REVISTA PROYECCIONES N° 12:71-96
Diciembre 1986 - ISSN 0716-0917
Jornada Matemáticas, Agosto 1986.

FUNDAMENTOS DE LA PROGRAMACION EN LOGICA UNA INTRODUCCION

Dr. FIDEL OTEIZA M.*

1. INTRODUCCION.

Este material tiene por objeto introducir los conceptos en que se fundamenta el uso de la lógica clausal como base para lenguajes computacionales.

En 1981 se inició en Japón un proyecto de investigación y desarrollo con el título de "Computadores de Quinta Generación". El propósito de ese proyecto es generar máquinas especializadas en el procesamiento del conocimiento, estas máquinas deberían -de acuerdo con el anuncio hecho ese año- estar disponibles dentro del decenio. Durante los últimos 25 años se ha desarrollado experimentalmente un área de las ciencias de la computación llamada inteligencia artificial, las máquinas de quinta gene-

* Académico Departamento de Matemáticas y Ciencias de la Computación - Universidad de Santiago de Chile.

ración tendrían un diseño adecuado para generalizar el uso del conocimiento adquirido en esta especialidad. Junto con anunciar el inicio del referido proyecto, la industria electrónica japonesa señaló que la base teórica de esos equipos sería un lenguaje de computación específico, PROLOG (Programación en Lógica). Este es un lenguaje adecuado para el tratamiento de conocimiento desarrollado en Francia en 1972 por A. Colmerauer y Ph. Roussel. Con posterioridad, un equipo británico en la Universidad de Edinburgo, desarrolló una versión para ser utilizada en microcomputadores (micro-PROLOG).

La comunidad europea primero y la industria electrónica norteamericana más adelante, iniciaron proyectos paralelos con el objeto de no quedarse atrás en una materia de tanta importancia como la informática.

A partir del anuncio japonés, la programación que tiene sus fundamentos en la lógica ha despertado un enorme interés. Es en efecto atractivo conocer, con diez años de anticipación, la orientación que se quiere dar a la tecnología que mayor impacto causa en un momento dado.

El Grupo de Estudios en Educación Matemática de la USACH, adelanta en la actualidad algunos trabajos en los que se diseñan sistemas de aprendizaje con el apoyo de la programación en lógica. Paralelamente se estudian los fundamentos teóricos de esta modalidad de programación y se buscan sus implicaciones educativas. En estos apuntes encontrará una traducción libre y adaptada de un trabajo de Robert Kowalski: LOGIC FOR PROBLEM SOLVING, Artificial Intelligence Series, J. Nilsson (ed), North Holland, 1979.

¿Por qué estudiar los fundamentos de la programación en lógica?

Desde su aparición en 1972, el uso y las aplicaciones de PROLOG se han extendido. Se lo usa en los ambientes de inteligencia artificial en el desarrollo de sistemas expertos, en el análisis de lenguajes naturales y también en matemática, en la resolución automática de problemas y

en la demostración automática de teoremas. Estos temas son ya una motiva
ción suficiente para conocer la programación en lógica.

Dar un paso más, y conocer sus fundamentos tiene -para este au-
tor- las ventajas siguientes: a) pone en contacto directo con ideas pode
rosas como lo son el principio de resolución de Robinson y la demostración
de teoremas por medios electrónicos; b) permite independizarse de las ver
siones o dialectos del lenguaje. En efecto, al entregar una visión más
amplia facilita la comprensión de diferencias sintácticas que de otro mo-
do se transforman en verdaderas barreras; y c) la programación en lógica
-por las razones dadas al comienzo de estos apuntes- es un área en rápida
expansión; para comprender los adelantos que son de prever en el área, es
indispensable conocer la teoría en que se fundamenta.

2. LENGUAJES DE COMPUTACION DECLARATIVOS Y ALGUNOS CONCEPTOS RELACIONADOS.

PROLOG es un lenguaje de computación declarativo descriptivo.
El concepto se opone a imperativo - algorítmico.

Lenguajes como FORTRAN, BASIC y PASCAL tienen carácter imperati
vo - algorítmicos. En estos últimos predomina la ejecución de órdenes:
PRINT X, LET Y = 6, GOTO N, esto es imprima, asigne, vaya, en un orden
preestablecido. El conjunto de órdenes conforma un algoritmo o procedi
miento. El procedimiento determina cómo se ejecuta el programa. También
contienen elementos "declarativos", en el sentido que el programador no de
be preocuparse del procedimiento. En efecto, cuando asignamos una expres
ión a una variable, por ejemplo el valor de $3*(y/5 + z**2)$, sólo expres
amos lo que queremos, esto es el objetivo, y no los procedimientos.

Los lenguajes que se usan en inteligencia artificial deben ser
adecuados para representar conocimiento. Tanto LISP (LIST Processing), co
mo PROLOG hacen uso de una alta proporción de comandos declarativos. En
estos ambientes predomina el enunciado del objetivo por sobre los procedi
mientos para lograrlos. PROLOG ataca el problema de la representación por

medio de una estrategia que se basa en el uso de relaciones entre objetos expresados como listas y un procedimiento -propio del sistema- que infiere sobre la base de las relaciones que define el usuario. A continuación una relación definida en micro-PROLOG haciendo uso de la sintaxis del módulo Simple (este es un módulo introductorio preparado para facilitar el ingreso al sistema).

```
x último-de (x)
x último-de (y|z) if x último-de z
```

Este es un "programa" que le describe al sistema lo que entendemos por "último-de" una lista. Según como se lo use permite que el sistema -siguiendo un procedimiento que no es responsabilidad del usuario- determine si un elemento dado es o no el último de una lista, o bien que encuentre el último elemento de una lista.

La ejecución del programa se logra por medio de preguntas. Podemos preguntar, por ejemplo, ¿es efectivo que "c" es el último elemento de la lista (a,b,c,)?; en este caso el sistema contestará: sí. También podríamos pedir que nos entregase el último elemento de una lista: en este caso nos mostraría el objeto "c"; en el primer caso se trataría de un procedimiento de verificación, en el segundo de uno de resolución de problemas.

El programa está formado, en este caso, por dos afirmaciones que en conjunto describen lo que entenderemos por la relación "último-de" que se define entre un individuo y una lista. La primera declara:

```
"un individuo es el último elemento
de una lista de la que es el único elemento".
```

La segunda hace uso de la estructura (y|z) que describe una lista no vacía, ya que comienza con un elemento "y", el que está seguido de una lista z. El símbolo "|" se usa en este caso para separar la "cabeza"

de la lista de su "cola". La segunda afirmación se puede leer del modo siguiente:

"un individuo es el último elemento de una
lista no vacía si es el último de su cola".

El lector habrá reconocido que se hizo uso de una definición que "resuelve una copia de sí misma antes de dar la respuesta final", esto es, se trata de una definición recursiva.

En síntesis diremos que un lenguaje es declarativo si está orientado de modo que el programador defina descriptivamente sus objetivos y que el sistema busque el procedimiento para lograrlo. Estos apuntes se refieren a los mecanismos y convenciones que permiten lograr este modo de actuar haciendo uso de los formalismos de la lógica clausal.

Objetos.

Los objetos que utilizaremos en este ambiente son:

1. **Números:** enteros y también punto flotante, según las implementaciones.
2. **Constantes:** cadenas de caracteres alfanuméricos que se utilizan para nombrar individuos y relaciones. Juan puede ser una constante, último-de, también lo es.
3. **Variables:** usualmente se las designa por x, y, z, X, Y, Z , o por medio de combinaciones de la forma $x_1, x_2, \dots, y_1, y_2, \dots$. En algunas versiones las variables se designan por cadenas de caracteres alfanuméricos que comienzan por una letra mayúscula. Edad, Padre, serían ejemplos.
4. **Listas:** estos son objetos que tienen la estructura de una n -tupla. (a, b, c) es una lista de tres elementos; $(1, (2, 3), 4, 5)$ es una de cua-

tro; $()$ es la lista vacía; (x) es una lista que tiene a x como único elemento y $(y|Z)$, es una lista de al menos un elemento, que tiene a ese elemento en la cabeza y contiene la lista Z como cola.

Relaciones.

Las estructuras más simples que se utilizan en la programación en lógica son los **átomos** y éstos expresan que un cierto número de objetos satisfacen una relación. Por ejemplo, $(a \text{ } r \text{ } b)$ es un átomo que expresa que "a" tiene la relación "r" con "b". Tratándose de relaciones binarias se puede usar la notación.

objeto 1	nombre-de-relación	objeto 2
d	último-de	(a, b, c)
Juan	padre-de	María
x	>	y

En una relación pueden participar uno, dos o n objetos. Puerto (Valparaíso) expresa que el objeto Valparaíso cumple con la relación de ser puerto. Esta es una relación unaria. También lo son Femenino(María) o Masculino(Juan). Padre-de(x, y, z), podría indicar que "x" es padre de "y" y de "z", una relación entre tres.

Llamaremos **símbolo predicado** a la constante que expresa al nombre de una relación.

Para expresar que ciertos objetos satisfacen una relación se dispone de tres tipos de notación: prefija (antes), infija (entre) o postfija (después), en general usaremos las dos primeras.

La notación infija genera expresiones bastante cercanas al lenguaje natural. A continuación algunos ejemplos que ilustran el uso de esta notación (los símbolos predicados están subrayados).

Valparaíso <u>es-puerto</u>	(unaria)
Juan <u>es-padre-de</u> x	(binaria)
Juan <u>saltó</u> 1,5 m <u>de</u> alto	(entre tres)

En esta notación el símbolo predicado de una relación unaria se coloca después del objeto; y para una relación entre varios el símbolo se reparte entre los objetos.

La notación prefija es algo menos natural pero notoriamente más fácil de generalizar. A continuación los mismos ejemplos en esta convención.

es-puerto(Valparaíso)	(unaria)
es-padre-de(Juan, x)	(binaria)
salto-de(Juan, 1,5 m, alto)	(entre tres)

En el programa de la página , se pueden reconocer varios de los conceptos recién introducidos.

1. "x último-de (x)" es un átomo, la segunda sentencia tiene dos átomos:
"x último-de (x z)" es uno y "x último-de z", el segundo.
2. "x" y "z" son variables.
3. "(x)" es una lista que contiene al elemento "x".
4. "último-de" es un símbolo predicado, por lo tanto, también una constante.
5. Se puede reconocer el uso de la notación infija.
6. "último-de" es una relación que está definida para un objeto y una lista. Note que "z", por ser la cola de una lista es también una lista.

En resumen se presentaron los siguientes conceptos:

1. Lenguaje computacional declarativo.
2. Objetos en uso en la programación en lógica; números, constantes, variables y listas.
3. La notación infija y la prefija.

Además se introdujeron las expresiones: lista, lista vacía, lista con al menos un elemento ($x|z$), cabeza de una lista, cola de una lista, átomo y símbolo predicado.

3. FORMAS CLAUSALES.

Una de las necesidades en el área de la inteligencia artificial, se refiere a la representación del conocimiento. Los sistemas expertos, por ejemplo, contienen información descriptiva, esto es afirmaciones del tipo: "x es el caso" y hacen uso de reglas para expresar el conocimiento que un experto tiene de un campo determinado, éstas tienen la forma: "A si C1 y C1 y ...y Ck. En esta sección veremos como se representan estas estructuras en programación en lógica.

Haciendo uso del concepto de átomo y los conectivos "o" e "y" y "la noción de implicación: si" (la que representaremos por una flecha o bien por ":-" al usar el teclado de un computador), se puede construir el concepto de "cláusula" o "forma clausal".

Una cláusula tiene la estructura:

$$A_1 \text{ o } A_2 \text{ o } \dots \text{ o } A_m \text{ :- } C_1 \text{ y } C_2 \text{ y } \dots \text{ y } C_n$$

Donde los A_i y los C_i son átomos, $i=0,1,2,\dots,m$ y $j=0,1,2,\dots,n$.

La estructura expresa: "Algún A_i si todos los C_j ".

El conjunto de los A_i recibe el nombre de conclusiones y el de los C_j , de condiciones.

Ejemplos:

1. María quiere-a Juan o María admira-a Juan si bueno(Juan) y alegre(Juan) y exitoso(Juan).

Esta oración tiene la forma: $A1 \text{ o } A2 :- C1 \text{ y } C2 \text{ y } C3$. Tiene dos conclusiones alternativas y tres condiciones conjuntas.

2. $\text{Par}(x) :- \text{Entero}(x) \text{ y } 2 * z = x$

Esta tiene la forma: $A :- C1 \text{ y } C2$

3. Elena madre-de Roberto.

Esta es una afirmación incondicional, un átomo en realidad. Comparando con la forma clausal se trata de una conclusión sin condiciones, tiene la forma: $A :-$.

Cláusulas de Horn: Reciben este nombre las cláusulas en que $m = 1$, esto es, aquellas que tienen sólo una conclusión.

Cláusulas y Listas.

Conocida la forma de una cláusula, la notación se puede simplificar eliminando los conectivos "o" e "y" usando alguna separación entre átomos y manteniendo algún símbolo que separe las conclusiones de las condiciones. Por ejemplo, se usa:

$$(A1, A2, \dots, Am :- C1, C2, \dots, Cn)$$

En particular si se trata de cláusulas de Horn, se sabe que el primer átomo es la única conclusión, de modo que basta con escribir:

$$(A, C1, C2, \dots, Cn)$$

y esa es una estructura de lista, en la que la cabeza es la conclusión de la cláusula.

En PROLOG, los comandos tienen la estructura de una cláusula de Horn:

$$(\langle \text{átomo} \rangle \text{ si } \langle \text{átomo1} \rangle, \langle \text{átomo2} \rangle, \dots, \langle \text{átomo n} \rangle)$$

o simplemente:

$$(\langle \text{átomo} \rangle, \langle \text{átomo1} \rangle, \langle \text{átomo2} \rangle, \dots, \langle \text{átomo n} \rangle)$$

Por ahora seguiremos utilizando "si" o ":-" para subrayar la implicación.

Representación por Medio de Cláusulas.

Antes de dar una definición algo más formal de cláusula, usaremos las convenciones introducidas para traducir expresiones del lenguaje natural al clausal. Usaremos ":-" para la implicación y la notación pre-fija.

1. Leng. Natural: alguien es femenino o masculino si es humano.

Leng. Clausal: $(\text{femenino}(x), (\text{masculino}(x) \text{ :- humano}(x)))$

2. Leng. Natural: alguien es femenino si es madre de alguna persona.

Leng. Clausal: $(\text{femenino}(x) \text{ :- madre-de}(x,y), \text{ humano}(y))$

3. Leng. Natural: alguien es progenitor de otro, si es su madre o su padre.

Leng. Clausal: Esta traducción nos plantea un problema, necesitamos un "o", pero esta forma lógica no se encuentra en la cola de una cláusula. La solución es sencilla y se la utiliza a menudo en la programación, se usan dos cláusulas

independientes que en conjunto actúan como un "O" no excluyente. Observe que la idea contenida en la oración 3 está expresada mediante las dos cláusulas siguientes:

(progenitor(x,y) :- madre-de(x,y))

(progenitor(x,y) :- padre-de(x,y))

4. Leng. Natural: alguien es abuelo de otro si es progenitor de un tercero que es progenitor del otro.

Leng. Clausal:

(abuelo-de(x,y) :- progenitor(x,z), progenitor(z,y))

Definición de Formas Clausales.

Una cláusula es una expresión de la forma:

$A_1 \text{ o } A_2 \text{ o } \dots \text{ o } A_m \text{ :- } C_1 \text{ y } C_2 \text{ y } \dots \text{ y } C_n$

Donde A_i y C_j son fórmulas atómicas y n y m son enteros mayores o iguales a cero.

Los átomos A_i son las conclusiones alternativas de la cláusula y los C_j son las condiciones conjuntas. Si la cláusula contiene las variables $x_1, x_2, \dots, x_k$, se las debe interpretar del modo siguiente:

Para todo $x_1, x_2, \dots, x_k$; $A_1 \text{ o } A_2 \text{ o } \dots \text{ o } A_m$ si $C_1 \text{ y } C_2 \text{ y } \dots \text{ y } C_n$

Si $n=0$ se debe interpretar que incondicionalmente, para todo $x_1, x_2, \dots, x_k$ se cumple $A_1 \text{ o } A_2 \text{ o } \dots \text{ o } A_m$.

Si $m=0$ interprétese del modo siguiente: para todo $x_1, x_2, \dots, x_k$, no es el caso que $C_1 \text{ y } C_2 \text{ y } \dots \text{ y } C_m$.

Si $n=m=0$, interprétese como una oración que es siempre falsa.

Un átomo o fórmula atómica es una expresión de la forma:

$$P(t_1, t_2, \dots, t_p)$$

Una p-tupla de símbolo predicado P , en la que los t_i son términos y p un entero mayor o igual a uno.

Un término puede ser una variable, un símbolo constante, o una expresión de la forma:

$$f(t_1, t_2, \dots, t_q)$$

f es un símbolo funcional de q lugares y q , un entero mayor o igual a uno.

Algunos comentarios acerca de esta definición.

1. Las cláusulas incondicionales ($n=0$) se pueden utilizar para representar hechos:

Leng. Natural: Juan mide 1,8 m

Leng. Clausal: (mide(Juan, 1,8 m):-)

Un hecho queda representado por una cláusula de la forma: $(A:-)$.

2. Para expresar la negación se usa una cláusula sin conclusiones ($m=0$).

Leng. Natural: nadie es padre de sí mismo.

Leng. Clausal: $(:- \text{padre-de}(x,x))$

Leng. Natural: no es el caso que alguien es femenino y masculino a la vez.

Leng. Clausal: $(:- \text{femenino}(x), \text{masculino}(x))$

3. Se introdujo el cuantificador "para todo". De este modo se dispone de

un símbolo único, la cláusula, que permite manejar la conjunción, la disyunción, la negación, la implicación y el cuantificador "para todo". Más adelante se verá cómo se maneja la existencia.

4. SEMANTICA Y FORMAS CLAUSALES.

En esta sección se introduce el tratamiento de los significados mediante formas clausales y se hace uso de la noción de consistencia e in consistencia que serán tratadas en la sección siguiente.

La sintaxis tiene que ver con las reglas gramaticales y/o forma les que debe cumplir un lenguaje, la semántica busca la relación entre los formalismos y los significados. Cuando en el lenguaje natural decimos fe menino, madre, hijo, etc., conocemos los alcances, limitaciones e implica ciones de esos conceptos. Si usamos las mismas palabras como nombres de relaciones, el alcance que queremos que esos términos tengan debe estar to talmente determinado por el conjunto de cláusulas que las define.

Para comenzar introducimos el concepto de "base de datos" o "ba se de conocimiento" o simplemente "base". Llamaremos base al conjunto de cláusulas que cumpliéndose simultáneamente definen un ambiente dado. En la literatura también se habla de "los supuestos" del ambiente. Kowalski propone un ejemplo en el que mediante 19 cláusulas define las relaciones de un ambiente tomado de la mitología griega. A la base la designaremos por F1-F19.

Relaciones Familiares entre Dioses Griegos (Kowalski, 1979, p3 y p8).

F1 padre(Zeus, Ares):-
 F2 madre(Hera, Ares):-
 F3 padre(Ares, Armonia):-
 F4 madre(Afroditia, Armonia):-
 F5 padre(Cadmus, Semele):-

```

F6   madre(Armonia, Semele):-
F7   padre(Zeus, Dionisio):-
F8   madre(Semele, Dionisio):-
F9   dios(Zeus):-
F10  dios(Era):-
F11  dios(Ares):-
F12  dios(Afrodita):-
F13  musa(Armonia):-
F14  femenino(x) :- madre(x,y)
F15  masculino(x) :- padre(x,y)
F16  progenitor(x,y) :- madre(x,y)
F17  progenitor(x,y) :- padre(x,y)
F18  abuelo(x,y) :- progenitor(x,z), progenitor(z,y)
F19  masculino(x), femenino(x) :- humano(x)

```

El lector puede reconocer que F1-F13 son hechos o cláusulas incondicionales, que F14-F19 son reglas y que F19 es la única cláusula no-Horn del conjunto. También es interesante hacer notar el uso de la variable "z" en la cláusula 18, esa cláusula se puede leer del modo siguiente: x es abuelo de y si existe un z en la base que ... Lo notable es el manejo de la existencia por medio de una variable en las condiciones. Regresaremos a este punto en la sección siguiente.

Alguien podría preguntarse si es efectivo que Zeus es madre de alguien, de Ares, por ejemplo. Esto lo podemos expresar por la cláusula, llamémosla F.

```

F   madre(Zeus, Ares):-

```

Si nos guiamos por el significado que nosotros damos a los conceptos de madre, masculino y femenino, "sabemos" que Zeus no puede ser madre. Si se analiza la base F1-F19 se observará que no hay nada en ella que permita negar F. Esto quiere decir que aquel aspecto del significado de madre, de masculino y de femenino no pertenece a la base. Si queremos

que ese aspecto del significado esté presente en la base, debemos agregar lo; en este caso sería suficiente con la cláusula:

F20 :-femenino(x), masculino(x)

Ahora podemos decir que el significado de F1-F20 incluye la negación:

F* :-madre(Zeus, Ares)

La afirmación F es, entonces, consistente con F1-F19 y es inconsistente con la base F1-F20. La segunda base contiene los elementos para que el significado de madre sea algo más cercano al que le damos en el lenguaje natural.

La traducción que podemos hacer de una cláusula al lenguaje natural sólo nos dará una orientación acerca del significado de los símbolos predicados. El significado asociado a un símbolo predicado, a una constante, a un símbolo funcional o a una cláusula, está dado por el conjunto de cláusulas que definen el ambiente.

De este modo se ha reducido el problema de la semántica al tratamiento de la sintaxis. Por último, la relación entre una base y una cláusula es similar a la que se da entre el conjunto de axiomas de una teoría matemática y un teorema.

5. CONSISTENCIA E INCONSISTENCIA.

Los significados, vimos, deben estar dados por el conjunto de los supuestos. Para determinar si una cláusula es o no parte del significado de una base, se la niega y se estudia la consistencia o inconsistencia de la cláusula negada con la base. De este modo la cuestión de los significados queda reducida al manejo de la inferencia. En esta sección estudiaremos los formalismos por medio de los cuales se puede tratar la

consistencia/inconsistencia entre una cláusula y una base.

El Ejemplo del Griego Falible.

Para mostrar que los supuestos:

```
G1    humano(Turing):-
G2    humano(Sócrates):-
G3    griego(Sócrates):-
G4    falible(x) :- humano(x)
```

implican la conclusión de que existe un griego falible, negamos esa conclusión:

```
G5    :- falible(u), griego(u)
```

y mostramos que el conjunto formado por G1-G5 es inconsistente. Se puede dar un paso más, se puede dar la razón para esa inconsistencia, esto es la existencia de la sustitución:

$u = \text{Sócrates}$

que señala un individuo que es, a la vez, falible y griego. De este modo la cláusula G5 puede interpretarse como expresando el problema de encontrar un individuo que es falible y griego, y el valor extraído de la demostración, como la solución al problema.

Universo del Discurso e Interpretaciones.

Si los individuos de una base se nombran usando sólo un término libre de variables, una sola constante para cada individuo, el universo del discurso puede ser comprendido, intuitivamente, como el conjunto de individuos descritos por las cláusulas.

La base del griego falible es un ejemplo simple. En este caso el universo del discurso está formado por Sócrates y Turing.

Una interpretación, en el mismo ejemplo, es una instanciación o asignación a cada uno de los tres símbolos predicados que participan en la base.

La diferencia entre la base G1-G4 y la que contiene a G5, es que para la primera existen instancias en que las sustituciones hacen que todas las cláusulas sean verdaderas mientras que en la segunda ninguna sustitución lo logra. Decimos que G5 es inconsistente con G1-G5.

Es interesante hacer notar que la inconsistencia de un conjunto de cláusulas puede ser demostrada "semánticamente" -y esto es lo que hemos hecho hasta ahora y lo que seguiremos haciendo en esta oportunidad- mostrando que no existe ninguna sustitución que haga, simultáneamente a todas las cláusulas de la base verdadera, o bien se puede demostrar "sintácticamente" construyendo un conjunto de pasos inferenciales. Esto último es lo que hace PROLOG mediante su propia máquina inferencial.

6. VARIABLES Y REPRESENTACION.

A continuación algunas situaciones en las que la traducción del lenguaje natural al clausal se logra mediante el uso de variables.

En primer caso se refiere al uso de expresiones como: algo, alguno, cualquier cosa, todas, algunas, ninguna. En efecto:

"alguien es bueno" lo podemos traducir por:

x es-bueno:-

"nadie es bueno y malo a la vez"

:- x es-bueno, x es-malo

"todo hombre es un animal"

x es-un-animal :- x es-un-hombre

El segundo se refiere al manejo de clases de objetos. La pertencia permite tratar una clase como objeto o como un símbolo predicado. Si nos referimos a humano, podemos considerarlo como individuo o como nombre de una relación.

Como individuo: Juan es-un humano

Como símbolo predicado: humano(Juan)

Por último, los pronombres hacen referencia a individuos ya mencionados en un discurso dado.

Por Ejemplo: "aquel que come animales es carnívoro".

x es-carnívoro :- x es-un humano,
 x come y,
 y es-un animal

7. **EXISTENCIA.**

En lógica clausal la existencia se expresa nombrando individuos mediante a) una constante, b) un símbolo funcional o c) mediante una variable. Distinguiremos dos casos según si la existencia debe ser declarada en una conclusión o en una condición.

Si la existencia debe ser declarada en conclusiones y el individuo en cuestión es uno independientemente de los valores que adopten los restantes elementos de la cláusula, se usa un símbolo constante que no se usa para nada diferente en la base.

Para expresar "algún hombre es animal", se puede usar la cláusula:

es-un hombre:-

es-un animal:-

El símbolo "#" denota la existencia de un objeto que cumple con las condiciones dadas.

Tener denota muchas veces existencia:

"Zeus tiene un progenitor que lo quiere".

En lógica clausal se necesita un símbolo para denotar al progenitor:

& es-un-progenitor-de Zeus:-

& quiere-a Zeus

Resulta diferente si se desea expresar que:

"todos tienen un progenitor que los quiere".

Si recurrimos a la misma estrategia y usamos una constante, "*" por ejemplo, las cláusulas serían:

* es-progenitor-de x :- x es-un humano

* quiere-a x :- x es-un humano

Termina en que "*" es un ser que existe y que es progenitor de todos y los quiere a todos (algo así como Dios). Para expresar proposiciones más modestas como que para cada humano "x" existe un progenitor y que éste es función de "x", se recurre a los símbolos funcionales.

P(x) es-un-progenitor-de x :- x es-un humano

P(x) quiere-a x :- x es-un humano

Ahora cada uno tiene un progenitor P(x) y éste lo quiere.

"Todos (cada uno) tiene una madre".

$M(x)$ es-madre-de x :- x es-un humano

En resumen la existencia de un individuo al que se refiere una o varias conclusiones debe ser expresado por medio de una constante o por un símbolo funcional.

Por el contrario, la existencia de un individuo que sólo aparece en las condiciones se expresa por medio de una variable. La oración siguiente sería un caso:

Una persona es abuelo de otra
si existe un hijo de la primera
que es progenitor de la segunda.

x es-abuelo-de y :- x es-un humano,
 y es-un humano,
 x progenitor-de z
 z progenitor-de y

La variable "z" denota la existencia en las condiciones.

8. EJEMPLOS Y APLICACIONES.

Para completar estos apuntes daremos algunos ejemplos más de traducción de expresiones del lenguaje natural al clausal y analizaremos relaciones escritas en la sintaxis del módulo simple de micro-PROLOG así como en la sintaxis de micro-PROLOG mismo.

1. Usando la notación del ejemplo de los dioses griegos (p9), exprese en formas clausales las relaciones siguientes:

x es la madre de z si

x es femenino y x es progenitor de z.

madre(x,z) :- femenino(x), progenitor(x,z)

x es humano si

x es progenitor de z y z es humano.

humano(x) :- progenitor(x,z), humano(z)

Un individuo es humano si

su madre es humana y

su padre es humano.

humano(x) :- madre(y,x), padre(z,x), humano(y), humano(z)

nadie es su propio padre

:-padre(x,x)

Si una persona es humana,

entonces su madre es humana o

su padre es humano.

humano(y) :- madre(y,x), humano(x)

humano(z) :- padre(z,x), humano(x)

2. Una relación escrita en la sintaxis del módulo simple y en la sintaxis de micro-PROLOG.

a) En Simple.

x pertenece-a (x|Z)

x pertenece-a (y|z) if x pertenece-a z

Se puede reconocer la notación infija.

La primera afirmación es una cláusula incondicional, afirma que un elemento x pertenece a una lista cuya cabeza es x .

$(x\ z)$ representa una lista de al menos un elemento.

En la segunda cláusula se usa "if" en vez de ":-"

Se trata de una definición recursiva.

La relación puede ser usada para confirmar si un elemento pertenece o no a una lista dada:

En simple se puede lograr la evaluación mediante "does".

```
does(a pertenece-a (1 2 a b))
```

También se la puede usar para generar elementos que cumplen la condición:

```
which(x x pertenece-a (1 2 a b) and x pertenece-a (1 2 3 4))
```

¿Cuál es el x que pertenece a $(1, 2\ a\ b)$ y a $(1\ 2\ 3\ 4)$?

b) En micro-PROLOG.

```
((pertenece-a x (x|z)))
((pertenece-a x (y|z) (pertenece-a x z)))
```

Se puede observar la notación prefija.

Observe los paréntesis y notará la estructura :

```
((átomo) (átomo) (átomo)...(átomo))
```

sin comas entre los átomos y sin ":-" entre el primero y los siguientes. El doble paréntesis de la primera afirmación recuerda que se trata de una cláusula con un sólo átomo.

3. Una relación que es necesario definir y luego experimentar por si mismo es la que permite formar una lista a partir de dos listas dadas. La designaremos por "concatenadas-en". Su estructura es:

$(L1 \ L2)$ concatenadas-en L ; y expresa que L está formada por la reunión de las listas $L1$ y $L2$ respetando sus elementos y el orden. En los textos en inglés y en los manuales de lenguaje se la designa por "appends-to".

La razón por la que es importante es que para el procesamiento de listas esta relación es una herramienta simple pero poderosa, tanto por las operaciones que permite hacer cómo porque es usada para construir otras relaciones que permiten analizar el contenido de una lista.

El lector puede preparar la propia sabiendo que lo que busca son dos cláusulas. La primera debe expresar qué va a entender como el resultado de "concatenar" la lista vacía y una lista arbitraria X . La segunda debe tratar el caso de una lista de al menos un elemento $(x|Z)$ concatenada con una lista arbitraria Y . Por último que la definición es recursiva.

Si dispusiera de la definición, ¿cuál sería la respuesta a las preguntas siguientes?

1. Which ($X \ ((1 \ 2 \ 3) \ (a \ b))$ concatenadas-en X)
2. Which ($X \ Y \ (X \ Y)$ concatenadas-en $(a \ b \ c)$)

En el primer caso sería la lista $(1 \ 2 \ 3 \ a \ b)$ y en el segundo las parejas de listas que cumplen la relación:

- $() \ (1 \ 2 \ 3)$
 $(1) \ (2 \ 3)$
 $(1 \ 2) \ (3)$
 $(1 \ 2 \ 3) \ ()$ estas son todas las biparticiones de la lista dada.

La definición en la sintaxis de Simple es la siguiente:

```
(( ) X) concatenadas-en X
((x|X) Y) concatenadas-en (x|Z) if
                                (X Y) concatenadas-en Z
```

Mediante qué pregunta se puede usar la relación definida para obtener todas las segundas partes de las biparticiones de la oración: "el hombre usó el computador para hacer el escrito", de modo que ellas comiencen por el artículo "el"?

El ejemplo muestra el tipo de análisis que este tipo de programación permite hacer. De pasada muestra que el computador está siendo usado en estos momentos como una herramienta analítica fuera del ámbito de las fórmulas matemáticas. Estoy recordando el origen de la palabra computador y que FORTRAN significa FORMula TRANlator.

La pregunta podría ser:

```
Which( (el|Y) ( X (el|Y)) concatenadas-en ( el hombre usó el computador para hacer el escrito))
```

Esto es: deme todas las listas que comienzan con "el" y siguen con cualquier cosa (el Y) tales que concatenadas, como segundo elemento, con una lista arbitraria X, dan como resultado la lista dada.

El lector puede usar esta relación para definir otras como: la relación: frente-de que entrega los primeros n elementos de una lista dada; la relación "compactada-de" que a partir de una lista dada entrega una nueva que no contiene repeticiones.

9. A MODO DE CONCLUSION.

Estos apuntes contienen algunos de los conceptos de la lógica

clausal que han sido utilizados para generar lenguajes de computación. Estos lenguajes representan un estilo diferente de programación, están relacionados con el campo de la inteligencia artificial y se supone que los equipos que se preparan en la actualidad para constituir lo que se llama la quinta generación, tendrán una arquitectura basada en estos lenguajes.

Si bien aquí se entrega una síntesis de los elementos que permiten comprender los elementos sintácticos de PROLOG, el lector interesado debería continuar con un estudio de las estrategias de búsqueda en estructuras representables por grafos y árboles, para comprender las operaciones de la "máquina inferencial" de PROLOG. Más adelante el lector encontrará algunas referencias bibliográficas.

El autor quiere agradecer en esta oportunidad al Dr. Enrique Cansado, tanto su participación, por dos semestres en nuestro grupo de estudios, como el habernos puesto en contacto con estas ideas y mostrarnos sus implicaciones. También a Víctor Araya, quien nos facilitó la entrada a los aspectos técnicos del lenguaje y ha sabido guiar a distintos alumnos de nuestro Departamento que realizan sus trabajos de titulación en esta área.

REFERENCIAS.

1. Como un primer texto, en el que se introduce el módulo simple por ejemplo, se puede usar:

C. ENNALS y MCCABE. A micro-PROLOG Primer
Londres: Logic Programming Ass., 1982.

2. En relación con los fundamentos, el texto que sirvió de base para estos apuntes:

R. KOWALSKI. Logic for Problem Solving.

North-Holland, Artificial Intelligence Series, 1979.

3. El clásico en cuanto a programación es:

W. F. CLOCKSIN y C.S. MELLISH. Programing in Prolog.

Segundda edición. Springe-Verlag, 1984.