Verificación de Conexiones Telefónicas con Esterel

Jorge Graña Gil Manuel Vilares Ferro Raphäel Bernhard

Resumen

El presente trabajo resume el proceso de diseño, implementación y verificación del comportamiento de una centralita telefónica digital en el entorno de programación de tiempo real síncrono Esterel. *Nuestra intención es mostrar la modularidad de la aplicación y la flexibilidad del proceso de verificación.*

Idéntica atención merecen los mecanismos de control que gestionan la sincronización de procesos concurrentes. El objetivo es detectar los fenómenos de abrazo mortal e interbloqueo en tiempo de compilación, una característica no disponible en todos los lenguajes de programación.

Palabras Clave*: Programación Síncrona, Sistemas Reactivos, Verificación de Autómatas.*

1 Introducción. Una conexión telefónica es un protocolo de comunicación universalmente conocido, pero no por ello fácil de implementar. En la fase de conexión, aparecen problemas de sincronización entre los teléfonos. Si no se dispone de un método eficaz para el manejo de las señales implicadas, el sistema podría verse afectado por un abrazo mortal entre ellas. Además, la no detección *a priori* de este tipo de fenómenos podría obligar a un rediseño total del sistema.

Los lenguajes de programación síncronos, tales como Esterel, se han revelado como adecuados para tratar este tipo de problemas en el marco de los sitemas reactivos. En efecto, su semántica facilita el proceso de verificación de las aplicaciones, y la detección y posterior eliminación de los problemas de sincronismo típicos de los protocolos de comunicación.

2 Sistemas reactivos. Los sistemas reactivos son sistemas de tiempo real que reaccionan instantáneamente a los eventos de su entorno. Pueden verse como una "caja negra" que recibe señales de entrada y emite respuestas en forma de señales de salida. Internamente, varios dispositivos funcionan en paralelo y se comunican entre sí por medio de señales locales, como se ve en la figura 1.

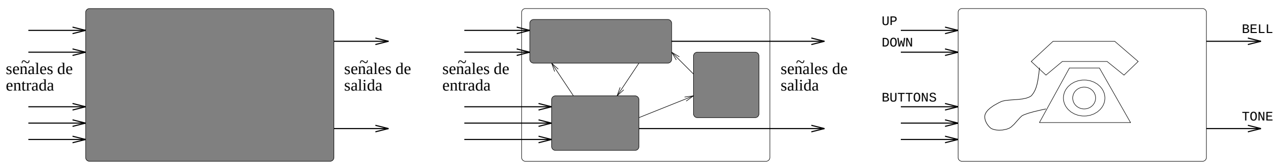

Figura 1: Visiones macroscópicas y visión interna de un sistema reactivo.

El procesamiento depende del orden de los eventos de entrada, lo cual implica mecanismos de control para gestionar la sincronización de actividades concurrentes. Las técnicas de programación tradicionales no resultan adecuadas. Una solución es resolver la comunicación entre procesos mediante el concepto de *difusión*, por el que una señal emitida se recibe en cualquier proceso dentro de su ámbito, y adoptar el paradigma de la *sincronía perfecta* [2], que establece que el tiempo está definido de forma externa a los programas a través del flujo de entradas, y que el manejo interno de éstas se realiza con retraso cero respecto a las unidades de tiempo externas [3].

Sin embargo, como consecuencia de la hipótesis de sincronía perfecta, surgen dos cuestiones: los *ciclos de causalidad*, "abrazos mortales" cuando hay dependencias circulares entre las señales, y los *bucles instantáneos*, bucles en los que ninguna sentencia consume tiempo. Una forma de afrontar estos problemas es que los sistemas los detecten en tiempo de ejecución, pero lo ideal sería detectarlos en tiempo de compilación, y no generar código incorrecto.

J. Graña y M. Vilares pertenecen al Departamento de Computación de la Universidad de La Coruña, Campus de Elviña S/N, 15071 La Coruña, España. E. mail: grana@dc.fi.udc.es, vilares@dc.fi.udc.es.

R. Bernhard pertenece al Centre National d'Etudes des Télécommunications de France Telecom, 905 rue Albert Einstein, 06921 Sophia-Antipolis Cedex, Francia. E. mail: bernhard@sophia.cnet.fr.

Este trabajo fue parcialmente subvencionado por el proyecto XUGA10501A93 de la *Xunta de Galicia*, y desarrollado en el Centre de Mathématiques Apliqueés de l'Ecole Superieure des Mines de Paris, Sophia-Antipolis, Francia.

Respecto a la verificación de las aplicaciones, el entorno de programación debería incluir mecanismos que permitan reducir el sistema global. El parámetro de esta reducción debería ser un criterio de abstracción definido por el usuario y que exprese su vista parcial del sistema.

Esterel es un lenguaje de programación síncrono para sistemas reactivos, que reune estas condiciones [4]. El compilador genera un autómata determinista, adaptado al manejo de actividades paralelas y a la sincronización de procesos concurrentes. El lenguaje dispone de un entorno gráfico que integra las tareas de edición, compilación, simulación y verificación formal.

3 El modelo básico. Nuestra primera aproximación conecta dos teléfonos directamente. En un primer momento, el compilador rechaza nuestro programa por existir un ciclo de causalidad. Una vez eliminado, el proceso de verificación nos permite detectar la existencia de comportamientos no deseados: en los estados 27 y 28, las señales `TALKING_1` y `TALKING_2` no han sido emitidas de forma síncrona. Como consecuencia, un teléfono puede estar hablando solo, tal y como se ve en la figura 2. Es importante resaltar que si el lenguaje de programación no fuera Esterel, estos comportamientos

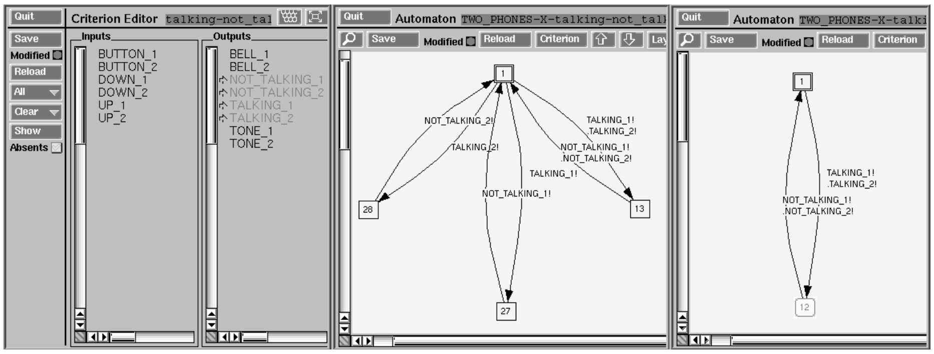

Figura 2: Criterio de abstracción, comportamiento anómalo y comportamiento correcto.

erróneos probablemente no habrían sido detectados y no hubieramos podido llegar a una solución totalmente correcta, cuyo comportamiento se muestra en el autómata reducido de la derecha.

4 La centralita digital. Realmente, los teléfonos no tienen inteligencia para establecer una comunicación. El relé de la centralita asume dicha funcionalidad. Esta es la característica esencial que incorporamos ahora. Se ha desarrollado una aplicación con tres teléfonos y con una centralita de tres relés. No obstante, esta concepción modular [1] podría usarse para añadir cualquier número arbitrario de teléfonos, lo cual proporciona extensibilidad al modelo. El comportamiento de esta nueva aplicación ha resultado ser correcto, lo cual hace válida la construcción propuesta.

5 Conclusión. Nuestro propósito ha sido modelizar una centralita telefónica digital. Esterel se ha mostrado como un entorno válido, facilitando el desarrollo modular e incremental del protocolo. Aún siendo un protocolo de comunicación sencillo, los problemas tratados en nuestra implementación son comunes a buen número de protocolos de comunicación más complejos. Esto prueba la buena disposición del entorno, para la modelización de los mismos.

Referencias

[1] R. Bernhard. Esterel v4: une extension modulaire d'Esterel. Thèse dínformatique, Université de Nice, 1992.

[2] G. Berry. The semantics of pure Esterel. In M. Broy, editor, *Program Design Calculi*, volume 118 of *Series F: Computer and System Sciences*, pages 361–409. NATO ASI Series, 1993.

[3] G. Berry and A. Benveniste. The synchronous approach to reactive and real-time systems. *Another Look at Real Time Programming, Proceedings of the IEEE*, 79:1270–1282, 1991.

[4] E. Madelaine and D. Vergamini. Auto: A verification tool for distributed systems using reduction of finite automata networks. In *Proc. FORTE'89 Conference, Vancouver*, 1989.