Hacia la definición de una representación para líneas de razonamiento

José Alfonso del Carmen Garcés-Báez

Benemérita Universidad Autónoma de Puebla,
Facultad de Ciencias de la Computación, Puebla,
México

agarces@cs.buap.mx, alfonso.garcesb@gmail.com

Resumen. Son más conocidos los métodos para la representación del conocimiento, tales como redes semánticas, el cálculo de predicados y el cálculo de situaciones que sirvió para introducir el problema del marco en Inteligencia Artificial [3] que los esquemas para representar razonamiento. En este trabajo se define una forma de representar el razonamiento o el plan a seguir para resolver un problema tal y como lo hacemos de forma cotidiana. La estructura que aquí se presenta es una quíntupla que contiene una condición que detona la valoración de evidencias para llevar a cabo ciertas acciones encaminadas al logro de un objetivo. La idea de la presente propuesta, bio-inspirada, es que permita hacer fácil y reducida la programación de tareas con base en la estructura *LR* que se presenta.

Palabras clave: Representación, conocimiento, razonamiento, inteligencia artificial, objetivo, evidencia y programación.

1. Introducción

Una de las tareas importantes que tiene la lógica, es la de encontrar un lenguaje común no sólo a gente de diferentes especialidades sino de diferentes idiomas y culturas. Norbert Wiener dijo en relación a una reunión con diferentes especialistas en México [6]:

Las discusiones eran interesantes y, en realidad, si aprendimos a hablar más o menos en el lenguaje de cada uno; pero existían grandes obstáculos que impedían una comprensión completa. Estas dificultades semánticas residían en el hecho de que en general no existe otro lenguaje que pueda sustituir la precisión de las matemáticas, y de que gran parte del vocabulario de las ciencias sociales es y debe ser empleado para la expresión de cosas que aún no sabemos expresar en términos matemáticos...me he dado cuenta en muchas otras ocasiones, de que uno de los principales deberes del matemático, al fungir como consejero de científicos de campos menos precisos, es el de desanimar a dichos hombres de ciencia de esperar demasiado de las matemáticas...

La experiencia de Wiener en 1944 sigue siendo la experiencia de muchos grupos interdisciplinarios de trabajo en la actualidad y en este sentido proponemos una forma

de representar la solución de problemas considerando el entendimiento y utilización en diversas áreas del conocimiento.

Son frecuentes los casos en los que ejecutamos una *acción* dependiendo del cumplimiento o no de un suceso (*valoración* de *evidencias)* en un momento determinado (*condición en el tiempo*), con el propósito de alcanzar un *objetivo*. Así mismo, no podemos dejar de considerar la posibilidad de que para la solución de un problema puede ser necesaria la solución previa de otros problemas que pudieran ser resueltos uno tras otro o al mismo tiempo.

El modelo que se presenta en este trabajo es recursivo y tiene los elementos que caracterizan a las estrategias de "divide y vencerás" [1]:

a. Divide.- Si el problema lo amerita puede dividirse en "problemitas".
b. Resuelve.- Se busca alcanzar los objetivos particulares.
c. Combina.- Las soluciones parciales ayudan a resolver los objetivos más generales.

2. Desarrollo

2.1 Estado del arte

Son numerosos los esfuerzos realizados en lo que se refiere a la representación del conocimiento y considero que en menor cantidad aquéllos dedicados a la representación del razonamiento por lo cual no debemos perder de vista lo que menciona Brachman [5]:

A widely recognized goal of artificial intelligence (AI) is the creation of artifacts that can emulate humans in their ability to reason symbolically, as exemplified in typical AI domains such as planning, natural language understanding, diagnosis, and tutoring.

Un avance especializado del uso de la representación del razonamiento y del conocimiento se encuentra en [7]:

KnowLang is a formal language providing a comprehensive specification model that must be able to address all the aspects of an ASCENS Knowledge Corpus and eventually some of the ASCENS Knowledge Base Inference Engine (ASCENS – Autonomic Service-Component Ensembles. 2010. http://www.ascens-ist.eu/) .

Every ASCENS Knowledge Corpus is structured into a domain specific ontology [5], logical framework and inter-ontology operators. The domain-specific ontology gives a formal and declarative representation of the knowledge domain in terms of explicitly described domain concepts, individuals and the relationships between those concepts/individuals:

- *facts – define true statements in the knowledge domains that can be used to discover situations;*
- *x rules – express knowledge such as: 1) if H than C; or 2) if H than C1 else C2; where H is hypothesis of the rule and C is the conclusion of the rule;*

- *x constraints – used to validate knowledge, i.e., to check its consistency. Can be positive or negative and express knowledge of the form: 1) if A holds, so must B; or 2) if A holds B must not.*

Por su parte, la programación lógica ha tenido importantes avances que facilitan la implementación de soluciones como se puede ver en [8]:

… At the same time, the fact that the typical applications of logic programming frequently involve irregular computations, make heavy use of dynamic data structures with logical variables, and involve search and speculation, makes the techniques used in the corresponding parallelizing compilers and run-time systems potentially interesting even outside the field.

2.2 Hacia una definición de LR

En [2] se encuentran algunos antecedentes del presente trabajo.

La estructura *LR* representa la posible solución a un problema *P* y se define mediante la quíntupla siguiente:

$$[Cond(t_0), v(e, s), a, o, m],$$

donde:

Cond(t_0) contiene una función booleana, muchas veces en términos del tiempo. También puede contener la inicialización de parámetros previamente a la *Condición* correspondiente.

- *v(e, s)* es una función de entero que valora la(s) evidencia(s), para el cumplimiento o no, del suceso *s* cuando *Cond(t_0)* es verdadera.
- *a* es la acción que se realizará en caso de que *v(e, s)* sea positivo.
- es el objetivo a lograr, puede ser una meta o la impresión de un texto.
- *m* es un arreglo opcional de *P´s* que puede ser vacío o puede verse como una matriz, esta opción se activa en caso de que *v(e, s)* sea negativo, refiriéndose a la posible necesidad de resolver uno o una secuencia de problemas previos.

Es necesario señalar que el valor de las evidencias que nos permite tomar una decisión es relativo y configurable ya que por claridad y conveniencia a veces debemos decidir teniendo evidencias con valor positivo o con valor negativo, es decir, si el valor de las evidencias es igual a lo que esperamos entonces decimos que la evaluación fue positiva en otro caso decimos que fue negativa.

Con respecto al arreglo *m*, se ha dicho que tiene la misma estructura que *P* y podrá tener las configuraciones siguientes:

a. m puede ser vacía (m = []). Esto significa que al cumplirse *Cond(t_0)*, se evaluarán las evidencias y de ser positivas se realizará la acción *a*, si las evidencias son negativas el procedimiento termina o, si se prefiere, se enviará una leyenda como una señal que ayude a la toma de decisiones o nos obligue a un nuevo acuerdo.

En este caso el tiempo total (T) para resolver el problema es igual a t_0 ($T = t_0$).

b. m puede estar compuesta de un solo elemento:

$$[\ [Cond(t_1), v(e_1, s_1), a_1, o_1, []]\ ].$$

Lo cual quiere decir que de ser negativa la valoración de la(s) evidencia(s) en el nivel anterior para realizar la acción *a*, es necesario realizar primero la acción a_1 y por lo tanto se requiere valorar e_1 con respecto a s_1 agotado el tiempo t_1 ($v(e_1, s_1)>0$) para alcanzar el objetivo o_1.

En este caso el tiempo total que tardamos en tomar la decisión para ejecutar la acción *a* es:

$$T = t_0 + t_1$$

c. m, con un renglón, se podrá interpretar como un conjunto de acciones en serie:

$$[$$
$$[Cond(t_{11}), v(e_{11}, s_{11}), a_{11}, o_{11}, [Cond(t_{12}), v(e_{12}, s_{12}), a_{12}, o_{12},$$
$$[\ldots[Cond(t_{1n}), v(e_{1n}, s_{1n}), a_{1n}, o_{1n}, []]\ldots]]]$$
$$]$$

Y en este caso el tiempo total que tardamos en tomar la decisión para realizar la acción *a* es:

$$T = t_0 + t_{11} + \ldots + t_{1n}$$

las $a_{i's}$ son acciones particulares y los $o_{i's}$ son objetivos particulares.

d. m podrá ser interpretado como un conjunto de elementos en paralelo:

$$[$$
$$[Cond(t_{11}), v(e_{11}, s_{11}), a_{11}, o_{11}, [\ldots[Cond(t_{1n}), v(e_{1n}, s_{1n}), a_{1n}, o_{1n}, []]\ldots]],$$
$$\ldots$$
$$[Cond(t_{m1}), v(e_{m1}, s_{m1}), a_{m1}, o_{m1}, [\ldots[Cond(t_{mn}), v(e_{mn}, s_{mn}), a_{mn}, o_{mn}, []]\ldots]]$$
$$]$$

Ahora, el tiempo total que tardamos en tomar la decisión de realizar la acción *a* es:

$$T = t_0 + \max\{t_{11}, \ldots, t_{m1}\} + \ldots + \max\{t_{1n}, \ldots, t_{mn}\}$$

2.3 Aplicaciones

1. Operaciones básicas

Substracción de dos números naturales a través de la suma:

$$R =$$
$$[(a=Rdm(100), b=Rdm(100),\ If\ a \neq b);\ v(e, a > b);\ c = 0,\ While\ (a \geq b+c):c{+}{+};$$
$$Print\ a, \text{“-“}, b, \text{”=”}, c;\ R]$$

- Se generan dos números aleatorios naturales entre 1 y 100, diferentes.
- Se valora la única evidencia ($a > b$). Si es positiva se ejecuta la *acción (While)*, si es negativa la estructura se llama a sí misma.
- Se termina con la impresión del resultado.

2. Sensores y acciones.

[(Sensor_humo_OK), v(e,"Incendio"), Activar: alarma y aspersores, "Dar protección",[]].

Si un sensor detecta cierta densidad de humo, se valoran las evidencias de incendio y en caso de ser positiva la valoración se activa la alarma y los aspersores con el objetivo de dar protección a las personas. Si la valoración de evidencias es negativa no procede acción alguna.

3. Cierre de operaciones de cobranza diaria.

Utilizando la estructura podemos indicar que:

"A las 20:00 hrs. se iniciará la verificación de programas en ejecución (procesos) y cuando todas las terminales hayan concluido sus operaciones se iniciarán los trabajos de cierre diario de la cobranza".

[(System_time = 20:00),v(e, Off_term_all), Close_job.batch, "Concluido el cierre diario de la cobranza",[]].

4. Venta de una casa.

Supongamos que queremos hacer la venta de un bien inmueble en un tiempo determinado, si el valor de la evidencia es diferente a lo que esperamos, es decir negativa, para ejecutar la acción requerimos nuevas evidencias y la valoración de las mismas que pudieran reunirse en forma seriada o paralela con acciones y objetivos particulares con la intención de obtener evidencias positivas que nos lleven a una valoración global positiva y como consecuencia la correspondiente ejecución de la acción principal.

Para indicar que deseamos tener la documentación completa en 25 días de un inmueble y poder buscar cliente para la venta, lo expresamos como:

[(Cond(25_días)), v(e, tener_doc), buscar_cliente, venta_casa,[]].

Si *v(e, tener_doc) > 0*, el valor de las evidencias coinciden con lo que esperamos. Eso quiere decir que "como máximo en 25 días fue posible tener toda la documentación necesaria y podemos buscar cliente para la venta de una casa"

Si *v(e, tener_doc) < 0*, hubiese sido necesario replantear la(s) acción(es) para lograr el objetivo, por ejemplo, de la siguiente manera:

[Cond(25_días), v(e, tener_doc), buscar_cliente, "venta de casa",
[
[Cond(20-días), v(e, tener_doc), tramitar, "eliminación de embargo",[]] $_{11}$
[Cond(7_días), v(e, tener_doc), pagar_serv, "pagos al corriente",[]] $_{21}$
[Cond(15_días), v(e, tener_doc), conseguir_ presupuesto, "gastos notaría",
[Cond(10_días), v(e, tener_doc), trámitar, "liberación de hipoteca", []] $_{32}$*]* $_{31}$
]

Como podemos darnos cuenta, nos vemos obligados a identificar sucesos para reunir nuevas evidencias en ciertos periodos, definir nuevas acciones y objetivos que se pueden ver en algunos casos como componentes de los principales y en otros como ajenos pero con un objetivo general común.

Ahora hemos agregado:

- Tener la documentación para el trámite de eliminación de un embargo en 20 días.
- Tener la documentación para el pago de los servicios y estar al corriente en 7 días.
- Tener la documentación para hacer un presupuesto de gastos notariales y cubrir el requisito de notaria en 15 días, para esto es necesario tener la documentación para el trámite de eliminación de hipoteca que tardará otros10 días.

Ejemplo para el cálculo del tiempo total en días:

T=25+ | 20 | 0 |
| 7 | 0 |
| 15 | 10 |

25 + max{20, 7, 15} + max{0, 0, 10} = 55 días

De un estimado inicial de 25 días pasamos a un estimado de 55 considerando más actividades por realizar. Se ha mencionado que, por la estructura recursiva de *LR*, la ejecución de tareas se puede hacer en serie o en paralelo, iniciando con una "raíz", de tal suerte que el trazo de la ejecución se puede interpretar como una estructura irregular multidimensional tipo fractal donde cada nodo tiene la misma estructura.

2.4 Relación entre *LR* y lenguajes de programación

La forma de escribir la quíntupla *LR* es natural y entendible para las diversas áreas del conocimiento. Una línea de la estructura se tendrá que traducir en más instrucciones en el lenguaje de programación que se tenga al alcance y dicha labor será transparente para quién use la estructura. Como una forma de aproximar la semántica de *LR*, se presenta el ejemplo que sigue.

Recordemos la primera aplicación de *LR* referente a operaciones básicas (1), donde:

$$R = [(a=Rdm(100), b=Rdm(100), \ If\ a \neq b);\ v(e, a > b);\ c = 0,$$
$$While\ (a \geq b+c):c{+}{+};\ Print\ a,"-",b,"=",c;\ R]$$

Esta instrucción se traduce en el siguiente programa en Python Ver. 2.7.6 (http://www.python.org):

```
#                  Programa LR
#     Substracción de dos números naturales
# aleatorios entre 1 y  100 a través de la suma
#                                        GB-JAC
#
# Primer parámetro
def Condicion(a, b):
      if (a != b):
            return 1
      elif (a == 0):
            return 0
#
# Segundo parámetro
```

```
def Verificacion(e, (a, b)):
      if (e==1):
            if (a >b):
                  return 1
            elif (a < b):
                  return -1
#
# Tercer parámetro
def Accion(a, b):
      c=0
      while (a > (b+c)):
            c=c+1
      return c
#
# Cuarto parámetro
def Objetivo(a, b, c):
      print a,"-",b,"=", c
#

# Quinto parámetro
# y procedimiento recursivo
#
def R():
  a, b = random.randint (1, 100), random.randint (1, 100)
  if (Condicion(a, b)):
        if (Verificacion(1, (a, b)) > 0):
              c=Accion(a, b)
              Objetivo(a, b, c)
        elif (Verificacion(1, (a, b)) < 0):
              print a,"-",b," ¡No es un numero natural!"
              R()
```

Casos de prueba:

1. LR.R()
   64 - 70 ¡No es un numero natural!
   62 - 40 = 22

2. LR.R()
   64 - 82 ¡No es un numero natural!
   25 - 92 ¡No es un numero natural!
   69 - 65 = 4

3. LR.R()
   88 - 49 = 39

Con el programa anterior se ejemplifica el uso de todos los parámetros que componen la estructura *LR* incluyendo su propiedad recursiva en los casos en que la *Verificación* de *evidencias* resulta negativa.

3. Conclusiones

Es necesario especificar con mayor precisión la semántica de cada uno de los parámetros que integran la quíntupla *LR* para llegar a la definición formal y a las demostraciones necesarias que fundamentan la misma.

Un trazo de la estructura multidimensional mostrará las posibilidades de uso de los patrones que puede generar *LR*.

La estructura *LR* es independiente del lenguaje de programación que se utilice y el uso de la misma se vislumbra funcional y de relativa facilidad.

La representación para líneas de razonamiento *LR*, es una propuesta para expresar tareas o procedimientos de diversas áreas de conocimiento, en particular, tiene gran potencial en las áreas de planificación como se presenta en [4], en robótica mediante el uso de Robot Operating System (http://www.ros.org/), en la plataforma JdeRobot (www.jderobot.org/) para la programación de Drones y en aplicaciones de las tecnologías emergentes, como las móviles, aprovechando la amplia gama de sensores disponibles (http://developer.android.com/tools/device.html).

Referencias

1. Cormen, Thomas H.: Introduction to Algorithms. MIT (2009)
2. Garcés Báez, JAC.: Interrelación de algunas lógicas intermedias y Answer Set. Tesis de Maestría, FCFM-BUAP (2002)
3. Nilsson, N.J.: Inteligencia Artificial: Una nueva síntesis. McGraw-Hill (2001)
4. Russell, S., Norvig, P.: Inteligencia Artificial: Un enfoque moderno. Prentice Hall (1996)
5. Brachman, R.J., Levesque, H.J.: Knowledge representation and reasoning. Elsevier, San Francisco (2004)
6. Ensayos Científicos. 3ª. Edición de Ciencia y Desarrollo. CoNaCyT (1982)
7. Vassev, E.: Requirements and Initial Model for KnowLang – A Language for Knowledge Representation in Autonomic Service-Component Ensembles. In: C3S2E´11: Proceeding of The Fourth International C* Conference on Computer Science and Software Engineering (2011)
8. Gupta, G.: Parallel execution of prolog programs a survey. Transactions on Programming Languages and Systems (TOPLAS), Vol. 23, Issue 4 (2011)