Propuesta de tesina para la obtención del grado Licenciado en Ciencias de la Computación

November 11, 2015

Postulante: José Luis Diaz
Directores: Hernán Ponce de León y Maximiliano Cristiá

1 Situación del postulante

El postulante aprobó 29 materias de la Licenciatura en Ciencias de la Computación (plan 1994) en marzo de 2015. Solamente le resta la tesina.

2 Título

Un modelo semántico para Actores basado en CSP.

3 Motivación y objetivo general

Las aplicaciones, con el incremento de la cantidad de núcleos por microprocesador, hacen un uso mas frecuente de la concurrencia. Una forma de atacar este tipo de problemas es el modelo tradicional de concurrencia que se basa en multi-hilos, variables compartidas, locks, etc. Este trabajo propone utilizar un enfoque diferente: el modelo de actores, utilizado en la industria particularmente en lenguajes como Erlang [3] y Scala [4] con la librería Akka [9].

La diferencia entre ambos modelos se puede notar mediante el problema del Jardín Ornamental. El enunciado del problema es el siguiente: supongamos que tenemos dos entradas a un parque y se requiere saber cuanta gente ingresa, para eso se instala un molinete en cada entrada. Se utiliza una computadora para registrar la información de ingreso.

Implementándolo en un lenguaje imperativo, siguiendo el modelo tradicional, incluiría una variable global que guarde la cantidad de visitantes y dos hilos representando los molinetes que incrementan esta variable. Sin ningún tipo de protección en la región critica planteada por la actualización de la variable se perderían incrementos, ya que cada hilo carga localmente el valor de la variable global, efectúa un incremento y finalmente guarda el valor en la variable global.

El mismo problema se puede describir utilizando el modelo de actores, que tiene como único mecanismo de comunicacion entre procesos el paso de mensajes. En este caso, el problema puede ser representado utilizando un actor que realiza la tarea de contador que incrementará su valor cuando reciba un mensaje *inc*, y otros dos actores que emitirán estos mensajes (los molinetes). En este caso el problema de la pérdida de la actualización no ocurre, por las garantías que tiene el paso de mensajes entre actores.

El objetivo de este trabajo es comprender en profundidad el modelo de actores y su semántica. Una buena herramienta para asistir a este proceso es utilizar métodos formales. Se propone modelar su semántica en CSP y efectuar algunas pruebas utilizando la herramienta FDR [8].

4 Fundamentos y estado de conocimiento sobre el tema

Como señala Rob Pike en su charla titulada "Concurrencia no es paralelismo" [5], muchas veces se pasa por alto la diferencia conceptual entre estas dos ideas. En programación, la concurrencia es la composición de los procesos independientemente de la ejecución, mientras que el paralelismo es la ejecución simultánea de cálculos (posiblemente relacionados).

El modelo de actores es originalmente propuesto por C. Heweeit [9], es un enfoque diferente a cómo estructurar programas concurrentes. Un actor, computacionalmente, es una entidad que de manera concurrente puede:

- Enviar y recibir un numero finito de mensajes a otros actores
- Crear un numero finito de actores
- Designar un nuevo comportamiento a ser usado cuando se reciba el próximo mensaje.

Gul Agha [1] en parte de su trabajo doctoral describe los actores usando una semántica operacional estructuada [6]. Define dos tipos de transiciones que representan la evolución de la configuración de un sistema de actores, la primera *transición posible* representa cuales son todas las posibles transiciones del sistema, esta relación es insuficiente para garantizar la entrega de estos eventos, también define *siguiente* que expresa justamente esta garantía.

Communicating Sequential Processes (CSP), fue propuesto por primera vez por C.A.R Hoare [7], es un lenguaje para la especificación y verificación del comportamiento concurrentes de sistemas. Como su nombre indica, CSP permite la descripción de sistemas en términos de componentes que operan de forma independiente que interactúan entre sí únicamente a través de eventos síncronos. Las relaciones entre los diferentes procesos y la forma en que cada proceso se comunica con su entorno, se describen utilizando un álgebra de procesos.

Comparando CSP con el modelo de actores, ambos mecanismos tienen procesos concurrentes que intercambian eventos. Sin embargo, los dos modelos

hacen algunas decisiones fundamentalmente diferentes con respecto a las primitivas que proporcionan:

- Los procesos de CSP son anónimos, mientras que los actores tienen identidades.
- Los eventos fundamentalmente consisten en una sincronización entre los procesos involucrados en el envío y la recepción del evento, es decir, el remitente no puede transmitir un evento hasta que el receptor está dispuesto a aceptarlo. Por el contrario, en los sistemas de actores, el paso de eventos es fundamentalmente asíncrona, es decir, la transmisión y la recepción de eventos no tienen que suceder al mismo instante.
- CSP utiliza canales explícitos para el paso de eventos, mientras que los sistemas de actores transmiten eventos a los actores de destino.

Estos enfoques pueden ser considerados duales de uno al otro, en el sentido de que los sistemas basados en *sincronización* pueden utilizarse para construir comunicaciones que se comporten como sistemas de mensajería asíncrona, mientras que los sistemas asíncronos se pueden utilizar para construir las comunicaciones síncronas utilizando algún protocolo que permita el encuentro entre los procesos. Lo mismo ocurre con los canales.

El $\pi - calculo$, es un álgebra de procesos que captura en esencia la simpleza del $\lambda - calculo$. Un posterior trabajo de Gul Agha [2] crea un sustento teórico algebraico para el modelo de actores, y le da semántica a un lenguaje concreto llamado Simple Actor Language (SAL). Este lenguaje permite definir los rudimentos necesarios para construir actores, definir nuevos comportamientos, crear/enviar/recibir comandos.

FDR es una herramienta para el análisis de los programas escritos en notación CSP de Hoare, en particular utilizando CSP_M, que combina los operadores de CSP con un lenguaje de programación funcional. FDR originalmente fue escrito en 1991 por Formal Systems (Europe) Ltd, que también lanzo la versión 2 a mediados de la década de 1990. La versión actual de la herramienta es FDR3, se liberó por primera vez en 2013 por la Universidad de Oxford.

5 Objetivos específicos

- Realizar un estudio exhaustivo del modelo de actores.
- Crear una semántica del modelo de actores en CSP, adaptar la misma al lenguaje SAL.
- Modelar en SAL, algunos programas tradicionales como el factorial, productor consumidor con buffer acotado, una pila. Generar el modelo asociado en CSP.
- Utilizar la herramienta FDR3, para comprobar que la especificación es deadlock free, o que las trazadas no divergen.

6 Metodología y plan de trabajo

Los objetivos específicos en sí, describen una secuencia adecuada del trabajo a realizar el cual parecería tener una carga equilibrada en complejidad y en tiempo de ejecución. Se propone trabajar durante 6 meses con una dedicación de 20 horas semanales. Se finalizará con la escritura de los resultados obtenidos.

References

[1] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA, USA, 1986.

[2] Gul Agha and Prasanna Thati. An algebraic theory of actors and its application to a simple object-based language. In Olaf Owe, Stein Krogdahl, and Tom Lyche, editors, *From Object-Orientation to Formal Methods*, volume 2635 of *Lecture Notes in Computer Science*, pages 26–57. Springer Berlin Heidelberg, 2004.

[3] Francesco Cesarini and Simon Thompson. *ERLANG Programming*. O'Reilly Media, Inc., 1st edition, 2009.

[4] Martin Odersky and al. An Overview of the Scala Programming Language. Technical Report IC/2004/64, EPFL, Lausanne, Switzerland, 2004.

[5] Rob Pike. Concurrency is not parallelism. `http://blog.golang.org/concurrency-is-not-parallelism`, 2013.

[6] G. D. Plotkin. A structural approach to operational semantics, 1981.

[7] Hoare C. A. R. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, August 1978.

[8] et al. Thomas Gibson-Robinson, Philip Armstrong. Fdr3 — a modern refinement checker for csp. In Erika Ábrahám and Klaus Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 8413 of *Lecture Notes in Computer Science*, pages 187–201, 2014.

[9] Derek Wyatt. *Akka Concurrency*. Artima Incorporation, USA, 2013.