Notas de Clase para IL

1. Introducción y Motivación

Robert Nieuwenhuis, Pilar Nivela, Albert Oliveras,
Enric Rodríguez

3 de septiembre de 2009

Esta primera entrega de los apuntes pretende servir de motivación del estudiante para el estudio de la lógica, dando alguna idea de por qué es tan importante en la informática.

1. La lógica en la informática

Durante muchos años se ha estado confundiendo la esencia intelectual de la Informática con las herramientas que se han utilizado. Por ejemplo, se dice: "no puede ser que el alumno de Informática no sepa Java, ni el lenguaje máquina del Pentium!". Pero hace 25 años no teníamos ni Windows, ni Pentiums, ni Java, y dentro de 25 años tampoco los tendremos (una muy acertada observación de Jordi Cortadella; véanse los libros del 25 aniversario de la FIB). Por lo tanto, debemos preguntarnos: ¿qué *fundamentos* deberían aprender los estudiantes para que en un futuro dispongan de madurez y agilidad al tener que asimilar los nuevos conceptos, lenguajes, técnicas y herramientas que surjan?

Gran parte de la respuesta está en el estudio de la Lógica. Ya lo dijo McCarthy[1] en los años 60, cuando aún era difícil apreciar su justeza, al considerar que "la lógica va a tener para la informática una importancia comparable a la que tuvo el análisis matemático para la física". Antes incluso que eso, el propio Alan Turing[2] ya había afirmado:

> "I expect that digital computing machines will eventually stimulate a considerable interest in symbolic logic (...). The language in which one communicates with these machines (...) forms a sort of symbolic logic."

Asimismo, informáticos actuales de la talla de Moshe Vardi han llamado a la lógica "el cálculo de la informática", debido a que su papel en la informática tanto teórica como práctica es similar al de las matemáticas en las ciencias físicas y, en particular, al del cálculo en ingeniería.

Al igual que los arquitectos e ingenieros, que analizan sus construcciones matemáticamente, los informáticos necesitan analizar las propiedades lógicas de sus sistemas mientras los diseñan, desarrollan, verifican y mantienen, especialmente cuando se trata de sistemas críticos económicamente (un fallo tendría un alto coste económico), o críticos en seguridad o privacidad. Pero también en sistemas cuya *eficiencia* resulta crítica, los análisis lógicos pueden ser reveladores y, en *todos* los tipos de sistemas, los métodos y herramientas basados en la lógica pueden mejorar la calidad y reducir costes.

Esta visión está ampliamente documentada en el artículo de Vardi y otros: "On the Unusual Effectiveness of Logic in Computer Science" (disponible en `www.lsi.upc.edu/~roberto`; ver allí también "The calculus of Computer Science"). En él se detalla el papel crucial de la lógica en la informática en general y, en particular, en áreas como la complejidad computacional, bases de datos, lenguajes de programación, inteligencia artificial, o la verificación de sistemas.

[1] Importante investigador y pionero de la Inteligencia Artificial, Premio Turing en 1971.

[2] 1912–1954, uno de los padres de la Informática. Formalizó los conceptos de algoritmo y computación mediante la Máquina de Turing. La hoy ampliamente aceptada Tesis de Church-Turing postula que ningún modelo computacional existente supera (en un sentido definido de manera precisa) las capacidades computacionales de la Máquina de Turing.

1.1. Ejemplos de usos de la lógica en la informática

Los métodos y herramientas basados en la lógica se utilizan cada vez más en todo tipo de sistemas de hardware y software. A menudo ha ocurrido que una técnica pensada para un uso muy concreto, *ad-hoc*, ha resultado ser un método lógico mucho más general, con muchas más aplicaciones. También ha pasado que técnicas desarrolladas independientemente, para aplicaciones muy distintas, han resultado ser "casualmente" el mismo principio lógico! Por ejemplo:

- *La lógica proposicional*, que veremos en IL, se usa para la especificación, síntesis y verificación de numerosas clases de circuitos y sistemas (reactivos, asíncronos, concurrentes, distribuidos). La deducción en lógica proposicional (resolver el problema de SAT, como veremos en IL) también sirve para resolver muy diversos problemas de la vida real: horarios, rutas de transporte, planificación de obras,...

- *La resolución para cláusulas de Horn* (una manera de hacer deducción en lógica de primer orden, que veremos en IL) es el principio lógico común resultante de, por un lado, la evolución de sistemas de bases de datos (que progresaron desde los jerárquicos y en red hacia los relacionales y deductivos), y, por otro lado, de los lenguajes de programación lógica, como Prolog, y sus extensiones a la programación lógica con restricciones y la programación lógico-funcional.

- *Las técnicas de reescritura*, la aplicación de *reglas* lógicas, son la base común de la programación funcional (LISP, ML, Miranda,...), de diversas aplicaciones en inteligencia artificial (sistemas expertos, bases de conocimiento), de ciertos (sub)sistemas de álgebra por ordenador (Maple, Mathematica), etc.

- La lógica también está jugando un papel central desde el primer momento en la llamada *web semántica*, que permite buscar con criterios semánticos; un ejemplo muy sencillo es poder buscar un herrero *de profesión*, sin tener que filtrar miles de respuestas sobre personas cuyo apellido es Herrero. En la web semántica se usan las *description logics* para dotar a las páginas de internet y de intranets de criterios de búsqueda semánticos, mecanismos deductivos, restricciones de consistencia o integridad, etc.

- Otro buen ejemplo es la necesidad absoluta de lógicas para la verificación de protocolos criptográficos, usados en aplicaciones que requieren privacidad, autentificación (firma electrónica), dinero electrónico anónimo, etc.

- Existen estrechas relaciones entre la lógica y la *complejidad computacional*, que estudia los recursos en cuanto a tiempo de cálculo o memoria necesarios para resolver clases de problemas. A veces es posible determinar la complejidad de un problema simplemente expresando el problema en una determinada lógica.

1.2. Lógica matemática vs. lógica informática

Para los matemáticos, el interés en la lógica surgió en parte del programa de Hilbert de dotar a las matemáticas de fundamentos sólidos[3]. Pero para la investigación actual en matemáticas la lógica ya no juega ese papel central, y hace énfasis en aspectos diferentes a los que interesan en la informática, donde, en cambio, la lógica adquiere un interés cada vez mayor. Hoy día, son de informática la mayoría de los trabajos de investigación en lógica que se publican.

No es lo mismo estudiar la lógica desde el punto de vista de un matemático que desde el de un informático. Por ejemplo, para un informático no sólo importa el poder expresivo de un lenguaje lógico, sino también la comodidad o la naturalidad al utilizarlo, porque su tarea consiste precisamente en crear formalizaciones (programas, especificaciones,...) y necesita hacerlo con la máxima facilidad; un informático sabe perfectamente que es preferible programar en un lenguaje de alto nivel que en un lenguaje máquina, aunque ambas tengan el mismo "poder expresivo".

En los aspectos de implementabilidad y eficiencia son también evidentes las diferencias: los informáticos desean ver sus lógicas funcionando, en forma de, por ejemplo, lenguajes de programación, bases de datos, herramientas de síntesis y verificación, o lenguajes de consulta de la *web semántica*. Los matemáticos tradicionales no se han interesado tanto por aspectos como los procedimientos de decisión viables en la práctica, o las clases de complejidad asociadas a las distintas lógicas, o lo que hoy día se denomina "Lógicas Computacionales", o la combinación modular de lógicas mediante *constraints*.

2. Objetivos para la asignatura de IL

El objetivo de esta asignatura es proporcionar al estudiante una base sólida de conocimientos teóricos y prácticos de lógica para informáticos, y, al mismo tiempo, contribuir a su formación general en razonamiento formal (sobre conjuntos, relaciones, funciones, estructuras de orden, recursión, conteo) y en técnicas de demostración (contrarrecíproco, contraejemplos, inducción, reducción al absurdo). En concreto, la asignatura ha de proporcionar los siguientes conocimientos específicos:

1. El concepto de qué es una lógica, solamente en términos de sintaxis: *¿qué es una fórmula F?*, y semántica: *¿qué es una interpretación I?*, y *¿cuándo una interpretación I satisface a una fórmula F?*.

2. La definición de dos lógicas: la proposicional y la de primer orden.

3. Las propiedades de tautología/validez, contradicción, consecuencia y equivalencia lógica (de forma independiente de la lógica concreta), cómo se utilizan para formalizar problemas prácticos en informática, y cómo se reducen al problema de satisfactibilidad.

[3] Programa parcialmente frustrado con los resultados negativos de Gödel, Church y Turing.

4. Los métodos de deducción para determinar estas propiedades de mayor relevancia para la informática: Davis-Putnam, y resolución; su corrección y completitud con respecto de la definición de la lógica.

5. El poder expresivo de las dos lógicas, y entender en qué sentido la lógica proposicional es una restricción de la otra; compromiso entre poder expresivo y buenas propiedades computacionales: primeras nociones intuitivas de complejidad y decidibilidad.

6. Algunas de las -cada vez más abundantes- aplicaciones a la informática de los métodos deductivos: fundamentos de la programación lógica (Prolog) bases de datos deductivas, circuitos, problemas de planificación, etc.

También ha de proporcionar las siguientes habilidades:

1. Entender, escribir y manipular ágilmente fórmulas en ambas lógicas, con especial énfasis en aplicaciones a la informática.

2. Saber demostrar formalmente propiedades sencillas sobre fórmulas, conjuntos, relaciones, funciones, etc, mediante técnicas de demostración como el contrarrecíproco, contraejemplos, inducción, o reducción al absurdo.

3. Ser capaz de aplicar a mano resolución y Davis-Putnam sobre ejemplos prácticos abordables y saber utilizar la resolución como mecanismo de cómputo (cálculo de respuestas).

4. Saber expresar algunos problemas prácticos NP-completos (sudokus, horarios, problemas de grafos, circuitos) como problemas de satisfacción de lógica proposicional y resolverlos con un SAT solver.

5. Saber expresar problemas sencillos utilizando el lenguaje Prolog y entender las extensiones “extra-lógicas” de Prolog (la aritmética predefinida, el operador de corte y la negación, la entrada/salida).