FAIA: Framework para un Agente de software que resuelve problemas de Inteligencia Artificial

Jorge Marcelo Roa
CIDISI - Universidad Tecnológica Nacional, Facultad Regional Santa Fe

Abstract
Este trabajo presenta un framework que permite resolver problemas de Inteligencia Artificial (IA) utilizando agentes basados en objetivos. El framework, denominado FAIA: Framework para un Agente que resuelve problemas de IA, presenta las interacciones necesarias entre ambiente y agente pudiendo especificar las características particulares para un problema dado. Con el uso de esta herramienta el alumno centra su atención en la definición de las propiedades del agente y su estrategia para decidir qué acción emprender, sin tener que preocuparse por el simulador del ambiente en el cual el agente se desenvuelve. Este framework fue desarrollado dentro de la cátedra Inteligencia Artificial, dictada en la UTN-FRSF en la carrera Ing. en Sistemas de Información, en el cual se le propone a los alumnos el desarrollo de un agente de software. El objetivo de este trabajo es proveer una herramienta flexible con la cual el alumno pueda aprender en la construcción de agentes inteligentes, comprendiendo la relación de éste con su ambiente. De esta forma pueden proponerse problemas a resolver más complejos, requiriendo menos tiempo en la implementación de una solución.

Palabras Clave
agente de software, inteligencia artificial, patrones de diseño, framework

Introducción

En la cátedra Inteligencia Artificial (IA) dictada en la carrera Ingeniería en Sistemas de información de la UTN-FRSF, uno de los objetivos que se persigue es que el alumno sea capaz de aplicar técnicas de IA en la resolución de problemas.

Así, desde la asignatura se trata de proveer oportunidades (en forma de situaciones problemáticas a resolver) para que el alumno vea cuáles son los problemas típicos de la IA y cómo éstos pueden ser resueltos desde un punto de vista ingenieril. Una de las actividades que se propone es el desarrollo de un agente de software que debe desenvolverse (interactuar) en un cierto ambiente (recibiendo percepciones y ejecutando acciones sobre el mismo) para tratar de lograr su objetivo de diseño.

El problema que surge aquí es que, para problemas complejos, los alumnos no logran su resolución en el tiempo que se les asigna, dado que no sólo tienen que tomar decisiones acerca del diseño del agente (tema desconocido para ellos hasta entonces) sino que también se requiere la simulación del ambiente donde el agente va a ser probado. Ante estos inconvenientes, se deben plantear problemas más simples que sí puedan ser tratados en tiempo y en forma.

Distintos autores proponen variadas definiciones de qué es un agente [1][2]. Según la definición de agente propuesta en [3] adoptada en la cátedra, un agente es todo aquello que puede considerarse que percibe su ambiente mediante sensores y que responde o actúa en tal ambiente por medio de efectores o actuadores. La figura 1 muestra una representación esquemática genérica de un agente según esta definición.

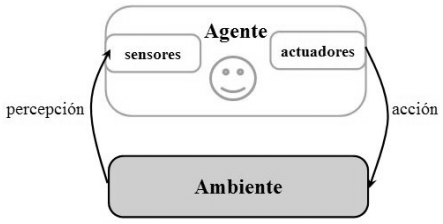

Figura 1: Esquema de un Agente.

En esta figura podemos observar que el agente percibe su entorno a través de sensores que le permiten conocer el universo o ambiente en donde actúa y decidir más acertadamente qué acción emprender sobre el mismo, esta acción será ejecutada por los actuadores modificando el ambiente. El ambiente en el cual se desenvuelve el agente es un elemento

importante en el diseño del mismo, dado que representa la situación problemática que el agente debe resolver.
Los sensores son implementados en hardware cuando el agente trabaja en un entorno físico (p.e. cámara de video, sensores de temperatura) o se simulan por software. La salida de los sensores es una percepción (p.e. imagen, temperatura), es decir, datos que le interesan al agente para llevar a cabo su misión. Por otro lado, el agente ejecuta acciones sobre el ambiente, modificándolo. El diseñador del agente tiene que definir cuáles son esas acciones.
El trabajo de la IA es diseñar el programa del agente. Éste es una función que implementa la relación que establece el agente entre percepciones y acciones. Se asumirá que este programa será ejecutado sobre algún tipo de dispositivo computacional. Las arquitecturas posibles para el programa del agente son realmente arquitecturas de software para sistemas de toma de decisión que están embebidas en un entorno. No todos los agentes son iguales y el diseñador debe seleccionar la arquitectura más conveniente de acuerdo a sus objetivos de diseño.
Una de las actividades prácticas que se le solicita a los alumnos es construir un agente basado en objetivos para resolver problemas, para comprender como éste se relaciona con el mundo en donde se desenvuelve y para comprender cómo, utilizando las técnicas para solución de problemas presentadas en la materia, va tomando decisiones sobre qué acciones ejecutar para lograr su objetivo de diseño: resolver un problema. Comparado con otros tipos de agentes, se puede decir que la implementación de este tipo de agente es de complejidad media.
En este trabajo se propone un framework para la solución de problemas académicos, orientado a que el alumno se familiarice con la temática de agentes y vea también plasmada en forma práctica y didáctica la interacción entre arquitecturas de software y sistemas de agente. La mayoría de los frameworks disponibles, de uso libre, no están orientados a la enseñanza de arquitecturas de agentes, o fueron específicamente desarrollados para modelar las relaciones entre sistemas multi-agente como JADE [4] o entre agente móviles, como Aglets [5], lo cual agrega complejidad al momento de querer definir simplemente un solo agente. Otro ejemplo es Zeus [6], una librería de componentes de software para desarrollo de agentes reactivos, que usan reglas y envío de mensajes. Sin embargo, ninguno de estos trabajos está orientado a la enseñanza académica, no poseen la flexibilidad suficiente que permita modelar agentes que se mueven en distintos tipos de ambientes, ni incorporan explícitamente los conceptos de IA que son necesarios para el diseño de un agente basado en objetivos resolvedor de problemas con búsqueda. Es por eso que se propone el framework FAIA explicado en detalle más adelante. A continuación el trabajo se organiza de la siguiente manera: se hace una introducción sobre agentes basados en objetivos que resuelven problemas, luego se explica la arquitectura del framework, sus clases y los patrones que fueron utilizados, mediante una instanciación del framework con un ejemplo clásico de IA y por último se presentarán las conclusiones y trabajos futuros.

Agente basado en objetivos que resuelve problemas

Un agente basado en objetivos necesita información sobre el objetivo que persigue, la cual describe las situaciones deseables. Este tipo de agente, para decidir qué hacer, tiene en cuenta el estado del ambiente así como también la descripción de su objetivo. De esta manera, para seleccionar la acción necesaria para cada momento, combina la información sobre el objetivo y el conocimiento del resultado de las posibles acciones.
En la toma de decisión, tiene en cuenta dos aspectos fundamentales:
i. ¿Qué ocurre si se ejecuta tal o cual acción? (resultado de las acciones)

ii. ¿Cuánto más lo complace cada acción? (utilidad)

En algunas situaciones, el resultado de una acción alcanza el objetivo deseado en forma directa. Pero en otros casos, el agente tiene que realizar algún proceso más complejo de búsqueda para determinar la secuencia de acciones que lo lleven a alcanzar su objetivo. Búsqueda y planificación son técnicas de IA encargadas de encontrar esta secuencia de acciones que le permitan al agente alcanzar sus objetivos.

Como muestra la figura 2, el agente tiene una memoria o estado interno que representa el conocimiento que tiene del entorno y de su experiencia en él. Cuando percibe, actualiza esta memoria con nuevo conocimiento, y elige la mejor acción basándose en la misma. Luego, vuelve a actualizar su estado interno de acuerdo al conocimiento que posee de cómo la acción modificará al ambiente. Finalmente retorna la acción elegida para que los efectores hagan su trabajo.

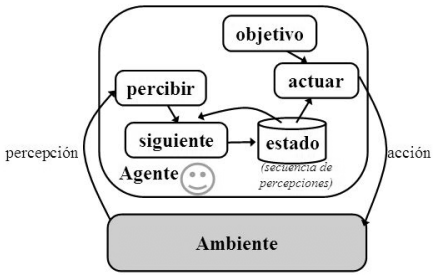

Figura 2: Esquema de un Agente basado en Objetivo.

En forma grupal, los alumnos deben definir un agente de este tipo, es decir, un agente que no tiene conocimiento del universo o ambiente en el cual interactúa pero que sí mantiene en su estado interno la secuencia de percepciones que recibe de éste para poder aprender de ellas y tomar mejores decisiones. En particular se les solicita que esta toma de decisiones se realice con búsqueda. Este trabajo práctico se divide en etapas, siendo la primera la definición de la arquitectura del agente de software, que será implementado en Java™, lo cual permite aprovechar las ventajas de portabilidad, software libre, escalabilidad y reuso que brinda este lenguaje.

Desde el punto de vista de la Ingeniería de software, con el objetivo de uniformar el diseño de la solución antes de pasar a su implementación y evitar errores en la etapa de análisis y diseño, se propone desde la cátedra un framework de agente de software, que será explicado en detalle en la siguiente sección, el cual fue diseñado aplicando patrones de diseño [7].

Desde el punto de vista de la IA, otro de los objetivos de este trabajo es el de disminuir la brecha (que a menudo a los alumnos les es difícil superar) entre resolver un problema de búsqueda en una hoja y codificar esa solución en un lenguaje de programación de tal manera que sea algo casi transparente.

En este trabajo, en particular, se muestra la arquitectura propuesta para un agente resolvedor de problemas con estrategias de búsqueda. Similares arquitecturas serán diseñadas para otros tipos de agentes tales como agentes lógicos, agentes que planifican, etc.

FAIA: Framework de Agente que resuelve problemas de IA

El framework propuesto FAIA para un agente de software que resuelve problemas de la IA, combina tanto conceptos de ingeniería de software, como son los patrones de diseño, y conceptos de IA, como son las distintas estrategias de búsqueda de soluciones en un espacio de estados.

Con este trabajo, pretendemos proponer a los alumnos un ejemplo concreto de aplicación de la ingeniería de software, definida según la IEEE como "*la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software*"[8]. Esta aplicación en concreto se ve realizada en el ámbito de la IA, para el desarrollo de un agente basado en objetivos que resuelve problemas.

Las clases dentro de FAIA están agrupadas en paquetes: *Agent*, *Util*, *Problem* y *Solve*.

En esta sección se describe la utilización del framework FAIA a través de un caso de estudio, el cual consiste en un agente-tortuga que se mueve en un tablero, como muestra la figura 3, y que debe encontrar un camino para pasar por las celdas, según ciertas restricciones.

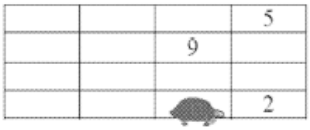

Figura 3: El problema de la tortuga.

El juego consiste en una tortuga que inicia su recorrido en una determinada celda, pasando una sola vez por todas las celdas. Los números que aparecen en el tablero indican el orden que debe seguir la tortuga para recorrer esas celdas. El objetivo del juego consiste en encontrar una ruta que contenga las celdas indicadas en el orden especificado. En el caso presentado la tortuga inicia su recorrido en la celda (4,3). La segunda celda visitada debe ser la (4,4), la quinta celda visitada debe ser la (1,4) y la novena celda visitada debe ser la (2,3).
Para resolver este problema con búsqueda, se deben definir los siguientes elementos:

- ✓ la representación interna (estado) que usará el agente-tortuga,
- ✓ las acciones que podrá realizar el agente-tortuga en el ambiente,
- ✓ el problema (estado + acciones),
- ✓ la prueba de meta (cómo evaluar cada estado para saber si ya llegó a la solución),
- ✓ la estrategia de búsqueda a emplear para encontrar la solución.

Definiendo la representación de estado

El diagrama de clases de la figura 4 muestra las clases del framework que pueden ser usadas para representar el estado del agente ya que son las estructuras de datos provistas. De esta manera se facilita el diseño del mismo, ya que no es necesario preocuparse por el desarrollo de estructuras de datos complejas que no son inherentes al problema del agente desde el punto de vista de la IA, brindando una mayor simpleza al diseñador.

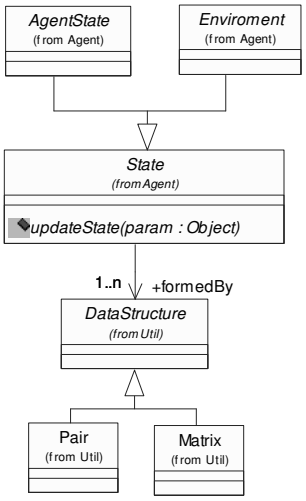

Figura 4: Diagrama de clases en FAIA que permite representar estado

La clase *EstadoTortuga* (figura 5) es usada por el agente para representar su estado. La misma fue definida usando las estructuras de datos propuestas en el framework. Esta clase está compuesta por un par. Su primer elemento es una matriz que representa el tablero, donde cada celda tendrá un valor que indica el orden en el cual el agente-tortuga la visita. El segundo elemento del par es otro par que representa la posición del agente dentro del tablero, identificada con coordenadas *(x,y)*.

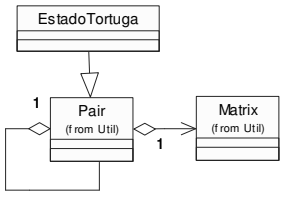

Figura 5: Diagrama de clases que representa el estado del agente

Definiendo el problema

Un agente basado en objetivos necesita información de la meta que persigue. El programa del agente puede combinar información sobre la meta y el resultado de cada una de las acciones posibles, para decidir cómo ir seleccionando acciones que lo lleven a alcanzar su objetivo.

En particular, un objetivo y un conjunto de medios para alcanzarlo se denomina *problema*. El diagrama de clases de la figura 6 muestra la representación de esta definición en el framework FAIA.

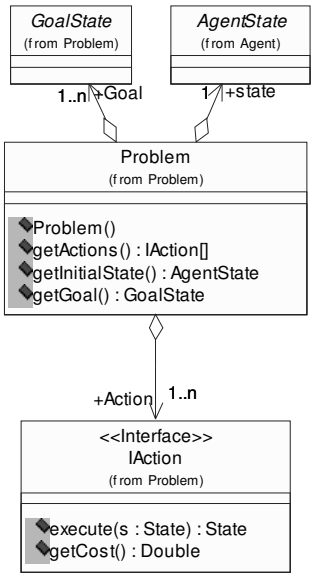

Figura 6: Diagrama de clases que representa el problema asociado a un agente.

Para generar una instancia de un problema es necesario definir un objetivo (*GoalState*), un estado inicial del agente (*AgentState*) e implementar las acciones (operadores) que efectúa el agente sobre el ambiente (interfaz *IAction*). En la figura 7 se muestran las extensiones e implementaciones que permiten definir el problema del agente-tortuga.

La clase *MetaTortuga* extiende a *GoalState*. En esta clase se debe definir el estado particular en el cual se espera que el agente alcance el estado objetivo. La clase *EstadoTortuga* extiende a *AgentState*. En la misma se debe definir el estado inicial del agente. Finalmente tenemos las acciones Arriba, Izquierda, Derecha y Abajo las cuales implementan la interfaz *IAction.* Cada una de estas clases representan una acción que puede realizar el agente-tortuga sobre el tablero. Las mismas deben implementar el método *execute*() de tal manera que la ejecución de la acción se vea reflejada en el estado.

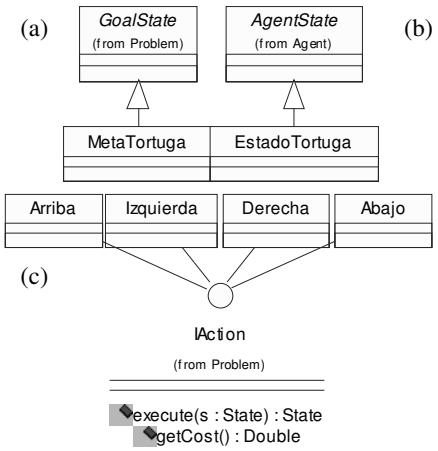

Figura 7: (a) Meta del problema. (b) Estado inicial del agente. (c) Conjunto de acciones del agente

Definiendo la prueba de meta

La prueba de meta es una función utilizada por el agente para determinar si un estado alcanzado es un estado objetivo. Esta función recibe un estado como argumento y retorna un valor booleano.

La figura 8 muestra la clase *CeldasEnOrden*, la cual implementa la interfaz *IGoalTest* que representa la prueba de meta utilizada por el agente-tortuga que se está definiendo.

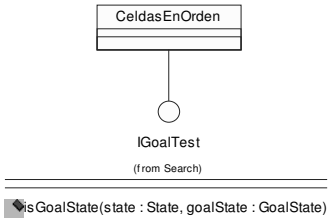

Figura 8: Implementación de la interfaz IGoalTest.

En esta implementación se debe definir el método *isGoalState*() el cual, basándose en un estado y una meta, debe determinar si se ha alcanzado el estado objetivo.

Definiendo las percepciones

Las percepciones le permiten al agente conocer el mundo donde se mueve cuando el mismo no es totalmente accesible [3]. Al recibirlas del ambiente, actualiza su estado interno y, de esta manera, debe decidir cuál es la mejor acción a ejecutar que le permita alcanzar el estado objetivo. En el framework FAIA se definió la clase

abstracta *Perception* para representar las percepciones.

Definiendo la estrategia

En el proceso de decir que acción emprender, el agente utiliza alguna estrategia de búsqueda. En el framework propuesto se definen las estrategias vistas en la cátedra tales como amplitud, profundidad, A* y costo uniforme (figura 10). En la representación se utiliza el patrón de diseño *Strategy* [7] que permite cambiar la estrategia en forma dinámica. Así es posible comparar los resultados obtenidos al aplicar distintas estrategias en la búsqueda de la solución.

Definiendo el agente

Una vez definido todos los elementos que un agente necesita, finalmente se debe extender la clase *GoalBasedAgent* propuesta en el framework que define el comportamiento mínimo que debe tener un agente basado en objetivos.

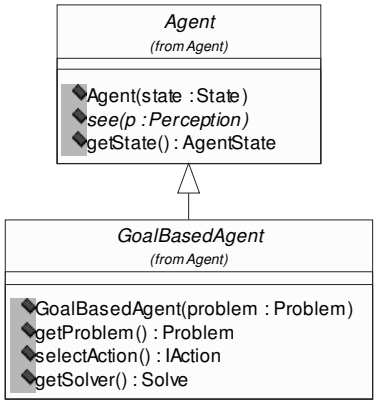

Figura 11: Diagrama de clases que representa un agente basado en objetivos.

Conclusiones

Se presentó un framework orientado a resolver problemas académicos en la enseñanza de Inteligencia Artificial en una carrera de Ingeniería. Se desarrolló un ejemplo mostrando cómo el framework fue instanciado para resolver un problema particular.

Se destaca que no sólo los alumnos se ven beneficiados en el uso de este framework sino que los docentes también encuentran sus ventajas dado que la visualización de los árboles de búsquedas pueden ser usados en el desarrollo de las clases.

Las principales ventajas en el uso de esta propuesta son:

- es una guía teórico-práctica que orienta al alumno a crear sin error los elementos que conforman un agente.
- sólo se deben definir cuestiones relacionadas con la resolución de un problema con búsqueda (estado, operadores, estrategias, etc.) ya que la estructura básica del agente ya está propuesta y se facilita la codificación.
- es posible comparar los resultados obtenidos al aplicar distintas estrategias en la búsqueda a la solución. El uso del patrón *strategy*, permite cambiar la estrategia en forma dinámica.
- es posible visualizar los distintos árboles de búsqueda que son generados durante el proceso de alcanzar el objetivo que permite al alumno entender dicho proceso.
- como asistencia al trabajo docente, en clase, es posible armar los árboles de búsqueda dinámicamente y mostrar como se van generando los mismos según las diferentes estrategias.

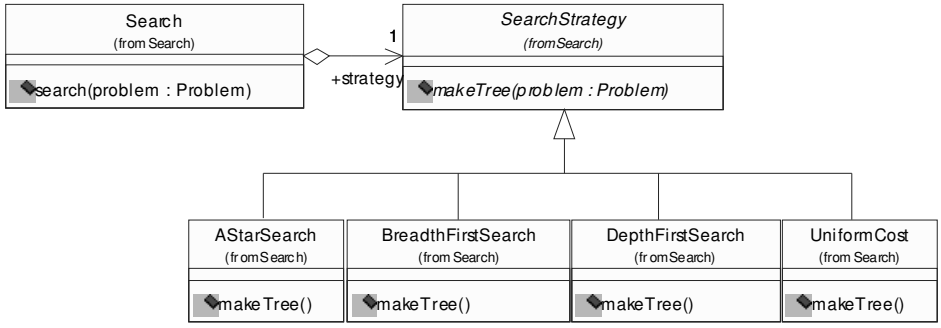

Figura 10: Definición de la estrategia de búsqueda mediante el patrón *Strategy*.

Trabajo futuro

Como trabajo futuro se propone:

- almacenar los árboles de búsqueda resultantes en formato XML, de manera tal de facilitar su visualización en cualquier explorador de la web.
- Incorporar otras estrategias que le permitan al agente resolver problemas tales como planificación.
- Incluir al framework como plugin de la herramienta de desarrollo *Eclipse*[9] creando un ambiente todo integrado donde el alumno pueda escribir el código para el agente particular que está diseñando y ejecutarlo.

Referencias

[1] M. Wooldridge (2002) An Introduction to Multiagent Systems. John Wiley & Sons, New York.

[2] P. Maes (1995) Intelligent Software, Scientific American, vol. 273, No. 3, pp. 84-86.

[3] S. Russell, P. Norvig (2002) Artificial Intelligence: A Modern Approach. Ed. Prentice-Hall.

[4] F.L. Bellifemine, G. Caire, D. Greenwood (2006) Developing Multi-Agent Systems with JADE. Ed. Wiley & sons, New York.

[5] Y. Aridor, D.B. Lange (1998) Agent Design Patterns: Elements of Agent Applications Design. Proceedings of the Second International Conference on Autonomous Agents, pp. 108-115.

[6] J. Collis, D. Ndumu, C. van Buskirk (2000) The Zeus. Agent Building Toolkit - ZEUS Methodology Documentation.http://labs.bt.com/projects/agents/zeus/

[7] E. Gamma, R. Helm, R. Johnson, J. Vlissides (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Ed. Addison-Wesley Professional Computing Series.

[8] http://standards.ieee.org/software/

[9] http://eclipse.org

Agradecimientos

El autor quisiera agradecer a las profesoras Milagros Gutierrez y Georgina Stegmayer, ambas docentes de la cátedra de IA en la UTN-FRSF, por su colaboración y guía en la redacción de este trabajo.

Datos de Contacto:

Jorge Roa. Centro de Investigación y Desarrollo de Ingeniería en Sistemas de Información (CIDISI), Universidad Tecnológica Nacional, Facultad Regional Santa Fe. Lavaise 610 (3000) Santa Fe. E-mail: jroa@arnet.com.ar