

PRACTICA 2 - INTELIGENCIA ARTIFICIAL

GPROLOG

INDICE

1. Introducción al problema y aproximación mediante agentes basados en conocimiento y LPO.
2. Aproximación Gprolog. Implementación de la LPO mediante Gprolog.
3. Explicación de los resultados obtenidos. Traza Gprolog.
4. Conclusiones.

Introducción al problema y aproximación mediante agentes basados en conocimiento y LPO.

Antes de comenzar con el análisis del problema que nos ocupa, comenzaremos tratando las bases sobre las que se asienta la solución de este problema, los agentes basados en conocimiento y la lógica proposicional.

El agente basado en conocimiento es el agente indicado para problemas en los que sea necesaria la capacidad de hacer uso de la lógica para llegar a la solución de un problema concreto, esto engloba una serie de características.

Es necesaria la capacidad de sintetizar y combinar información para adquirir nuevo conocimiento, esta actualización del conocimiento que posee mediante el uso de la lógica amplía su perspectiva sobre el entorno haciendo posible que sus capacidades y comportamiento varíen conforme el agente avanza en la resolución del problema.

Los agentes basados en conocimiento tienen un componente principal axiomáticamente necesario, la base de conocimiento KB (Knowledge base).

Este componente está formado por sentencias, todas representadas en el lenguaje de representación del conocimiento, en este caso concreto trataremos con la lógica de primer orden o LPO. Estas sentencias representan afirmaciones del entorno en el que trabaja el agente.

La lógica de primer orden es como hemos visto anteriormente un lenguaje de representación del conocimiento. Es un lenguaje muy popular ya que cuenta con un enfoque de la realidad en el cual entran hechos objetos y relaciones entre estos.

Este lenguaje cuenta con todos los elementos necesarios para la representación del conocimiento ya que además de contar con constantes predicados y funciones también tenemos a nuestra disposición las variables, que nos permiten la representación de un objeto genérico que nos servirá para construir aserciones generales.

El problema propuesto para esta práctica es el siguiente:

“Todos los coches que sean de la categoría A, tengan una incidencia en el motor y sean antiguos han de pasar una revisión. Los coches 1,2 y 3 son de categoría A mientras que el coche 4 es de categoría B. Todos los coches en los que el sensor S1 y el sensor S2 estén desactivados tienen una incidencia. Los coches 1 y 3 tienen el sensor S1 desactivado y los coches 2 y 3 tienen el sensor S2 desactivado. Para que un coche sea considerado antiguo debe tener el certificado de coche de interés histórico. Los coches 1 y 3 tienen el certificado de interés histórico.”

A continuación, representaremos la base de conocimiento del supuesto agente basado en conocimiento mediante lógica de primer orden.

1. categoría(A, x) Λ incidencia(x) Λ antiguo(x) → revisión(x)
2. categoría (A, C1)
3. categoría(A, C2)
4. categoría(A, C3)
5. categoría(B, C4)
6. sensorDesactivado(S1, x) Λ sensorDesactivado(S2, x) → incidencia(x)
7. sensorDesactivado(S1, C1)
8. sensorDesactivado(S1, C3)
9. sensorDesactivado(S2, C2)
10. sensorDesactivado(S2, C3)
11. certHist(x) → antiguo(x)
12. certHist(C1)
13. certHist(C3)

Utilizaremos el algoritmo de encadenamiento hacia adelante para la resolución de este mediante LPO.

Inicio – Ciclo 1

* No puede extraerse conocimiento de 1).
* Unificando C3/x Se agrega 14) incidencia(C3) por 6), aplicando 8) y 10).
* Unificando C3/x Se agrega 15) antiguo(C3) por 11), aplicando 13).
* Unificando C1/x Se agrega 16) antiguo(C1) por 11), aplicando 12).

Ciclo 2

* Unificando C3/x Se llega a la **respuesta revision(C3)** por 1), aplicando 4), 14) y 15).

Respuesta:

Revisión(C3)

Aproximación Gprolog. Implementación de la LPO mediante Gprolog.

A continuación, se muestra la traducción de las sentencias de LPO anteriores a lenguaje prolog:

revision(X):-categoria(a, X), incidencia(X), antiguo(X).

categoria(a, c1).

categoria(a, c2).

categoria(a, c3).

categoria(b, c4).

incidencia(X):-sensorDesactivado(s1, X), sensorDesactivado(s2, X).

sensorDesactivado(s1, c1).

sensorDesactivado(s1, c3).

sensorDesactivado(s2, c2).

sensorDesactivado(s2, c3).

antiguo(X):-certHist(X).

certHist(c1).

certHist(c3).

Contamos con 3 reglas y el resto son sentencias atómicas no derivadas, es decir axiomas. Cabe destacar el cambio en sintaxis, tanto en el uso de mayúsculas y minúsculas como en el cambio de conectores.

Conectores:

Λ -- ,

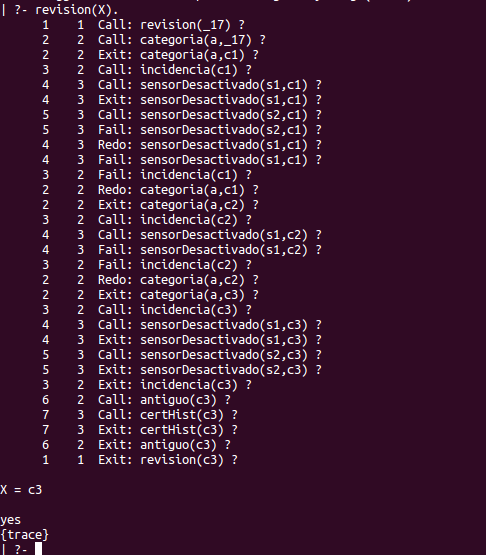
V -- ;

→ -- :-

Las variables en el caso de Gprolog hay que escribirlas con mayúscula, las constantes y relaciones han de comenzar por minúscula y todas las sentencias han de acabar en un punto.

La aproximación al problema en lenguaje prolog es bastante parecida a la que se tomó mediante lógica proposicional, contamos con variables que utilizamos para sentar aserciones generales sobre hechos, lo que permite, en este caso al intérprete Gprolog, descubrir nuevo conocimiento mediante el uso de la lógica.

Explicación de los resultados obtenidos. Traza Gprolog.

En este punto veremos la traza, o el camino recorrido por la herramienta prolog hasta llegar a la solución. La traza que nos ofrece el programa es la siguiente:

El lenguaje prolog utiliza una mezcla de dos técnicas para solucionar las problemáticas que se le presentan, una es la unificación, que fue utilizada anteriormente en la resolución mediante encadenamiento hacia adelante en el primer punto de este trabajo y la segunda es el backtracking.

Es importante destacar el modo en el que prolog aplica estas técnicas, a saber, la herramienta intenta aplicar unificación de forma secuencial en el orden en que las sentencias se encuentran en la base de conocimiento y cuando una unificación no da más de sí en términos de que no va a aportar más conocimiento nuevo válido, aquí es donde entra en juego el backtracking.

La unificación es la búsqueda de una constante dentro de los axiomas o hechos alojados en la base de conocimiento que sustituya o unifique correctamente con las variables de una regla de forma que se pueda adquirir una nueva sentencia válida que añadir a la base de conocimiento, esto es aplicar la lógica para la obtención de nuevo conocimiento, parte fundamental de los agentes basados en conocimiento.

Backtracking es una técnica que como hemos dicho anteriormente entra en juego cuando prolog llega a la conclusión de que no puede continuar con la unificación que estaba realizando, esto es, traducido literalmente “dar marcha atrás” hasta llegar a la sentencia donde comenzó determinada unificación para intentar otra distinta, si no hubiese más unificaciones posibles, el programa acabaría y estaría resuelto con una, múltiples o ninguna solución.

En nuestro caso podemos ver como prolog intenta 3 unificaciones distintas c1, c2 y finalmente c3. Como podemos ver el proceso comienza intentando unificar c1 para categoría a con éxito, pero falla en la siguiente comparación, incidencia(c1), ya que c1 no tiene el sensor s2 desactivado. Aquí prolog utiliza el backtracking para volver hasta la primera comprobación, e intenta empezar a unificar con c2, en este caso hay éxito en la comprobación de categoría, pero al no tener desactivado s1 falla de nuevo en la sentencia incidencia(c2), aquí entra otra vez en juego el backtracking y prolog unifica satisfactoriamente c3 en todas las sentencias, llegando a la solución:

revisión(X) 🡪 X = c3

Conclusiones

Las conclusiones sobre esta práctica pretenden hacer brillar la importancia de los conceptos teóricos dados en clase como son los agentes inteligentes, en este caso concreto estamos tratando con un tipo de agente bastante avanzado, el agente basado en conocimiento. Estos agentes son un concepto abstracto que servirá de base para entender y poder diseñar sistemas en el mundo real.

Uno de los tipos de estos sistemas reales son los sistemas expertos, lo cuales integran el concepto de agente basado en conocimiento en su funcionamiento, esto no quiere decir que los sistemas expertos se limiten a las funcionalidades descritas en los agentes basados en conocimiento, sino que el agente es la base desde la que empiezan a diseñarse sus funcionalidades, el sistema real requerirá de una complejidad mucho mayor. Por ejemplo, el agente basado en conocimiento se conforma con aplicar la lógica sobre una base de conocimiento dada para intentar llegar a una solución, en cambio los sistemas expertos han de ser capaces de, por ejemplo, almacenar todos los resultados de ejecuciones anteriores e incluir estos a sus conocimientos a la hora de razonar en futuras problemáticas, o sintetizar la información de forma que la solución sea inteligible para un usuario que esté utilizando el sistema.

En el caso de nuestro problema, el agente abstracto implementado mediante prolog está claro que es una visión muy simplificada de lo que podría ser un caso real, pero imaginemos que contamos con cientos de sensores y precisas mediciones de ciertos aspectos o componentes de los coches, todo esto podría ser integrado en un complejo agente basado en conocimiento y este a su vez servir como base para la construcción de un sistema experto el cual podría ser una gran ayuda en entornos muy complejos como podrían ser talleres de Fórmula 1 o Nascar donde la precisión y el tener en cuenta el histórico de los datos y los sucesos es muy importante para tanto reparaciones, como mejoras o innovaciones en el futuro.

Estos nuevos sistemas podrían eliminar en gran medida errores humanos que serían inevitables de otra forma, por ahora siempre es necesaria la participación de humanos expertos e ingenieros para la construcción, supervisión y mejora de estos sistemas, pero puede que llegue el día que ninguno de estos puestos sea necesario y con esto la eliminación de todo error humano posible dentro de áreas donde la precisión y la efectividad son clave.