Incompletitud: nociones preliminares

Hugo Ryckeboer

27 de junio de 2015

1. Nociones básicas

Rumbo a una prueba de incompletitud de la lógica de primer orden, necesitamos tener en claro algunos conceptos.

1.1. Reflexiones sobre los lenguajes

Es de notar que los símbolos lógicos que usamos son convencionales.

Todo el edificio de definiciones y teoremas se seguiría conservando si en lugar de '$\wedge$' se utilizara '&'. Inclusive si se usara 'A' en lugar de '$\forall$' y '$\forall$' en lugar de '('. Por supuesto que acostumbrarse a cambios tan poco intuitivos llevaría un rato y se prestaría a confusiones.)

Lo mismo pasaría con la aritmética. En primer lugar la computación ya acostumbró a que la base del sistema de numeración no tiene porque ser 10. Pero los dígitos no tienen porque ser 0, 1, 2, 3,... y bien podrían haber sido: a, b, c, d,... o α, β, γ,... como efectivamente usaban los griegos (arrancando en 1 ya que no conocían el 0).

Es más, la distinción entre letra, dígito, separador,... es cultural y es cómodo respetarlo; pero puede ser ignorado en bien de un mayor grado de abstracción.

Una vez fijado los grafismos de los cuales disponemos, queda delimitado el conjunto de hileras que se pueden formar con ellos. Si a esos grafismos le imponemos un orden, por ejemplo el que propone una tabla del código ASCII, es factible darle un buen orden a todas las hileras finitas.

Ese orden no puede ser el lexicográfico porque si nuestros grafismos fueran { `a`, `b` }, "`a`" sería la primera hilera, "`aa`" la segunda; pero la hilera "`b`" no tiene un predecesor, con lo cual en una recorrida sistemática nunca aparecería.

Pero cambiando el orden de modo tal que primero se ubiquen las hileras más cortas y a igualdad de longitud resolverlo lexicográficamente se tiene un buen orden y se podría calcular el número de orden de una hilera cualquiera sin necesidad de escribir los elementos que le preceden.

Esto es una de las muchas formas en las cuales se puede conceptualizar que una hilera es representable por un número. Tiene la ventaja de ser biyectiva. Sin embargo nos basta con que la aplicación de textos en naturales sea inyectiva y tengamos la capacidad de reconocer que un número no representa una hilera.

1.2. ¿Qué significa que algo sea decidible?

En temas teóricos de computación, y la lógica matemática es uno de ellos, aparece con alguna frecuencia la afirmación de que una propiedad de cierto clase de objetos es *decidible*.

Con ello se quiere indicar que es factible definir un sistema de representación textual de tales objetos y es factible escribir un programa que alimentado con la descripción de uno de los objetos concluya egresando —por ejemplo con un 0 o un 1— que el objeto ingresado no tiene o sí tiene la propiedad deseada.

La pregunta de decibilidad sólo aporta algo si el conjunto de objetos es infinito.

Así es *decidible* si la suma de dos números da un tercero. En primer lugar se dispone de sistemas de representación de números. Disponiendo de un símbolo separador se puede construir una representación de una terna de números. Se dispone además de algoritmos que construyen la representación de un número suma de otros dos ya representados. En el último paso habría que comparar la hilera construida con la tercera componente de la hilera dato.

1.3. ¿Qué significa que un conjunto sea enumerable?

Con similar frecuencia aparece la calificación de *enumerable*. Con ello se quiere indicar que se dispone de un sistema de representación de los elementos del conjunto y de un aprograma —no sería adecuado hablar de un algoritmo— que produzca uno tras otro las representaciones de los elementos del conjunto y una demostración de que un elemento de ese conjunto aparecerá en un tiempo finito.

Es de notar que cuando se piensa en tales programas hay que desprenderse de todos las finitudes de las máquinas reales. No hay cotas superiores en los números representables ni en los largos de las hileras almacenables. Las máquinas de Turing son máquinas ideales aptas para probar enumerabilidades con rigor.

Como ejercicio se puede demostrar que los autómatas finitos son enumerables y con un poco más de trabajo escribir un programa C que efectivamente lo haga (mientras los recursos de la máquina lo permitan).

1.4. Pruebas de decibilidad y de enumerabilidad

Las pruebas positivas de estas propiedades son constructivas.

La afirmación de que este o aquel conjunto es enumerable debe venir acompañado de un programa que lo enumere.

Del mismo modo la afirmación de una decibilidad debe venir acompañada de un algoritmo capaz de contestar si la propiedad existe o no.

Más laboriosas son las pruebas de no decibilidad o no enumerabilidad, ya que el hecho de no conocerse programas positivos que lo hagan, podría tener por explicación no la ausencia de la propiedad sino la falta de ingenio de quienes lo intentaron hasta ahora.

La prueba consiste en exponer una técnica que ponga en contradicción a cualquiera que venga con una supuesta prueba positiva. Es lo que en matemáticas se suele llamar una prueba por reducción al absurdo. La más antigua documentación de una de estas demostraciones la provee Euclídes para demostrar la infinitud de los números primos. Un poco más moderno Aristóteles lo utiliza para demostrar la validez de dos de los silogismos.

1.5. El uso de decibilidad y enumerabilidad en lógica

Ambas propiedades, para existir, requieren que los conjuntos donde se las quieren evaluar sean representables. Pero los objetos lógicos son distintas construcciones sintácticas: términos, fórmulas, etc. y estos ya son textos, con lo cual se auto-representan.

Se utilizará con frecuencia la posibilidad de construir subconjuntos de conjuntos enumerables. Esto está garantizado si la función característica del subconjunto es decidible. Interceptando cada elemento producido por la máquina enumeradora, se lo envíaconjunto global paso cada elemento producido por el algoritmo que determina si posee o no la propiedad, la decibilidad asegura que en un tiempo finito estará la respuesta y por lo tanto la eliminación o incorporación del elemento.

Subconjuntos de conjuntos enumerables con función característica decidible adquieren auto-

máticamente la propiedad de ser enumerables.

Y esta acción de dividir se puede repetir teniendo en cuenta otras propiedades decidibles.

Una vez elegidos los símbolos de la lógica de predicados y los propios de una teoría, es factible enumerar todas las hileras que se pueden formar con las mismas. Hay muchas formas de hacerlo. Producirlas en el buen orden expuesto al reflexionar sobre los lenguajes puede ser uno de ellos y es un ejercicio sencillo programarlo en una máquina de Turing que tenga esos símbolos y el espacio de modo tal de disponer sobre una cinta una hilera tras otra separadas por un espacio.

Es decidible si una hilera representa una fórmula bien formada. Con un uso generoso de paréntesis se puede construir una gramática para la escritura de las fórmulas que sea LR,la que guiará en la construcción de un analizado sintáctico de la hilera.

Luego, el conjunto de fórmulas bien formadas es enumerable.

Las fórmulas bien formadas pueden tener variables libres. Un algoritmo no muy complejo puede determinar cuales son las variables libres y por lo tanto su cantidad. En particular interesará para la prueba de indecibilidad separar las que tienen una variable libre. Por lo dicho el conjunto de las fórmulas bien formadas con una variable libre es enumerable.

También las demostraciones son enumerables porque es decidible si una secuencia de fórmulas bien formadas es una demostración. Si la yuxtaposición de fórmulas pudiera tener alguna ambigüedad sintáctica habrá que agregar un símbolo separador al alfabeto.

Disponiendo de una representación hay que proponer el algoritmo que decide. Una por una hay que verificar si las fórmulas constituyentes son axiomas o surgen de un esquema deductivo aplicado a algunas de las anteriores.

A modo de ejemplo se supone que se está analizando la k-ésima fórmula y que ya quedó verificado que está bien formada. Si le tocara ser verificado si surge de la aplicación de un modus ponens, suponiendo que éste sea uno de los esquemas deductivos.

$$(((\mathcal{A} \rightarrow \mathcal{B}) \wedge \mathcal{A}) \rightarrow \mathcal{B})$$

1. el algoritmo deberá recorrer las $k-1$ fórmulas precedentes
2. en cada una determinar si es del tipo:

   $$(\mathcal{A} \rightarrow \mathcal{B})$$

   Esto se puede lograr contando paréntesis interiores hasta localizar el operador.
3. Si no fuera '$\rightarrow$' pasar a la siguiente [1].
4. Si lo es, comparar el operando derecho con la k-ésima fórmula (que está tomando el papel de $\mathcal{B}$).
5. Si no son iguales pasar a la siguiente [1].
6. Si son iguales separar el operando izquierdo que hace el papel de $\mathcal{A}$.
7. Iniciar un ciclo interior comparando las $k-1$ fórmulas que preceden a la a la k-ésima con este operando izquierdo.
8. Ante una desigualdad seguir iterando [7]
9. En caso positivo la k-ésima fórmula esta justificada. Incrementar k y empezar nuevamente
10. si alguno de los dos ciclos se agota sin éxito efectuar un trabajo similar con otros esquemas deductivos o con generadores de axiomas.
11. si todos fracasan la secuencia no es una demostración
12. si avanzando k logramos satisfacer todas las fórmulas la secuencia es una demostración.