FROB: Un lenguaje de programación funcional reactivo para robots con bajas capacidades de cómputo

Guillermo Pacheco

28 de abril de 2015

Resumen

El proyecto consiste en la creación de un lenguaje de programación para robots, cuyas capacidades de cómputo son limitadas. Para esto se escogió el paradigma de Programación Funcional Reactiva (*FRP*) el cuál permite expresar naturalmente reacciones a valores que varían en función del tiempo.

El objetivo es utilizarlo con fines educativos, por lo tanto debe ser simple y fácil de usar por usuarios inexpertos y no familiarizados con la electrónica y la informática.

Para resolver el problema, se dividió en dos etapas.

La primera consiste en la implementación de un compilador que traduce el lenguaje cuyo nivel es alto a un lenguaje de bajo nivel (*Bytecode*) más simple e interpretable.

La segunda etapa consiste en implementar una máquina virtual, que sea capaz de interpretar el lenguaje de bajo nivel. Por cada plataforma objetivo, es posible realizar una implementación de la máquina, lo que permite ejecutar los mismos programas en alto nivel, en diferentes plataformas.

El lenguaje se implementará como un DSL embebido en Haskell.

El diseño de la máquina consiste de un núcleo común capaz de interpretar instrucciones, y módulos bien definidos de entrada/salida los cuáles varían de una plataforma a otra. Ésto permite mayor portabilidad y extensibilidad.

Debe ser posible ejecutar programas en dicho lenguaje dentro de plataformas de hardware reducido. Considerando ésto el lenguaje de programación elegido para la implementación de la máquina virtual es C/C++.

De ésta forma se implementación de un lenguaje reactivo con las características deseadas, y una implementación modelo de una máquina virtual que permite su ejecución en una arquitectura objetivo deseada. La misma es fácil de mantener, portable y cuenta con suficiente flexibilidad para ser extendida.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Este documento desarrolla la construcción de herramientas que permitan programar robots utilizando el paradigma de programación funcional reactiva.

Para ello se define el lenguaje *Frob* de alto nivel, que permite expresar los comportamientos de un robot y sus interacciones. El lenguaje permite expresar los mismos en base a *Comportamientos* y *Eventos*.

Los eventos son

Capítulo 2

Programación Funcional Reactiva

2.1. Programación Funcional Reactiva

Acá voy a hablar de programación funcional reactiva.

Tradicionalmente los programas son formados a partir de una secuencia de acciones imperativas. Los programas reactivos suelen formarse por eventos y código iterativo que se corre cuando un evento ocurre.

Dicho código iterativo suele hacer referencia y manipular variables compartidas con diferentes rutinas. Ésto lleva a que como un valor puede ser manipulado desde diferentes lugares, puedan producirse problemas de concurrencia y algunos valores pueden quedar en un estado inconsistente.

En el paradigma FRP no existen valores compartidos, sinó que dichos valores dependientes del tiempo, tienen una representación llamada Comportamiento y la única forma de modificarlos, es a partir de como fueron definidos.

Definición 1. *Comportamientos (Behaviours).*
Un comportamiento es un valor contínuo que depende del paso del tiempo. Los comportamientos se pueden definir, combinar, pasarlos como argumentos a funciones, retornarlos. Un comportamiento puede ser un valor constante, el tiempo mismo (un reloj), o puede formarse combinando otros comportamientos, por ejemplo secuencialmente o paralelamente.

Definición 2. *Eventos (Streams).*
Son valores discretos dependientes del tiempo, que forman una secuencia finita o infinita de ocurrencias. Cada ocurrencia está formada por el valor y el instante de tiempo.

La principal diferencia entre Comportamientos y Eventos, es que los comportamientos son valores contínuos y los eventos son discretos.

Los comportamientos representan cualquier valor en función del tiempo, por ejemplo:

- *entrada* sensor de distancia, temperatura, video
- *salida* velocidad, voltaje
- *valores* intermedios calculados

Las operaciones que se pueden realizar sobre los comportamientos incluyen:

- *Operaciones genéricas* Aritmética, integración, diferenciación
- *Operaciones específicas de un dominio* como escalar video, aplicar filtros, detección de patrones.

Los eventos pueden ser sensores específicos de un dominio, por ejemplo un botón, un click, una interrupción o mensaje asincrónico. También puede ser generado a partir de valores de un comportamiento, como ser *Temperatura alta*, *Batería baja*.

Las operaciones que se pueden realizar sobre los eventos incluyen:

- fmap, filter
- Modificar un *Comportamiento* reactivo

2.2. Ejemplo

Para entender un poco más las ideas de Comportamiento y Evento, se puede plantear el siguiente ejemplo.

En una cuenta de un banco, el saldo se puede definir como un comportamiento, el cuál solo se modifica cuando ocurre un movimiento. Un movimiento puede ser depositar dinero o extraer dinero de la cuenta. Un dato importante a ver, es que no es posible asignar un valor sin que sea por medio de su propia definición, por lo que nadie podría realizar la asignación $saldo = 1000000$. La misma operación sería posible creando un movimiento, el cuál afectaría al saldo.

Usaremos la notación $< nombre >:: Event < tipo >$ para definir eventos y la notación $< nombre >:: Behaviour < tipo >$ para definir comportamientos.

```
movimiento :: Event Number
movimiento = read(0)

saldo :: Behaviour Number
saldo := saldo + movimiento

alert :: Event Bool
alert = saldo > 1000

on alert:
  write(0, 'El saldo es mayor a 1000')
```

Un comportamiento, solo se modifica cuando cambian sus componentes,

Capítulo 3

Conclusiones

Bibliografía

[1] *Principles of Model Checking.*
Christel Baier y Joost-Pieter Katoen.
The MIT Press, 2008.

[2] Sitio web de *Python 2.7*:
`http://docs.python.org/2/`
Último acceso: 31/10/2013.

[3] Sitio web de *GraphML File Format*:
`http://graphml.graphdrawing.org/`
Último acceso: 31/10/2013.

Apéndices