Computational Logic and Human Thinking: How to be artificially intelligent

Robert A. Kowalski

Department of Computing
Imperial College

Traduce: Jacinto Dávila

Contenido

Atomos

En todas las variantes de la lógica, el bloque de construcción básico es la *fórmula atómica* o *átomo*, para decirlo en breve. Al igual que en la física, en donde un átomo puede ser visto como una colección de electrones que son mantenidos juntos por un núcleo, los átomos en lógica son colecciones de *términos*, como "tren", "conductor" y "estación", mantenidos juntos por *símbolos de predicados*, como "en" o "parada". Los ímbolos de predicados son como los verbos en Inglés, y los términos son como los sustantivos o frases de sustantivos.

Átomos

Donde hemos escrito informalmente:

El conductor detiene el tren

en Lógica Simbólica, se escribiría normalmente así:

detiene(conductor, tren).

Átomos

En esa forma, el símbolo del predicado se escribe primero, seguido de los términos del átomo, que son llamados sus *argumentos*, rodeados por paréntesis y separados por comas. Cada símbolo de predicado tiene un número estándar de argumentos, escrito un órden fijo, pero arbitrario. En el ejemplo, el símbolo *detiene* tiene dos argumentos, con el sujeto *conductor* primero y el objeto *tren* de segundo.

La ventaja de la forma simbólica de la lógica para escribir átomos es que distingue, sin ambigüedad, entre el predicado y sus argumentos y, además, identifica los diferentes roles (tales como sujeto u objeto) de sus argumentos por su posición dentro de los paréntesis. Es esta precisión lo que hace de la Lógica Simbólica apropiada para el procesamiento computacional.

Átomos

Sin embargo, esta ventaja tiene un costo: tener que sobredefinir los componentes de un átomo. Por ejemplo, una representación igualmente apropiada de la oración *el conductor detiene el tren* es la fórmula atómica:

ocurre(detiene, conductor, tren)

Esta otra representación trata a *detiene* como un término en lugar de un símbolo de predicado. Es también posible, aún si no muy útil, representar la misma oración con un predicado sin argumentos, digamos *ocurre-detiene-conductor-tren()* o sin los paréntesis, simplemente *ocurre-detiene-conductor-tren*.

Átomos

De hecho, la representación más cercana a lo pretendido, subrayando el significado de la oración en lenguaje natural, es la colección de oraciones atómicas:

ocurre(evento-0014). tipo(evento-0014, detiene). agente(evento-0014, 007). objeto(evento-0014, el-escocés-volador). esun(007, conductor-de-tren). esun(el-escocés-volador, tren).

Esta representación vuelve explícito que el conductor *007* es un identificador individual, y que el tren es un tren particular con su propio identificador único *el-escocés-volador*. El mismo evento es un evento único, con un identificador *evento-0014* que lo distingue de cualquier otro evento en el cuál el mismo conductor detiene el mismo tren en alguna otra ocasión.

Átomos

Aún cuando tales representaciones son un tanto incómodas en comparación con las oraciones en lenguaje natural, suele ser necesarias en las implementaciones computacionales de la lógica, donde las distinciones que ellas introducen son inevitables. Se puede argumenta que tales distinciones son inevitables también en un lenguaje del pensamiento para un agente humano.

La representación informal que usamos en la mayor parte de este libro tiene la ventaja de que oculta esa complejidad subyacente en las representaciones más precisa. Sin embargo, el lector debe tener presente que, para representar el significado de aparentemente simples oraciones en lenguaje natural, tendrían que ser traducidas esas oraciones a una representación más precisa como la ilustrada acá.

Símbolos de Predicado (o simplemente predicados)

Los símbolos de predicados o predicados tienen cero, uno o más argumentos. Las fórmulas atómicas cuyos predicados tienen cero argumentos son llamadas por algunos *fórmulas proposicionales*. Estas incluyen los dos átomos especiales *verdadero* y *falso*. El caso especial de la Lógica Simbólica en el cual todos los átomos son fórmulas proposicionales es conocido como *lógica proposicional*. El caso más general, en el que los predicados tienen cualquier número de argumentos, se denomina *lógica de predicados*.

Símbolos de Predicado (o simplemente predicados)

Las fórmulas proposicionales son oraciones que denotan directamente *proposiciones*. Los predicados con un argumento denota *propiedades* de individuos y los predicados con más de un argumento denotan *relaciones* entre individuos. Esta distinción entre proposiciones, propiedades y relaciones es significativa en el lenguaje natural, pero es una complicacion innecesaria y no es bienvenida en la Matemática. Es más simple y más conveniente referirse a esas tres nociones como *relaciones*, que pueden mantener cero, uno o más individuos. Con esta convención, podemos decir directamente que los *símbolos de predicados denotan (o representan) relaciones*.

Símbolos de Predicado (o simplemente predicados)

No obstante, no todas las relaciones deben ser representadas por símbolos de predicados. Las relaciones pueden ser también representadas por predicados, propiamente dichos, que son expresiones compuestas construidas a partir de expresiones más simples que se combinan usando las conectivas lógicas como "y", "o", "no" y "si". Por ejemplo, la propiedad de ser alto y atractivo puede denotarse con un predicado, digamos alto(X) y atractivo(X), que no tiene porque ser expresado a través de un nuevo símbolo de predicado. Será conveniente, con frecuencia, hablar de tales predicados, sin tener que forzarles una representación con símbolos de predicados particulares.

La *Denotación* es una relación semántica entre los símbolos y los objetos que esos símbolos representan. Es uno de los grandes logros de la Lógica Simbólica, que atrae la envidia incluso de muchos de sus críticos, el tener una *semántica* apropiada. Pero antes de discutir la semántica, debemos completar la discusión de la *sintaxis*.

Términos

El tipo de término más simple es una *constante*, como 007, que denota un *individuo*, digamos la persona nacida el 1 de Abril del 200, hijo de Mary Smith y John Smith de Petworth, Inglaterra. Pero también es una término una *variable*, que representa a una clase completa de individuos. Es común en Lógica Simbólica usar letras como la X y la Y para designar variables, como en la fórmula algebraica:

$$X + Y = Y + X$$

que se cumple para todos los números X y Y. En este libro, usamos la convención, tomada en préstamo del lenguaje de programación Prolog, de que las variables comienzan con un letra en mayúscula, como esa X y esa Y, y que las constantes y los símbolos de predicados comienzan con minúsculas.

Términos

Términos más complejos se pueden construir a partir de los más simples, como *madre de X*, escrito así *madre(X)*, o $2 + 3$, escrito como $+(2,3)$. En estos casos *madre* y $+$ son *símbolos de funciones*. Sin embargo, las funciones son un caso especial de relaciones, así que los símbolos de funciones son, estrictamente hablando, innecesarios. En lugar de escribir, por ejemplo:

$$madre(cain) = eva$$

$$+(2,3) = 5$$

podemos escribir:

$$madre(cain,\ eve)$$

$$+(2,3,5)$$

Términos

Representar funciones como relaciones tiene la ventaja de que los símbolos de funciones pueden permanecer reservados para construir nombres de individuos. Los símbolos de funciones usando de esa manera son algunas veces conocidos como *funciones Skolem*, en honor al lógico Thoralf Skolem.

Usados para nombrar, los símbolos de funciones hacen posible en nombramiento de una infinidad de individuos con un vocabulario finito. Por ejemplo, en lógica matemática, es común nombrar los *números naturales* *0,1,2,...* con los términos *0, s(0), s(s(0)), ...* en donde el símbolo de función *s* es llamado la *función sucesor*.

Términos

El término *s(X)* es equivalente a *X+1*. Usando la función sucesor y representando la suma como una relación, podemos representar *2+3=5* con:

$$+(s(s(0)), s(s(s(0))), s(s(s(s(s(0))))))$$

Esta última no es muy fácil de leer, pero es mucho más apropiada para los estudios teóricos que el uso de sistemas numéricos alternativos como el decimal, el binario o los números romanos.

Los términos que no contienen variable alguna son llamados *términos básicos* (literalmente, *términos a tierra*). Juegan un papel especial en la semántica, pues constituyen el reservorio del cual se obtienen los nombres de los individuos.

Condicionales

Estrictamente hablando, una *condicional* es una oración con esta forma: $A \rightarrow B$, en donde A y B son oraciones. Sin embargo, podemos usar el término *condicional* con un poco más de flexibilidad para referirnos a oraciones que pueden contener variables. Más aún, en general, limitaremos la atención a condicionales que puedan escribirse en cualquiera de estas dos formas equivalentes:

$$C_1 \wedge ... \wedge C_n \wedge \neg D_1 \wedge ... \wedge \neg D_m \rightarrow E$$

es decir, *si* C_1 *y .. y* C_n *y no* D_1 *y .. y no* D_m *entonces* E

y

$$E \leftarrow C_1 \wedge ... \wedge C_n \wedge \neg D_1 \wedge ... \wedge \neg D_m$$

es decir, E *si* C_1 *y .. y* C_n *y no* D_1 *y .. y no* D_m

Condicionales

en donde la conclusión E es una fórmula atómica, las *condiciones* Ci son fórmulas atómicas y las *condiciones* $\neg Dj$ son las negaciones de fórmulas atómicas. A tales condicionales también se les conoce como *cláusulas* y cuando se las reune en conjuntos de tales condicionales se les llama a esos conjuntos *programas lógicos*.

Como se suele hacer con las definiciones matemáticas, la cantidad de condiciones positivas n y la cantidad de condiciones negativas m pueden una o la otra o ambas ser 0. Si m es 0, entonces la condicional es una *cláusula definida*.

Las cláusulas definidas son importantes por dos razones. Primero, son la forma adecuada de representar cualquier predicado computable. Segundo, como veremos en el capítulo A2, tienen una semántica muy simple en términos de modelos mínimos.

Condicionales

Si la cantidad de condiciones $n + m$ es 0, entonces la condicional degenerada $E \leftarrow (\text{o} \rightarrow E)$ es, de hecho, simplemente una oración atómica, y normalmente se le escribe sin la flecha. Simplemente como E.

La flecha hacia atrás $\leftarrow$ se lee *si*, y la flecha hacia adelante $\rightarrow$tiene el mismo significado, pero en dirección opuesta. El símbolo $\wedge$ es usando en lugar de la conectiva *y*. Las expresiones conectadas por $\wedge$ se denominan *conjunciones*.

Los símbolos de predicados y los símbolos constantes que aparecen en cláusulas diferentes son una especie de pegamento que atra esas cláusulas entre sí. Las variables son también un tipo de pegamento, interno a las cláusulas.

Condicionales

Por ejemplo, la variable X en la cláusula:

$$sorprendente(X) \leftarrow puede\_volar(X)$$

tiene el efecto de expresar que *todo lo que puede volar es sorprendente*.
Por otro lado, sin son dos variables en la cláusula como en:

$$sorprendente(X) \leftarrow puede\_volar(Y)$$

se tiene la expresión de *si algo puede volar entonces todo es sorprendente*!.

Condicionales

Las variables en las cláusulas se dice entonces que están *cuantificadas universalmente dentro del alcance de la cláusula en la que aparecen*. En Lógica Simbólica, la cuantificación de variables se escribe normalmente con los símbolos $\forall$ que significa *para todo* y $\exists$ que significa *existe*, y el alcance de esos cuantificadores es indicado por los paréntesis. De esta forma, las condicionales anteriores se escribirían:

$$\forall X(sorprendente(X) \leftarrow puede\_volar(X))$$

$$\forall X \forall Y(sorprendente(X) \leftarrow puede\_volar(Y))$$

Puesto que todas las variables en una cláusula están universalmente cuantificadas y su alcance es toda la cláusula, no se introduce ninguna ambigüedad si se omiten completamente los cuantificadores.

Condicionales

Puesto, además, que las condicionales pueden no tener condicionales, las oraciones atómicas pueden contener también variables universalmente cuantificadas. Considere este ejemplo simpático:

$$le\_gusta(bob, X).$$

Las oraciones atómicas que no contienen tales variables se llaman también *hechos*.

Condicionales

En la versión más simple de la Lógica Simbólica, las variables como X y Y se pueden referir a cualquier tipo de individuos. Así, por ejemplo, la cláusula $sorprendente(X) \leftarrow puede\_volar(Y)$ implica que si una piedra puede volar entonces esa piedra es sorprendente. De forma similar, la ecuación matemática $X + Y = Y + X$, si fuese escrita en notación lógica, implicaría que podría Ud sumar dos piedras en cualquier orden y el resultado sería el mismo.

Para superar ese uso poco natural de variables sin restricciones, se han desarrollado *lógicas con tipo* o *tipeadas*, en las cuáles las variables si están restringidas, de manera que se refieren solamente a individuos de ciertas clases, llamadas *tipos*. Un efecto similar puede obtenerse, con algo de dificultad, en una lógica sin tipos incluyendo una condición extra para cada variable, en cuyo predicado se exprese el tipo de la variable.

Condicionales

Por ejemplo, para establecer que todo animal que puede volar es sorprendente, tendríamos que escribir algo como esto en una lógica sin tipos:

$$sorprendente(X) \leftarrow puede\_volar(X) \wedge animal(X)$$

Para concluir que toda persona que puede volar es sorprendente, necesitaríamos una cláusula adicional que exprese que las personas son animales:

$$animal(X) \leftarrow persona(X)$$

O, como los seguidores de la orientación por objetos en Computación preferirían que dijeramos, la clase de todas las personas hereda la propiedad de volar de la más abstracta clase de todos los animales.

Condicionales

En la versión informal de la Lógica Computacional que usamos en este libro, no sólo omitimos los cuantificadores universales, sino que también usamos palabras como cualquiera o todos, en lugar de variables sin tipo. Las con tipo suelen también se reemplazadas por nombre comunes, como un animal, una estación o un ave. La virtud de este uso informal es que es neutral respecto a si se le formaliza en alguna versión de lógica con tipo o se usa, en su lugar, una lógica sin tipos con predicados para representar los tipos correspondientes. Por ejemplo, en lugar de escribir:

$$\forall X(sorprendente(X) \leftarrow puede\_volar(X) \wedge animal(X))$$

escribimos:

Si un animal puede volar entonces ese animal es sorprendente

o *cualquier animal que pueda volar es sorprendente*

Condicionales

Más aún, la versión informal es compatible con otras representaciones formales, tales como:

$$sorprendente(X) \leftarrow puede\_volar(X) \wedge esun(X, animal)$$

$$esun(X, animal) \leftarrow esun(X, persona)$$

Definiciones Recursivas

Las condicionales se usan frecuentemente para definir predicados. Por ejemplo, aquí tenemos la definición del predicado *número_natural*:

número_natural(0)

número_natural($s(X)$) $\leftarrow$ *número_natural*(X)

Se dice que esta es una definición recursiva, porque el predicado *número_natural*, que se define en la conclusión de la segunda oración, aparece entre sus condiciones (y viceversa). La habilidad de expresar las definiciones recursivas le da a las condicionales el poder completo de un lenguaje de programación de propósito general.

Definiciones Recursivas

Considere esta definición recursiva de la suma:

$+(0, Y, Y)$

$+(s(X), Y, s(Z)) \leftarrow +(X, Y, Z)$

Por simplicidad, omito las condiciones calificadoras de X, Y y Z como números naturales. En la notación funcional, la definición es mucho más simple y luce algo así como:

$0 + Y = Y$

$s(X) + Y = s(X+Y)$

Definiciones Recursivas

Eso también puede ser escrito en una forma todavía más simple $(X+1)+Y = (X+Y)+1$. Pero esto puede confundir, porque ese signo $+$ en la expresión $+1$ es diferente del signo $+$, por ejemplo, en $(X+Y)$. Diré otras cosas sobre la relación entre funciones y relaciones más tarde en este capítulo.

Cláusulas Meta

En la Lógica Computacional, usamos las condicionales (incluyendo hechos y otras oraciones atómicas) para representar creencias, cuyas variables están todas universalmente cuantificadas. Además, usamos conjunciones para representar metas cuyas variables están todas existencialmente cuantificadas.

En general, una cláusula meta es una conjunción existencialmente cuantificada de átomos o negaciones de átomos:

$$\exists X_1 ... \exists X_m (C_1 \wedge ... \wedge C_n \wedge \neg D_1 \wedge ... \wedge \neg D_m)$$

es decir, existe X_1.. y existe X_m tales que C_1y ... y C_ny no D_1y .. y no D_m

Si m es 0, entonces la cláusula meta se conoce como una cláusula meta definida.

Puesto que todas la variables en una cláusula meta están existencialmente cuantificadas dentro del alcance de la cláusula, es normal omitir el uso explícito del cuantificador existencial.

Cláusulas Meta

Por ejemplo, la cláusula meta:

$$le\_gusta(bob, X)$$

expresa realmente que

$$\exists X \; le\_gusta(bob, X)$$

Tales metas existencialmente cuantificadas son suficientes para representar las metas por lograr de un agente. Sin embargo, como veremos en mayor detalle más tarde, no alcanzan a representar metas de mantenimiento y restricciones.

Cláusulas Meta

Tanto las cláusulas definidas (que incluyen las oraciones atómicas) como las cláusulas metas definidas son normalmente etiquetadas como cláusulas de Horn, en honor al Lógico Alfred Horn, quien estudió algunas de sus propiedades matemáticas. Las cláusulas de Horn tienen el poder expresivo de las Máquinas de Turing, el modelo matemático estandar de computación mecánica.

En programación lógica, las cláusulas metas representan la computación que se habrá de realizar. Por ejemplo, la cláusula meta:

$$+(s(s(0)), s(s(0)), X) \wedge +(X, Y, s(s(s(s(s(0))))))$$

representa el problema de computar la suma X de 2 mas 2 y calcular un número Y que sumado a X de 5.

Otras Clases de Oraciones

Los condicionales, usados para representar creencias, y las cláusulas metas, usadas para representar metas por lograr, tienen todas una sintaxis muy simple. Sin embargo, las condicionales son lógicamente equivalente a oraciones más complejas en la sintaxis de la lógica clásica. Acá van algunos ejemplos de tales equivalencias:

$$\forall X \forall Y (sorprendente(X) \leftarrow puede\_volar(Y))$$

es equivalente a:

$$\forall X (sorprendente(X) \leftarrow \exists Y \; puede\_volar(Y))$$

Otras Clases de Oraciones

$$sorprendente(X) \leftarrow puede\_volar(X)$$

$$sorprendente(X) \leftarrow estrella\_de\_cine(X)$$

es equivalente a:

$$sorprendente(X) \leftarrow (puede\_volar(X) \vee estrella\_de\_cine(X))$$

$$generoso\_con(X, Z) \leftarrow le\_gusta(X, Y) \wedge da(X, Y, Z)$$

es equivalente a:

$$(generoso\_con(X, Z) \leftarrow le\_gusta(X, Y)) \leftarrow da(X, Y, Z)$$

Otras Clases de Oraciones

El símbolo$\vee$ es usado en lugar de la conectiva *o*. Las expresiones conectadas por$\vee$son denominadas disjunciones. En general, un disjunción tiene la forma:

$$C_1 \vee ... \vee C_n$$

es decir, C_1o .. o C_n

(NT: En Español, es la o no acentuada. La ó acentuada forzaría la exclusión entre las condiciones).

Otras Clases de Oraciones

Veremos más adelante que, además de permitir el uso de los cuantificadores existenciales y las disjunciones, es útil extender la sintaxis de la lógica condicional para representar metas y creencias más complejas. En particular, es útil incluir los cuantificadores existenciales y las disjunciones en las conclusiones de metas de mantenimiento. Por ejemplo:

Las metas de mantenimiento:

$$hambriento(yo) \rightarrow \exists X\ come(yo, X)$$

$$ataca(X, yo) \rightarrow huye(yo) \vee ataca(yo, X)$$

Los cuantificadores existenciales en las conclusiones de las metas condicionales son tan comunes que es conveniente omitirlos, conviniendo que las variables en la conclusión de una meta condicional que no aparecen en las condiciones de la meta están existencialmente cuantificadas, con un alcance sobre la conclusión de la meta. Por ejemplo, meta de mantenimiento:

Otras Clases de Oraciones

La inclusión de disjunciones en las conclusiones de las condicionales otorga a la lógica de las condicionales en poder de la lógica clásica. Diremos algunas otras cosas acerca de la relación entre la lógica de los condicionales y la lógica clásica en el capítulo A2. Nos enfocamos en la forma condicional de la lógica en este libro, porque es más fácil de entender tanto para los computadores como para los humanos.

Se puede argumenta que la relación entre la lógica clásica y la lógica de las condicionales es como la relación entre el lenguaje de comunicación entre humanos y el lenguaje del pensamiento humano.

Otras Clases de Oraciones

Una forma de entender esta relación es ver que el razonamiento involucra dos clases de reglas de inferencia, aplicadas en dos etapas. El primer tipo de reglas, aplicada en la primera etapa, transforma oraciones complicadas en oraciones más simples. El segundo tipo de reglas, aplicado en la segunda etapa, razona con esas oraciones más simples.

Este proceso de razonamiento de dos etapas es usado en muchos procedimientos de prueba desarrollados para la lógica clásica en Computación. En sistemas basados en el principio de Resolución (Robinso, 1965) en particular, la primera etapa transforma las oraciones de la lógica clásica a su forma clausal. La segunda etapa procesa las cláusulas usando refinamientos de la regla de inferencia llamada Resolución. Discutimos el principio de Resolucion en el capítulo adicional A5.

Otras Clases de Oraciones

Las comunicaciones en lenguaje natural pueden ser vistas como un proceso similar de dos etapas. La primera etapa transforma (o compila) oraciones del lenguaje natural en oraciones más simples en el lenguaje del pensamiento. La segunda etapa procesa esas oraciones más simples usando reglas de inferencia, como las de razonamiento hacia adelante y hacia atrás, que son casos simples de Resolución. Mientras más cercana estén las oraciones del lenguaje natural al lenguaje del pensamiento, menos esfuerzo requerirán para ser transformadas en oraciones en ese lenguaje del pensamiento, y serán más fáciles de entender.

La Negación

En la lógica clásica, las oraciones positivas y negativas tiene el mismo estatus. *Ser o no ser* - no hay razón para preferir una a la otra. Pero en la Lógica Computacional, las oraciones positivas son más básicas que las negativas, y estas normalmente sólo llenan los espacios entre oraciones positivas. Este estatus más básico de las oraciones positivas se refleja en las sintaxis de las condicionales, las cuáles normalmente sólo tienen conclusiones positivas, pero pueden tener condiciones negativas $\neg C$ (que también se escribe *no C*). Por ejemplo:

$$sancionable(X) \leftarrow presiona\_alarma(X) \wedge no\ hay\_emergencia$$

$$puede\_volar(X) \leftarrow ave(X) \wedge no\ pingüino(X)$$

La Negación

Tal como se explica en el capítulo 5 y en otros lugares, es natural concluir que una condición negativa *no C* se cumple si la condición *C* correspondiente no se cumple. Esta interpretación de la negación se llama *negación por falla*. De esta forma, dada una situación en la cual se nos dice que *ave*(*juan*), pero en la que no tenemos ningún indicio para creer *pingüino*(*juan*), se deriva por negación por falla que *puede_volar*(*juan*).

Considere esta definición de los números pares e impares, que usa sólo conclusiones positivas y una condición negativa:

$$par(0)$$

$$par(s(s(X))) \leftarrow par(X)$$

$$impar(X) \leftarrow no\ par(X)$$

La Negación

Puesto que no se puede demostrar que $par(s(0))$, se deriva de esas cláusulas y por negación por falla que $impar(s(0))$se cumple.

Además de las condiciones negativas interpretadas por la negación por falla, las oraciones negativas puede tener la forma de restricciones, que son metas condicionales con *falso* por conclusión. Por ejemplo, en el contexto de un agente que supervisa sus acciones candidatas, la restricción:

$$sancionable(X) \rightarrow falso$$

es decir, *no seas sancionable,*

sirve como una prohibición que impide acciones, como el que Ud presione el botón de la señal de alarma inapropiadamente o que Ud deje de pagar sus impuestos, que lo convertirían en sancionable.

La Negación

Más aún, como se ve en el capítulo sobre la Abdución, una restricción, tal como:

$$par(X) \wedge impar(X) \rightarrow falso$$

es decir, *nada es par e impar*,

que es una propiedad de la definición de los números pares e impares, puede ser usada para eliminar explicaciones incorrectas de las observaciones.

Veremos en su momento que ambos tipos de negaciones (la negación por falla y las restricciones) tienen la misma semántica de la negación en lógica clásica. Sin embargo, cumplen funciones muy diferentes en la representación del conocimiento y en el razonamiento.

Las Funciones, las Relaciones y la Igualdad

En este libro raramente usamos símbolos de funciones y lo hacemos solamente para construir nombres compuestos de individuos. Otros tipos de funciones son tratadas como relaciones (o predicados), como en las bases de datos relacionales. En lugar de escribir $f(X) = Y$, donde f es un símbolo de función, escribimos $f(X, Y)$, en el que f es un símbolo de predicado (o relación). En esta representación relacional, el hecho de que la relación es una función se representa con la restricción:

$$f(X, Y_1) \wedge f(X, Y_2) \rightarrow Y_1 = Y_2$$

Hemos logrado combinar esa representación relacional de las funciones con una noción simple de igualdad, comprendida como identidad, y definida por un sólo axioma:

$$X = X$$

Las Funciones, las Relaciones y la Igualdad

Esta representación, de las funciones como relaciones y de la igualdad como la identidad, funciona muy bien solamente si los individuos tienen nombres únicos. Así, por ejemplo, no es suficiente decir *bob detuvo el tren* si la misma persona se llama *robert* y si más de una persona se llama también *bob*. Tenemos que darle a *bob* un nombre único, digamos *007* por ejemplo, y decir algo como:

$$detiene(007, el\_tren)$$

$$nombre(007, bob)$$

$$nombre(007, robert)$$

$$nombre(008, bob)$$

Consideraciones similares se deben aplicar al nombre del tren, desde luego, y quizás también a el nombre del evento, como se mostró antes en este capítulo.

Las Funciones, las Relaciones y la Igualdad

La definición de igualdad como identidad significa que dos individuos son idénticos si y sólo si tienen el mismo nombre único. Esto constrasta con la noción más convencional de igualdad, en la que el mismo individuo puede tener varios nombres. Por ejemplo:

$$el\ lucero\ de\ la\ mañana = la\ estrella\ de\ la\ tarde$$

$$doctor\_jekyll = mister\_hyde$$

Las Funciones, las Relaciones y la Igualdad

Para razonar con las igualdades de este tipo, es normal usar axiomas adicionales, tales como las cláusulas definidas siguientes:

$$X = X$$

$$f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n) \leftarrow X_1 = Y_1 \wedge \ldots \wedge X_n = Y_n$$

$$p(X_1, \ldots, X_n) = p(Y_1, \ldots, Y_n) \leftarrow X_1 = Y_1 \wedge \ldots \wedge X_n = Y_n$$

para cada símbolo de función f y para cada símbolo de predicado p. Sin embargo, razonar con tales axiomas es costoso computacionalmente. Más aún, su uso requiere cierta precaución, si es que queremos hacer distinciones como:

$$bueno(doctor\_jekyll) \wedge malo(mister\_hyde)$$

La Lógica Clásica

La sintaxis de la lógica clásica es una extensión de la sintaxis de la forma condicional de la lógica usada en este libro. Los términos y fórmulas atómicas en lógica clásica son los mismos que en la lógica de los condicionales. Sin embargo, las oraciones no atómicas pueden construirse usando una combinación arbitraria de las conectivas lógicas $\rightarrow$, $\wedge$, $\vee$ y $\neg$, y los cuantificadores $\forall$y $\exists$.

La lógica clásica está peor estructurada que la forma condicional de la lógica. Por ejemplo, en la forma condicional, hay una sola forma de expresar que *todas las aves pueden volar* y que *Juan es un ave*:

$$puede\_volar(X) \leftarrow ave(X)$$

$$ave(juan)$$

La Lógica Clásica

Pero en la lógica clásica, las mismas creencias pueden ser descritas en muchas formas equivalentes, incluyendo:

$$\neg(\exists X((\neg puede\_volar(X) \wedge ave(X)) \vee \neg ave(juan)))$$

$$\neg(\exists X((\neg puede\_volar(X) \vee \neg ave(juan)) \wedge (ave(X) \vee \neg ave(juan))))$$

La Lógica Clásica

Para traducir la lógica clásica en oraciones en la forma condicional de la lógica, es preciso emplear reglas de inferencia que preservan tales equivalencias, como:

reemplace $\neg\exists X\ \neg A$ con $\forall X\ A$

reemplace $\neg A \vee \neg B$ con $\neg(A \wedge B)$

reemplace $A \vee \neg B$ con $A \leftarrow B$

La Lógica Clásica

La lógica clásica y la lógica condicional también se distinguen en el uso de los cuantificadores. En la lógica condicional, todas las variables en las condicionales están cuantificadas universalmente, y todas las variables en las cláusulas metas están existencialmente cuantificadas, por lo que los cuantificadores pueden ser omitidos. Pero en la lógica clásica, todas las variables pueden estar en cualquier caso cuantificadas universal o existencialmente, por lo que los cuantificadores deben ser explícitos.

La Lógica Clásica

En la lógica condicional, los cuantificadores existenciales se evitan otorgando un nombre a cada cosa que existe, el cual es o bien una constante o bien un símbolo de función que se aplica a otros nombres. En lugar de decir, por ejemplo, $\exists X$ *ave*(X), decimos *ave*(*juan*)o *ave*(007). Hacemos eso porque otorgar nombres explícitos a los individuos transmite más información. Si Ud sabe que *juan es un ave*, porque habría de ocultar la identidad de Juan diciendo solamente que *alguien es un ave*, especialmente cuando está Ud hablando con Ud mismo en su propio lenguaje de pensamiento.

La Relación entre La Lógica Clásica, Lógica Clausal y La Lógica Computacional

Cualquier cosa que se puede decir en lógica clásica puede también decirse en la forma condicional de la lógica, pero tiene que ser dicha usando solamente variables cuantificadas universalmente y permitiendo disjunciones en las conclusiones de las condicionales. Para decirlo mejor, cualquier oración de la lógica clásica puede ser traducida en un conjunto de *cláusulas* con la forma:

$$C_1 \wedge \ldots \wedge C_n \rightarrow D_1 \vee \ldots \vee D_m$$

donde cada condición C_i y cada conclusión D_j es una fórmula atómica, y toda variable en la clásula está implicitamente cuantificada universalmente, con alcance en toda la cláusula. Si n es 0, entonces $C_1 \wedge \ldots \wedge C_n$ es equivalente a *cierto*. Si m es 0, entonces $D_1 \vee \ldots \vee D_m$es equivalente a *falso*.

La Relación entre La Lógica Clásica, Lógica Clausal y La Lógica Computacional

Tradicionalmente, tales cláusulas están escritas en la forma lógicamente equivalente de una disjunción universalmente cuantificada (llamada *forma clausal*):

$$\neg C_1 \vee \ldots \vee \neg C_n \vee D_1 \vee \ldots \vee D_m$$

Aún cuando las oraciones de la lógica clásica pueden ser siempre traducidas a la forma clausal, la oración original y su traducción no siempre son logicamente equivalentes. Por ejemplo, la oración $\forall X \exists Y \, (mother(X, Y) \leftarrow person(X))$puede ser traducida como la cláusula $mother(X, mom(X)) \leftarrow person(X)$. Esta cláusula usa una función Skolem para nombrar un nombre y es, en cierta forma, más informativa que la original. Pero no es equivalente.

La Relación entre La Lógica Clásica, Lógica Clausal y La Lógica Computacional

En teoría, el uso de las funciones Skolem para reemplazar cuantificadores existenciales implica razonar con la igualdad. Por ejemplo, $mom(cain) = eve$. Sin embargo, tal tipo de de calificadores existenciales ocurre en la conclusión de las metas, en lugar de en las creencias. El procedimiento de prueba del capítulo A6 opera con cuantificadores existenciales explícitos en las conclusiones de las metas. De forma que los problemas de razonar con la igualdad, creados por las funciones Skolem, no parecen sugir mucho en la práctica.

En la lógica clausal, las metas de logro se resuelve por *reductio ad absurdum*, suponiendo su negación y derivando *falso* del o con el conjunto de cláusulas que resulta.

La Relación entre La Lógica Clásica, Lógica Clausal y La Lógica Computacional

Por ejemplo, la negación de la meta de logro:

$$\exists X_1 ... \exists X_m \, (C_1 \wedge \ldots \wedge C_n)$$

es equivalente tanto a la negación (universalmente cuantificada) siguiente:

$$(C_1 \wedge \ldots \wedge C_n) \rightarrow \mathit{falso}$$

como a esta cláusulas ordinaria (universamente cuantificada):

$$\neg C_1 \vee \ldots \vee \neg C_n$$

La Relación entre La Lógica Clásica, Lógica Clausal y La Lógica Computacional

Las metas de mantenimiento en lógica clausal se resuelven de la misma manera, convirtiendo su negación a la forma clausal y derivado *falso*. Sin embargo, puesto que las metas de mantenimiento están universalmente cuantificadas, su negaciones estarán existencialmente cuantificadas y estos cuantificadores existenciales deben ser reemplazados por constantes Skolem. Por ejemplo, para resolver la meta de mantenimiento:

$$ataca(X, yo) \rightarrow huye(yo) \vee ataca(yo, X)$$

es necesario reemplazar la variable X con una constantes Skolem, digamos : $-($ y convertir la negación de la condicional skolemizada en las siguientes cláusulas:

La Relación entre La Lógica Clásica, Lógica Clausal y La Lógica Computacional

$$ataca(' : -(', yo)$$

$$\neg huye(yo)$$

$$\neg ataca(yo,' : -(')$$

Si esta forma de resolver metas de mantenimiento tiene éxito (deriva *falso*), entonces tiene éxito al resolverlas de una vez y para siempre.

Sin embargo, en este libro resolvemos las metas de mantenimiento de una forma diferente, demostrando que cuando quiera quer sus condiciones son *cierto*, sus conclusiones también son *cierto*. Esta forma alternativa de las metas de mantenimiento se discute informalmente en el capítulo 8 y se formaliza en el capítulo A6.

La Relación entre La Lógica Clásica, Lógica Clausal y La Lógica Computacional

El tratamiento diferente de las metas de mantenimiento refleja el hecho de que ni la lógica clásica, ni la lógica clausal establecen distinción fundamental alguna entre metas y creencias. Por el contrario, nosotros distinguimos entre metas y creencias por medio de pequeñas variantes de la forma clausal, para las metas, y la forma de programa lógico:

$$C_1 \wedge \ldots \wedge C_n \wedge \neg D_1 \wedge \ldots \wedge \neg D_m \rightarrow E$$

ó

$$E \leftarrow C \wedge \ldots \wedge C_n \wedge \neg D_1 \wedge \ldots \wedge \neg D_m$$

para las creencias. Como se dijo antes, las conclusiones de las metas (pero no las de las creencias) pueden contener tanto disjunciones como variables existencialmente cuantificadas.

La Relación entre La Lógica Clásica, Lógica Clausal y La Lógica Computacional

Agregando confusión, cosa que suele pasar en la literatura, yo uso el término *cláusula* para referirme a las cláusulas escritas como condicionales, a las cláusulas escritas como disjunciones y a cláusulas para programación lógica. Todavía peor, uso el término *condicional* tanto para las cláusulas escritas como condicionales con conclusiones disjuntivas como para las cláusulas en programación lógica. También llamo a la combinación resultante de las dos clases de condicionales, *la forma condicional de la lógica* y *la forma de la Lógica Computacional usada en este libro*. Con suerte, en la mayoría de los casos el contexto hará que el significado que pretendo sea obvio.

Conclusiones y otras referencias

Este recorrido a vuelo de pájaro sobre la sintaxis de la forma condicional de la lógica y sus relaciones con las formas estándar y clausal de la lógica clásica ha cubierto mucho terreno, pero sólo ha tocado la superficie.

La forma condicional de la lógica es tan poderosa, pero más simple, que la oraciones sin forma definida de la lógica clásica. Las reglas de inferencia de la forma condicional son también más simples. Las reglas de inferencia de la lógica clásica son más complejas porque, de hecho, además de las reglas necesarias para razonar con condicionales, también incluyen reglas para traducir oraciones de la lógica clásica en oraciones equivalentes en la forma condicional.

Conclusiones y otras referencias

Esta distinción entre las dos clases de reglas de inferencia en la lógica clásica corresponde a la distinción entre las dos clases de razonamiento en el lenguaje natural. Las reglas de inferencia necesarias para traducir la lógica clásica en condicionales corresponde a el razonamiento necesario para traducir el lenguaje natural en lenguaje del pensamiento, LOT. Y las reglas de inferencia necesarias para razonar con condicionales corresponden a las necesarias en el LOT.

Conclusiones y otras referencias

Para defender esta perspectiva de la relación entre la lógica clásica y la lógica condicional y entre el lenguaje natural y el LOT, me apoyo en las guías de buen estilo al escribir propuestas en libros como los de Williams (1990, 1995). Esas guías, que plantean la necesidad de claridad, simplicidad y coherencia en los escritos, pueden ser reinterpretadas como la defensa de un estilo de escritura que minimiza la diferencia entre la sintaxis de las comunicaciones en lenguaje natural y la representación de sus significados en LOT.

Conclusiones y otras referencias

La forma condicional de la lógica evolucionó de la forma clausal de la lógica, y esta a su vez evolución a partir de la lógica clásica estandar. Uno de los primeros usos de la forma clausal se le debe a Martin Davis y Hillary Putnam (1960) en uno de los primeros procedimientos de prueba mecánicos para la lógica clásica. También se le usó con la regla de resolución desarrollada por Alan Robinson (1965a).

Conclusiones y otras referencias

La aplicación de la forma clausal a la representación del conocimiento y de la regla de resolución a la resolución de problemas fue propuesta inicialmente por Cordell Green (1969). Sin embargo, los probadores de teoremas con resolución que estuvieron disponibles en esa época no fueron muy efectivos y, así, se hicieron objeto de ataques y críticas dirigidas al propio método de resolución por los defensores de las representaciones del conocimiento procedimentales, contrarias a las representaciones declarativas (Hewitt, 1971; Winograd, 1971, 1972).

Conclusiones y otras referencias

En defensa de la lógica clausal, Kowalski y Kuehner (1971) mostraron que la llamada resolucion-SL, básicamente una interpretación por resolución del procedimiento de prueba del modelo de eliminación, de Loveland (1968), podía ser interpretado procedimentalmente en términos de reducción de metas. En 1971 y 1972, tuve la oportunidad de colaborar con Alain Colmenauer en Marseille, resultando en el desarrollo que hiciera Colmenauer de Prolog en 1972, y en la interpretación procedimental de Resolución-SLD (Kowalski, 1974), una variante de resolución-SL aplicada a las cláusulas de Horn.

Conclusiones y otras referencias

En *Lógica para Resolver Problemas* (Kowalski, 1974, 1979), defendí con mayor generalidad el uso de la forma clausal para representación del conocimiento y razonamiento. Un análisis detallado de la relación entre la lógica clausal y la lógica clásica se hace en los capítulos 2 y 10 de aquel libro. La combinación, en la Lógica Computacional, de la lógica clausal para las metas y de los programas lógicos para la creencias proviene de la programación lógica abductiva (ALP) (Kakas, Kowalski y Toni, 1998). Los detalles técnicos de ALP se discuten en el capítulo A6.