol de prueba se encuentra en la figura 13.

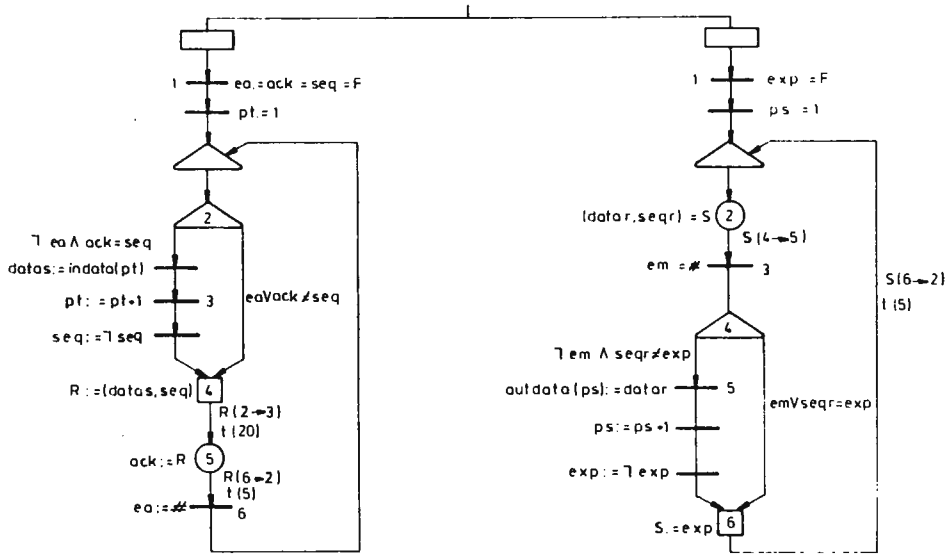


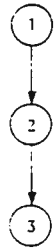
Figura 12: Descripción gráfica LDP del protocolo de la figura 3

La rafz (nudo 1) fija el punto de control en la primera sentencia de cada proceso y establece el tiempo de ejecución 0 para las transiciones de ambos. Su vector de estado adopta valores simbólicos arbitrarios y la lista de predicados contiene un único elemento que restringe los valores simbólicos x_0 e y_0 para que se cumpla la aserción asociada a este punto de paro (los valores simbólicos de outdata (m) e indata (n) son respectivamente, y_m y x_n). La generación del árbol termina en el nudo 3, al ser su punto de control el 3-2, punto de paro especificado en la relación de simulación. En este punto debe probarse la aserción.

$$seq = exp \wedge pt = ps \wedge outdata(ps-1) = indata(pt-1)$$

que en este caso se convierte en el predicado

ARBOL



DESCRIPCION DE LOS NUDOS

Nudo	Estado	ea	ack	seq	pt	datas	datar	seqr	em	exp	ps
1	1 1 0 0	a	b	c	d	e	f	g	h	i	j
2	2 2 0 -	F	F	F	1	e	f	g	h	F	1
3	3 2 0 -	F	F	F	1	e	f	g	h	F	1

LISTA DE PREDICADOS IGUAL EN LOS TRES NUDOS

$$1=y_0=x_0$$

PRUEBA

$$y_0=x_0 \supset F=F \wedge 1=1 \wedge y_0=x_0$$

Figura13: Arbol de prueba y condición de verificación correspondiente al primer punto de paro de la relación de simulación.

$$y_0 = x_0 \supset F = F \wedge 1 = 1 \wedge y_0 = x_0,$$

trivialmente cierto.

Las figuras 14 y 15 muestran el árbol de prueba correspondiente a la segunda componente de la relación de simulación. El punto de control de la raíz es el 3-2, en el que

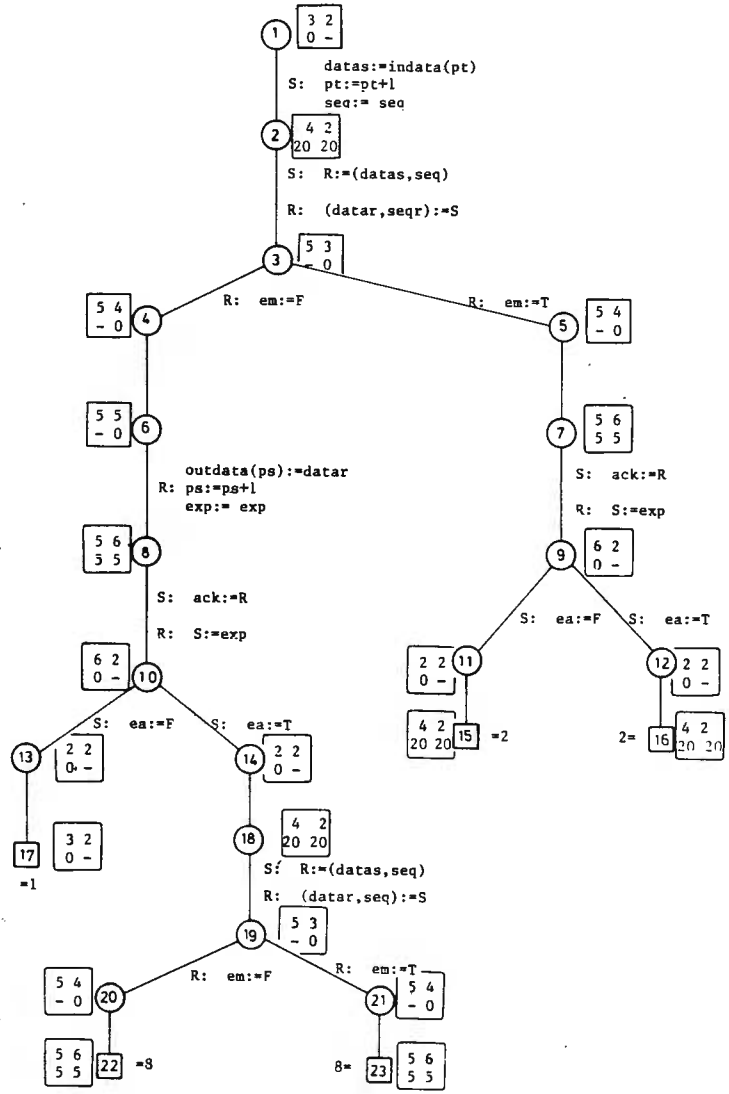


Figura 14 :Arbol de prueba del protocolo correspondiente a la segunda componente de la relación de simulación.

únicamente el proceso S puede evolucionar. Las variables del vector de estado toman valores simbólicos y la lista de predicados impone a éstos las condiciones que hacen que se cumpla la aserción correspondiente a este punto. Para facilitar el seguimiento de la construcción de este árbol se ha escrito junto a los nudos de la figura 14 el punto de control del nudo y el tiempo necesario para ejecutar la siguiente transición y, junto a éstas, las sentencias que ejecuta simbólicamente cada proceso.

En la figura 14 se observa que el subárbol que comienza en el nudo 5, correspondiente a la recepción errónea de un mensaje, conduce a los estados 15 ó 16. Ambos estados son iguales al estado 2, por lo que esta situación conduce a la retransmisión del mensaje. El subárbol que se indica en el nudo 4 corresponde a la recepción correcta del mensaje. En la transición del nudo 6 al 8 el mensaje se pasa al vector outdata y en de 8 a 10 se envía el ack. Si el ack llega correctamente se llega al nudo 17 que es un punto de paro. En él ha de probarse la aserción (seq=exp)∧ (pt=ps)∧ (outdata(ps-1)= indata(pt-1)) que se traduce en el predicado siguiente

$$(c=i)\wedge(d=j)\wedge(y_{j-1}=x_{d-1})\supset(c=i)\wedge(d+1=j+1)\wedge(y_j=x_d),$$

que, dado que el elemento j de outdata se ha hecho igual al d de indata, es siempre cierto. El subárbol que comienza en el nudo 14 corresponde a la recepción incorrecta del ack.Tras retransmitir el mensaje se llega al nudo 22 ó 23 que son idénticos al 8, a partir del cual se retransmite el ack.

De la verificación del protocolo de bit alternante se concluye que su comportamiento es cíclico, carece de bloqueos y tiene dos bloqueos temporales. Efectivamente, si la línea que transmite los mensajes de S a R produce siem

pre errores, el protocolo sigue los ciclos 2-3-5-7-9-10-2 ó 2-3-5-7-4-12-2. Igualmente, si la línea que transmite el ack de R a S introduce sistemáticamente errores, el protocolo entra en los ciclos 8-10-14-18-19-20-8, 8-10

Nudo	Estado	ea	ack	seq	pt	datas	datar	seqr	em	exp	pa	
1	3 2 0 -	a	b	c	d	e	f	g	h	i	j	
2	4 2 20 20	a	b	lc	d+1	x_d	f	g	h	i	j	
3	5 3 - 0	a	b	lc	d+1	x_d	x_d	lc	h	i	j	
4	5 4 - 0	a	b	lc	d+1	x_d	x_d	lc	F	i	j	
5	5 4 - 0	a	b	lc	d+1	x_d	x_d	lc	T	i	j	
6	5 5 - 0	a	b	lc	d+1	x_d	x_d	lc	F	i	j	
7	5 6 5 5	a	b	lc	d+1	x_d	x_d	lc	T	i	j	
8	5 6 5 5	a	b	lc	d+1	x_d	x_d	lc	F	li	j+1	$y_j=x_d$
9	6 2 0 -	a	i	lc	d+1	x_d	x_d	lc	T	i	j	
10	6 2 0 -	a	li	lc	d+1	x_d	x_d	lc	F	li	j+1	
11	2 2 0 -	F	i	lc	d+1	x_d	x_d	lc	T	i	j	
12	2 2 0 -	T	i	lc	d+1	x_d	x_d	lc	T	i	j	
13	2 2 0 -	F	li	lc	d+1	x_d	x_d	lc	F	li	j+1	
14	2 2 0 -	T	li	lc	d+1	x_d	x_d	lc	T	li	j+1	
15	4 2 20 20	F	i	lc	d+1	x_d	x_d	c	T	i	j	
16	4 2 20 20	T	i	lc	d+1	x_d	x_d	c	T	i	j	
17	3 2 0 -	F	li	lc	d+1	x_d	x_d	lc	F	li	j+1	
18	4 2 20 20	T	li	lc	d+1	x_d	x_d	lc	T	li	j+1	
19	5 3 - 0	T	li	lc	d+1	x_d	x_d	lc	T	li	j+1	
20	5 4 - 0	T	li	lc	d+1	x_d	x_d	lc	F	li	j+1	
21	5 4 - 0	T	li	lc	d+1	x_d	x_d	lc	T	li	j+1	
22	5 6 5 5	T	li	lc	d+1	x_d	x_d	lc	F	li	j+1	
23	5 6 5 5	t	li	lc	d+1	x_d	x_d	lc	T	li	j+1	

Figura 15 : Descripción de los nudos de la figura 6

-14-18-19-21-8. Fuera de estos casos triviales de funcionamiento incorrecto, el protocolo se comporta correctamente respecto a la relación de simulación dada, lo que garantiza que el protocolo transfiere una única copia de cada mensaje de indata a outdata y en el debido orden.

Autor	Formalismo de modelación	Técnicas de análisis	Aspectos considerados	Propiedades estudiadas	Aspectos complicados	Trabajo adicional necesario
DANTHINE BREMER	Modelo local de estados más algoritmo	Caminos compatibles	Control	Bloqueos	Definición del protocolo	Función de transferencia, bloqueo temporal, asimetría, tiempo
BRAND JOYNER	Algoritmos	Ejecución simbólica, aserciones	Función de transferencia	Bloqueos, bloqueos temporales, f. transf	Definición de aserciones	Inicializaciones, medios complejos, tiempo
RUDIN WEST ZAFIROPULO	Modelo local de estados	Caminos compatibles, estado global	Control	Bloqueos, compleción	Definición del protocolo	Medios complejos, bloqueo temporal, tiempo, f. transf
BOCHMANN	Modelo local de estados con variables y algoritmos	Estado global aserciones y estados adjuntos	Ambos	Bloqueos, bloqueo temporal, vida	Definición protocolo, prueba aserciones	Automatización, medios complejos, tiempo
HAJEK	Algoritmos	Estado global	Ambos	Bloqueos temporales, termina ción	Definición algoritmos, aserciones	Medios complejos, control nª estados, tiempo
HARANGOZO	Gramáticas formales		Ambos		Definición gramática	Verificación, tiempo
GOUDA	Modelo local de estados	Estado global, caminos compatibles	Control	Bloqueos, acotación	Definición protocolo, prueba	Medios y protocolos complejos, automatizac.,tiempo
VALIDACION PROPUESTA EN /10/	Algoritmos en LDP	Estado global, análisis de alcanzabilidad	Control	Bloqueos, bloqueo temporal, comportamiento cíclico,tiempo		Automatización
VERIFICACION PROPUESTA	Algoritmos en LDP	Ejecución simbólica, aserciones	Ambos	Bloqueos, bloqueo temporal, comp. cíclico, tiempo, f. de transferencia	Definición y prueba de aserciones	Automatización

Figura 16: Comparación de las técnicas de verificación más significativas (referenciadas en /9/)

4. CONCLUSIONES

Con el LDP puede especificarse el funcionamiento de los procesos que integran un protocolo y su entorno. La ejecución concurrente de los procesos y el no determinismo se expresan mediante la sentencia paralela y la alternativa. El lenguaje permite describir la estructura de datos de los procesos así como su lógica interna, medios de comunicación, estructuras jerárquicas, temporizaciones, etc. La comunicación entre procesos se realiza a través de mensajes mediante la ejecución simultánea de una sentencia de salida en un proceso y una de entrada en otro. El LDP permite especificar sencillamente otras formas de comunicación como son la comunicación a través de memoria común, monitores, etc. El aspecto del LDP es el de un lenguaje de alto nivel por lo que las conocidas técnicas de programación estructurada y modular pueden utilizarse en la descripción de protocolos. En particular, el LDP permite realizar corutinas, subrutinas, procedimientos, etc.

En la validación no temporal se generan todas las posibles secuencias de transiciones sin tener en cuenta la velocidad de ejecución de los procesos por los que en el árbol de ejecuciones aparecen más caminos de los que el protocolo real puede seguir, al considerarse todas las formas en que los procesos pueden entrelazar su ejecución tomando como cero el tiempo de ejecución de las transiciones.

La validación temporal resulta más simple que la no temporal. El árbol de ejecución contiene todos los caminos de ejecución que el protocolo puede seguir en su funcionamiento, sin considerar las limitaciones impuestas por la estructura de datos de los procesos, pero tomando en cuenta su tiempo de ejecución.

La aplicación manual de uno u otro algoritmo de validación

es extraordinariamente sencilla. Sin embargo, para protocolos de cierta complejidad la construcción manual del árbol de ejecuciones se hace inabordable. Es interesante considerar la posibilidad de automatizar el proceso de validación mediante un programa que partiendo de la descripción LDP de un protocolo genere su árbol de ejecución.

La limitación fundamental de este tipo de programas se encuentra en el número de estados diferentes que pueden aparecer en el árbol de ejecución. Por un lado, el almacenamiento en memoria de los estados puede encontrar limitaciones de capacidad, por otro, al generar un nudo sucesor es necesario comprobar si se trata de un nuevo nudo o si es un nudo ya existente. Esta comprobación resulta tanto más lenta cuanto mayor sea el número de nudos del árbol. Este número depende fundamentalmente del grado de acoplamiento (interacciones) entre los distintos procesos que integran el protocolo.

De la validación de la estructura gráfica de un protocolo puede concluirse la validación de la descripción LDP. En efecto, si el sistema se comporta correctamente cuando en las sentencias alternativas se elige aleatoriamente la lista de sentencias que ha de ejecutarse, también lo hará si en la selección se considera el valor de las guardias. Contrariamente, los errores que se encuentran en la validación de la estructura no siempre serán errores de diseño del protocolo. De la descripción LDP y del camino de ejecución que conduce al error se determina de forma inmediata si el protocolo puede realmente llegar en su funcionamiento al estado erróneo.

El algoritmo de verificación propuesto permite generar sistemáticamente las condiciones de verificación de una descripción LDP respecto a una relación de simulación dada. La prueba de estas condiciones pertenece al cálculo de predicados de primer orden, que carece de la propiedad de ser decidible (no existe ningún procedimiento de decisión que permita determinar si un predicado de primer orden es o no válido). Esto supone que

puede resultar imposible demostrar si una condición es válida o no. Por tanto, el método puede fallar de dos maneras en la verificación: no siendo capaz de probar la corrección de una descripción correcta o la incorrección de una descripción incorrecta. Si el algoritmo es capaz de decidir si una descripción LDP determinada es correcta (o incorrecta), puede afirmarse sin lugar a dudas que la descripción es correcta (o incorrecta) respecto a la relación de simulación considerada. Esta limitación es común a todos los sistemas de verificación de programas basados en el cálculo de predicados de primer orden.

Una limitación adicional es la posibilidad de que el algoritmo no termine. Efectivamente, si las variables de una descripción LDP pueden tomar valores arbitrariamente grandes, resulta imposible garantizar que la verificación termine.

Estas limitaciones teóricas impiden el poder afirmar que el algoritmo de verificación permita siempre probar la corrección o incorrección de procesos concurrentes descritos en LDP. Sin embargo, si puede garantizarse la limitación en el crecimiento del número de estados del árbol de prueba y la complejidad de las condiciones de verificación son moderadas, la prueba podrá llevarse a cabo. Tal como se ha indicado, estas condiciones suelen darse en la verificación de protocolos de comunicación y en muchos otros procesos concurrentes de características análogas.

La verificación manual de descripciones de cierta complejidad resulta inabordable. Puede pensarse en la construcción de un sistema de verificación automático, o semiautomático (interactivo), que genere las condiciones de verificación y las pruebe. La realización de este tipo de programas constituye un tema permanente de investigación. Las dos limitaciones fundamentales de estos sistemas automáticos son la dificultad de encontrar aserciones inductivas adecuadas y la falta de potencia de los probadores de teoremas que, en muchos casos, son incapa

ces de demostrar las condiciones de verificación generadas /54/

Resulta interesante comparar el sistema de descripción y verificación desarrollado con otros métodos existentes. La figura 16 recoge, en términos similares a los de /71/ las características de las técnicas más importantes.

De la observación de la figura puede concluirse que el sistema propuesto en este trabajo se compara favorablemente con las técnicas más significativas que existen en la actualidad.

5. AGRADECIMIENTO.

La realización de este trabajo ha sido posible gracias a una Beca de Estudios Científicos de Ingeniería concedida por la Fundación Juan March al autor, quien desea por ello expresar aquí su reconocimiento.

6. REFERENCIAS.

/1/ BARTLETT,K.A. et alt.: "*A Note on Reliable Full-Duplex Transmission over Half-Duplex Links*". Comm. ACM, v.12, n.5., May 1969, pp. 260-261.

/2/ BIRMAN, A., JOYNER, W.H.: "*A Problem Reduction Approach to Proving Simulation Between Programs*". IEEE Trans. on Soft Eng., v.SE-2, nº2, June 1976, pp. 87-96.

/3/ BOCHMANN, G.V., "*Logical verification and implementation of protocols*" Proc. of the 4th Data Communication Symposium, Québec, 1975.

/4/ BOCHMANN, G.V. "*Communication Protocols and Error Recovery Procedures*". Proc. of the ACM Sigcomm/Sigops Interprocess Communications Workshop, Santa Mónica, Ca., 1975.

/5/ BOCHMANN, G.V. "*Finite State Description of Communication Protocols*". Departement d'Informatique, Univ. de Montreal Pu. ## 236, julio 1976.

/6/ BOCHMANN, G.V., CHUNG, R.J., "*A Formalized Specification of HDLC Classes of Procedures*". University of Montreal, Tr nº 265, 1977.

/7/ BOCHMANN, G.V., GECSEI, J. "*A Unified Method for the Specification and Verification of Protocols*". Proc. of IFIP'77 Toronto, 1977.

/8/ BOCHMANN, G.V. "*Finite State Description of Communication Protocols*" Proc. Symposium on Computer Communication Protocols, Liège, 1978.

/9/ BRAND, D., JOYNER, W.H.: "*Verification of Protocols Using Symbolic Execution*". Proc. Symp. on Computer Network Protocols, Liège 1978, pp. F2-1/F2-7.

/10/ BREMER, J. "*Representation Axiomatique d'un Protocol. Description du Programme Reduction*". S.A.R.T. 76/19/10, Univ. de Liège, Septiembre 1976,

/11/ BREMER, J. "*Verification de la Logique d'un Protocol. Description du Programme Verify*". S.A.R.T. 77/03/10, Univ. de Liège, Enero 1977.

/12/ CAMPBELL, R.H., HABERMANN, A.N. "*Specification of Process Synchronization by Path Expressions*" Proc. of the International Symposium on Operating Systems, Rocquencourt, 1974.

/13/ CCITT Libro Naranja, vol. VII-2, Redes Públicas de Datos, Ginebra, 1977.

/14/ CLEVELAND, J.C., UZGALIS, R.C. "*Grammars for Programming Languages*". Elsevier North-Holland, Amsterdam, 1977.

/15/ DAHL, O.J. et al.: "*SIMULA 67, Common Base Language*". Norwegian Computing Center. Forksningveien, Oslo, 1967.

/16/ DAVIS, D.W., BARBER, D.L.A.: "*Communication Networks for Computers*". Wiley & Sons, Inc. London. 1973.

/17/ DANTHINE, A.S., BREMER, J. "*Communication Protocols in a Network Context*". Proc. of the ACM Sigcom/Sigops Interprocess Communications Workshop, Santa Mónica, Ca., 1975.

/18/ DANTHINE, A., BREMER, J. "*An Axiomatic Description of the Transport Protocol of Cyclades*". Professional Conference on Computer Networks and Teleprocessing, Aachen, 1976.

/19/ DANTHINE, A.S. "*Petri Nets for Protocol Modelling and Verification*". Proc. of the Computer Networks and Teleprocessing Symposium, Budapest, Octubre, 1977.

/20/ DANTHINE,A.S., BREMER, J.: "*Modelling and Verification of End-to-End Transport Protocols*". Proc. Symp. on Computer Network Protocols, Liège 1978. pp. F5-1/F5-12.

/21/ DAYTON, J.D. "*A Bibliography on the Formal Specification and Verification of Computer Network Protocols*". Proc. Symposium on Computer Networks Protocols, Liège, 1978.

/22/ DEUTSCH, P. : "*An Interactive Program Verifier*". Ph.D dissertation, Univ. California, Berkeley, 1973.

/23/ DIJKSTRA, E.W. : "*Guarded Command, Nondeterminacy and Formal Derivation of Programs*". Comm. ACM, v.18, n.8. Aug., 1975, pp. 453-457.

/24/ DOLL, D.R. : "*Data Communications*", John Wiley & Sons,1978.

/25/ DONNAN, R.A., KERSEY, J.R.: "*SDLC: A Perspective*". IBM Systems Journal, v.15, n.1, 1976, pp. 140-162.

/26/ FLOYD, R.W.: "*Assigning Meaning to Programs*". Proc. Symp. in Aplied Mathematics, v.19, 1967, pp. 19-32.

/27/ GARCIA HOFFMANN, M. "*Aportación al estudio de la descripción, validación y verificación de protocolos de comunicación*". Tesis Doctoral, E.T.S. Eng. Industrials de Barcelona.

/28/ GOOD, D.I., et al.: "*An Interactive Program Verification Method*". IEEE Trans. on Soft. Eng., v. SE-1, Mar.1975, pp. 59-67.

/29/ GOUDA, M.G., MANNING, E.G. "*On the Modelling, Analysis and Design of Protocols. A Special Class of Software Structures*" Proc. of the 2nd International Conference on Software Engineering, 1976.

/30/ GOUDA, M.G, MANNING, E.G. "*Protocol Machines: a Concise Formal Model and its Automatic Implementation*", Proc. of the 3rd. Int. Conf. on Comp. Communication, Toronto, 1976.

/31/ GOUDA, M.G. "*Protocol Machines: towards a Logical Theory of Communications Protocols*". Honeywell Systems Research Center, 2600 Ridgway Parkway Minneapolis, Minnesota 55413, Nov. 1977.

/32/ GREIF, I.: "*A Language for Formal Problem Specification*". Comm. ACM, v.20, n.12, Dec. 1977, pp. 931-935.

/33/ HABERMANN, A.N.: "*Synchronization of Communicating Processes*". Comm. ACM, v.15, n.3, March 1972, pp. 171-176.

/34/ HABERMANN,A.N.: "*Path Expressions*". TR Computer Science Dep. Carnegie Mellon Univ., 1975.

/35/ HAJEK, J.: "*Automatically Verified Data Transfer Protocols*". Proc. of ICCC 78, 1978, pp. 749-756.

/36/ HANSEN, P.B. : "*The Programming Language Concurrent Pascal*". IEEE Trans. on Soft. Eng., v.SE-1, n.2, June 1975, pp. 199-207.

/37/ HANSEN, P.B. : "*The Architecture of Concurrent Programs*". Prentice-Hall, 1977.

/38/ HANSEN, P.B.: "*Distributed Processes: A Concurrent Programming Concept*". Comm. ACM, v.21, n.11, Nov. 1978, pp. 934-941

/39/ HANTLER, S.L., KING, J.C.: "*An Introduction to Proving the Correctness of Programs*". ACM Computing Surveys, v.8, nº3, Sept. 1976, pp. 331-353.

/40/ HARANGOZO, U. "*An Approach to Describing a Data Link Level Protocol with a Formal Language*". Proc. of the 5th Data Communications Symposium, Snowbird, 1977.

/41/ HARANGOZO, J. "*Protocol Definition with Formal Grammars*". Proc. Symposium on Computer Communication Protocols, Liège, 1978.

/42/ Von HENKE, F.W., LUCKHAM, D.C.: "*A Methodology for Verifying Programs*" Proc. Int. Joint Conf. Reliable Software , Apr. 1975, pp. 156-164.

/43/ HOARE, C.A.R.: "*Communicating Sequential Processes*". Comm. ACM, v.21, n.8, Aug. 1978, pp. 666-677.

/44/ ISO : "*High Level Data Link Control Procedures*". ISO/TC/ /SGG, N-1005, 1975.

/45/ KELLER, R.M. "*Formal verification of Parallel Programs*". Comm. of the ACM, v.19, n.7, 1976, pp.371-384.

/46/ KING, J.C. : "*A Program Verifier*". Proc. IFIP, Aug. 1971, pp. 235-249.

/47/ KING, J.C. : "*Symbolic Execution and Program Testing*". Comm. ACM, v.19, n.7, July 1976, pp. 385-394.

/48/ KLEINROCK, L.: "*Queueing Systems: Computer Application*". John Wiley & Sons, 1976.

/49/ LAMPORT, L.: "*Time, Clock and the Ordering of Events in a Distributed System*". Comm. ACM, v.21, n.7, July 1978, pp. 558-565.

/50/ LAYTON, C., "*Computer Network Protocols. Their Place in European Strategy for Computer Communications*". Proc. of the Computer Network Protocols Symposium. Liège 1978.

/51/ LE LANN, G. : "*Note sur la Synchronisation et l'Observation des Evenements dans les systèmes de Traitament Repartis*", IRIA, Project Sirius, Bigre 9, 1978.

/52/ LE LANN, G.: "*Distributed Systems. Towards a Formal Approach*". Information Processing 77, IFIP, North-Holland Pub., Co.,1977.

/53/ LE MOLI, G. "*A Theory of Colloquies*". 1st. European Workshop on Computer Networks, Arles, 1973.

/54/ MANNA,Z., WALDINGER, R.: "*The Logic of Computer Programming*". IEEE Trans. on Soft. Eng., v.SE-4, nº 3. May 1978, pp.199-229.

/55/ MARTIN, J.: "*Systems Analysis for Data Transmission*". Prentice-Hall, 1972.

/56/ MERLIN, P., FARBER, D.J. "*Recoverability of Communication Protocols*". IBM Research Report RC 5415 (23664), 1975.

/57/ MERLIN, P.M. "*A Methodology for the Design and Implementation of Communication Protocols*". IEEE Trans. on Communications, v. COM-24, n.6, 1976, pp.614-621.

/58/ MERLIN, P.M., FARBER, D.J. "*Recoverability of Communication Protocols-Implications of a Theoretical Study*". IEEE Trans. on Communication , v. COM-24, nº 9, 1976, pp.1036-1043.

/59/ MEZZALIRA, L., SCHREIBER, F.A. "*Designing Colloquies*". 1st. European Workshop on Computer Networks, Arles, 1973.

/60/ MEZZALIRA, L., SCHREIBER, F.A. "*A proposal for a Formal Description of Colloquies as a Form of Interaction of Sequential Machines*". Instituto di Electrotecnica e d'Electronica del Politêcnico di Milano, Abril, 1973.

/61/ NAUR, P.: "*Proof of Algorithms by General Snapshots*". BIT, v.6, 1966, pp.310-316.

/62/ NOE, J.D., NUTT, G.J. "*Macro E-Nets for Representation of Parallel Systems*" . IEEE Trans, on Computer, v. C-22, n.8, 1973, pp. 715-727.

/63/ NUTT, G.J. "*Evaluation Nets for Computer System Performance Analysis*". AFIPS Conf. Proc. Vol. 41, Part I, 1972, pp.275-286

/64/ PETERSON, J.L.; BREDT, T.H. "*A Comparision of Models of Parallel Computation*", Proc. IFIP Congress 74, 1974.

/65/ PETERSON, J.L., "*Petri Nets*". ACM Computing Surveys, v.9, n.3, 1977, pp. 223-252.

/66/ POSTEL, J.B. "*A Graph Theory Analysis of Computer Communications Protocol*" UCLA-ENG 7410, Univ. California, Los Angeles, 1974.

/67/ RUDIN, H. et alt. "*Automated Protocol Validation: One Chain of Development*". Proc. Symposium on Computer Communication Protocols, Liège, 1978.

/68/ SCHWARZ, M.: "*Computer Communication Network Design and Analysis*". Prentice-Hall, 1977.

/69/ STENNING, N.V. "*A Data Transfer Protocol*", Computer Networks, v.1, n.2, 1976, pp. 99-110.

/70/ SUNSHINE, C. : "*Interprocess Communication Protocols for Computer Networks*". Tech. Rep. 105 Digital Systems Lab., Stanford Univ. , 1975.

/71/ SUNSHINE, C.A.: "*Survey of Protocol Definition and Verification Techniques*". Proc. Symp. on Computer Network Protocols, Liège, 1978, pp. F1-1/F1-4.

/72/ SYMONS, F.J.W. "*Modelling and Analysis of Communication Protocols Using Numerical Petri Nets*". Ph. D. Thesis, Dep. of Electrical Engineering Science, Univ. of Essex, G.B., 1978.

/73/ WALDINGER, R.J., LEVITT, K.N.: "*Reasoning About Programs*". Artificial Intelligence, v.5, Fall 1974, pp. 235-316.

/74/ WEST, C.H., ZAFIROPULO, P. : "*Automated Validation of a Communication Protocol: The CCITT x.21 Recommendation*". IBM J. Res. Develop. , v.22, n.1, 1978, pp. 59-71.

/75/ WEST, C.H. "*General Technique for Communications Protocol Validation*". IBM Journal of Research and Development, v.22, n.4, 1978, pp. 392-404.

/76/ WEST, C.H. "*An Automated Technique of Communications Protocol Validation*", IBM Journal of Research and Development, v.26, n.8, 1978, pp. 1271-1275.

/77/ ZAFIROPULO, P. "*Protocol Validation by Duologue-Matrix*", IEEE Trans. on Communications, v. COM-26, n.8, 1978, pp. 1187-1194.

FUNDACION JUAN MARCH
SERIE UNIVERSITARIA

TITULOS PUBLICADOS **Serie Roja**

(Geología, Ciencias Agrarias, Ingeniería, Arquitectura y Urbanismo)

3 Velasco, F.:
Skarns en el batolito de Santa Olalla.

6 Alemán Vega, J.:
Flujo inestable de los polímeros fundidos.

9 Fernández-Longoria Pinazo, F.:
El fenómeno de inercia en la renovación de la estructura urbana.

13 Fernández García, M.ª P.:
Estudio geomorfológico del Macizo Central de Gredos.

15 Ruiz López, F.:
Proyecto de inversión en una empresa de energía eléctrica.

23 Bastarreche Alfaro, M.:
Un modelo simple estático.

24 Martín Sánchez, J. M.:
Moderna teoría de control: método adaptativo-predictivo.

31 Zapata Ferrer, J.:
Estudio de los transistores FET de microondas en puerta común.

33 Ordóñez Delgado, S.:
Las Bauxitas españolas como mena de aluminio.

35 Jouvé de la Barreda, N.:
Obtención de series aneuploides en variedades españolas de trigo común.

36 Alarcón Alvarez, E.:
Efectos dinámicos aleatorios en túneles y obras subterráneas.

38 Lasa Dolhagaray, J. M., y Silván López, A.:
Factores que influyen en el espigado de la remolacha azucarera.

41 Sandoval Hernández, F.:
Comunicación por fibras ópticas.

42 Pero-Sanz Elorz, J. A.:
Representación tridimensional de texturas en chapas metálicas del sistema cúbico.

43 Santiago-Alvarez, C.:
Virus de insectos: multiplicación, aislamiento y bioensayo de Baculovirus.

46 Ruiz Altisent, M.:
Propiedades físicas de las variedades de tomate para recolección mecánica.

58 Serradilla Manrique, J. M.:
Crecimiento, eficacia biológica y variabilidad genética en poblaciones de dípteros.

64 Farré Muntaner, J. R.:
Simulación cardiovascular mediante un computador híbrido.

79 Fraga González, B. M.:
Las Giberelinas. Aportaciones al estudio de su ruta biosintética.

81 Yáñez Parareda, G.:
Sobre arquitectura solar.

83 Díez Viejobueno, C.:
La Economía y la Geomatemática en prospección geoquímica.

90 Pernas Galí, F.:
Master en Planificación y Diseño de Servicios Sanitarios.

97 Joyanes Pérez, M.ª G.:
Estudio sobre el valor nutritivo de la proteína del mejillón y de su concentrado proteico.

99 Fernández Escobar, R.:
Factores que afectan a la polinización y cuajado de frutos en olivo (Olea europaea L.).

104 Oriol Marfá I Pagés, J.:
Economía de la producción de flor cortada en la Comarca de el Meresme.

109 García del Cura, M.ª A.:
Las sales sódicas, calcosódicas y magnésicas de la cuenca del Tajo.

112 García-Arenal Rodríguez, F.:
Mecanismos de defensa activa en las plantas ante los patógenos. Las Fitoalexinas en la interacción Phaseolus vulgaris-Botrytis cinerea.

114 Santos Guerra, A.:
Contribución al conocimiento de la flora y vegetación de la isla de Hierro (Islas Canarias).

120 Vendrell Saz, M.:
Propiedades ópticas de minerales absorbentes y su relación con las propiedades eléctricas.

123 Pulido Bosch, A.:
Datos hidrogeológicos sobre el borde occidental de Sierra Nevada.

137 Berga Casafont, L.:
Estudio del comportamiento reológico de la sangre humana. Aplicaciones al flujo sanguíneo.

146 Arribas Moreno, A.:
Distribución geoquímica de los elementos en trazas de los yacimientos españoles del tipo B. G. P. C.