NOTOS: Un Generador de Prototipos para Especificaciones LOTOS

J.A. Gallud
DISCA
Universidad Politécnica
46071-VALENCIA

J.M. GarcÍa
Departamento de Informática
Universidad de Castilla-La Mancha
02071-ALBACETE
e-mail:utjmgarcia@02ccv1.ucma.es
Tel.: + 34-67- 50 61 84
Fax: + 34-67- 50 66 50

Resumen

Se describe en este artÍculo el diseño de una herramienta que permite obtener prototipos ejecutables a partir de especificaciones escritas en un subconjunto del LOTOS. Dicha herramienta, denominada NOTOS, analiza la especificaci\n formal de un sistema, permitiendo su verificación por medio de la simulación del comportamiento de dicho sistema. Hasta la fecha, se han realizado diversas implementaciones de especificaciones para diversos sistemas siendo los resultados obtenidos muy buenos.

Palabras claves: concurrencia, autómatas, especificación, lenguajes formales, compiladores, flexibilidad, prototipos.

1. Introducción.

Uno de los problemas frecuentes en el entorno de los Sistemas Distribuidos es la obtención de prototipos, en particular aquellos prototipos relacionados con los estándares de la Interconexión de Sistemas Abiertos. Son bastantes los protocolos y servicios OSI cuya descripción formal se ha realizado utilizando LOTOS.

Una especificación formal es una sentencia que expresa la conducta (el comportamiento) que se desea para un sistema, expresado con un alto nivel de abstracción (se muestra qué es lo que se quiere más que cómo se tiene que realizar).

Ultimamente se está haciendo mucho énfasis en el uso de la especificación formal para la construcción y desarrollo de un sistema, pues presenta una serie de ventajas. La más importante es que las propiedades de la especificación pueden ser analizadas matemáticamente para asegurar que la especificación es consistente internamente y su comportamiento el requerido. Ello permite que los costes de

desarrollo y mantenimiento de un sistema sean menores.

Tres técnicas importantes han sido desarrolladas para la especificación formal: la especificación orientada a modelos, la especificación algebraica y el álgebra de procesos. El LOTOS es un lenguaje incluido en ésta última técnica.

El principal problema para un mayor uso de las técnicas de especificación formal es la falta de herramientas con las que validar y chequear las diferentes especificaciones. Para el caso del LOTOS, tenemos noticia de una implementación dentro del proyecto SEDOS [Mañ89]. Otra referencia interesante se puede encontrar en [Hol93], dónde se usa el LOTOS junto con el lenguaje META-IV para programar robots off-line.

Esta circunstancia nos animó al estudio y desarrollo de un generador de prototipos para especificaciones formales escritas en LOTOS. Nuestra experiencia en el desarrollo de compiladores y lenguajes paralelos [Gar92] ayudó a la consecución de este objetivo.

En este artículo presentamos un generador de prototipos para especificaciones LOTOS denominado NOTOS. Nuestro generador toma como entrada una especificación de un sistema escrita en LOTOS y como salida obtiene un prototipo ejecutable escrito en lenguaje C con el que simulamos el comportamiento del sistema. De esta forma, podemos verificar si la especificación que habíamos realizado en LOTOS del sistema es correcta.

El resto del artículo se estructura como sigue. En la sección 2 se ofrece una visión del LOTOS y se explican los motivos que impulsaron su desarrollo. Posteriormente, en la sección 3 se especifica el lenguaje que acepta el generador de prototipos que hemos desarrollado, que es un subconjunto del LOTOS, mientras que en la sección 4 detallamos el funcionamiento del NOTOS. Finalmente, en la sección 5 se ofrecen unas conclusiones al trabajo realizado y se muestran las posibles líneas de trabajo a realizar en el futuro.

2. El lenguaje LOTOS.

El LOTOS [ISO87] (Language Of Temporal Ordering Specification) es un lenguaje formal de especificación por medio del álgebra de procesos de un sistema,

permitiéndonos realizar a través de un análisis técnico la verificación, validación y desarrollo de dicho sistema. Este fue diseñado para permitir la especificación formal de los sistemas distribuidos, en particular para aquellas especificaciones relacionadas con las arquitecturas OSI (Open Systems Interconnection) en las redes de computadores. Fue desarrollado por expertos de distintas nacionalidades en FDT entre los años 1981-86.

La idea básica que desarrolló el LOTOS fue que los sistemas pueden ser especificados definiendo la relación temporal entre las interacciones que constituyen el comportamiento observable externamente del sistema. El aspecto más importante del lenguaje es que incluye métodos del álgebra de procesos, concepto que introdujo Milner en el trabajo sobre el CCS [Mil89]. Así mismo, también está influenciado por el trabajo de Hoare en este campo [Hoa85].

En una especificación LOTOS, un sistema es modelado como una colección de procesos que se componen secuencialmente y en paralelo para definir el comportamiento observable del sistema. Un mecanismo de tipos abstractos de datos está también disponible para realizar prototipos en los que sea necesario cualquier estructura de datos.

En LOTOS un sistema concurrente y distribuido, es visto como un proceso, consistente probablemente de varios subprocesos. Un subproceso es, en si mismo, un proceso, por lo que en general una especificación LOTOS describe un sistema mediante la definición jerárquica de procesos. Un proceso es una entidad capaz de realizar acciones internas, no observables, así como interactuar con otros procesos, lo cual constituye su entorno. Las complejas interacciones entre los procesos se construyen a partir de las unidades elementales de sincronización, conocidas como eventos, interacciones acciones ó señales.

Los eventos (señales) sirven para la sincronización de los procesos. Tal sincronización puede incluir el intercambio de datos. Los eventos se consideran atómicos, en el sentido en que ocurren instantáneamente, sin consumir tiempo. La *ocurrencia* de estos eventos se realiza mediante unos puntos de interacción o puertas, que en el caso de una sincronización de dos procesos sin intercambio de datos coinciden los nombres del evento y de la puerta. El entorno de un proceso cualquiera, dentro de un sistema genérico S, se forma con el conjunto de procesos de S con el que aquel actda. Además se considera que siempre existe un observador imparcial que está preparado para observar cualquier cosa que el sistema pudiera hacer. Esta es una de las características de las especificaciones LOTOS: la descripción del

comportamiento observable del sistema. De ahí que sea necesaria para la observación, la interacción, ya que al decir que el proceso P realiza una acción observable nos estamos refiriendo a la interacción entre el proceso P y al menos el observador.

Lo importante en la definición de los procesos en LOTOS es tratar el comportamiento observable, sin necesidad de reparar en como actda internamente el proceso. En la definición de un proceso se especifica su comportamiento mediante la secuencia de acciones observables que se realizan a través de las puertas o puntos de interacción. Los diseñadores se apoyan en los árboles de acción para describir las secuencias de acciones.Cada expresión del comportamiento define un árbol de sincronización o árbol de transición. En un árbol tenemos nodos y transiciones, los nodos representan la expresión del comportamiento y las transiciones representan a las acciones. Así es como define la semántica operacional del LOTOS, mediante los axiomas y las reglas de inferencia, las expresiones de un modo formal. Este esfuerzo por asegurar el formalismo, tiene la utilidad que ya hemos mencionado de comprobar analíticamente las propiedades de las especificaciones para asegurar que el comportamiento que definimos está correctamente "capturado", y comprobar también la consistencia de la especificación.

Se puede encontrar en detalle una explicación sobre el LOTOS y cada uno de los operadores en [Bol89]. íesumiendo diremos que la idea de los diseñadores fue que se pudieran realizar descripciones de los estándares OSI -aunque se puede generalizar al modelado de cualquier sistema concurrente- de forma no ambigua, precisa, completa e independiente de la implementación. También se deseaba desarrollar documentos legibles para los usuarios de entornos OSI, así como para los implementadores y verificadores. El poder contar con un lenguaje con unas bases formalmente bien definidas, permite soportar un riguroso análisis de protocolos a nivel de diseño, con lo que se disminuyen el ndmero de errores que suelen proliferar en las distintas etapas de implementación.

3. El subconjunto del lenguaje adoptado.

El lenguaje LOTOS tiene dos características importantes: la especificación de la concurrencia y la definición de tipos abstractos de datos. En el subconjunto del lenguaje que hasta ahora hemos implementado, no hemos tenido en cuenta los tipos abstractos de datos. Sin embargo si que hemos tratado de abordar todos los

operadores, incluidos los operadores de la composición paralela, de manera que se pueden desarrollar especificaciones con expresiones del comportamiento casi de cualquier tipo. Hay que tener en cuenta que hemos tratado de hacer una herramienta lo más flexible y lo más potente posible y esto es imposible si no partimos de unas restricciones de diseño.

En la implementación del LOTOS que hemos realizado tenemos un conjunto de procesos y una serie de acciones de esos procesos entre si y también de los procesos con el entorno. Tenemos un sistema genérico que dividimos en procesos, en el comportamiento en su conjunto del sistema descubrimos estados o etapas claramente diferenciadas las cuales pueden constituir procesos. Estos los definiremos según el mismo esquema siempre con la visión del sistema o de los subsistemas como cajas negras.

Por último vamos a presentar la gramática de nuestro subconjunto del LOTOS en notación BNF:

Especificación = **specification** Ident;
behaviour ExpComp
[**where** { DefineProc }]
endspec Ident .

DefineProc = **process** IdentProc **[** Lista **]** [**(**Lista**)**] **:=**
ExpComp
[**where** { DefineProc }]
endproc

Op = >> * ****** * **[]** * ***[**Lista**]*** * ******* * **[>** * **i**

Lista= IdentSig { **,** IdentSig } *
IdentVar**:** IdentTipo {**,**IdentVar**:**IdentTipo }

ExpComp = IdentSig **;** {IdentSig **;**} **exit** *
IdentSig **;** {IdentSig **;** } **stop** *
ExpComp Op ExpComp *
IdentProc **[**Lista**]** [**(**Lista**)**]
{ Op IdentProc [**[**Lista**]**][**(**Lista**)**] } *

hide Lista **in** ExpComp *
IdentSig**?**Lista**;** { IdentSig**?**Lista**;** } ExpComp *
IdentSig**!**Exp**;** { IdentSig**!**ExpComp**;** } ExpComp

Exp = IdentVar * Valor
Valor = Num * **"** Let **"**
Num = **0** Num * **1** Num * ... **9** Num * **0** ... * **9**
Let = **A** Let * ... **z** * **0** Let ... * **9**
IdentVar = Ident
IdentTipo = Ident
IdentProc = Ident
IdentSig = Ident
Ident = **A** Ident * **B** Ident * ... * **z** Ident * **A** * ... * **z**

Se ha intentado ser lo más claro posible a pesar de la coincidencia de los caractéres * , [y] como operadores del lenguaje, símbolos terminales y como metasímbolos de la notación BNF.

Se puede comprobar la flexibilidad que proporciona el lenguaje en la definición de expresiones. Podría decirse que se puede representar cualquier sistema con los operadores que facilita esta técnica. éunto a esta característica que hace que empleemos el calificativo de lenguaje formal, hay que señalar la importante ventaja de la descripción de la concurrencia.

4. El generador NOTOS.

El generador de prototipos NOTOS analiza una especificación de un sistema escrita en el subconjunto del LOTOS descrito en el apartado anterior, realiza un chequeo léxico, sintáctico y semántico generando un prototipo. Esto representa una gran ayuda, ya que una vez realizada la labor de especificación de un sistema cualquiera se puede verificar el comportamiento mediante una simulación con el prototipo.

Una cuestión a tener en cuenta al hablar de verificar una especificación escrita en LOTOS mediante la simulación de su modelo, es la condicional dependencia del especificador. Esto significa que, debido al alto grado de abstracción proporcionado por las características del lenguaje, una especificación puede no ser directamente

"implementable" o incluso puede ocurrir el que de una especificación puedan establecerse varias implementaciones válidas. Es tarea de la persona que realiza la especificación garantizar la coherencia de la especificación.

Nuestro generador de prototipos NOTOS se compone de dos grandes fases o bloques: la fase de análisis y la fase de generación del prototipo. En la fase de análisis se estudia que la especificación de entrada sea sintácticamente correcta. Para ello, hemos implementado las distintas fases de análisis de un compilador tal y como vienen descritas en [Aho86].

El diseño de esta etapa ha sido dirigido por la estructura sintáctica de una especificación LOTOS, lo que hemos definido en el apartado anterior con las reglas de producción. Dos aspectos en esta etapa destacables han sido, por una parte, el conseguir que las etapas de análisis se ajustaran con el mayor rigor posible a las reglas de producción de una gramática muy flexible y con pocas restricciones, y por otra parte adoptar en la filosofía de diseño un criterio con el que, el generador NOTOS, fuera capaz de realizar un seguimiento a distintos niveles de las expresiones del comportamiento.

Son dos aspectos diferentes que reflejan dos de las características que diferencian a un lenguaje formal de otros lenguajes. El primero de ellos tiene que ver con la 'libertad' de que goza el especificador al definir un comportamiento, lo que complica el análisis al tener que utilizar una recursividad fuerte. El segundo aspecto está relacionado con la posibilidad de, en una especificación, tener distintos niveles de definición del comportamiento, cada uno de ellos detallando el nivel superior. Algunas ideas tradicionales [Wir90] sobre el manejo de los procedimientos y funciones en otros lenguajes han sido utilizadas, aunque el tratamiento de fondo es distinto ya que el comportamiento implementado, mediante el autómata que obtiene el generador NOTOS, se define con información sacada de las definiciones de los comportamientos en los distintos niveles.

La salida de las etapas de análisis, que hemos explicado someramente -recordemos que la entrada al generador era la especificación escrita en LOTOS-, consiste en una tabla de transición, en la que tenemos cada uno de los procesos definidos, por una parte, y cada una de las acciones que intervienen en el comportamiento del sistema por la otra. En este caso tenemos dos opciones: generar un prototipo (escrito en C) que pudiera ser compilado y a partir de ahí obtener el fichero ejecutable que simula el sistema, o bien realizar dicha simulación directamente

en la memoria del ordenador. En ambos casos es clave la fase de simulación, que es la que nos permite verificar si el comportamiento del sistema es el adecuado. La fase de simulación trata de ir recorriendo el árbol de acción o árbol de transición de acuerdo con las posibles secuencias de acciones. En este recorrido del árbol de transición podemos validar el comportamiento del sistema. Uno de los aspectos que conviene explicar es el modo en que se han implementado algunas de las expresiones del comportamiento más características del lenguaje, como es la posibilidad de describir la concurrencia mediante el operador que define la composición paralela. La idea que se ha empleado para simular el paralelismo consiste en el empleo de la técnica del tiempo compartido, de manera que asignamos a cada una de las expresiones que se realizan en paralelo un tiempo o cuanto. Cuando una expresión consume su cuanto se conmuta a la siguiente, esto implica que hay un orden de ejecución, o lo que es lo mismo hay una cola. Este mismo mecanismo, transparente al especificador, emplea también un planificador. Es necesario también desdoblar la tabla de transición, una por expresión, en el momento en el que el sistema debe realizar una serie de acciones en paralelo.

Más detalles sobre la implementación del generador se pueden encontrar ampliamente en [Gal91]. Sólamente señalar que se ha buscado como lenguaje para la implementación uno que reuniera las características de ser claro y modular, la elección ha recaido sobre el lenguaje Módula-2.

Se han realizado diversas implementaciones -como la especificación de un protocolo de transporte, la especificación del comportamiento paralelo y otros ejemplos [Gal91]- que han demostrado la utilidad del generador NOTOS como herramienta de validación de especificaciones de sistemas. Otra de las experiencias de esta etapa de prueba del generador es la necesidad de ajustar los criterios del especificador a los de la filosofía del generador, esto es, asumir las restricciones sin las que sería imposible desarrollar herramientas de este tipo.

5. Conclusiones y trabajo futuro.

Se ha intentado presentar un generador de prototipos que implementa la técnica formal LOTOS como una máquina de estados. También se han explicado las principales ideas que deben emplearse para conseguir una implementación de la especificación de una forma correcta. Hemos señalado las herramientas existentes, las ventajas de aquellas y las facilidades que tiene nuestra herramienta.

Gracias a nuestro generador NOTOS, tenemos la posibilidad de especificar sistemas, mediante un lenguaje formal, de una manera mucho más clara que con un lenguaje de alto nivel, lo que significa no estar sujeto a restricciones. La principal ventaja que aquí se descubre es que podemos manejar, con la flexibilidad que da el entorno de la informática personal, un lenguaje formal.

Como trabajo futuro, tenemos varias vias abiertas. Por una parte, ampliar nuestra gramática de entrada para que admita especificaciones escritas en LOTOS, permitiendo por tanto los tipos abstractos de datos y las instanciaciones de los procesos. Por otra parte, nos gustaría dotar a la herramienta desarrollada -NOTOS- con un entorno que hiciera amigable su manejo por el usuario.

Referencias

[Aho86] Aho, Sethi y Ullman. Compilers: Principles, techniques and tools. Addison-Wesley, 1986.

[Ber84] G.Berry y L.Cosserat. The Synchronous Programming Language Esterel and Mathematical semantics. Seminar on Concurrency. Springer-Verlag LNCS 197, 1984.

[Bol89] Tommaso Bolognesi. Introduction to the ISO Specification Language LOTOS. The Formal Description Technique LOTOS, results of the ESPíIT/SEDOS project, pp. 23-71. Elseveir Science Publisher B.V. (North-Holland) 1989.

[Gal91] é.A. Gallud. Diseño e Implementación de un Generador Automático de Prototipos a partir del Lenguaje LOTOS. Proyecto Final de Carrera. Facultad de Informática de Valencia. 1991.

[Gar92] é.M. García y é. Duato. An advanced environment for programing transputer networks with dynamic reconfiguration. M.Valero et al. (Eds). Parallel Computing and Transputers Applications. IOS Press/CIMNE, pp. 601-610, 1992.

[Hoa85] C.A.í. Hoare. Communicating Sequential Processes. Prentice Hall, 1985.

[Hol93] D.í.W. Holton et al. Experiences with then íigorous Development of a Parallel Program, in Proc. Euromicro Workshop on Parallel and Distributed Processing, pp. 375-382. Gran Canaria, 1993.

[ISO87] ISO. Information Processing Systems -Open Systems Interconnection- LOTOS, A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. ISO 1987.

[Mañ89] éosé A. Mañas, Tomás de Miguel y Huub van Thienen. The implementation of Specification Language for OSI Systems. The Formal Description Technique LOTOS, results of the ESPíIT/SEDOS project, pp. 409-421. Elsevier Science Publishers B.V. (North-Holland) 1989.

[Mil89] íobin Milner. Communication and Concurrency. Prentice Hall, 1989.

[Wir90] Niklaus Wirth. Compiladores. íueda, 1990