UNIVERSIDAD DE PANAMÁ
FACULTAD DE CIENCIAS NATURALES Y EXACTAS
ESCUELA DE MATEMÁTICA
CENTRO REGIONAL UNIVERSITARIO DE VERAGUAS

ASOCIACIÓN NACIONAL DE ESTUDIANTES DE MATEMÁTICA (A.N.E.MAT.)
CAPÍTULO DE VERAGUAS

SEMINARIO: LA MATEMÁTICA, SU ENSEÑANZA, FUNDAMENTOS Y PERSPECTIVAS

CONFERENCIA: ***EL CÁLCULO DE PREDICADOS DE PRIMER ORDEN APLICADO A LA PROGRAMACIÓN LÓGICA.***

EXPOSITOR: ***RAÚL ENRIQUE DUTARI DUTARI.***

FECHA: ***18 DE NOVIEMBRE DE 1997.***

HORA: ***11:30 A. M. - 12:20 P.M.***

LUGAR: ***AULA A-5 DEL CENTRO REGIONAL UNIVERSITARIO DE VERAGUAS.***

DIRIGIDA A: ***PROFESORES UNIVERSITARIOS DEL DEPARTAMENTO DE MATEMÁTICA DEL C.R.U.V., ASÍ COMO DOCENTES DE ENSEÑANZA MEDIA DE MATEMÁTICA QUE PARTICIPARON EN EL EVENTO.***

DURACIÓN: ***50 MINUTOS.***

OBJETIVOS GENERALES

1. Elevar el nivel de cultura informática de los participantes.

2. Despertar el interés de los participantes en la temática de la informática educativa.

3. Caracterizar a la programación lógica como herramienta para el desarrollo de software.

OBJETIVOS ESPECÍFICOS

1. Definir los modelos de programación procedural y declarativo.

2. Comparar los modelos de programación procedural y declarativa.

3. Caracterizar a la lógica proposicional y al cálculo de predicados de primer orden, como la herramienta base de la programación lógica.

TABLA DE CONTENIDOS

1. Observaciones preliminares.

Los lenguajes de programación se diseñan de acuerdo a ciertos criterios generales[1] acerca de cómo organizar:

- ☑ Los modelos matemáticos o lógicos que nos permiten almacenar y manipular datos e información dentro de la computadora[2];
- ☑ Las instrucciones que da el programador a la máquina.

En esta conferencia, analizaremos dos de ellos: la programación orientada a los procedimientos frente a la programación orientada a las declaraciones.

Dentro del contexto de esta conferencia, interesa aclarar lo concerniente a la programación orientada a las declaraciones, en una de sus representaciones menos conocida: la programación lógica, que está fundamentada en conceptos que rompen con las estructuras de control tradicionales de los lenguajes orientados a los procedimientos, tales como Pascal y C.

Para comprender cabalmente lo que representan los párrafos anteriores, debemos comprender ciertos conceptos básicos de los lenguajes de programación, así como de cálculo de predicados de primer orden.

---

[1] Algunos autores los llaman “Paradigmas de la programación”.

1.1. La programación orientada a los procedimientos o procedural.

Un lenguaje de programación se dice ***orientado a los procedimientos*** si ***las órdenes contenidas en el código fuente de sus programas, son ejecutadas instrucción por instrucción, estrictamente de acuerdo a la secuencia lógica de aparición de los mandatos***. El código fuente le indica a la computadora lo que debe hacer paso a paso, detalladamente[3]. Entiéndase, se requiere que el programador especifique el procedimiento que ha de seguir la computadora para llevar a cabo una tarea[4].

Tales lenguajes obligan al programador a decirle a la computadora lo que tiene que hacer, paso a paso, según lo planteado por el algoritmo que resuelve el problema, hasta completar su solución, siguiendo estrictamente los planteamientos de la programación estructurada y del diseño arriba - abajo.

Recordemos que la programación estructurada es una estrategia de diseño de software disciplinada, basada en el uso exclusivo de unas cuantas estructuras de codificación básicas y la aplicación de conceptos descendentes para

---

2 Mejor conocidas como “Estructuras de datos”, según LIPSCHUTZ, Seymour. Estructura de datos. Página 2.

3 NORTON, Peter. Introducción a la computación. Página 384.

4 PFAFFENBERGER, Bryan. Diccionario para usuarios de computadoras. Página 403.

descomponer las funciones principales del programa, en componentes de nivel más bajo, con el objetivo de codificarlo en forma modular[5].

Puesto que uno de los principios de la programación estructurada establece que el programa tendrá un único inicio, un único fin y una sola línea de flujo, podemos afirmar que: ***en la programación orientada a los procedimientos, la secuencia de ejecución de las instrucciones queda establecida de antemano por el programador y no se puede modificar sin modificar el programa.***

En conclusión, bajo el enfoque de la programación orientada a los procedimientos, el programador ha de preocuparse de la especificación de:

- ☑ Las estructuras de datos que manipulará el programa (variables, arreglos, archivos, registros, punteros, etc.),

- ☑ El control del flujo de instrucciones dentro del programa; es decir, la forma en que se deben manipular las entradas del sistema, para que generen las salidas respectivas.

Adicionalmente, cuando el programa se ejecuta, la secuencia de ejecución de las instrucciones no puede alterarse. Para alterarla, debe modificarse el código fuente del programa y reconstruir nuevamente el archivo ejecutable.

Históricamente, la mayoría de los lenguajes que se han usado en computadoras —ALGOL, BASIC, COBOL, C, MÓDULA - 2, Pascal, entre otros— son lenguajes orientados a los procedimientos.

---

5 SANDERS, Donald H. Informática: Presente y futuro. Página 864.

1.2. La programación orientada a las declaraciones o declarativa.

Un lenguaje de programación se dice ***orientado a las declaraciones o declarativo*** si está diseñado de manera tal que ***libera al programador de especificar el procedimiento exacto que la computadora necesita seguir para llevar a efecto una tarea.*** Los programadores usan este tipo de lenguaje para describir un conjunto de hechos y relaciones a fin de que el usuario pueda consultar al sistema para obtener un resultado específico[6].

Dentro de la programación declarativa, el equivalente de las instrucciones o comandos procedurales, se encuentra en las reglas o relaciones.

La principal característica de la programación declarativa está en que en ella: ***las reglas se plantean independientemente de su secuencia de aplicación***.

Otra característica relevante de la programación declarativa está en la existencia de un "Motor de Inferencia"[7]. El motor de inferencia, es un mecanismo interno del lenguaje de programación, y se encarga de:

---

6 PFAFFENBERGER, Bryan. Diccionario para usuarios de computadoras. Página 142.

7 También conocido como motor inferencial o intérprete de las reglas. En realidad, está alojado a nivel del software que implementa al lenguaje de programación.

- ☑ Definir un subconjunto de reglas sobre las cuales actuará esa corrida del programa.

- ☑ Decidir la secuencia de ejecución de las reglas.

- ☑ Ejecutar, según el orden preestablecido en los pasos previos, a las reglas que se deben ejecutar.

Todas estas acciones se realizan independientemente de la secuencia que las caracteriza dentro del código fuente del programa.

La programación declarativa es una tendencia de diseño de software relativamente nueva, implantada en los sistemas expertos y por ciertos lenguajes de programación, como el PROLOG y los lenguajes de consulta estructurados (S.Q.L.).

Como ejemplo más sencillo de este tipo de lenguaje de programación, tenemos a los lenguajes de consulta estructurados. Ellos permiten realizar búsquedas; por ejemplo, solicitando ver una lista de registros que muestre información específica; en vez de indicarle a la computadora que busque todos los registros, los que tienen las entradas apropiadas en los campos especificados.

1.3. Diferencias entre la programación procedural y declarativa.

Gran número de las características de los sistemas expertos se derivan de los principios fundamentales que acabamos de plantear para los modelos de programación declarativa y procedural. Sólo mencionaremos las más importantes.

En primer término, los programas escritos bajo un lenguaje declarativo permiten añadir, modificar y eliminar reglas[8] dentro del listado, sin ocuparse por sus efectos sobre la ejecución del programa, como producto del orden en que se coloquen[9]. Esto se debe a que el motor de inferencia se encarga de "decidir", con base a los datos y las reglas provistas "en bruto", cual es la salida correspondiente para el programa.

En cambio, en un programa escrito bajo un lenguaje procedural, generalmente no se puede modificar la secuencia de las órdenes, sin alterar a la ejecución del programa, y más estrictamente, a los resultados que él debe generar. En estos sistemas, simplemente no existe un motor inferencial que facilite la forma en que se obtienen las salidas dentro de un sistema experto.

En consecuencia, los programas escritos bajo un lenguaje declarativo, disfrutan de un nivel de modularidad superior al que tienen los programas escritos bajo un lenguaje procedural.

Por otro lado, como no hay un camino procedural pre - establecido para contestar, de manera única, a cada conjunto arbitrario de datos de entrada, el programa puede responder a muchos planteamientos distintos sin modificar, en absoluto, el código fuente.

Finalmente, la elevada modularidad de los programas escritos bajo un lenguaje declarativo, hace que las reglas sean más fáciles de leer, que en los programas escritos bajo un lenguaje procedural. Los programas escritos bajo un

---

8 Que son equivalentes a las órdenes dentro de un lenguaje procedural.

9 Se asume que todas las reglas son escritas de manera secuencial.

lenguaje declarativo resultan más sencillos y legibles, al momento de comprobar su consistencia, que los escritos bajo un lenguaje procedural.

Obsérvese que estas ventajas teóricas son algunas veces parcialmente inconvenientes. En muchos casos, el orden en que el motor de inferencia considera las reglas no es significativo y los resultados pueden ser diferentes si varios caminos conducen a una misma conclusión. Por otro lado, añadir reglas a conjuntos de reglas sin ningún tipo de control puede llegar a ser peligroso.

Para aclarar la última oración, nos adelantaremos un poco en la exposición para presentar un ejemplo de esta situación. Supongamos que tenemos el conjunto de reglas, planteadas dentro de un sistema que involucre el cálculo de probabilidades:

$(R_1)P \Rightarrow Q(0.4)$
$(R_2)Q \Rightarrow R(0.6)$

que conduce, en el caso de que P sea cierto, a que la probabilidad de la probabilidad de R sea de $0.4*0.6=0.24$ (al aplicar la regla del producto de probabilidades). Ahora, si introducimos en el sistema la regla:

$(R_3)P \Rightarrow R(0.5)$

que asigna a R el valor de 0.5, directamente. Así, tendremos que las reglas R_1 y R_2 se pueden eliminar. Adicionalmente, estaremos ante una situación en que el sistema puede obtener dos valores distintos 0.24 y 0.5, para un único valor de P.

Además, podemos plantear situaciones en las que ciertos conjuntos de reglas nos llevarían a obtener conjuntos de reglas contradictorios. Por ejemplo, si tenemos el sistema de reglas definido por:

$(R_1) P \Rightarrow Q$
$(R_2)(R \wedge Q) \Rightarrow U$

si introducimos en el sistema, la regla:

$(R_3) P \Rightarrow \neg U$

logramos que nuestro sistema de reglas se torne contradictorio. La contradicción está en que si P y Q son verdaderos, (R_1) y (R_2) implican a U, en contradicción con lo que señala la regla (R_3).

Lo cierto es que si nunca ha usado un lenguaje declarativo, aprendiéndolo conocerá más aspectos distintos de la programación de computadoras[10] que aprendiendo cualquier otro lenguaje procedural. Pero el esfuerzo merecerá definitivamente la pena, porque un lenguaje declarativo ahorrará mucho trabajo de programación.

Hablando simbólicamente, si utilizamos un programa de computadora para preparar una receta: mientras que un lenguaje procedural exige que se introduzca el recipiente y los ingredientes, un lenguaje declarativo sólo pide los ingredientes y el objetivo a lograr con ellos. Se declara la situación con la que quiere trabajar y dónde se quiere ir. Luego, el propio lenguaje —ya sea compilador o intérprete, a

---

[10] En el caso particular del Prolog: se debe desarrollar fuertemente el manejo de reglas recursivas, de una forma más completa que en los lenguajes procedurales; las variables sólo pueden recibir asignación de valores una vez; las variables al recibir asignaciones, se convierten en predicados; entre otras cosas.

través del motor de inferencia— realiza el trabajo de decidir cómo alcanzar dicho objetivo, si es posible lograrlo.

2. Introducción al cálculo proposicional y cálculo de predicados de primer orden, como fundamento de la programación lógica.

En los lenguajes de programación procedurales resulta relativamente fácil identificar las estructuras de control básicas. Por ejemplo, si usted es un programador en BASIC que quiere conocer algo de Pascal, pronto observara que muchas construcciones son similares en uno y otro lenguaje. Pronto reconocerá las sentencias asignación, la elección IF - THEN, los bucles FOR - NEXT y demás estructuras fundamentales.

Pero si usted va a pasar de BASIC o Pascal a un lenguaje de programación declarativo, tiene que dar un paso mayor. No reconocerá muchas de las estructuras del lenguaje. Un programa escrito en un lenguaje de este tipo[11] se parece a una base de datos de hechos —muchos paréntesis sobre líneas no relacionadas sin ritmo ni razón—. Pero no debemos preocuparnos. Después de un poco de práctica encontraremos que son tan fáciles de leer como un Programa en BASIC. Además, conforme se avance en su estudio, descubriremos algunas órdenes que son muy similares a las sentencias avanzadas del Pascal u otros lenguajes procedurales. Nos asombraremos de todo lo que puede realizar con sólo unas cuantas líneas de código.

---

[11] Prolog, List, S.Q.L., entre otros.

No obstante, la filosofía de programación de los lenguajes declarativos tiene fuertes fundamentos teóricos en el cálculo de proposiciones de la Lógica Proposicional —Cálculo de Predicados de Primer Orden— de la Matemática, más que en las estructuras de control algorítmicas tradicionales. En consecuencia, es recomendable que aprendamos o recordemos algunos conceptos básicos del Cálculo de Predicados de primer orden.

Para que nuestro discurso tenga el carácter axiomático que requiere, debemos empezar definiendo el concepto de "Lenguaje formal"[12].

Intuitivamente, un ***"lenguaje formal"*** es un conjunto de términos y reglas que se siguen para estructurar dichos términos[13] que se usa para modelar el lenguaje natural del ser humano. Se usan para comunicarse con las computadoras, porque se pueden definir completamente las reglas que siguen, sin ambigüedad.

Sin embargo, para los efectos de nuestro análisis, requerimos una definición más formal del término.

2.1. Definición de lenguaje formal.

Un ***"Lenguaje formal"*** L, sobre un conjunto cualquiera A finito, es un subconjunto de A^*: el conjunto de todos los arreglos o cadenas de elementos de

---

12 KORFHAGE, Robert R. Lógica y Algoritmos : Con aplicaciones a las ciencias de la computación e información. Página 167 y siguientes.

13 Normalmente conocidas como "Gramática del Lenguaje".

A, que se pueden formar de acuerdo a otro conjunto finito G de reglas acerca de cómo estructurar dichos arreglos. Se dice que la gramática G genera al lenguaje L y se representa como $L(G)$.

A nivel práctico, la gramática de un lenguaje formal $L(G)$ está integrada por los siguientes elementos:

1. Un ***"alfabeto":*** Consiste en un conjunto finito de símbolos que pueden integrarse en sucesiones finitas y representan expresiones dentro del lenguaje.

2. Un ***"número finito de reglas o producciones":*** Dentro del lenguaje formal existirá un conjunto finito de reglas que se aplican a los símbolos del lenguaje y especifican cómo combinar sucesiones de símbolos correctamente, en palabras, frases, oraciones, sentencias o instrucciones. Este conjunto se conoce como: "***sintaxis"*** o ***"gramática del lenguaje"***.

3. Un ***"número finito de axiomas":*** Son un conjunto no vacío de sucesiones de símbolos del lenguaje. Son la materia prima, a la que se puede aplicar las producciones del lenguaje. La elección del conjunto de axiomas afecta materialmente el conjunto de sucesiones de símbolos que se generan.

2.2. La lógica proposicional.

Para los efectos de nuestro estudio consideraremos a la "Lógica Proposicional o de predicados" como un lenguaje formal. En consecuencia, su gramática tendrá un alfabeto, un conjunto de reglas de sintaxis y un número finito de axiomas.

2.2.1. Alfabeto de la lógica proposicional.

El alfabeto contiene los siguientes símbolos:

- ☑ ***Variables proposicionales o proposiciones simples:*** Denotadas con letras mayúsculas individuales, por ejemplo: A,B,C. Se usan para identificar a las expresiones básicas del Lenguaje Formal.

- ☑ ***Operadores o conectores lógicos:*** Son los objetos: $\neg,\wedge,\vee,\Rightarrow$. Permiten expresar operaciones entre proposiciones simples, para generar las denominadas ***proposiciones compuestas***.

- ☑ ***Los paréntesis:*** $(\ )$, para precisar el significado de las operaciones entre proposiciones; de manera similar a como se utilizan en el álgebra ordinaria.

2.2.2. Sintaxis de la lógica proposicional.

La sintaxis define las reglas que utilizaremos para construir sucesiones de símbolos del alfabeto, que tendrán sentido gramatical. Estas sucesiones se llamarán, en adelante, ***"fórmulas bien formadas"***.

DEFINICIÓN DE LAS REGLAS DE SINTAXIS DE LA LÓGICA PROPOSICIONAL, PARA CREACIÓN DE FÓRMULAS BIEN FORMADAS.

Una fórmula se dice que está bien formada si está construida de acuerdo con las siguientes reglas recursivas:

1. Una sucesión de símbolos del alfabeto es una fórmula bien formada si y sólo si el serlo se sigue de un número finito de aplicaciones de las reglas (2) y (3),

2. Las proposiciones se definen como fórmulas bien formadas,

3. Si A y B son fórmulas bien formadas, entonces: $A \wedge B$, $A \vee B$, $\neg A$, $\neg B$, $A \Rightarrow B$, serán fórmulas bien formadas.

EJEMPLO PRÁCTICO.

Tenemos que $(A \Rightarrow B) \Rightarrow C$ es una fórmula bien formada, pero $(A \Rightarrow B) \Rightarrow$ no lo es.

Obsérvese que los paréntesis pueden eliminarse cuando no exista ambigüedad: $(A \Rightarrow B)$ podría haberse escrito como $[(A) \Rightarrow (B)]$. A partir de este momento, sólo consideraremos las fórmulas bien formadas, a las que denominaremos simplemente fórmulas.

2.2.3. Interpretación semántica de la lógica proposicional.

Para ***interpretar el sentido*** de este lenguaje formal y hacer de él una herramienta útil para el estudio del razonamiento, tenemos que definir que: ***Toda proposición es una afirmación que toma exclusivamente, uno de dos valores verdadero o falso***.

EJEMPLO PRÁCTICO.

Para un agente de bolsa, la proposición: “La compañía $INTEL,S.A.$ opera en la industria de las $COMUNICACIONES$” es una proposición cierta.

2.2.4. Axiomas de la lógica proposicional.

A continuación, definiremos el conjunto de axiomas que fundamentan a la lógica proposicional. Dichos axiomas son:

1. $A \Rightarrow (B \Rightarrow A)$

2. $(A \Rightarrow B) \Rightarrow [\{A \Rightarrow (B \Rightarrow C)\} \Rightarrow (A \Rightarrow C)]$

3. $A \Rightarrow [B \Rightarrow (A \wedge B)]$

4. $A \wedge B \Rightarrow A$

5. $A \Rightarrow A \vee B$

6. $(A \Rightarrow C) \Rightarrow [(B \Rightarrow C) \Rightarrow \{(A \vee B) \Rightarrow C\}]$

7. $(A \Rightarrow B) \Rightarrow [(A \Rightarrow \neg B) \Rightarrow A]$

8. $\neg(\neg A) \Rightarrow A$

Estos axiomas determinan y fundamentan las reglas de demostración de teoremas en este lenguaje formal.

2.2.5. Reglas de deducción de la lógica proposicional.

Una fórmula en el lenguaje formal descrito anteriormente es verdadera[14] si coincide con un axioma o puede comprobarse a partir de los axiomas por medio de la ***regla de deducción simple***[15] dada por:

REGLA DE DEDUCCIÓN SIMPLE.

Sean A y B dos fórmulas bien formadas. Si A es verdadera y si $A \Rightarrow B$ es verdadera, entonces B es verdadera.

La regla de deducción simple también se puede interpretar como: Dadas las fórmulas bien formadas A y $A \Rightarrow B$, entonces B es otra fórmula bien formada. Se expresa simbólicamente como: $[(A \Rightarrow B) \wedge A] \Rightarrow B$.

Una fórmula que se obtiene a partir de axiomas y de reglas de deducción será una fórmula verdadera.

Uno de los métodos que generalmente se emplea para probar cuando una fórmula es verdadera, consiste en utilizar las conocidas ***tablas de verdad*** de la Lógica Matemática. Ellas nos permiten, además verificar cuando dos proposiciones tienen el mismo valor lógico (cierto o falso), al resultar:

---

[14] BENCHIMOL, Guy; LEVINE, Pierre; y POMEROL, Jean Charles. Los sistemas expertos en la empresa. Página 41.

[15] Mejor conocida como ***modus ponens, regla de eliminación, o de razonamiento directo.***

- ☑ ***Tautologías:*** si en cualquier caso, la proposición asume valores verdaderos.

- ☑ ***Contradicción:*** si en cualquier caso, la proposición asume valores falsos.

- ☑ ***Contingencia:*** si la proposición no es tautología o contradicción.

La tabla de verdad mostrada a continuación, a manera de ejemplo, nos recuerda los valores verdadero (V) o falso (F), para los conectores lógicos o ***conectivas*** $\neg$, $\vee$, $\wedge$, $\Rightarrow$, para dos proposiciones dadas A y B.

A	B	$\neg A$	$A \vee B$	$A \wedge B$	$A \Rightarrow B$
V	V	F	V	V	V
V	F	F	V	F	F
F	V	V	V	F	V
F	F	V	F	F	V

Observemos que: puesto que cada proposición que interviene en la tabla de verdad puede asumir 2 valores únicamente (cierto o falso, V o F), si n es el número de proposiciones, entonces habrá 2^n posibles combinaciones veritativas.

De todo lo que hemos expresado hasta este momento, podemos percatarnos que una proposición es, en esencia, una frase o declaración que

puede ser verdadera o falsa[16]. De hecho, este es uno de los conceptos más importantes dentro de nuestro estudio.

2.3. Ejemplo del cálculo proposicional aplicado a la formalización del razonamiento.

En este punto conviene que aclaremos el sentido de nuestra exposición en cuanto al cálculo de predicados. Para tal efecto, haremos una breve referencia a temas que serán desarrollados con más detalle en capítulos posteriores, pero que tenemos que mencionar para que se pueda apreciar el sentido de nuestra discusión.

Supongamos que deseamos expresar el siguiente razonamiento de un corredor de valores, en términos manejables para un sistema experto basado en el cálculo proposicional:

> "Si los *precios del petróleo bajan*, y el *índice de empresas petroleras permanece estable*, entonces venderemos a *ESSO* y compraremos a *SHELL*".

Para resolver el problema, utilizaremos los conceptos previamente analizados, acerca de la lógica proposicional. Utilizaremos proposiciones para representar los hechos establecidos en nuestro razonamiento - modelo. A dichas proposiciones les asignaremos valores, que dependerán de los que expresa el razonamiento base. En consecuencia, podemos plantear las siguientes proposiciones o hechos:

---

[16] LIU, C. L. Elementos de matemáticas discretas. Página 28.

- ☑ *precios del petróleo = bajando*
- ☑ *índice de empresas petroleras = estable*
- ☑ *vender = ESSO*
- ☑ *comprar = SHELL*

En tanto, la regla de deducción será de la forma $(A \wedge B) \Rightarrow (C \wedge D)$, es decir:

- ☑ Si (*precios del petróleo = bajando* y *índice de empresas petroleras = estable*) entonces (*vender = ESSO* y *comprar = SHELL*).

2.4. Deficiencias de la lógica proposicional o de predicados.

El sistema que hemos descrito para representar el conocimiento, a la larga, resulta molesto, tedioso y repetitivo porque ***se tiene que escribir una regla por cada hecho que se considere***.

Por ejemplo, si consideramos que en nuestro razonamiento debemos considerar una compañía adicional, del ramo de las computadoras, denominada *IBM* , y un índice de sus ventas, que están en ascenso.

Observemos que la primera regla descrita en el ejemplo práctico no nos libra de tener que escribir:

☑ Si $índice\ de\ ventas = aumenta$ entonces $comprar = IBM$

para expresar que si el índice de ventas de IBM aumenta, debemos comprarla.

Observemos que el sistema que plantea la lógica proposicional es bastante limitado, pues se observa que para manejar este tipo de conocimiento, deberíamos introducir algún tipo de construcción lógica válida que nos permita representar funciones[17] tales como $PERTENECE(IBM, informatica)$ o $INDICE(precios, estable)$. Pero esto no es posible, dentro del sistema que hemos descrito relativo a la lógica proposicional.

En consecuencia, debemos ampliar el grado de generalidad de nuestro lenguaje formal. Esta ampliación de su cobertura la logramos en la lógica de primer orden, que pasamos a considerar de inmediato.

2.5. La lógica de primer orden.

La lógica de primer orden puede considerarse, al igual que el cálculo de predicados, como un lenguaje formal. También se le conoce como ***"cálculo de predicados de primer orden".***

Nuestro punto de partida será el lenguaje definido en la sección precedente. De hecho, la lógica de primer orden es una extensión natural de los conceptos previamente planteados.

---

[17] Desde el punto de vista matemático.

2.5.1. Alfabeto de la lógica de primer orden.

Los elementos que definen el alfabeto de la Lógica Proposicional, serán reconsiderados, para definir el siguiente alfabeto:

- ☑ ***Constantes individuales:*** Denotadas con las primeras letras minúsculas individuales, por ejemplo: a,b,c,d .

- ☑ ***Variables individuales:*** Denotadas con las últimas letras minúsculas individuales, por ejemplo: x,y,z .

- ☑ ***Predicados:*** Denotados por letras mayúsculas, por ejemplo: $P_n,Q_n,R_n,S_n,\ldots$. Cada predicado está asociado a un ***"peso"***[18] n, donde n es un entero positivo o cero. Se asumirá, por definición, que $P_0 = P$. Es decir: un predicado de peso 0 será equivalente a una proposición.

- ☑ ***Operadores o conectores lógicos:*** Nuevamente son los objetos: $\neg,\wedge,\vee,\Rightarrow$.

- ☑ ***Funciones:*** Denotadas por letras minúsculas, por ejemplo: $p_n,q_n,r_n,s_n,\ldots$. Cada función está asociada a un ***"peso"***[19] n, donde n es un entero positivo, no nulo.

---

[18] Aridad, según otros autores.

[19] Aridad, según otros autores.

- ☑ Los símbolos $\forall(\ ),\exists(\ )$, (los cuantificadores ***"para todo"*** y ***"existe",*** respectivamente). Son los equivalentes formales de las palabras ***"todo"*** y ***"algún"***.

- ☑ La coma, (,) además de los paréntesis $(\ )$.

2.5.2. Sintaxis de la lógica de primer orden.

La sintaxis de este lenguaje se define por la adición de las siguientes reglas, al lenguaje que plantea la lógica proposicional:

- ☑ Los predicados tendrán un peso $n \geq 0$.

- ☑ Las funciones tendrán un peso $n \geq 1$.

- ☑ El peso de un predicado o función, definirá cuantas constantes o variables tendrá como ***argumentos***, considerados desde el punto de vista matemático[20].

Los ***"términos"*** se definen a continuación, recursivamente:

1. Las constantes individuales y las variables individuales son términos.

2. Si f_n es una función de peso n y si $t_1,t_2,...,t_n$ son n términos, entonces $f_n(t_1,t_2,...,t_n)$ es un término.

3. Los términos son definidos por (1.) y (2.), exclusivamente.

Las ***"fórmulas atómicas o átomos"*** se definen a continuación, recursivamente:

1. Una variable proposicional aislada es un átomo.

2. Si P_n es un predicado de peso n y si $t_1,t_2,...,t_n$ son n términos, entonces $P_n(t_1,t_2,...,t_n)$ es un átomo.

3. Los átomos son definidos por (1.) y (2.), exclusivamente.

Las ***"fórmulas bien formadas"*** se definen por inducción y recursión como sigue:

1. Una fórmula atómica es una fórmula bien formada.

2. Si F y G son fórmulas bien formadas, entonces $F \wedge G$, $F \vee G$, $\neg F$, $\neg G$, $F \Rightarrow G$, son fórmulas bien formadas.

3. Si G es una fórmula bien formada y x es una variable individual, entonces $\forall(x)(G)$, $\exists(x)(G)$, son fórmulas bien formadas, aparezca o no x en G.

---

[20] Por ejemplo, en la función $F(x)=y$, la variable x es un argumento de la función F. Además, la función F tiene peso o aridad 1.

4. Las fórmulas bien formadas son definidas por (1.), (2.) y (3.), exclusivamente.

2.5.3. Interpretación semántica de la lógica de primer orden.

Para ***interpretar*** correctamente el sentido del cálculo de predicados de primer orden, debemos agregar a la interpretación semántica de la lógica proposicional, las definiciones que enunciamos a continuación:

- ☑ La lógica de primer orden se debe interpretar dentro de un ***dominio de interpretación*** D.

- ☑ Las constantes individuales representan a elementos específicos (determinados) dentro del dominio de interpretación D.

- ☑ Las variables individuales representan a elementos (indeterminados) dentro del dominio de interpretación D.

- ☑ Emplearemos el término ***asignación de*** x**,** para cualquiera de los valores que pueda tomar x en D, siendo x una constante o variable individual.

- ☑ Un predicado de peso $n(n>0)$ es una función de D_n en el conjunto verdadero, falso.

- ☑ Un predicado, es verdadero o falso dependiendo de los valores de sus argumentos.

- ☑ Un predicado de peso 0 es una proposición que puede ser verdadera o falsa.

- ☑ Las funciones representan mapeos del dominio de interpretación sobre sí mismo.

2.5.4. Axiomas de la lógica de primer orden.

Además de los axiomas de la lógica de predicados, adoptamos los axiomas:

9. $\forall(x)(P(x)) \Rightarrow P(a)$

10. $P(a) \Rightarrow \exists(x)(P(x))$

El axioma 9., llamado de la ***especialización universal***, establece que si para toda x del dominio de interpretación D, la propiedad $P(x)$ es verdadera, entonces $P(a)$ ***es verdadera, para toda*** a ***en*** D.

Del mismo modo el axioma 10., o de la ***generalización existencial*** dice que si un elemento en un dominio de interpretación tal que $P(a)$ es verdad, entonces ***existe un*** x ***tal que*** $P(x)$ ***es verdadero***.

Ambos cuantificadores están relacionados a través de las siguientes fórmulas de negación de cuantificadores:

- ☑ $\neg[\forall x P(x)] = \exists x[\neg P(x)]$

- ☑ $\neg[\exists x P(x)] = \forall x[\neg P(x)]$

Los predicados no pueden cuantificarse, es decir, que $\forall PP(x,y) \Rightarrow Q(z)$ no es una fórmula bien formada.

2.5.5. Reglas de deducción de la lógica de primer orden.

Al igual que en el caso de la lógica proposicional, una fórmula es verdadera si coincide con un axioma o puede comprobarse a partir de los axiomas por medio de la ***regla de deducción simple***.

2.6. Ejemplos de la lógica de primer orden, aplicados a la formalización del razonamiento.

No proseguiremos más en la teoría de predicados y de sus interpretaciones —no es necesario para los fines de esta monografía—, y nos quedaremos en la exposición de algunos ejemplos de fórmulas aplicadas a casos generales y concretos de formalización del razonamiento, que es lo que en realidad persigue este trabajo.

Para separar fórmulas que pueden tener igualmente el valor de verdadero o falso, emplearemos el signo $=$ como lo hicimos en la sección precedente.

Volvamos ahora con nuestro ***"experto"*** agente de bolsa y veremos como el cálculo de predicados permite que tal razonamiento se pueda formalizar. Nos interesa redactarlo en términos de que se pueda aplicar a cualquier industria de cualquier ramo.

En primer término, consideremos un predicado definido por $PERTENECE(x,y)$. Dicho predicado tiene peso 2. Será verdadero si la variable

representa a x una empresa que pertenece al sector representado por la variable y. Por ejemplo, los predicados:

- ☑ $PERTENECE(IBM, petróleo)$ y
- ☑ $PERTENECE(ESSO, informatica)$ son falsos.

Del mismo modo:

- ☑ $PERTENECE(IBM, informatica)$ es un predicado verdadero.

Adicionalmente, debemos considerar el predicado $INDICE(x, y)$, donde la variable x representa a un sector específico, y la variable y representa los estados en que se puede encontrar un sector cualquiera: sube, baja, estable. Será verdadero cuando el valor de y corresponde al estado del sector x. De ese modo, si deseamos representar que el índice de precios de la industria del petróleo esta en alza, podríamos utilizar el predicado:

- ☑ $INDICE(petróleo, sube)$ que es verdadero.

También debemos considerar el predicado $PRECIO(x, y)$, donde la variable x representa a un producto específico, y la variable y representa los estados en que se puede encontrar dicho producto: alza, baja, estable. Será verdadero cuando el valor de y corresponde al estado del producto x. De ese modo, si deseamos representar que el los precios del petróleo esta en alza, podríamos utilizar el predicado:

- ☑ $PRECIO(petróleo, alza)$ que es verdadero.

Finalmente, debemos considerar los predicados $VENDER(x)$ y $COMPRAR(x)$, donde la variable x representa a una empresa específica. Serán verdaderos cuando se desee vender o comprar a la empresa x, respectivamente. De ese modo, si deseamos representar que deseamos vender a la empresa IBM y comprar a la empresa NCR, podríamos utilizar los predicados:

- ☑ $VENDER(IBM)$ y
- ☑ $COMPRAR(NCR)$ que serán verdaderos.

Ahora el razonamiento original del experto:

> "Si los *precios del petróleo bajan*, y el *índice de empresas petroleras permanece estable*, entonces venderemos a $ESSO$ y compraremos a $SHELL$".

puede plantearse en términos más generales como:

> Si $PRECIO(petróleo, baja)$ y $INDICE(petróleo, estable)$ y $PERTENECE(ESSO, petróleo)$ entonces $VENDER(ESSO)$ y $COMPRAR(SHELL)$.

considerando, adicionalmente, los siguientes hechos:

- ☑ $PRECIO(petróleo, baja)$
- ☑ $INDICE(petróleo, estable)$

- ☑ $PERTENECE(ESSO, petróleo)$
- ☑ $VENDER(ESSO)$
- ☑ $COMPRAR(SHELL)$

La mayor diferencia con la lógica proposicional es que una vez escrita la regla anterior, se puede aplicar a cualquier otra empresa en cualquier sector, al actualizar los hechos correspondientes en nuestro razonamiento.

Podemos utilizar el mismo proceso para representar otra forma razonamiento del experto. Nos referimos a razonamientos del tipo: ***"cualquier empresa que pertenezca al sector del acero, véndala"***. Esta especie de consejo la podemos representar, utilizando los predicados antes establecidos, de la forma:

Si $PERTENECE(x, acero)$ entonces $VENDER(x)$.

3. Consideraciones finales.

Para finalizar, aclararemos un poco más el sentido de las funciones dentro del Cálculo de Predicados, puesto que en nuestra exposición no se ha profundizado en ellas.

Las funciones dentro del Cálculo de Predicados pueden interpretarse naturalmente como funciones $D_n \mapsto D$ (siendo n cualquier entero positivo y D el conjunto de interpretaciones). Las funciones en el Cálculo de Predicados desempeñan el mismo papel que las variables y las constantes, es decir, si f es una función de segundo orden, x es una variable, a es una constante y P es

un predicado de peso 2 entonces $P(x, f(x,a))$ es una fórmula atómica. Ellas nos permiten ampliar más la posibilidad de formalizar la representación del razonamiento humano.

4. Bibliografía.

1. AHO, Alfred V; SETHI, Ravi y ULLMAN, Jeffrey D. Compiladores: Principios, técnicas y herramientas. Traducido por Pedro Flores Suárez y Pere Botella y López. Primera edición. U. S. A., Wilmington, Delaware: Addison - Wesley Iberoamericana, 1990. 820 páginas.

2. BENCHIMOL, Guy; LEVINE, Pierre; y POMEROL, Jean Charles. Los sistemas expertos en la empresa. Sin traductor. Macrobit Editores, S. A. de C. V. México, D. F., México. Primera edición, 1990, 199 páginas.

3. CHINON AMÉRICA, INC. Enciclopedia interactiva Santillana. Primera edición. Primera edición. 1995. Sin numeración de páginas.

4. CORNEJO JORDÁN, Arsenio. Introducción a la Lógica Matemática. Primera edición, Panamá, Panamá: Universidad de Panamá, 1993. 98 hojas.

5. GÓLCHER, Ileana. Escriba y sustente su tesis: Metodología para la investigación social. Panamá, Panamá, Servicios Gráficos. 1995, 166 páginas.

6. JOHNSONBAUGH, Richard, Matemáticas discretas. Traducido por Reinaldo Giudici E. y María Rosa Brito. Primera Edición. México D.F., México: Grupo Editorial Iberoamérica, 1992. 506 páginas.

7. KORFHAGE, Robert R. Lógica y Algoritmos: Con aplicaciones a las ciencias de la computación e información. Traducido por Federico Velasco C. Primera edición. México, D.F., México: Limusa-Wiley 1970. 222 páginas.

8. LIPSCHUTZ, Seymour. Estructura de datos. Traducido por Manuel Ortega Ortíz de Apodaca y Luis Hernández Yañez. Primera edición. México, D.F., México: McGraw-Hill, 1987. 390 páginas.

9. LIU, C. L. Elementos de matemáticas discretas. Traducido por Luis Armando Díaz Torres. Segunda edición. México D.F., México: McGraw-Hill. 1995. 432 páginas.

10. MULLER, Horst. Formas del trabajo Heurístico en la Enseñanza de la Matemática. Boletín Nº 6, Sociedad Cubana de Matemática. Cuba, 1986.

11. NORTON, Peter. Introducción a la computación. Traducido por Leslie Charles Dawe Barnett, Saúl Ramón Flores Soto y Lucas Marreron Razo. McGraw-Hill/Interamericana de México, S. A. de C. V., México D. F., México. Primera edición, 1995, 567 páginas.

12. PFAFFENBERGER, Bryan. Diccionario para usuarios de computadoras. Traducido por Oscar Palmas Velasco. Quinta edición. México, D.F., México: Prentice-Hall, 1994. 590 páginas.

13. ROBINSON, Phillip R. Aplique Turbo PROLOG. Traducido por José María Troya Linero. Libros mcGraw-Hill de México, S. A., de C. V., México, D. F., México. Primera edición., 1987, 338 páginas.

14. ROHN, Karl. Consideraciones acerca de la “Enseñanza problémica” en la Enseñanza de la Matemática (I) y (II). Boletín Nº4. Sociedad Cubana de Matemática. Cuba, 1985.

15. SANDERS, Donald H. Informática: Presente y futuro. Traducido por Roberto Luis Escalona. Tercera Edición. México D.F., México: McGraw-Hill, 1991. 887 páginas.

16. SCOTT, Patrick B. Las computadoras y la enseñanza de las Matemáticas. Revista Educación Matemática. Volumen 2, Número 1, Abril de 1990, páginas 46-49.

17. SMITH, Karl J., Introducción a la Lógica. Traducido por Eduardo M. Ojeda Peña. Primera Edición. México D.F., México: Grupo Editorial Iberoamérica, 1991. 117 páginas.