Revista Educación en Ingeniería
ISSN 1900-8260
Enero a Junio de 2013, Vol. 8, N°. 15, pp. 62-68 • © 2013 ACOFI • http://www.educacioneningenieria.org
Recibido: 19/09/2013 • Aprobado: 09/06/2013

LA LÓGICA EN LAS CIENCIAS COMPUTACIONALES

LOGIC IN COMPUTER SCIENCE

Edgar Serna M.
Instituto Tecnológico Metropolitano, Medellín (Colombia)

Resumen

En este trabajo se hace un análisis a la necesidad de incluir a la lógica en los procesos formativos en Ciencias Computacionales (CS por sus siglas en inglés). Se parte de un recorrido sobre la historia de la lógica en estas ciencias, posteriormente se describe la relación y la necesidad de incluirla en los procesos formativos relacionados, y al final se analiza qué, cuándo y qué tan profundo se debería trabajar en la formación en CS. Se trata de una revisión al estado del tema y a la importancia de incluirlo en los pregrados y en los posgrados en áreas de ciencias computacionales y tecnologías de información.

Palabras clave: lógica, ciencias computacionales, informática, formación

Abstract

In this paper it is analyzed the need to include the logic in training processes in Computer Science (CS). It is part of a tour of the history of logic in these sciences, later it's described the relationship and the need to include it in the training processes related, and finally analyzed that, when and how deep it should work training CS. This is a revision to issue state and the importance of including this subject in undergraduate and graduate programs in computer sciences and Information Technologies areas.

Keywords: *logic, computer science, computing, training*

Introducción

En los últimos años se han desarrollado poderosas herramientas para verificar especificaciones formales de sistemas *hardware* y *software*; la industria de las tecnologías de información (TI) se ha dado cuenta del impacto y la importancia que tienen en sus propios procesos de diseño e implementación, y empresas de tecnología las investigan e incorporan en sus departamentos de planeación y producción. Para trabajar en estos procesos se requiere una formación formal básica que les permita a estudiantes y profesionales obtener suficientes competencias para utilizar, *razonar* y potencializar los sistemas. El cambio de las TI para

favorecer al acceso de datos con base en Internet y el procesamiento, también generó un incremento en la demanda por profesionales calificados que puedan razonar acerca del *software* sofisticado basado en agentes autónomos y capaces de interactuar con otros agentes para recopilar la información necesaria en las grandes redes.

Para aportar al hecho de que el trabajo en Ciencias Computacionales (CS por sus siglas en inglés) requiere la aplicación y el manejo adecuado de la lógica como componente formal, en este artículo se realiza un análisis al campo temático para responder a esas necesidades formativas, y se analizan las bases de la formación en lógica que requieren los estudiantes y profesionales para desempeñarse en este campo laboral. Además, se hace una introducción a los marcos lógicos utilizados para modelar y razonar acerca de los sistemas informáticos. El objetivo es proporcionar un contenido que se pueda utilizar para diseñar cursos de formación en lógica computacional, como una contribución que permita adaptarlos rápidamente a los actuales entornos profesionales en medio del acelerado y cambiante entorno de las TI.

Recorrido histórico

La Magna Charta Universitatum (Bolonia, 1988), emitida con ocasión de los 900 años de la fundación de la Universidad de Bolonia, resumía la misión de las universidades en la sociedad moderna. Ese mismo año y por iniciativa del ministro francés de educación, la *Sorbonne Declaration* (Sorbonne, 1988) estableció el reconocimiento mutuo de las respectivas titulaciones como un objetivo común de los cuatro países signatarios: Francia, Alemania, Italia y Reino Unido. Otros países aceptaron esas ideas y expresaron su disposición de unirse al proyecto.

Para enfatizar su compromiso, los ministros de educación de 29 países se reunieron en Bolonia en 1999 y se comprometieron, en una declaración conjunta (CRE, 2000), a establecer un espacio europeo en educación superior para el año 2010. El objetivo del *Bologna Process* era lograr que los sistemas europeos de formación superior confluyeran hacia un sistema más transparente, mediante el cual los diferentes sistemas nacionales utilizaran un marco común basado en tres ciclos formativos: Licenciatura, Maestría y Doctorado. Desde entonces, los países signatarios han organizado varias reuniones —Praga 2002, Berlín 2005, Londres 2007, Leuven 2009, Budapest 2010— para revisar los objetivos del proceso a corto plazo y clarificar algunas cuestiones en su implementación.

Posteriormente, aunque de manera tangencial, otros países en el mundo entraron al proyecto con el objetivo de adoptar los principios que en él se discuten e implementarlos en sus propios sistemas formativos. En América Latina se está trabajando seriamente al respecto, y algunos países han comenzado a modificar sus sistemas en pregrado y posgrado; la idea es no quedarse rezagados con respecto a los progresos que se hacen en Europa y EE.UU., porque las TIC requieren actualización constante y profesionales capacitados para su explotación. La revisión y re-construcción de los planes de estudio en Ciencias Computacionales se realiza bajo la supervisión de investigadores interesados en el tema, y se ha llegado a concluir que es necesario contar con una profesionalización en desarrollo de *software*, es decir, una ingeniería de *software* (Serna, 2011, 2011a).

Aunque esta ingeniería se considera actualmente como sucesora directa de la ingeniería de sistemas y los planes de estudio desarrollados están fuertemente basados en las experiencias locales relacionadas con el mantenimiento de más de 30 años de la misma, los cambios en el sistema formativo siempre brindan la oportunidad de hacer una revisión importante de los objetivos y del contenido en ambas áreas de formación. La revisión que se presenta en este trabajo se llevó a cabo teniendo en cuenta el papel y el contenido de la lógica en la formación en Ciencias Computacionales, especialmente en el área de la ingeniería de *software*.

La lógica en las Ciencias Computacionales del siglo XX

A comienzos de siglo pasado, Hilbert (1920) consideraba a la lógica como una teoría axiomatizada. Según este enfoque, es posible demostrar teoremas por medio de los métodos matemáticos tradicionales, sin embargo, no existían algoritmos que soportaran la construcción de tales deducciones. El primer avance significativo en este sentido se debe a Gentzen (1934), con el desarrollo de la técnica natural de deducción y el cálculo sucesivo, con los que creó un

kit de herramientas sintácticas especiales para probar teoremas automáticamente. Por otro lado, Herbrand (1968) aportó la probabilidad de insatisfacción sobre los universos de Herbrand mediante un modelo teórico e hizo posible la re-escritura del problema de decisión de primer orden, como una fórmula proposicional que permitía expandirlo sobre dichos universos, con lo que el problema de demostrar un teorema se redujo a la revisión de las fórmulas a través de un kit de herramientas de lógica proposicional.

En los años 50, cuando los computadores fueron accesibles, Davis y Putnam (1960) utilizaron los resultados de Herbrand y elaboraron el primer algoritmo de computador para demostrar el teorema. Newell y Simon (1956) desarrollaron el *General Problem Solver* y Newell, Shaw y Simon (1959) diseñaron el sistema *Logic Theorist*, con los que impactaron la Inteligencia Artificial contemporánea. El primero utiliza el algoritmo del *British Museum*, un método de búsqueda horizontal a ciegas de bajo rendimiento, con base en los axiomas y reglas de inferencia dadas por Russell y Whitehead (1997).

La investigación del problema de insatisfacción de las fórmulas canónicas, *Conjunctive Normal Form (CNF)* y la generalización de la regla de resolución proposicional para las fórmulas de primer orden, dio origen a la resolución lógica proposicional o resolución básica. Robinson (1965) definió la noción de la resolución de primer orden con base en que, si la resolución existe, es posible utilizar dos cláusulas que contengan las instancias básicas cuando aparece una pareja literal básica complementaria. Además, reconoció que el poder unificar el par literal original es la condición para que una resolución básica exista, principio con el que creó el cálculo de resolución de primer orden. Posteriormente, otros trataron de desarrollar estrategias de resolución para simplificar la implementación, pero hasta el momento las principales estrategias resultantes son la resolución semántica y la lineal —cálculo completo—, y la resolución de entrada lineal y la resolución de unidad, que son posibles de implementar pero que actualmente son cálculo incompleto.

En los años 60 surgió una demanda por la aplicación de la lógica en conexión con el análisis y la síntesis, lo que significó que las propiedades de un programa se pudieran describir mediante fórmulas lógicas —axiomas—, y que fuera posible tratar de responder las preguntas acerca del funcionamiento correcto del mismo. La base formal de este enfoque fue principalmente la lógica de Hoare (1969), cuyos aportes contienen los resultados más importantes en esta área (Manna & Waldinger, 1971), (Manna & Pnueli, 1974) y (Pnueli, 1977). Los llamados sistemas pregunta/respuesta fueron herramientas útiles, en los que la respuesta era una supuesta consecuencia/inferencia de las fórmulas que describen las propiedades del programa. Estos sistemas fueron dotados de una técnica especial que adicionalmente podía generar cierto tipo de respuestas acerca de la consistencia del teorema, siempre que se hubiera demostrado a sí mismo. En este proceso se utilizó la lógica de segundo orden, e incluso la temporal, en la formalización (Manna & Pnueli, 1974) y (Kröger, 1987). Los trabajos de Cliff (1980), Hoare & Shepherdson (1985), Loeckx & Sieber (1987) y Goos et al. (2003), ilustran que ésta es un área importante en el desarrollo y aplicación de la lógica.

A principios de los 80, y utilizando la estrategia de entrada lineal incompleta, Colmerauer (1970) y Kowalski (1979) desarrollaron el teorema de pruebas del Prolog para las cláusulas de Horn de primer orden, que se utilizaban para definir las declaraciones lógicas de un programa. La investigación acerca de las capacidades de este sistema concluyó que, sobre la base del principio del modelo dado por los átomos de la primera capa, era posible desarrollar la noción del modelo mínimo de Herbrand (1968) y de cierto tratamiento de negación. Debido a que el resultado de un trazo de asignación, ordenado y definido en un reticulado completo, siempre tiene un punto fijo, entonces una de las interpretaciones de Herbrand es un subconjunto del conjunto de átomos de la primera capa sobre su universo, es decir, un sub-conjunto de la base de Herbrand, donde el conjunto de todas sus interpretaciones es un super-conjunto de toda la base. El resultado del conjunto de todas las interpretaciones es un reticulado completo, con dos operadores y una relación del sub-conjunto. Una consecuencia directa de la función asignada a la lógica de un programa sobre esta red es la preservación de la ordenación, por lo que al menos tiene un punto fijo; además, se ha demostrado como el modelo mínimo de Herbrand de la lógica de un programa, con lo que se definió la extensión del punto fijo de la lógica de primer orden (Abiteboul

et al., 1995), el cual juega un papel significativo en el enfoque DATALOG de las bases de datos relacionales y las bases del conocimiento.

La investigación de los sistemas de deducción lógica tuvo amplio impacto en el desarrollo de la teoría de modelos, principalmente en la evaluación de las cuestiones de la axiomatización (Ebbinghaus et al., 1994). La aparición de los lenguajes de programación funcionales, y el desarrollo de las tecnologías de programación paralela y concurrente, condujeron a la introducción de nuevas herramientas en la teoría de la programación. Una de ellas es l-cálculo, presentada por Church (1932) y utilizada para el tratamiento unificado de lógicas de orden diferente —nulo, primero, superior— (Andrews, 2002). La evolución de los lenguajes de programación involucró la evolución de varias herramientas, como π-cálculo y μ-cálculo, entre otras.

Desde los años 60, los *kits* de herramientas de Inteligencia Artificial, la teoría de la programación y la teoría basada en el conocimiento, incluyen como herramientas lógicas no clásicas a las diversas versiones de la lógica temporal, la lógica modal, las lógicas de valores diversos, las lógicas relevantes, la lógica no-monótona y a la lógica difusa.

La lógica en la formación en ciencias computacionales

Inicialmente se formaba en lógica pero sólo vista como una técnica descriptiva. Para los 60 sus aplicaciones suponían el conocimiento de amplias áreas de la misma, lo que implicaba que en los planes de estudio de los cursos avanzados se debían incluir los temas básicos de la lógica matemática. Desde entonces, la lógica se convirtió en una herramienta para otros campos de la informática, como la Inteligencia Artificial, la teoría de bases de datos relacionales y el análisis y síntesis de programas. Desde finales de esta década las bases de la lógica comenzaron a hacer parte de la formación en los primeros años de CS, y las lecturas eran obligatorias debido a que estos campos del conocimiento comenzaron a jugar un papel importante en las aplicaciones; posteriormente se desplazaron a años superiores.

Desde finales de los 70 se comenzaron a publicar trabajos, para estudiantes de informática y para especialistas avanzados, acerca de la lógica (Bergmann & Noll, 1977), (Richter, 1978), (Gallier, 1987), (Schönig, 1987), (Börger, 1989), (Fitting, 1996), (Nerode & Shore, 1997), (Pásztor & Várterész, 2003) y (Huth & Ryan, 2004), al mismo tiempo que se editaban algunos manuales de resúmenes (Barr & Feigenbaum, 1981), (Bledsoe & Loveland, 1984), (Boyer & Moore, 1988), (Gabbay et al., 1993), (Abramsky et al., 1995), (Agostino et al., 1999) y (Krantz, 2002).

Qué, cuándo, cuánta lógica

En la fase actual del desarrollo de las Ciencias Computacionales se conoce cuál es el conocimiento y qué parte de la lógica es el que debe adquirir un graduado en informática, y se llegó a la conclusión que tanto en el currículo como en la formación básica se le debe capacitar en lógica proposicional y de primer orden, a través de un enfoque orientado al lenguaje, el alfabeto, la sintaxis, la semántica, las reglas de re-escritura, la noción de consecuencia semántica, el teorema de demostración de problemas, el problema de la decisión, la solución semántica de la prueba de teoremas, el tratamiento sintáctico de la lógica y el principio de los sistemas de deducción. Además, que se debe incluir el cálculo de resoluciones y el método arbitrario, conectados por el sistema de Hilbert (Goguen & Goubault, 2000), como el más aceptado de los dos sistemas de deducción conocidos.

También se deben abarcar las nociones de correctitud y completitud, de tal manera que se sienten las bases necesarias para la introducción de la lógica temporal de Hoare, la lógica de unidad, la lógica de punto fijo y la lógica descriptiva (Küsters, 2001), utilizadas en la formación en diversas áreas de las Ciencias Computacionales. En cierta medida se puede utilizar en la formación en bases de datos, la teoría de la programación y la Inteligencia Artificial, y son un punto de partida para introducir las lógicas no clásicas (Qualls & Sherrell, 2010). Después de esta fundamentación se podrán introducir los lenguajes de programación funcionales (Hoare, 2004). Los cursos de pregrado enfocados en la teoría de base de datos o la Inteligencia Artificial no son los únicos que requieren el conocimiento de la lógica fundamental, también algunos cursos de posgrado se deben fundamentar en ella, y luego de analizar los currículos relevantes de algunas universidades en las que se imparte la lógica en la formación en posgrados, se concluye

que los campos más importantes de formación en esta área son:

- Los fundamentos teóricos y los métodos de prueba de teoremas
- Los problemas de formalización y axiomatización: problemas de las teorías axiomatizadas
- La programación lógica y sus fundamentos teóricos: lógica de punto fijo y el modelo Herbrand de un programa
- Cálculo en el marco de la programación funcional
- Lógicas no clásicas: lógica temporal, lógica polivalente, lógica difusa, lógica descriptiva, entre otras.

Estos temas son importantes para incluirlos en el marco de un programa de posgrado, al tiempo que se justifica la existencia de temas como lógica no clásica a nivel de pregrado, al menos como curso electivo.

Conclusiones

En este trabajo se presenta un breve recorrido histórico acerca de la lógica y las posibilidades y ventajas de la formación en Ciencias Computacionales con base en ella. Estas ideas parten de la conceptualización que introdujo el Tratado de Bolonia para la formación en informática, pero que redunda en todo el planeta. El acercamiento propuesto aquí hace hincapié en la formación acerca de los principales métodos de inferencia, incluyendo la resolución no causal, y ofrece una introducción a la teoría y aplicación de las lógicas de muchos valores, y de otras que se difunden y aplican en la industria de TI.

La formación en lógica se debe realizar no sólo en el marco de las matemáticas, sino también en el de la formación en Ciencias Computacionales, con un aspecto inter y multi-disciplinar para lograr la eficiencia, lo que significa que ambas área del conocimiento tienen que ser colaborativas en el currículo (Wing, 2008). Lo básico debe ser inherente a la formación lógico-matemática, como la abstracción, los lenguajes, las semánticas, los conectores, las interpretaciones, las tautologías, el cálculo proposicional y la lógica computacional. Pero en los posgrados se deben incluir aspectos avanzados, como sintaxis vs semánticas, representación de conocimiento, tareas problemáticas de la *vida real*, deducción y métodos formales, resoluciones y lógicas de muchos valores (Denning, 2009). Además, se deben utilizar las herramientas informáticas actuales para que la formación sea más interesante para los estudiantes, como la lógica de programación, las máquinas de inferencia visual con ejemplos preparados, y las herramientas CASE, entre otras. También es necesario mantener una actualización constante mediante seminarios y asignaturas de aplicación práctica en la industria.

Referencias

Abiteboul, S., Hull, R. & Vianu, V. (1995). *Foundations of Databases*. USA: Addison-Wesley.

Abramsky, S., Gabbay, D. & Maibaum, T. (1995). *Handbook of Logic in Computer Science*, vols. 1-4. Oxford: Oxford University Press.

Agostino, M., Gabbay, D., Hähnle, R. & Possega, J. (1999). *Handbook of Tableau Methods*. London: Kluwer.

Andrews, P. (2002). *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. New York: Springer.

Barr, A. & Feigenbaum, E. (1981). *Handbook of Artificial Intelligence*, vols. 3. Stanford: Heuristic Press.

Bergmann, E. & Noll, H. (1977). *Mathematische Logik mit Informatik-Anwendungen*. Berlin-New York: Springer-Verlag.

Bledsoe, W. & Loveland, D. (1984). *Automatic Theorem Proving after 25 years*. Providence: American Mathematical Society.

Bolonia (1988). *Magna Charta Universitatum*. Online: http://www.magna-charta.org /cms/cmspage.aspx?pageUid={d4bd2cba-e26b-499e-80d5-b7a2973d5d97} (Aug. 2012).

Börger, E. (1989). *Computability, Complexity, Logic*. Amsterdam: North-Holland.

Boyer, R. & Moore, J. (1988). *A Computational Logic Handbook*. Boston: Academic Press.

Church, A. (1932). A set of Postulates for the Foundation of Logic. *Annals of Mathematics, second series*, 33, 346-366.

Cliff, B. J. (1980). *Software development. A rigorous Approach*. New York: Prentice Hall.

Colmerauer, A. (1970). *Les systèmes-q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur.* Internal publication 43, Département d'informatique de l'Université de Montréal.

CRE (2000). *The Bologna Declaration on the European space for higher education: An explanation*. Online: http://ec.europa.eu/education/policies/educ/bologna/bologna.pdf, (Jul. 2012)

Davis, M. & Putnam, H. (1960). A Computing Procedure for Quantification Theory. *Journal of the ACM*, 7(3), 201-215.

Denning, P. (2009). The Profession of It: Beyond Computational thinking. *Communications of the ACM*, 52(6), 28.

Ebbinghaus, H., Thomas, W. & Flumm, J. (1994). *Mathematical Logic*. London: Springer.

Fitting, M. (1996). *First-Order Logic and Automated Theorem Proving*. New York: Spinger-Verlag.

Gabbay, D., Hogger, C. & Robinson, J. (1993). *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford: Oxford University Press.

Gallier, J. (1987). *Logic for Computer Science*. USA: John Wiley.

Gentzen, G. (1934). Untersuchung Äuber das logische Schliessen. *Mathematische Zeitschrift*, 39, 176-210.

Goguen, H. & Goubault, J. (2000). Sequent combinators: a Hilbert system for the lambda calculus. *Mathematical Structures in Computer Science*, 10(1), 1-79.

Goos, G., Hartmanis, J. & van Leuwen, J. (2003). *Verification: Theory and Practice*. Berlin: Springer.

Herbrand, J. (1968). *Logical writings*. Holland: Reidel Publishing.

Hilbert, D. (1920). *Die Grundlagen Der Elementaren Zahlentheorie*. Mathematische Annalen. Traducido al inglés por Ewald W. como *The Grounding of Elementary Number Theory*, en From Brouwer to Hilbert: The debate on the foundations of mathematics in the 1920's de Mancosu (ed., 1998), 266-273. New York: Oxford University Press.

Hoare, C. (2004). *Communicating Sequential Processes*. New York: McGraw Hill.

Hoare, C. & Shepherdson, J. (1985). *Mathematical Logic and Programming Languages*. USA: Prentice-Hall.

Hoare, C. (1969). An Axiomatic Basis for Computer Programming. *Communications of the ACM*, 12(10), 576-580.

Huth, M. & Ryan, M. (2004). *Logic in Computer Science - Modelling and Reasoning about Systems*. Cambridge: Cambridge University Press.

Kowalski, R. (1979). Algorithm = Logic + Control. *Communications of the ACM*, 22(7), 424-436.

Krantz, S. (2002). *Handbook of Logic and Proof Techniques for Computer Science*. Boston: Birkhäuser.

Kröger, F. (1987). *Temporal Logic of Programs*. London: Springer.

Küsters, R. (2001). *Non-Standard Inferences in Description Logics*. Berlin: Springer-Verlag.

Loeckx, J. & Sieber, K. (1987). *The Foundations of Program Verification*. London: Wiley.

Manna, Z. & Pnueli, A. (1974). Axiomatic approach to total correctness of programs. *Acta Informática*, 3(3), 243-264.

Manna, Z. & Waldinger, R. (1971). Toward Automatic Program Synthesis. *Communications of the ACM*, 14(3), 151-165.

Nerode, A. & Shore, R. (1997). *Logic for Applications*. New York: Springer-Verlag.

Newell, A. & Simon, H. (1956). The logic theory machine: A complex information processing system. *IRE Transactions on Information Theory*, 2(3), 61-79.

Newell, A., Shaw, J. & Simon, H. (1959). Report on a general problem-solving program. *Proceedings of the International Conference on Information Processing*. Paris, France, 256-264.

Pásztor, K. & Várterész, M. (2003). *Mathematical Logic: An Application Oriented Approach*. Budapest: Panem Kiadó.

Pnueli, A. (1977). The temporal logic of programs. *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*. Rhode Island, USA, 46-57.

Qualls, J. & Sherrell, L. (2010). Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges*, 25(5), 66-71.

Richter, M. (1978). *Logikkalküle*. Stuttgart: Teubner.

Robinson, J. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1), 23-41.

Russell, B. & Whitehead, A. (1997). *Principia Mathematica to *56*. USA: Cambridge University Press.

Schönig, U. (1987). *Logik für Informatiker, Spektrum Akad*. Heildelberg-Berlin-Oxford: Verlag.

Serna, M.E. (2011). Systems engineering for the XXI century: A proposal from the academy. *Proceedings Ninth LACCEI Latin American and Caribbean Conference (LACCEI'2011)*, August 3-5, Medellín, Colombia.

Serna, M.E. (2011a). "Software Engineering" is Engineering. *Revista RACCIS*, 1(1), 34-43.

Sorbonne (1988). *Sorbonne Declaration*. Online: http://www.bologna-bergen2005.no/

Docs/00-Main_doc/980525SORBONNE_DECLARATION.PDF, (Aug. 2012).

Wing, J. (2008). Five deep questions in computing. *Communications of the ACM*, 51(1), 58-60.

Sobre el autor

Edgar Serna M.

Científico computacional teórico con más de 10 años de experiencia como líder de proyectos en Sistemas de Información y Arquitecto de Software, y profesor universitario e investigador con más de 20 años de trayectoria. Sus áreas de investigación son la Lógica, la Ingeniería de Software, las Ciencias Computacionales, los Métodos Formales y las Matemáticas en la Computación, alrededor de las cuales ha publicado libros y artículos, y participado con ponencias y conferencias en eventos nacionales e internacionales. Actualmente es investigador de la Facultad de Ingenierías del Instituto Tecnológico Metropolitano (ITM), Medellín, Colombia.
eserna@eserna.com