Programación de Robots Físicos mediante Interfases Visuales. Reporte Preliminar*

Nicolás Passadore nicolasale@gmail.com
Gerardo Parra gparra@uncoma.edu.ar
Rodolfo Del Castillo rolo@uncoma.edu.ar

Grupo de Investigación en Robótica Inteligente
Depto. de Ciencias de la Computación
Facultad de Economía y Administración
Universidad Nacional del Comahue
Buenos Aires 1400 - 8300 Neuquén - Argentina
Tel/Fax (54) (299) 4490312/313

Resumen

Este trabajo tiene como objetivo realizar un análisis preliminar de la programación de robots físicos por medio de interfases gráficas. Se describe el diseño de un compilador que toma como entrada un programa generado desde una interfase visual y genera como salida un programa en código intermedio. Luego, este programa intermedio puede ser interpretado por el robot para ejecutar las acciones indicadas en el programa fuente.

1. Introducción

El vertiginoso avance de la robótica en los últimos años, ha provocado un desarrollo notorio en las herramientas de software relacionadas. Una de ellas, los lenguajes de programación, han sido adaptados para soportar la programación de robots. El caso más común se da en los lenguajes de programación tradicionales. Estos fueron, en su mayoría, extendidos con el fin de proveer al programador las funciones necesarias que le permitan programar robots físicos.

*Este trabajo está parcialmente financiado por la Universidad Nacional del Comahue, en el contexto del Proyecto de Investigación "Técnicas de Inteligencia Computacional para el diseño e Implementación de Sistemas Multiagentes" (COD 04/E062), por el Grupo de Investigación en Robótica Inteligente y por la Universidad Politécnica de Madrid a través del Proyecto AL05_PID_0040, "Implementaciones y Modelos de Razonamiento basado en Programación Lógica".

La utilidad y flexibilidad de un robot está determinada, en gran medida, por la potencia del software que posee. Esto ha dado lugar a una clasificación basada en el nivel del lenguaje de programación en el que fueron programados. Esta clasificación[5] se presenta de la siguiente manera:

- Sistemas guiados: el usuario conduce el robot a través de los movimientos a ser realizados.
- Sistemas de programación de nivel-robot: el usuario escribe un programa en un determinado lenguaje, especificando los movimientos y sensores.
- Sistemas de programación de nivel-tarea: el usuario especifica la operación por medio de las acciones sobre los objetos que el robot manipula. Esto es, solo se programan las acciones que el robot realizará sobre determinados objetos. Por ejemplo, un robot grúa tendrá funciones para levantar un determinado objeto y funciones para girar, siempre en un lugar fijo.

La segunda categoría de esta clasificación es de especial interés, puesto que involucra a lenguajes de programación tradicionales, extendidos para soportar la programación de robots. Estos lenguajes contienen funciones que permiten programar las distintas acciones que el robot debe realizar. La programación con lenguajes tradicionales es ampliamente conocida[6]. En

este caso, el programador introduce sentencias (código propio de cada lenguaje), que le permiten especificar las acciones que el robot debe realizar. Estas sentencias permiten controlar los motores y sensores y determinan el comportamiento mediante el cual, el robot se desempeñará en un entorno determinado y predefinido. De esta manera, el software que controla las acciones del robot puede ser construído como cualquier otro programa, solo debe conocerse el lenguaje en el que se va a programar y las funciones que permiten controlar los distintos elementos que conforman un robot, sus sensores y motores.
Pero la manera tradicional de programar robots tiene una limitante importante. El interesado en realizar esta tarea debe conocer en detalle el software que controla el robot.
Una alternativa a la programación de robots físicos con lenguajes tradicionales extendidos, es la de programar mediante una interfase gráfica visual. En este caso, no es necesario introducir código de programación, sino elementos que representen bloques de código que realizan alguna acción concreta.
El objetivo de este trabajo es mostrar cómo, la utilización de la interfase gráfica de Lego MindStorms[7], nos permite facilitar la programación de robots físicos. Con este objetivo como guía, realizamos el análisis de la interfase gráfica de Lego y presentamos el diseño de un compilador. La tarea fundamental de nuestro compilador es tomar como entrada un programa generado desde la interfase visual y generar otro programa equivalente en código intermedio que luego puede ser interpretado por el robot.
El artículo está organizado de la siguiente manera. En la sección siguiente describimos la interfase gráfica de Lego. En la sección 3 introducimos los conceptos que hacen referencia al código intermedio LASM. La siguiente sección contiene el principal aporte de este trabajo. Se presenta el diseño de un compilador que nos permite generar código intermedio LASM, a partir del programa fuente realizado desde la interfase gráfica de Lego. La sección 5 contiene las conclusiones del trabajo y se describen las líneas de trabajo futuro.

2. La Interfase Gráfica de Lego

El Sistema de Invención Robótica de Lego Mindstorms[7], fue pensado, en principio, para niños[1]. El lenguaje oficial del RCX[2] se caracteriza por su simplicidad y legibilidad.
Este sistema propone programar robots montando bloques de comandos gráficos. Estos bloques de comandos se conocen como *Lego bricks* y representan instrucciones para el robot. Mediante estos bloques puede lograrse que el robot se mueva, emita un sonido o que reaccione a distintos eventos.
El entorno de programación es una interfase de programación visual orientada a objetos. Existen íconos que representan bloques de código que realizan acciones concretas. El usuario puede elegir de una lista de comandos y colocarlos en un orden lógico que el RCX puede ejecutar.

2.1. Tipos de Comandos

Los tipos de comandos que se pueden utilizar desde la interfase son los siguientes:

- Movimientos y sonido: Con estos comandos se logra animar al robot. Permiten que se mueva en diferentes sentidos, que gire y realice determinados sonidos.
- Variables: Las variables son utilizadas para personalizar al robot. Por ejemplo, las variables pueden ser utilizadas para especificar la duración de algunas de las acciones.
- Esperar y repetir: Estos comandos son usados para hacer que el robot responda y reaccione al mundo que lo rodea.
- Condicionales: Los comandos condicionales permiten seguir diferentes cursos de acción.
- Comandos personalizados: Estos comandos sirven para crear bloques de códigos personalizados.
- Sensores: Los sensores son usados para que el robot realice testeos y reaccione a su entorno.

[1] Según Lego, *"... es un entorno de programación visual que permite a los niños recoger, soltar y apilar comandos y trozos de código."*
[2] RCX es el microprocesador de cada robot Mindstorms.

2.2. Descripción del Recipiente de Bloques

El recipiente de bloques contiene los distintos comandos que pueden ser utilizados. Determinados bloques están habilitados o no, dependiendo del tipo de robot que se requiere programar.

Los distintos bloques que se encuentran en el recipiente de bloques son:

- Bloques Grandes: Con estos se logra controlar las partes móviles del robot. Aquí tenemos comandos que, por ejemplo, determinan movimientos hacia delante o hacia atrás.
- Bloques Chicos: Hacen posible un control total sobre el robot. Permiten manipular variables y personalizar al robot.
- Mi Bloque: Este tipo de bloques permiten mantener un programa de manera eficiente y ordenada.
- Esperar (Wait): Este bloque permite que el programa espere una determinada cantidad de tiempo o que espere a que un sensor sea disparado.
- Repetir: Este bloque permite repetir un conjunto de comandos.
- Condicionales: Estos bloques permiten adherir un salto en el programa basado en la lectura de un sensor.
- Sensores: Estos bloques sirven para monitorear los diferentes sensores y disparar una tarea de comandos cuando se cumple determinada condición.

La programación de los robots con la interfase gráfica se basa en apilar estos bloques en el orden en que serán ejecutados. Cada uno de estos, como ya fue mencionado, representan un conjunto de comandos que conforman una acción concreta que el robot puede realizar. De esta forma, en ningún momento el programador introduce código, solo interactúa con los elementos (bloques) que ofrece la interfase.

3. El Código Intermedio LASM

Como se describió en la introducción, el objetivo de este trabajo es mostrar cómo la interfase gráfica de Lego nos permite programar robots físicos. Nuestra propuesta de compilador toma como entrada un programa realizado desde la interfase visual y genera un programa equivalente en código intermedio. Luego, el programa intermedio es enviado, a través de la herramienta MindScript[8], al procesador del robot, para ejecutar las acciones indicadas en el programa fuente.

Para este trabajo, el código intermedio elegido es el LASM (Lego Assembler)[1]. LASM contiene instrucciones que permiten manipular los distintos componentes de un robot. Componentes físicos, como motores y sensores y componentes no físicos, como los eventos.

Los comandos de LASM, como por ejemplo, WAIT, EVENT y SETV, tienen 2 (dos) parámetros. Uno, el denominado fuente, indica el tipo de parámetro. El otro parámetro es el valor e indica el índice del tipo del parámetro. A continuación, se exhibe un comando típico de LASM:

wait 0,5

donde 0 (cero) indica que el tipo de parámetro es una variable. El segundo parámetro es el índice donde esa variable está alojada.

El código LASM no contiene instrucciones que permitan almacenar valores en una pila. Todos los valores están alojados en variables que luego son accedidas mediante un índice como se explica arriba.

Algunas de las instrucciones más importantes en LASM, son las que permiten controlar los sensores, los eventos y los motores del robot. La instrucción EVENT, para el caso de los eventos, permite inicializar un evento. La instrucción DIR permite controlar los motores del robot.

Para mayores detalles referidos al código LASM se recomienda [1].

4. El Compilador

Los programas generados desde la interfase visual de Lego contienen una estructura similar a un programa realizado en lenguaje C. Esto es, luego que un programa es diseñado desde la interfase visual, se genera un archivo con un programa fuente que contiene la estructura que se muestra a continuación:

program ejemplo {
declaraciones
main {*comandos*}
watcher {}
}

Un bloque global (*program ejemplo* {}), que encierra todo el código que el robot ejecutará y en donde se declaran las variables, sensores, motores y eventos. Un bloque principal (*main* {}), que contiene las acciones específicas que realizará el robot, por ejemplo, las acciones que hacen que un evento se dispare o aquellas que se ejecutan ante la percepción de algún sensor. Por último, un bloque opcional de tareas (*watcher* {}), permite ejecutar tareas en paralelo. El lenguaje del programa generado por la interfase es el lenguaje Script[1].

Nuestro compilador toma como entrada un programa, con la estructura indicada arriba, realizado desde la interfase gráfica. El primer paso en la construcción de la herramienta, fue especificar la gramática que genera los programas que se pueden obtener desde la interfase. El diseño general del compilador sigue los lineamientos básicos de [9], referencia obligada en el tema.

El compilador cuenta con cuatro etapas bien definidas. La primera de ellas es la que corresponde al análisis léxico. En esta primera etapa, el compilador lee el programa de entrada caracter a caracter identificando componentes léxicos. La segunda etapa es la que corresponde al análisis sintáctico. En esta fase el compilador, a partir de la secuencia de componentes léxicos, impone una jerarquía de acuerdo a la gramática especificada. La tercera instancia está marcada por el análisis semántico. Aquí se analiza si existen errores semánticos y se genera la información necesaria para la última etapa de este compilador. Esta última fase genera el código intermedio LASM, teniendo en cuenta la información generada en la etapa anterior.

El programa de entrada del compilador es recorrido en su totalidad, realizando las verificaciones sintácticas y semánticas. Conjuntamente, se genera el programa de salida en código LASM. En el recorrido del programa fuente se genera una tabla de símbolos que almacena, tanto los nombres de las variables, como el de los eventos, sensores y motores. De esta tabla se extrae la información necesaria que es utilizada para generar el código LASM. A su vez, la información contenida por la tabla también es usada para realizar el análisis semántico.

5. Conclusiones

Hemos podido comprobar, después de varias pruebas, que los programas realizados desde la interfase Lego son tomados correctamente por el compilador y que se genera el código LASM correspondiente. De esta manera, usando una interfase visual como la del Lego, se puede programar robots físicos sin introducir código de programación. La interfase visual permite al usuario independizarse de ciertos detalles de los lenguajes de programación.

Las distintas salidas generadas por el compilador muestran a los programas generados desde la interfase gráfica en código LASM. Cada una de estas salidas fue enviada al robot a través del software MindScript[8] y fueron ejecutadas con éxito. Esta característica permite programar los robots desde el ambiente visual de Lego sin necesidad de introducir código.

Un trabajo futuro es realizar un ambiente visual de prueba que tome el código LASM generado desde el compilador y simule el funcionamiento que el robot realizaría. De esta manera, no será necesario un robot físico para realizar las pruebas de lo programado desde la interfase. Un trabajo adicional a considerar sería modificar la etapa inicial del *front-end* del compilador para permitir tomar como entrada programas desde alguna otra interfase visual.

Referencias

[1] The Lego Mindstorms Scout Software Developers. htt://minstorms.lego.com/scoutsdk/default.asp LASM byte code. Lego P-Brick Script Code Lenguage.

[2] Technical Suport Lego http://www.legomindstorms.com/help

[3] Robots, Virtual Reality Touted as Mine-Safety Solutions.

http://news.nationalgeographic.com/news/2006/01/0127_0601227_mines.html

[4] RCX Internals http://grafics.stanford.edu/~kekoa/rcx/

[5] Francisco Armando Dueñas Rodriguez. La Robótica. Universidad La Salle. http://www.monografias.com/trabajos6/larobo/larobo.shtml#intro

[6] Samuel Candelas Rodriguez. Lenguaje de programación de los robots. Escuela Campus Aragon. http://www.monografias.com/trabajos3/progrob/progrob.shtml

[7] Lego MindStorms. http://www.mindstorms.lego.com

[8] ScriptEd - MindScript Editor. Lego MINDSTORMS SDK 2.0. htt://minstorms.lego.com/scoutsdk/default.asp

[9] Aho, Sethi and Ullman. *Compilers. Principles, Techniques and Tools*.