Análisis e Implementación del lenguaje AgentSpeak(L) para agentes inteligentes con arquitectura BDI [1]

Santiago Giusti
sgiusti@argentina.com

Alejandro J. García
ajg@cs.uns.edu.ar

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Avda. Alem 1253 – C. P. 8000 – Bahía Blanca
Tel: (0291) 4595135 – Fax: (0291) 4595136

Palabras clave: Inteligencia Artificial, Agentes Inteligentes, Arquitectura BDI.

El objetivo de este trabajo es presentar una línea de investigación en el área de agentes inteligentes. Esta investigación se basa en el estudio de la arquitectura BDI (belief-desire-intention) y en particular del lenguaje de programación de agentes AgentSpeak(L). En este trabajo presentaremos un análisis del lenguaje AgentSpeak(L) mostrando características salientes y algunas falencias. Se presentará además una implementación reducida y simplificada de AgentSpeak(L) realizada en Prolog.

Descripción de AgentSpeak(L)

AgentSpeak(L) es un lenguaje de programación basado en un lenguaje de primer orden con acciones y eventos. AgentSpeak(L) fue diseñado por Anand S. Rao en [1] para poder caracterizar agentes bajo la arquitectura BDI [2] y para cubrir las distancias entre los resultados teóricos y los problemas prácticos que el modelo BDI presenta. AgentSpeak(L) es una abstracción de dos sistemas BDI que han sido implementados: PRS (Procedural Reasoning System) [3] y dMARS (Distributed Multi-Agent Reasoning System) [4]. A continuación se describirá brevemente los conceptos principales de AgentSpeak(L), más detalles pueden encontrase en [1, 5, 6].

Un agente programado en AgentSpeak(L) podrá actuar en un entorno cambiante donde recibirá continuamente percepciones acerca del entorno en el que se desenvuelve. Como AgentSpeak(L) está basado en la arquitectura BDI, el agente hará un análisis de sus actitudes mentales representadas por los estados de información (creencias), de motivación (deseos o metas) y deliberativo (intenciones). En base a este análisis y con el fin de lograr sus metas, el agente podrá decidir que acciones realizar, las cuales afectaran al entorno.

Específicamente en Agentspeak(L), los agentes disponen de un conjunto de creencias y un conjunto de planes, donde estos últimos se activan mediante eventos que se producen debido a cambios o estímulos en el entorno del agente. Los planes organizan de manera jerárquica la ejecución de metas (indicadas por el disparo de eventos) y las acciones del agente. Informalmente, un agente AgentSpeak(L) consiste de:

- un *conjunto de creencias* B, que incluye hechos concernientes a las propiedades y características del dominio de aplicación (es decir, una representación del mundo), como así también a los nuevos hechos que son generados por acción del agente;

[1]Financiado parcialmente por SeCyT Universidad Nacional del Sur (subsidio: 24/ZN09) y por la Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 Nro 13096)

- un *conjunto de planes* P, que es un repositorio que contiene los planes disponibles para usar por el agente. Estos planes están pre-compilados y no requieren ningún tipo de planificación en tiempo de ejecución por parte del agente[2];
- un *conjunto de eventos* E, que almacena las percepciones del agente acerca de los cambios en su entorno y la solicitud de metas a cumplir;
- un *conjunto de acciones* A, que son las que pueden modificar el estado del entorno. Las acciones están explícitamente ordenadas en el cuerpo de cada plan para su futura instanciación y ejecución;
- un *conjunto de intenciones* I, donde una intención es una pila de planes parcialmente instanciados[3] y donde un pedido de ejecución fue realizado;
- una *función de selección de eventos* $\mathcal{S}_\mathcal{E}$, que elige un evento para procesar de un conjunto de eventos E. Los eventos son acumulados en E a medida que se perciben;
- una *función de selección de plan aplicable* $\mathcal{S}_\mathcal{O}$, que es la responsable de elegir un plan de la librería de planes P y
- una *función de selección de intención* $\mathcal{S}_\mathcal{I}$, que elige una intención para ejecutar del conjunto de intenciones actuales I.

En la figura 1 puede verse gráficamente el ciclo que representa el funcionamiento de un agente en AgentSpeak(L).[4]

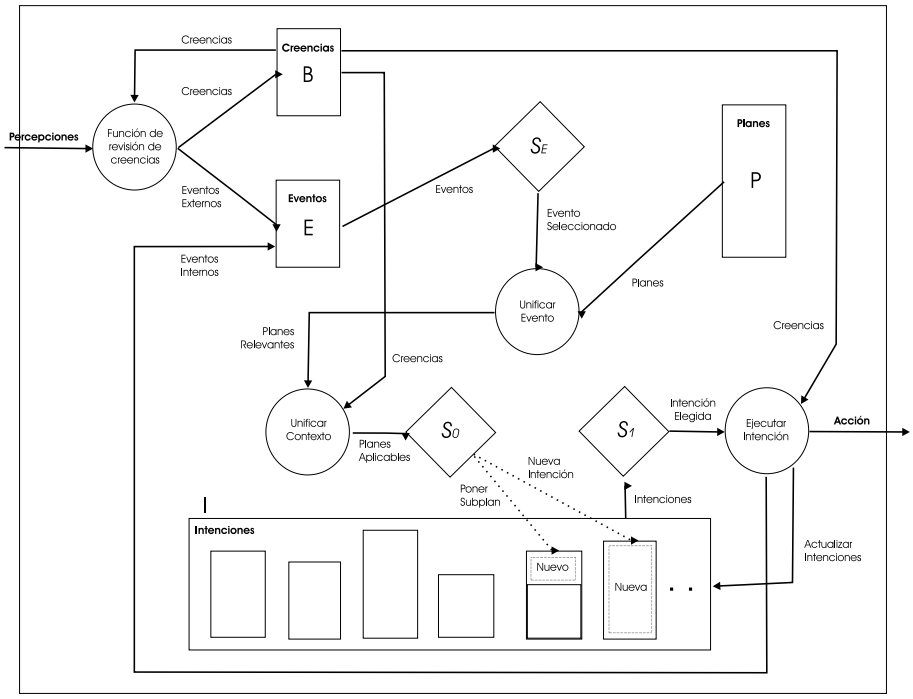

Figura 1: Gráfico del intérprete de AgentSpeak(L) [6].

[2]La tarea de generar planes automáticamente por el agente no está contemplada en AgentSpeak(L).

[3]Un plan parcialmente instanciado es una serie de *medios propuestos* que representa una copia del plan original, ahora bajo el rol de *actitud mental* que dirige el comportamiento. Notar la diferencia con el plan como *norma* de comportamiento, que está en el conjunto P.

[4]La función de revisión de creencias no es parte de la definición original de AgentSpeak(L). Este agregado pertenece al trabajo de Machado *et al.* en [6]

Cuando el agente nota un cambio en el entorno o cuando un usuario externo requiere que el sistema adopte una meta, se genera un evento disparador apropiado. Este tipo de eventos son denominados como **eventos externos**. Un agente también puede generar eventos internos. Un **evento interno** es aquel que se dispara por estar indicado en el conjunto de fórmulas a ejecutar de un plan. Los eventos, internos o externos, son asincrónicamente agregados al conjunto de eventos E.

La función de selección $\mathcal{S}_{\mathcal{E}}$ selecciona del conjunto E un evento para procesar. Este evento es removido de E y es usado para unificar con los eventos disparadores de los planes del conjunto P. Los planes cuyos eventos disparadores así unifican son llamados **planes relevantes** y el unificador es llamado **unificador relevante**. Luego, el unificador relevante es aplicado al **contexto** de los planes relevantes obtenidos. En un plan, la condición de contexto indica cuál debe ser la situación del agente y de su dominio para que las acciones del mismo puedan ser ejecutadas. Luego, al aplicar el unificador relevante sobre los contextos se obtiene una sustitución de respuesta correcta sobre los planes, tal que el contexto es una consecuencia lógica del conjunto de creencias base B. Los planes que sustituyen satisfactoriamente su contexto son llamados como **planes aplicables** u opciones y la composición del unificador relevante con la sustitución de respuesta correcta es llamada **unificador aplicable**.

Por cada evento puede haber muchos planes aplicables u opciones. La función $\mathcal{S}_{\mathcal{O}}$ elige uno de estos planes. Aplicando el unificador aplicable a la opción elegida se obtienen los medios propuestos para responder al evento disparador. Cada **intención** es una pila de planes parcialmente instanciados o de marcos de intención. En el caso de un evento externo, los medios propuestos son usados para crear una nueva intención, la cual es agregada al conjunto de intenciones I. En el caso de un evento interno para agregar una meta, los medios propuestos son puestos en el tope de una intención existente (la que generó el disparo del evento interno).

Luego, la función de selección $\mathcal{S}_{\mathcal{I}}$ selecciona una intención para ejecutar. Cuando el agente ejecuta una intención, ejecuta lo que se encuentra en el tope del cuerpo de la intención. En el tope puede haber tres tipos de fórmulas: metas de logro y de testeo y acciones. La ejecución de una **meta de logro** es equivalente a generar un evento interno para agregar dicha meta a la intención actual. La ejecución de una **meta de testeo** es equivalente a encontrar una sustitución para dicha meta que la haga consecuencia lógica de las creencias base. Si tal sustitución es encontrada, la meta de testeo es removida del cuerpo del tope de la intención y la sustitución es aplicada al resto del cuerpo del tope de la intención. Si se encuentra una acción entonces ésta se ejecuta y se elimina del cuerpo del tope de la intención.

El agente ahora vuelve al conjunto de eventos E y el ciclo entero continúa hasta que no haya más eventos en E o hasta que no queden intenciones ejecutables.

AgentWhisper(L): Una implementación en Prolog de AgentSpeak(L)

El objetivo de este trabajo es investigar la aplicabilidad de los resultados teóricos desarrollados para la arquitectura BDI. Para el estudio de AgentSpeak(L) se decidió realizar una implementación en Prolog que permita disponer de un prototipo para analizar las características de este lenguaje. A continuación se describe muy brevemente a AgentWhisper(L), una implementación en Prolog de un subconjunto de AgentSpeak(L). Cabe destacar que existen otros análisis e implementaciones de AgentSpeak(L), como por ejemplo los mencionados en [5, 6].

En AgentWhisper(L), las creencias, los planes, los eventos y las intenciones son representadas mediante los predicados dinámicos `bel\1`, `pln\3`, `evt\2` y `int\2` respectivamente, con el objetivo de poder agregarlos o suprimirlos a lo largo del ciclo de ejecución.

Las creencias que representan el dominio del agente se representan de la forma `bel(b(t))` donde `b` es un predicado y `t` representa un término o una serie de términos `t1,...,tn`.

Los eventos son representados con la siguiente estructura:

```
evt(EventoDisparador, NombreIntencion).
```

donde `EventoDisparador` es un predicado de la forma `+!g(t)` y `NombreIntencion` refiere mediante el nombre, a la intención que generó dicho evento disparador. En el caso de que fuera un evento externo, el nombre de la intención es `trueIntention`. En caso contrario, `NombreIntencion` es `intention_X`, donde `X` es un número natural.

La estructura de un plan es la siguiente:

```
pln(EventoDisparador, Contexto, Cuerpo).
```

donde `EventoDisparador` es una fórmula, `Contexto` es una lista de predicados de creencia que de ser verdaderos en el momento de la selección del plan, determinan la aplicabilidad del mismo. `Cuerpo` es una lista de predicados indicando acciones o metas de logro que, por motivos de implementación, su último elemento es `true`.

El conjunto de acciones A es específico del dominio de aplicación del agente. Igualmente, se puede decir que las acciones se representan mediante predicados de la forma

```
nombreAccion(ListaParametros):- <definicionAccion>
```

En este punto es importante aclarar algunas cuestiones de implementación. Debido a la complejidad de formular una función de revisión de creencias, la actualización de la base de datos de creencias y la posible generación de sus eventos indicadores respectivos, queda a cargo de la implementación de las acciones.

Cada intención se representa como una pila de planes instanciados. Para lograr el efecto pila, se utiliza el predicado dinámico `int\2` para indicar un elemento de la pila/intención. La inserción y supresión de elementos se logra mediante el uso del predicado `asserta\1` y del `retractall\1` respectivamente. Un elemento de la pila es un plan con sus variables instanciadas.

```
int(NombreIntencion, pln(EventoDisparador, Contexto, Cuerpo)).
```

Las funciones de selección del modelo, al igual que las acciones, dependen del dominio de aplicación del agente. En AgentWhisper(L), la función $\mathcal{S}_\mathcal{E}$ utiliza un criterio de pila de eventos para seleccionar uno; es decir, el primer evento en entrar es el último evento en salir. La función de selección de planes u opciones aplicables $\mathcal{S}_\mathcal{O}$ se comporta de manera similar. $\mathcal{S}_\mathcal{O}$ toma el primer plan aplicable que encuentra en la base de datos.

Para la función de selección de intenciones $\mathcal{S}_\mathcal{I}$ se especificó un comportamiento un poco más inteligente. Al momento de elegir una intención para ejecutar en un determinado ciclo del agente, $\mathcal{S}_\mathcal{I}$ evaluará cuál de las intenciones disponibles en I es la más conveniente.

Luego de que la función $\mathcal{S}_\mathcal{I}$ eligió una intención para ejecutar, el intérprete de AgentWhisper(L) analiza qué tipo de fórmula es la que se encuentra primera en el cuerpo del plan tope de la intención. Si la primera fórmula es una meta de logro, el accionar del intérprete no varía. Se coloca un evento en el conjunto E correspondiente a la meta de logro recién evaluada y se actualiza la intención removiendo esta fórmula del cuerpo del plan tope. Ahora, si la primera fórmula es una acción, el intérprete la ejecuta, luego la remueve y evalúa la nueva fórmula que ahora está primera en el cuerpo del plan. Si es una acción, repite este proceso nuevamente. Si es una meta de logro, entonces el intérprete abandona esta intención y continúa con el ciclo habitual de ejecución.

En resumen, las fórmulas de un plan de una intención serán ejecutadas de corrido siempre y cuando dichas fórmulas sean todas acciones. De esta manera se previene que la secuencia u ordenamiento de las intenciones a ejecutar no influya en el resultado de las acciones.

Conclusiones y trabajo futuro

Gracias a la implementación de AgentWhisper(L), se pudo ver con mayor claridad algunos puntos oscuros que se encuentran tanto en las definiciones del modelo BDI como en la formalización de AgentSpeak(L). Uno de los puntos no considerados por Rao en [1] es el hecho de no incluir explícitamente el proceso de revisión de intenciones que Bratman *et al.* en [2] describen como el proceso de filtrado. Quizá AgentSpeak(L) permita esta funcionalidad con el manejo de los distintos tipos de metas de logro pero, de todas formas, esta consideración queda a cargo del programador de agentes y por lo tanto, fuera de la arquitectura. Este problema, conlleva a la ejecución de acciones incoherentes o fuera de contexto cuando hay más de una intención en el conjunto de intenciones I.

Otro punto que merece atención es la falta de definiciones operacionales de las funciones de selección $\mathcal{S}_\mathcal{E}$, $\mathcal{S}_\mathcal{O}$ y $\mathcal{S}_\mathcal{I}$. Si bien es cierto que son elementos de la arquitectura que son *específicos del dominio de aplicación* de cada agente, los autores que hablan del modelo AgentSpeak(L) [7, 1, 5] se abstraen de definirlas, a excepción de Machado *et al.* [6] que brinda una especificación de las funciones extremadamente simple para la implementación de AgentSpeak(L) que realiza. El argumento contra la no explicación del accionar de las funciones se debe a que son procesos del agente donde reside la codificación de una gran parte de su *accionar inteligente*. Por lo tanto, el hecho de no definir más específicamente la operatoria de dichas funciones hace que los mecanismos de análisis de oportunidades y los mecanismos de revisión de Bratman *et al.* [2] no estén concretamente visibles en AgentSpeak(L).

Como trabajo futuro se planea avanzar en dos direcciones: extender a AgentWhisper(L) para cubrir todos los aspectos de AgentSpeak(L), y estudiar otros desarrollos teóricos y prácticos de la arquitectura BDI.

Referencias

[1] Anand S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In Rudy van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

[2] Michael E. Bratman, David Israel, and Martha Pollack. Plans and resource-bounded practical reasoning. In Robert Cummins and John L. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 1–22. The MIT Press, Cambridge, Massachusetts, 1991.

[3] M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit (Michigan), 1989.

[4] M. Luck M. d'Inverno, D. Kinny and M. Wooldridge. A formal specification of dMARS. In A. Rao M. Singh and M. Wooldridge, editors, *Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages*, pages 155–176. Springer-Verlag, 1998.

[5] Mark d'Inverno and Michael Luck. Engineering AgentSpeak(L): A formal computational model. *Journal of Logic and Computation*, 8(3):233–260, 1998.

[6] Rodrigo Machado and Rafael H. Bordini. Running AgentSpeak(L) agents on SIM_AGENT. *Lecture Notes in Computer Science*, 2333:158–174, 2002.

[7] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems*, San Francisco, 1995.