

Implementación de la Arquitectura BDI basada en el Cálculo de Situaciones

Robert Demolombe¹ and Pilar Pozos Parra²

¹ ONERA-Toulouse

2, Avenue Edouard Belin BP 4025, 31055 Toulouse Cedex 4, France

Robert.Demolombe@cert.fr

² Benemérita Universidad Autónoma de Puebla

Río Verde y Avenida San Claudio, Puebla, Mexico

ppozos@cfm.buap.mx

Resumen. La arquitectura BDI (Beliefs, Desires and Intentions) Creencias, Deseos e Intenciones ha sido aceptada por gran parte de la comunidad científica para modelar agentes, su ventaja principal es que permite representar el comportamiento racional de agentes de una manera simple. En este trabajo presentamos una implementación "simple" de las tres nociones (Creencias, Deseos e Intenciones) basada en el cálculo de situaciones. Hasta donde sabemos no existen implementaciones eficaces de la arquitectura BDI. Nuestro enfoque ha podido ser implementado utilizando las ventajas de la programación declarativa. Mencionamos algunos resultados de un programa hecho en Prolog concerniente a la generación de planes.

Palabras Clave: Inteligencia Artificial, Agentes, Arquitectura BDI, Lógica, Generación de Planes.

1 Introducción

Algunos autores han propuesto formalizar las nociones de la arquitectura BDI bajo el marco de la lógica modal [6, 9, 10, 7, 8, 4]. Aunque este enfoque es intuitivo y elegante, sobreestima las capacidades de razonamiento de los agentes. Por ejemplo, su estructura define únicamente agentes "sábelo todo" cuyo comportamiento es el siguiente: si el agente sabe "p" entonces el agente conoce todas las consecuencias lógicas que se siguen de p (logical omniscience). En la vida real este tipo de agentes no existen. Además la implementación de las lógicas modales ha sido ampliamente estudiada y su complejidad es conocida. Por otro lado, se tienen los enfoques que resuelven problemas similares pero que no definen inferencias entre los aspectos cognitivos.

Un enfoque que intenta encontrar un balance entre el poder de expresión y la posibilidad de implementación de escenarios reales es mostrado en [3]. Este utiliza el cálculo de situaciones como teoría de base.

El cálculo de situaciones permite representar, de una forma muy "simple", la evolución del mundo real [5], así como la evolución del mundo de creencias de los

agentes [2]. En [3] una extensión del formalismo es presentada para considerar también las creencias del pasado y del futuro. Aunque la extensión considera tanto pasado como futuro, nuestro interés porta sobre el futuro dado que el agente basa su comportamiento sobre proyecciones de situaciones futuras, es decir sobre un modelo del mundo que es capaz de manipular para evaluar las consecuencias de sus actos y seleccionar las acciones a emprender. Esta toma de decisión se presume racional, es decir que corresponde a los objetivos o metas del agente.

[3] presenta también esquemas formales para representar la evolución de metas e intenciones. Aquí presentamos la relación que existe entre dichos esquemas y una implementación válida en Prolog. Como muchas implementaciones del cálculo de situaciones, ésta considera sólo cierto tipo de teorías, las teorías completas.

El documento está dividido como sigue: iniciamos con una breve descripción del modelado de agentes racionales, seguida de la propuesta para representar la evolución del mundo real, creencias, metas e intenciones, posteriormente describimos los resultados de una implementación hecha para el caso de un robot móvil y finalmente comparamos nuestro enfoque con otros existentes y mencionamos algunas posibles extensiones.

2 Modelando agentes bajo la arquitectura BDI

En la actualidad el término agente es ampliamente usado en computación. Este puede tener diferentes niveles de concepción desde un simple conjunto de líneas de código que se ejecuta automáticamente cada vez que una condición es satisfecha hasta un sofisticado programa que puede razonar acerca de su propio comportamiento y satisfacer sus metas en un lapso de tiempo. Nuestro interés porta sobre un tipo particular de agentes, los racionales.

Los agentes racionales son entidades de software que perciben su ambiente utilizado sensores, poseen un modelo y pueden razonar acerca de su ambiente, basados en sus estados mentales toman decisiones y realizan acciones que modifican su ambiente. Este tipo de agentes han sido propuestos para aplicaciones complejas tales como: recuperación inteligente de información, modelar diferentes estrategias de combate, control y monitoreo de terminales aéreas, diagnóstico de fallas de naves espaciales, etc.

En 1987, Bratman presenta el trabajo filosófico [1] que ha tenido la mayor influencia en la concepción de agentes racionales. Este hace énfasis en el papel que juega la intención dentro del razonamiento práctico humano. Argumentando que la intención no puede ser representada utilizando otro tipo de estados mentales, define tres actitudes mentales irreducibles: creencias, deseos e intenciones. Las creencias capturan el estado de información del agente, los deseos afectan las acciones de un agente y las intenciones controlan las acciones que son tomadas por el agente.

En la arquitectura BDI, la intención es el concepto complejo a modelar. Las intenciones consideran un análisis racional de la situación presente y de

las situaciones futuras posibles así como las acciones causantes del cambio de situación. Los agentes cognitivos son capaces de prever, en cierta medida, las consecuencias de las acciones que han elegido llevar a cabo.

Cuando un agente decide realizar una secuencia de acciones es porque el supone que dicha ejecución va a permitirle satisfacer sus metas, por lo tanto, toda teoría de la intención necesita una buena definición de la noción de acción, así como de estados del mundo y de cómo éstos evolucionan gracias al efecto una acción (hecho que lleva implícito la representación del tiempo), en consecuencia existe una doble dificultad: la manipulación de operadores temporales y la consideración de los aspectos concernientes la noción de acción. El cálculo de situaciones permite manejar ambos aspectos de manera formal cuya interpretación intuitiva es simple.

3 BDI basada en el cálculo de situaciones

La metodología empleada es intuitivamente simple: se proponen esquemas de axiomas que permiten modelar la evolución de las tres nociones y posteriormente se utilizan metapredicados para representar cada noción en Prolog. La transcripción en Prolog de los esquemas es, en cierto sentido, literal, es decir, dado un axioma representado con conectivas lógicas (\wedge , \vee , \neg), éste es escrito en formato Prolog reemplazando \wedge por ";", \vee por ";" y \neg por "not" (negación por falla). Cabe resaltar que esta última al no coincidir con la negación clásica (\neg) nos impone la condición de tratar únicamente con teorías completas (ver [5]).

Antes de presentar los esquemas de axiomas para las creencias, deseos e intenciones, mostramos los esquemas que permiten representar el cambio en el mundo real que puede no coincidir con el mundo imaginado por el(los) agente(s). De esta manera podemos tener una representación del mundo real y simular el desenvolvimiento de los agentes en este mundo "real".

3.1 Evolución del mundo real

Tomemos el ejemplo del robot que avanza y retrocede, el hecho que el robot esté en la posición x en la situación¹ s es representado por: $posicion(x, s)$. Supongamos que en toda situación, si se realiza la acción de *avanzar*, el resultado es que el robot está en x si estaba en $x - 1$ y si se realiza la acción de *retroceder*, el resultado es que el robot está en x si estaba en $x + 1$. Además si realiza la acción de *avanzar* o *retroceder*, el resultado es que el robot no está en x si estaba en x . Expresado formalmente por el axioma de estado sucesor ("successor state axiom" en [5]) de *posicion*:

$$\begin{aligned} - \text{posicion}(x, do(a, s)) \leftrightarrow & [(a = \text{avanzar} \wedge \text{posicion}(x - 1, s)) \vee \\ & (a = \text{retroceder} \wedge \text{posicion}(x + 1, s))] \vee \\ & (\text{posicion}(x, s) \wedge [\neg a = \text{avanzar} \wedge \neg a = \text{retroceder}]), \end{aligned}$$

Con este podemos calcular el cambio, así si tenemos $posicion(5, S_0)$, podemos deducir $\neg \text{posicion}(5, do(\text{avanzar}, S_0))$ y $posicion(6, do(\text{avanzar}, S_0))$. Lo

¹ Una situación puede verse de manera intuitiva como un instante de tiempo.

que significa que después de avanzar el robot ya no está en la posición 5 sino más bien en la posición 6.

Como puede verse la implementación en Prolog es "casi" literal, salvo el manejo de variables para representar las situaciones.

```
position(X,do(A,S)):- position(Y,S),
    ( (A=avancer,X is Y + 1) ;
      (A=reculer,X is Y - 1) );
    ( position(X,S), \+A=avancer, \+A=reculer ) .
```

3.2 Evolución de creencias

Una extensión a este formalismo que se aplica a las creencias de varios agentes es presentada en [2], la cual introduce "modalidades"² para representar las creencias.

En general, un agente puede tener cuatro actitudes correspondientes a sus creencias con respecto a un propiedad. Consideremos la presencia de otro agente, el piloto del robot, el cual al observar el tablero de control del robot y ver que el robot está en x cree que la posición de éste es x o bien que no cree que la posición no es x . Por otro lado, el hecho de observar y ver que el robot no está en la posición x , implica que el piloto no crea que la posición es x o bien que crea que el robot no está en x , representado formalmente por los axiomas de estado sucesor de las creencias del piloto con respecto a $posicion(x, s)$ y $\neg posicion(x, s)$:

$$\begin{aligned}
 - B_p(posicion(x, do(a, s))) &\leftrightarrow a = obs_pos \wedge posicion(x, s) \\
 &\quad \vee B_p(posicion(x, s)) \wedge \neg(a = obs_pos \wedge \neg posicion(x, s)), \\
 - B_p(\neg posicion(x, do(a, s))) &\leftrightarrow a = obs_pos \wedge \neg posicion(x, s) \\
 &\quad \vee B_p(\neg posicion(x, s)) \wedge \neg(a = obs_pos \wedge posicion(x, s)).
 \end{aligned}$$

Podemos extender la notación para considerar las creencias en general (pasadas, presentes y futuras). Así $B_i(p(s_1), s_2)$ denota el "predicado especial" $B_i p(s_1, s_2)$ que debe leerse como en la situación s_2 el agente i cree que el predicado p es verdadero en la situación s_1 . En [3] se introducen los esquemas de axiomas extendidos, donde eventualmente aparecen acciones de información que son una generalización de las acciones de monitoreo (sensing actions) que permiten realizar la revisión de creencias. Utilizando la noción de creencias generalizadas se pueden calcular las proyecciones a futuro.

3.3 Evolución de metas

Para representar la evolución de las metas, extendemos el lenguaje con predicados de metas tales como $G_i p(s)$ denotado por $G_i(p(s))$ que significa intuitivamente que en la situación actual s , el agente i tiene como meta que p sea

² En realidad son predicados especiales formados por un "operador" B_i y un predicado p así, $B_i p(s)$ debe leerse como la creencia de i con respecto a p en s y es denotado por $B_i(p(s))$. $B_i \neg p(s)$ debe leerse la creencia de i con respecto a $\neg p$ en s y es denotado por $B_i(\neg p(s))$.

verdadera en una situación futura. De manera similar a las creencias se pueden obtener los axiomas de estado sucesor de las metas del robot con respecto a $posicion(x, s)$ y $\neg posicion(x, s)$, representados por:

- $G_r(posicion(x, do(a, s))) \leftrightarrow a = decide.pos(x) \vee G_r(posicion(x, s)) \wedge \neg a = decide.non.pos(x)$,
- $G_r(\neg posicion(x, do(a, s))) \leftrightarrow a = decide.non.pos(x) \vee G_r(\neg posicion(x, s)) \wedge \neg a = decide.pos(x)$.

La significación intuitiva del primero es: si el robot decide estar en la posición x o bien ya tenía dicha meta y no decide no estar en x entonces su meta es de estar en la posición x . Similarmente para el segundo. Conceptos tales como la indiferencia ($\neg G_r(posicion(x, s)) \wedge \neg G_r(\neg posicion(x, s))$) de un agente puede ser representada utilizando estas modalidades.

3.4 Evolución de intenciones

La intención es un concepto que permite relacionar las metas con las creencias y los compromisos (pactos) de un agente. La intención supone implícito un mecanismo de planificación, es decir un mecanismo que anticipe situaciones futuras. Decimos que un agente i tiene la intención de realizar una secuencia de acciones $T = [a_1, a_2, \dots, a_n]$ en una situación s para satisfacer su meta p ($\neg p$), $I_i p(T, s)$ ($I_i \neg p(T, s)$) abreviado como $I_i(T, p(s))$ ($I_i(T, \neg p(s))$) si ciertas condiciones se cumplen (ver [1]). Representamos la evolución de la intención del robot respecto a su posición con:

$$\neg I_r(T, posicion(x, do(a, s))) \leftrightarrow G_r(posicion(x, do(a, s))) \wedge [(a = comprometer \wedge Bfposs_i(posicion(x, do(T, s)), s)) \vee I_i([a, a_1, \dots, a_n], posicion(x, s)) \vee I_i(T, posicion(x, s))]$$

Intuitivamente significa que el robot tiene la intención de realizar la secuencia de acciones $T = [a_1, \dots, a_n]$ que permitirán que su posición sea x si y sólo si ciertas condiciones se cumplen. Un axioma similar se introduce para la intención de no estar en la posición x .

4 Generación automática de Planes

Como mencionamos anteriormente la implantación de los esquemas se puede transcribir de manera simple a código Prolog. La validez de implantación, para los axiomas de estado sucesor, es demostrada por Reiter en [5], utilizando una metodología similar y metapredicados puede extenderse al caso de las creencias, deseos e intenciones.

Algunos resultados obtenidos de las ejecuciones del programa que representa el caso del robot son: si al inicio el robot está en la position 1, hay un obstáculo en 5 y el robot cree que está en 1 y que hay un obstáculo en 4 y además tiene como meta estar en la posición 6, cuyos hechos en Prolog son representados por: `posicion(1,s0). obstacle(5,s0). br(position(1,s0)). br(obstacle(4,s0)). gr(position(6,s0))`. Si deseamos saber cuál es la intención del robot respecto a su posición, hacemos la siguiente pregunta `Ir(T, posicion(X, do(comprometer, s0)))`., el programa responde con `T=[avancer, avancer, quita.obs, avancer, avancer, avancer]`,

cuya interpretación es: avanzar dos veces, quitar el obstáculo y finalmente avanzar tres veces. Note que el plan que hizo el robot va fallar pues tiene una creencia que no corresponde con la realidad, es decir cree que el obstáculo está en 4 sin embargo está en 5. Si ahora cambiamos `br(obstacle(4,s0))` por `br(obstacle(5,s0))`, el resultado es `T=[avancer,avancer,avacer,quita.obs,avancer,avancer]`, que corresponde a un plan "seguro".

5 Conclusión y posibles extensiones

Hemos presentado una implementación de la arquitectura BDI que permite verificar diferentes escenarios, cabe resaltar que dicha implementación considera también la simulación del mundo real, con lo cual se tiene la oportunidad de probar el modelo antes de que los agentes se enfrenten a condiciones reales y se puedan corregir las actitudes de dichos agentes sin necesidad de dañar entidades reales (equipo o personal). Otra característica de la implementación es su facilidad para la obtención de planes, dados los esquemas este proceso es un resultado colateral del esquema de la intención. Una de las ventajas que posee respecto a otras implementaciones, es el hecho de considerar agentes "normales", agentes que se puedan equivocar. Como pudo comprobarse en los ejemplos expuestos, está permitido que el agente posea creencias que no correspondan con la realidad y por ende que sus actitudes sean erróneas. Un trabajo futuro considera la extensión del formalismo para considerar el caso en el que si el agente observa que está en un error pueda corregirlo bajo el límite de sus posibilidades.

References

1. M. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.
2. R. Demolombe and P. Pozos Parra. A simple and tractable extension of situation calculus to epistemic logic. In Z. W. Ras and S. Ohsuga, editors, *Proc. of 12th International Symposium ISMIS 2000*. Springer. LNAI 1932, 2000.
3. R. Demolombe and P. Pozos Parra. BDI architecture in the framework of Situation Calculus. In *Proc. of International Joint Conference on Artificial Intelligence, Mexico, 2003* (to appear).
4. A.N. Rao and M.P. Georgeff. Modeling Rational Agents within a BDI Architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 1991.
5. R. Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. Technical report, University of Toronto, 1999.
6. M. P. Singh. *Multiagent Systems. A Theoretical Framework for Intentions, Know-How, and Communications*. Springer LNAI 799, 1994.
7. M. P. Singh, A. Rao, and M. Georgeff. Formal method in dai : Logic based representation and reasoning. In G. Weis, editor, *Introduction to Distributed Artificial Intelligence*, New York, 1998. MIT Press.
8. B. van Linder. *Modal Logics for Rational Agents*. PhD thesis, University of Utrecht, 1996.
9. M. Wooldridge. *Reasoning about Rational Agents*. MIT Press, 2000.
10. M. Wooldridge. *Introduction to Multi Agent Systems*. J. Wiley and Sons, 2002.