EVI 12: Arquitectura de Subsunción

Miguel Angel Astor Romero

16 de octubre de 2016

Outline

Objetivos

Conocer, comprender y practicar el uso de la arquitectura de subsunción como patrón de diseño para Inteligencia Artificial reactiva.

Objetivos Específicos

- Presentar el paradigma de Inteligencia Artificial reactiva.
- Presentar la arquitectura de subsunción como patrón de diseño para Inteligencia Artificial.
- Practicar la implementación de un controlador reactivo basado en la arquitectura de subsunción en el lenguaje Java.

I. A. Reactiva

- La Inteligencia Artificial se puede catalogar de dos maneras:

I. A. Reactiva

- El agente reacciona directamente a su percepción del mundo.
- No suele ser necesario almacenar una representación abstracta del mundo.

I. A. Deliberativa

- El agente construye y mantiene una representación abstracta del mundo, que usa para deliberar sobre su siguiente acción.

Historia de la Arquitectura de Subsunción

- La arquitectura de subsunción es una manera de estructurar controladores para robots y agentes inteligentes propuesta por Rodney Brooks en 1986.
- Se define como un patrón de diseño arquitectónico para la estructuración e implementación de controladores reactivos, asociada a una filosofía de diseño enfocada en producir comportamientos emergentes.
  - La idea es combinar múltiples comportamientos simples para producir comportamientos globales más complejos.

Arquitectura de capas

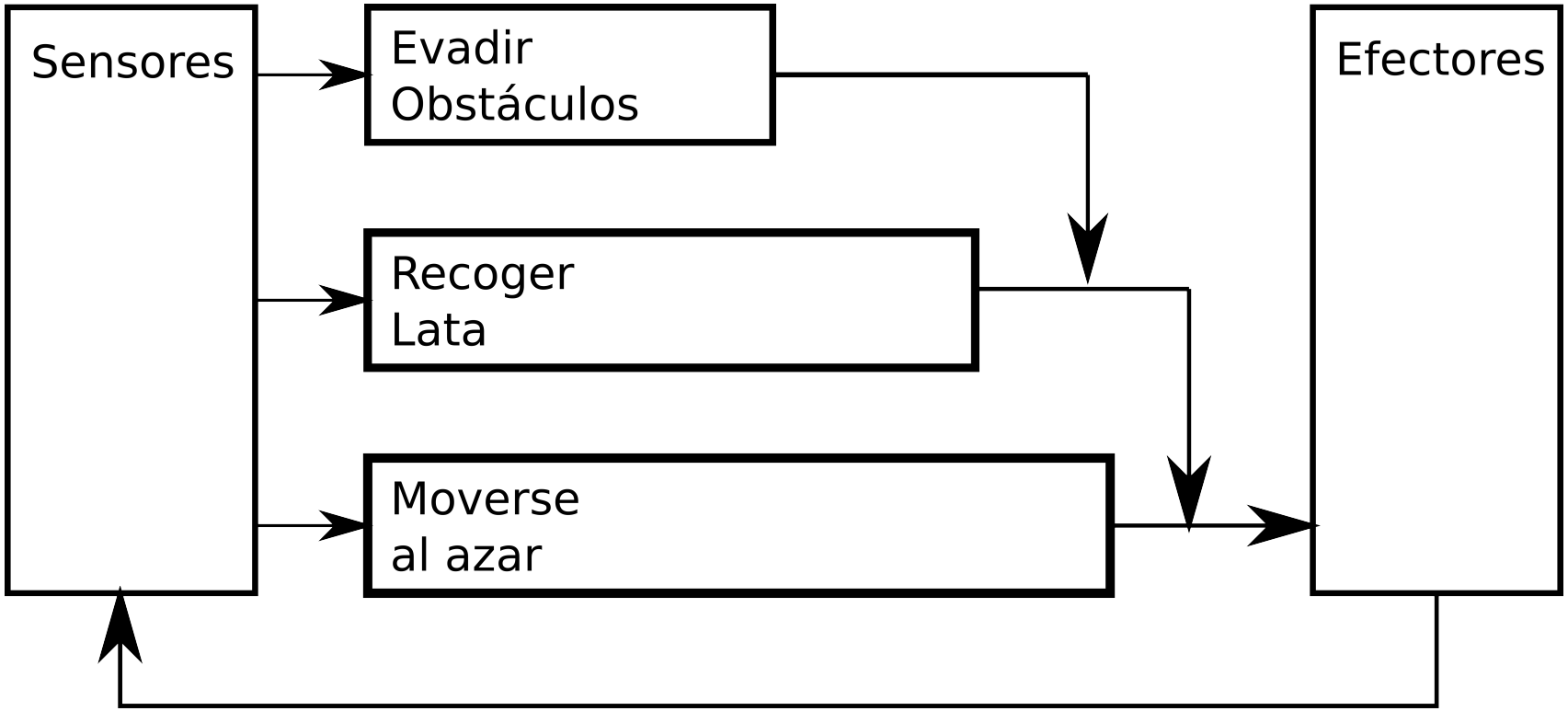
- La arquitectura se estructura en capas.
  - Cada capa es un comportamiento.
- Idealmente todas las capas se ejecutan en paralelo.
- Todas las capas reciben información de todos los sensores simultaneamente.
- En base a las medidas sensadas, se decide cual capa toma el control del agente en cada instante dado.
- Si un sensor capta una nueva medida antes que la anterior sea procesada, esta última se descarta.

Autómatas Finitos Aumentados

- Las capas se implementan internamente como máquinas de estados finitos (MEF).
- Las MEF se aumentan con la capacidad de almacenar información en variables propias.

Visualmente

Supongamos que tenemos un robot que recolecta latas de refresco vacías.



Determinación de la acción a realizar

El estructurar la arquitectura por capas tiene sus razones:

- Las capas forman una jerarquía con capas superiores e inferiores.
- El nivel de cada capa determina su prioridad.
- Cualquier capa superior puede suprimir la acción de una capa inferior en ejecución en cualquier momento si determina que es imperativo que tome el control del agente.
  - De esta funcionalidad es que se deriva el nombre de la arquitectura, las capas superiores "subsumen" a las capas inferiores.

Estilos de implementación

Estilo Brooks

- Cada capa corresponde a un hilo de ejecución.
- Se acualizan los sensores de forma centralizada, y cualquier capa los puede consultar.
- Las capas establecen lineas de comunicación con los efectores.
- La subsunción de una capa consiste en cortar temporalmente dicha linea de comunicación.

Estilo LeJOS

- Se define un árbitro que controla la ejecución y supresión de las capas.
- El sistema posee solo dos hilos de ejecución:
  - El árbitro de subsunción.
  - La capa actualmente en control del agente.
- La subsunción de una capa consiste en detener su hilo de ejecución.

Diseño del Arbitro de Subsunción

```
import java.util.List;
import java.util.LinkedList;

public class SubsumptionController {
    List<SubsumptionBehavior> m_behaviors;

    public void addBehavior(SubsumptionBehavior behavior);
    public void removeBehavior(SubsumptionBehavior behavior);
    public void start();
}
```

Diseño de los Comportamientos

```
public abstract class SubsumptionBehavior {
    public abstract boolean takeControl();
    public abstract void action();
    public abstract void supress();
}
```

Arquitectura de Subsunción en LeJOS

Ahora procederemos a una revisión del código fuente del módulo de subsunción de LeJOS.

Simplificaciones al modelo de LeJOS

Es posible simplificar la arquitectura para evitar tener que lidiar con múltiples hilos de ejecución, aplicando los siguientes cambios:

- Los comportamientos son ejecutados directamente por el controlador.
  - Es necesario que el método `action()` de los comportamientos se ejecute en un tiempo finito preferiblemente corto.
- Eliminar el método `supress()` de los comportamientos.
  - Cuando un comportamiento debe entrar en acción, el controlador simplemente comienza a ejecutar su método `action()` en lugar del correspondiente en el comportamiento suprimido.

Ejercicio

Dada la forma en que trabaja RoboCode, trabajaremos con el modelo simplificado. El objetivo es programar un robot controlado por una arquitectura de subsunción con las siguientes capas (ordenadas de inferior a superior):

- Avanzar en linea recta 200 píxeles, luego rotar al azar 30 grados a la izquierda o derecha.
- Al percibir a otro robot, moverse un poco hacia este.
- Al estar a 200 píxeles o menos de distancia con el otro robot, comenzar a disparar.
- Al chocar con una pared, retroceder 100 píxeles y girar 180 grados.

Referencias

- R. Brooks, "**A robust layered control system for a mobile robot**", IEEE journal on robotics and automation, vol. 2, no. 1, pp. 14–23, 1986.
- R. C. Arkin, Behavior-based robotics. MIT press, 1998.
- S. J. Russell y P. Norvig, "**Artificial intelligence: a modern approach**", 3a edición, Prentice Hall, 2009.

Contactos

Prof. Miguel A. Astor

- miguel.astor@ciens.ucv.ve
- miguel.a.astor@ucv.ve

¿Preguntas?