

Curso de Postgrado: Filosofía y Ciencia Computacional

Docentes a cargo: Javier Blanco, Pío García.

Fundamentación

Siendo la ciencia computacional una disciplina relativamente nueva, recién desde hace pocos años se ha comenzado a realizar una reflexión filosófica acerca de sus alcances y límites. Entre los temas que han formado parte de la reflexión filosófica acerca de esta disciplina se pueden mencionar la verificación formal, la individuación de programas, el pancomputacionalismo, el funcionalismo como modelo para entender la computación, los programas como mecanismos, la especificidad de la ciencia computacional como disciplina autónoma etc.

Muchas de las discusiones se han planteado en términos de problemas que provienen de otros campos. En este sentido, la filosofía de la mente ha sido una fuente de sugerencias y problemas para la reflexión acerca de la noción de computadora, de programa y de las relaciones entre máquinas físicas y abstractas. Algunos problemas típicos de la filosofía de la matemática también han sido vistos como relevantes para una filosofía de la ciencia computacional. Esta última cuestión está directamente ligada con el problema de si la ciencia computacional es sólo una rama de la matemática o si, por el contrario, hay suficientes aspectos para considerarla una disciplina aparte. En este curso presentaremos algunas de las discusiones filosóficas recientes en relación con la ciencia computacional.

Objetivo:

En este curso se evaluarán las principales corrientes filosóficas que han tomado a la ciencia computacional como problema. Asimismo se analizarán cuestiones básicas como la forma en la cual se entiende y usa el proceso de abstracción, la posibilidad de la verificación formal o la noción de mecanismo para comprender la computación.

Temas y bibliografía específica.

1. Paradigmas de la ciencia de la computación. ¿Es la informática una ciencia formal, una ingeniería o una ciencia empírica? Posiciones epistemológicas, ontológicas y metodológicas. Contexto de descubrimiento y contexto de justificación. En principio y en la práctica. Aspectos metodológicos de la ciencia computacional.

Raymond Turner, Amnon H. Eden. "Philosophy of computer science." In *The Stanford Encyclopedia of Philosophy* (Winter 2008 edition), Edward N. Zalta (ed.).

Amnon H Eden. "Three Paradigms of Computer Science." *Minds and Machines*, Special issue on the Philosophy of Computer Science, Vol. 17, No. 2 (Jul. 2007), pp. 135–167. London: Springer.

Peter Wegner. "Research paradigms in computer science." Proc. 2nd Int'l Conf. Software Engineering—ICSE 1976, San Francisco, CA, pp. 322–330.

Colburn, T., 2004, "Methodology of Computer Science", The Blackwell Guide to the Philosophy of Computing and Information, Luciano Floridi (ed.), Malden: Blackwell, pp. 318–326. Survey sobre epistemología.

2. Epistemología de las ciencias de la computación: El debate sobre verificación formal. Argumentos en contra de la verificación formal. Algoritmos y programas. Corrección de programas y de sistemas. Causalidad.

DeMillo, R.A., Lipton, R.J. and Perlis, A.J., 1979, "Social Processes and Proofs of Theorems and Programs", Communications of the ACM 22(5): 271–280.

Fetzer, J.H., 1988, "Program Verification: The Very Idea", Communications of the ACM 31(9): 1048–1063.

Smith, B.C., 1996, "Limits of Correctness in Computers", Computerization and Controversy, Kling, R. (ed.), Morgan Kaufman, pp. 810–825.-Barwise.

Blanco, J. y García P. A categorial mistake in the formal verification debate (extended abstract). E-CAP. 2008.

Barwise, J. Mathematical Proofs of Computer System Correctness. Notices of the American Mathematical Society.

3. La ciencia computacional desde los problemas de la filosofía de la matemática: Intuicionismo, realismo, formalismo. La relación de la matemática y de los programas con el mundo. Programación y matemática aplicada.

Klimovsky G. y Boido G. Desventuras del conocimiento matemático

Hempel C. On the Nature of Mathematical Truth. American Mathematical Monthly 52, 1945.

Hempel C. Geometry and Empirical Science. American Mathematical Monthly 52, 1945.

4. La naturaleza de los programas. Programas como manipuladores abstractos de símbolos. Programas y demostraciones constructivas. Justificación racional del comportamiento de los programas. Ontología de los programas.

Dijkstra, E.W. On the Cruelty of really teaching computer science.

Bornat, R. Is computer science science?

Janlert, L. Dark programming and the case for the rationality of programs. *Journal of Applied Logic*.

Amnon H. Eden, Raymond Turner. "Problems in the ontology of computer programs." *Applied Ontology* Vol. 2, No. 1 (2007), pp. 13–36. Amsterdam: IOS Press.

5. Lenguajes de programación, paradigmas de la programación y la computación. Abstracción en ciencia de la computación. Implementación e interpretación semántica. Semánticas de los lenguajes de programación.

Rapaport, W.J., 2005b, "Implementation is Semantic Interpretation: Further Thoughts." *Journal of Experimental and Theoretical Artificial Intelligence* 17(4): 385–417.

Turner, Raymond, 2007, "Understanding Programming Languages". *Minds and Machines* 17(2): 129-133

Piccinini, G. "Computation without Representation," *Philosophical Studies*, 137.2 (2008). This is a paper on how to individuate computational states, inputs, and outputs without appealing to semantic properties.

Piccinini, G. "Computing Mechanisms," *Philosophy of Science*, 74.4 (2007), pp. 501-526. An articulation and defense of the mechanistic account of computing mechanisms.

6. Individuación de los sistemas computacionales. Condiciones para que un mecanismo sea considerado una computadora. Tesis de Turing-Church y teoremas de Godel.

Chalmers, D. Does a Rock Implement Every Finite-State Automaton? (1996)

Sieg, W. (1994). *Mechanical Procedures and Mathematical Experience*. Mathematics and Mind. A. George. New York, Oxford University Press: 71-117.

Copeland, B. Jack, 2004, "Computation", *The Blackwell Guide to the Philosophy of Computing and Information*, Luciano Floridi (ed.), Malden: Blackwell, pp. 3–17.

Gandy, R. (1980) Church's Thesis and principles for mechanisms; in: *The Kleene Symposium* (edited by J. Barwise, H.J. Keisler and K. Kunen), North-Holland, 123-148.

Metodología de trabajo.

Los estudiantes presentarán cada semana un trabajo de a lo sumo una carilla defendiendo alguna de las posiciones contrapuestas en cada uno de los temas a considerar. Estos trabajos serán discutidos en cada encuentro, además de presentar los temas del siguiente trabajo. La evaluación del curso se

complementará con un coloquio final. Algunos de los cursos consultados fueron:

Philosophy of Computer Science” (CSE 410/510 & PHI 498). Bill Rapaport, Department of computer science and engineering, State University of New York at Buffalo, USA

William J. Rapaport. “Philosophy of Computer Science: An Introductory Course.” *Teaching Philosophy*, Vol. 28, No. 4, (Dec. 2005), pp. 319–341.

Philosophy of Computer Science” (175616). Matti Tedre, Department of Computer Science and Statistics, University of Joensuu, Finland