2747 - E 1534/13 Lenguajes de patrones dinámicos: fundamentos e implementación

Integrantes: Bonelli, Eduardo; Lombardi, Carlos; Barenbaum, Pablo; Steren, Gabriela; Ríos, Alejandro; Kesner, Delia; Viso, Andrés.

Resumen: Los patrones ("patterns") tienen una fuerte presencia en computación, incluyendo áreas tales como ingeniería de software (específicamente diseño de arquitecturas), Inteligencia Artificial, Data Mining y Procesamiento de Imágenes (específicamente en la mecánica de reconocimiento). En lenguajes de programación funcionales el uso de patrones para descomponer datos y acceder a sus partes para poder procesarlas es estándar. También variantes de ese mecanismo de implementación se están incorporando a lenguajes de programación orientado a objetos (como Scala). El procesamiento de datos semi-estructurados (como documentos XML) también se beneficia del uso de patrones para su procesamiento. El problema fundamental de cómo especificar, en un marco riguroso, patrón, datos y finalmente la correspondencia entre ellos ("pattern matching") es el objetivo que se persigue con los *cálculos de patrones*.
Los cálculos de patrones son lenguajes de programación minimales en los que se estudian patrones en su estado más puro. Estos lenguajes de programación se caracterizan por ser concisos, rigurosos en su formulación y desprovistos de facilidades que no contribuyen al comportamiento de los patrones. Los patrones que modelan pueden ser de una de dos clases: *estáticos* o *dinámicos*. Los primeros son aquellos que se especifican en tiempo de compilación mientras que los segundos pueden ser creados en tiempo de ejecución. Los patrones en general y los dinámicos en particular, comienzan a ganar interés principalmente por la facilidad que brindan para escribir programas genéricos en el sentido de poder recorrer de manera *uniforme* distintos tipos de estructuras de datos (como ser árboles, listas, documentos XML, etc.). Como contraparte de la flexibilidad que ofrecen, poco se conoce sobre sus propiedades. El objetivo de este proyecto es remediar esta situación de modo tal que el uso y comprensión de estos lenguajes no se limite a prueba y error a partir de implementaciones ad-hoc. En efecto, a pesar de que existen algunos prototipos que incorporan patrones expresivos a lenguajes como Java (Tom) o extensiones de ML (TomML) o lenguajes funcionales (Bondi), poco se conoce de las propiedades de estos lenguajes. En el caso de los patrones dinámicos esto se debe a que las nuevas formas de polimorfismo que ofrece ("path polymorphism" y "pattern polymorphism") son difíciles de tipar. Asimismo, los cálculos de patrones dinámicos no cumplen con una propiedad fundamental de muchos lenguajes funcionales, a saber la *secuencialidad*. Esta propiedad significa que hay más de una manera de generar redexes que no necesariamente son compatibles entre sí. Esto hace que prácticamente toda la teoría desarrollada de sistemas de reescritura y lambda cálculo no sean aplicables a estos cálculos.