Teórico 2 – Fundamentos de lógica

Introducción

Toda ciencia está (o debería estar) basada en la lógica. Toda teoría es un sistema de sentencias o declaraciones, que se aceptan como verdaderas y pueden ser utilizadas para obtener nuevas declaraciones utilizando reglas bien definidas.

Conceptos fundamentales

Valores, variables y tipos

Necesitaremos distinguir los términos de Valor, Variable y Tipo; términos que muchas veces se confunden.

- Un Valor es una constante individual, con un significado bien definido (por ejemplo, el entero 17). Un Valor no se puede modificar, ya que no sería el mismo Valor. Los Valores se codifican de alguna manera y pueden ser arbitrariamente complejos.

- Una Variable mantiene un Valor, pero puede mantener múltiples Valores a lo largo del tiempo. Las Variables sí se pueden modificar; cambiando el Valor que mantienen. Las Variables tienen nombre, lo que nos permite hablar (o predicar) sobre ellas.

- Diremos que todas las Variables tienen un Tipo, y diremos que el Tipo de una Variable es el conjunto de todos los Valores que la Variable puede mantener. Ejemplos comunes de tipo son strings, números enteros o fechas.

Una base de datos se puede ver como una variable; en cada momento del tiempo puede tener un determinado valor (de un tipo muy complejo, determinado por el esquema de la base).

Proposiciones y predicados

- Una proposición es una sentencia declarativa que puede ser verdadera o falsa, y se debe poder decidir cuál de las dos. Una forma de determinar si una sentencia S es declarativa es verificando que tenga sentido la pregunta ¿es cierto que S?. Diremos que las proposiciones tienen un valor de verdad, que puede ser verdadero o falso. Un ejemplo de proposición es "1 + 1 = 3", la cual obviamente, es falsa.

- Un predicado, a diferencia de una proposición, admite variables de las que no se conoce su valor de verdad. Estas variables se conocen como parámetros del predicado. Un ejemplo de predicado es "X aprobará Base de Datos 1", y podemos suponer que el valor de verdad de este predicado dependerá del parámetro X, que varía en el conjunto de estudiantes. Si sustituye el lector la variable X por su nombre (esto se denomina instanciar un predicado con valores) obtendrá una proposición. Queda como tarea averiguar el valor de verdad de la proposición que obtiene.

Los parámetros de un predicado también se denominan variables libres, y se pueden cuantificar (esto es, vincular de alguna forma a un conjunto de valores).

Antes de continuar, diremos que no permitiremos algunos tipos especiales de declaraciones como "esta proposición es falsa", ya que son declaraciones auto referenciadas (se referencian a sí mismas) y generan paradojas con las que no queremos lidiar. También descartaremos declaraciones mal formadas, o sea, que no adhieran a nuestras reglas sintácticas. Por ejemplo, no admitiremos la proposición "3 es un elemento de 4" ya que 4 no es un conjunto.

Recordemos los conectivos lógicos básicos de la lógica proposicional: conjunción (AND), disyunción (OR) y negación (NOT). Los conectivos lógicos son operadores lógicos; toman

predicados y devuelven predicados, permitiendo construir (en un proceso inductivo) predicados compuestos a partir de predicados simples (predicados sin conectivos).

El significado preciso de los conectivos puede ser definido mediante tablas de verdad, que son estudiadas en los cursos de lógica.

P	Q	P AND Q
V	V	V
V	F	F
F	V	F
F	F	F

P	Q	P OR Q
V	V	V
V	F	V
F	V	V
F	F	F

P	NOT P
V	F
F	V

Queda como tarea recordar las tablas de verdad de la implicancia y de la equivalencia.

Una implicancia se puede considerar como un orden entre predicados, determinando un predicado más fuerte y uno más débil. Si se tiene una implicancia verdadera P → Q se dice que P es más fuerte que Q (o que Q es más débil que P).

Por ejemplo, en la implicancia "X > 42 → X > 0", decimos que el predicado "X > 42" es más fuerte que el predicado "X > 0". Esto es consistente con la noción de cantidad de información, el primer predicado contiene más información (es más fuerte) que el segundo.

Note el lector que dados dos predicados P y Q no siempre se cumple que P → Q o que Q → P, por lo que la implicancia es un orden parcial en el conjunto de los predicados, pero no es un orden total.

Note el lector, además, que hay un predicado más fuerte que todos los demás y es valor F (falso), ya que para cualquier Q, se cumple que F → Q. De manera análoga, hay un predicado más débil que todos los demás y es el valor V (verdadero), ya que para cualquier P, se cumple que P → V.

Recordará el lector de los cursos de lógica que hay proposiciones (y de la misma forma predicados) que son equivalentes. Estaremos de acuerdo, por ejemplo, en que los predicados "X < 10 OR X > 20" y "NOT(X >= 10 AND X <= 20)" son equivalentes.

Recordará también que hay proposiciones (y de la misma forma predicados) que siempre son verdaderas, a los que llamamos tautologías; y en oposición contradicciones, que siempre son falsas. Por ejemplo "1 = 1" es una tautología, mientras que "1 = 2" es una contradicción.

A estas alturas, debe ser evidente que una declaración puede ser rescrita en una forma equivalente, pero que tal vez requiera menos trabajo de cómputo para ser evaluada por una computadora. Por ejemplo, el predicado "X < 2 OR X > 1" es una tautología, y es equivalente a V. Lo que se puede ganar con una reescritura, es que si X varía en un conjunto de 100 elementos sería necesario evaluar 200 desigualdades, y calcular 100 veces un conectivo OR para hallar el valor de verdad del predicado en función de cada valor del parámetro X. Si se rescribe el predicado, no importa dónde varíe X, se sabe de antemano que para cualquier X el predicado será verdadero. Normalmente, se puede evaluar de forma sistemática si se puede aplicar alguna tautología conocida, o regla de reescritura, como idempotencia (P AND P es equivalente a P, así como P OR P es equivalente a P), doble negación (NOT NOT P es equivalente a P), etc.

Valores nulos

Hasta aquí todo es sencillo, las proposiciones lógicas pueden ser verdaderas o falsas, y decimos que trabajamos en una lógica bi-valorada. Sin embargo, en sistemas de información, muchas veces desconocemos alguna información, o simplemente ésta no aplica (por ejemplo,

fecha de deceso en una base de personas). Para estos casos, admitiremos la existencia de un valor que no es verdadero ni falso, sino que representa la ausencia de información, sea porque es desconocido (UNKNOWN) o simplemente porque no aplica. Llamaremos NULL a este valor.

Es importante notar que el valor desconocido es un valor que aplica a todos los tipos de datos, y de naturaleza diferente al resto de los valores del tipo. Por ejemplo, NULL no debería ser igual que el string vacío o al número entero 0 (se debería codificar de una forma diferente).

La introducción del valor de verdad desconocido hace que debamos repensar la semántica de las operaciones lógicas. Por ejemplo, ¿qué debería devolver el OR entre falso y desconocido? Diremos que cada vez que el valor de verdad del resultado de una operación lógica dependa del valor que pudiera tener algún operando desconocido, el resultado será desconocido (para la semántica de la lógica tri-valuada conviene pensar en NULL como un valor desconocido).

Así, debemos construir nuevas tablas de verdad, extendidas, para admitir valores desconocidos.

P	Q	P AND Q
V	V	V
V	NULL	NULL
V	F	F
F	V	F
F	NULL	F
F	F	F
NULL	V	NULL
NULL	NULL	NULL
NULL	F	F

P	Q	P OR Q
V	V	V
V	NULL	V
V	F	V
F	V	V
F	NULL	NULL
F	F	F
NULL	V	V
NULL	NULL	NULL
NULL	F	NULL

P	NOT P
V	F
NULL	NULL
F	V

Más allá de las operaciones lógicas, al admitir el valor desconocido en variables de cualquier tipo, se nos presentan otros problemas como ¿cuál debería ser la suma de 5 y NULL? En general, cualquier operación que involucre un NULL devolverá NULL.

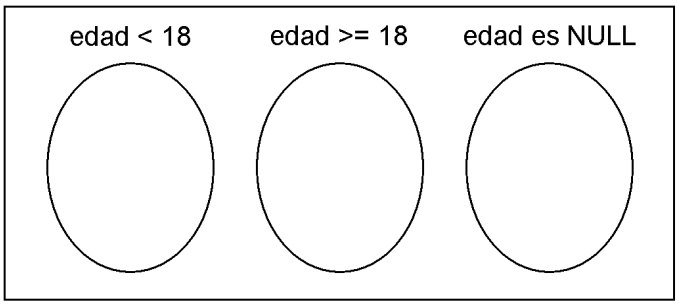
En ocasiones se debe devolver un valor de verdadero o falso, pero no desconocido. Por ejemplo, si tenemos que decidir si NULL es mayor que 9 y no podemos devolver NULL, tendremos que tomar una decisión. En general, se asume que cualquier comparación con NULL es falsa, pero por supuesto que esto nos traerá problemas.

En nuestra lógica bi-valuada teníamos que proposiciones (y admitamos que predicados) del tipo P OR NOT P eran tautologías, pero consideremos ahora que P tiene el valor NULL... y haga cuentas !!

Sólo para hacer énfasis en la complejidad del asunto, imaginemos que queremos realizar una partición del conjunto de personas; los mayores de edad y los menores de edad. Supongamos que tenemos una función para contar (COUNT), y que al contar las personas de 18 años o más el resultado nos da 37 personas. Por otro lado, las personas de menos de 18 años resultan ser 11. Podríamos deducir que el total de personas era de 37 + 11 = 48, sin embargo esto es cierto solamente si no había personas con edad desconocida (NULL). Para las personas con edad NULL, las comparaciones de desigualdad se asumirán falsas, por lo que no sumarán en ninguno de los dos conjuntos, y el total de personas podría ser mayor del que se dedujo.

En otras palabras, antes podíamos asumir que un conjunto se podía dividir en una partición de dos subconjuntos, los que cumplían P y los que cumplían NOT P. Con valores nulos, un

conjunto se divide en los que cumplen P, los que cumplen NOT P y los que por tener valores nulos no cumplen P ni tampoco NOT P.



En resumen, la introducción del valor NULL es un mal necesario, y se debe tener mucho cuidado cuando los datos pueden admitir valores NULL. En los manejadores de bases de datos existen formas de restringir los tipos de datos para que no admitan valor NULL, y a veces se recomienda hacer esto para evitar el valor NULL tanto como sea posible, especialmente para evitar errores como el del ejemplo anterior.