Inicio 02/05/2013 - Finalización 30/04/2017

Director: Bonelli, Eduardo

Co- Director: Martínez López, Pablo

Título: Programación funcional: fundamentos revisados

Integrantes: Lombardi, Carlos; Barenbaum, Pablo; Arévalo, Gabriela Beatriz; Passerini, Pablo Nicolás; San Martín, Hernán Javier; Ríos, Alejandro; Ciolek, Daniel; Viso, Andrés

Resumen: La programación funcional (PF) es uno de los paradigmas más difundidos de programación, entre los cuales también debe mencionarse a la programación imperativa (PI) y la programación orientada a objetos (POO). Precede históricamente a las dos últimas y se destaca por su alto nivel de abstracción y sus robustos fundamentos matemáticos. El costo de disponer de un alto nivel de abstracción es que el modelo de ejecución se aleja de la arquitectura física del computador y por lo tanto incurre en una penalidad en tiempos de ejecución. Sin embargo, a medida que el software se hace más complejo y que el poder de procesamiento y almacenamiento aumenta, los beneficios de la abstracción comienzan a compensar holgadamente las penalidades antes mencionadas. Es así como garbage collection (Java, Smalltalk, etc.), generics (Java, Scala), continuaciones (C#, Perl, Ruby, etc.), funciones anónimas (C#, Perl, Ruby, etc.), etc. tienden a adoptarse en lenguajes de escala industrial (indicados entre paréntesis). Otros conceptos igualmente importantes como alto-orden, currificación, esquemas de recursión, etc. también son de un alto valor agregado a la hora de programar, en cualquier paradigma. Este notable impacto de la PF está motivado de manera crítica en el hecho de que se erige sobre el concepto de "función": un concepto simple, general y bien entendido desde el punto de vista matemático.

El proyecto propuesto tiene por finalidad revisar los fundamentos de la PF. La PF tiene como contraparte formal o matemática el llamado Cálculo Lambda (CL), una teoría formal de funciones. En cierto sentido, el CL resume la esencia de un lenguaje de PF. Recientemente [AK:2010] se ha introducido una variante de granularidad más fina que promete ofrecer una visión igualmente refinada de varios de los resultados, técnicas y teorías que se basaban en el CL. Dicha variante se conoce como LSC (Linear Substitution Calculus) y en este trabajo nos interesa revisar algunos de los resultados del CL desde esta nueva perspectiva. En particular, nos interesa hallar estrategias para implementar lenguajes de programación con altos niveles de eficiencia: proponemos rever la teoría de optimalidad de LC desde el ángulo de la LSC.