SUBTIPOS Y OBJETOS EN TEORÍAS Y HERRAMIENTAS DE PROGRAMACIÓN BASADAS EN TEORÍA DE TIPOS

2. Resumen de la investigación

La investigación propuesta en este proyecto tiene su origen en la teoría formulada en [Tas97, BT98] la cual es a su vez una extensión de la *Teoría Constructiva de Tipos* de Per Martin-Löf [ML84, Mar87]. Consiste en el desarrollo de los siguientes temas: la noción de inclusión de tipos, modelos formales de la programación orientada a objetos e implementación de asistentes de desarrollo de derivaciones formales.

La teoría de Martin-Löf puede ser entendida como un lenguaje de programación funcional con tipos dependientes. La extensión mencionada arriba incorpora la noción de tipos de registros dependientes y la relación de inclusión de tipos por ella inducida. Sin embargo, no contempla la inclusión entre tipos inductivamente definidos. Un primer objetivo de este proyecto es investigar la posibilidad de formular esta última extensión.

Los conceptos de registro e inclusión de tipos son fundamentales en varias teorías relevantes de la programación orientada a objetos. En nuestra teoría los tipos de registros y de funciones poseen un mayor poder de expresividad por el hecho de ser dependientes. Es asíque luego de haber adquirido experiencia en el manejo de estos conceptos parece natural aplicarlos para proveer una formulación alternativa de la noción de *objeto*. Este es el segundo objetivo de la investigación propuesta.

La investigación de las posibilidades de la utilización práctica de la teoría constructiva de tipos en programación ha dado origen a una corriente de investigación abocada a la construcción de programas que proveen asistencia en el desarrollo de derivaciones formales en esta teoría. El objetivo final es llevar a la práctica métodos de desarrollo de programas de corrección certificada. En este plano, nuestro punto de partida es un prototipo, presentado en [Bet98], que se basa en la extensión de la teoría constructiva de tipos citada al comienzo. Este proyecto se plantea también el objetivo de refinar este prototipo, eventualmente incorporando a él los mecanismos que se justifiquen a partir de los estudios teóricos descriptos.

3. Antecedentes

3.1. **Estado actual del conocimiento.** En esta última década, varios equipos de investigación, principalmente en Europa, han dedicado un considerable esfuerzo al diseño e implementación de editores de prueba interactivos basados en teorías constructivas de tipos. Ejemplos de estos sistemas son ALF [Mag95], Coq [Bar97] y LEGO [Pol94], desarrollados en la Universidad Tecnológica Chalmers de Gotemburgo, en el INRIA de París y en el LFC de la

Universidad de Edimburgo, respectivamente. Uno de los principales ingredientes de estos sistemas es el carácter unificador de la teoría que implementan, en la cual pueden ser expresados programas, teoremas y pruebas de estos teoremas. Otro punto destacable es que el usuario es guiado en forma interactiva por el sistema en el proceso de construcción de un programa o prueba, siendo verificada inmediatamente la validez de cada paso del desarrollo. El principal objetivo de estos sistemas es convertirse en sofisticadas herramientas que asistan en la tarea del desarrollo incremental de programas correctos. Sin embargo, el marco conceptual necesario para desarrollar software verificado es de una muy alta complejidad y requiere cubrir muchos aspectos que en realidad escapan al problema concreto de la construcción del asistente de pruebas.

Las teorías constructivas de tipos han sido desarrolladas con el propósito de proveer un fundamento de la Matemática Constructiva. Cada una de ellas puede ser descripta como un lenguaje de programación funcional con tipos dependientes. Esta última propiedad permite expresar especificaciones de programas como tipos. Por lo tanto, que un programa tenga un tipo P equivale a que el programa satisface la especificación P. El problema de verificar si una expresión dada tiene un tipo dado en cada uno de estos lenguajes es decidible, lo cual convierte al lenguaje en cuestión en una teoría formal de programación. Por otra parte, y en concordancia con la motivación original con la que han sido propuestas, las teorías proveen un marco formal para el desarrollo riguroso de construcciones matemáticas. Es entonces natural pensar en el uso del editor de prueba para *hacer* Matemática. Una razón muy importante para utilizar el sistema con este último propósito es la enorme influencia que esta disciplina tiene en el área de Ciencia de la Computación: subyacente a un gran número de algoritmos existen sutiles argumentos matemáticos; buenos ejemplos pueden encontrarse en teoría de grafos o en el área de álgebra computacional. Se considera entonces de gran interés también la posibilidad de hacer un desarrollo extensivo de experimentos matemáticos y la consecuente construcción de bibliotecas de pruebas. En el caso concreto de la Teoría de Tipos de Martin-Löf la experiencia ha mostrado que para lograr adecuadas representaciones formales de, por ejemplo, construcciones algebraicas, el poder expresivo del lenguaje de la teoría en su formulación original no es suficiente, o al menos no suficientemente adecuado. En el plano de la programación esto se manifiesta en restricciones a la posibilidad de expresar y manejar Tipos Abstractos de Datos. En [Tas97, BT98] se ha formulado una extensión de esta teoría con tipos de registros dependientes (dependent record types) y subtipado, que va en la dirección de subsanar esta deficiencia. Esta extensión ha sido puesta en práctica en varias experimentaciones mostrando un potencial interesante de expresividad.

Por otro lado esta extensión ha despertado interés en cuanto a su utilización como marco conceptual para modelar un cálculo o teoría de objetos, en el sentido de [AC96].

3.2. **Antecedentes del equipo investigador.** El grupo de investigación esta compuesto por el Dr. Alvaro Tasistro y el Dr. Gustavo Betarte. Ambos son docentes del Instituto de Computación del grupo de Métodos Formales

y desarrollaron sus estudios de posgrado en el Departamento de Ciencia de la Computación de la Universidad Tecnológica Chalmers de Gotemburgo.

El grupo de Métodos Formales del Instituto de Computación (InCo) está formado por cuatro doctores con cargos de Prof. Agregado de 40 horas de dedicación semanal, un Magister en Informática del PEDECIBA y dos estudiantes de posgrado. Los doctores han realizado sus estudios de posgrado en la Universidad Tecnológica Chalmers de Gotemburgo, Suecia y en el INRIA de Rocquencourt, Francia. Tres de ellos comenzaron su carrera docente en el InCo alrededor del año 1985 y han retornado al país en el lapso que va de Setiembre de 1997 a la fecha.

Las principales áreas de trabajo del grupo son las siguientes:

- la investigación de métodos y herramientas de producción de programas de corrección certificada, con particular interés en aquellos basados en la Teoría Constructiva de Tipos.
- la verificación formal de sistemas reactivos.

Está entre los objetivos generales de este proyecto contribuír a la consolidación de este grupo de investigación. Es de especial interés el mantenimiento y fortalecimiento de los vínculos existentes con grupos de investigación extranjeros, a saber: el PMG de Gotemburgo, el grupo Coq del INRIA-Rocquencourt, el LSV de la ENS de Cachan y el grupo de Métodos Formales de la PUC de Río de Janeiro.

Para este proyecto se contará además con la estrecha colaboración de miembros del grupo de Lógica de la Programación de la Universidad Chalmers de Gotemburgo (Prof. Bengt Nordström) y del Departamento de Filosofía de la Universidad de Helsinki, (Dr. Petri Mäenpää). Esta colaboración no incluye apoyo financiero.

El estado actual de nuestras investigaciones en este tema puede resumirse de la siguiente manera:

- Hemos desarrollado una extensión de la Teoría Constructiva de Tipos con registros e inclusión de tipos [Tas97, BT98].
- Se han implementado en máquina verificadores de tipo de expresiones para la teoría original (en Montevideo) y para la extensión mencionada en el ítem anterior (en Gotemburgo, [BT98, Bet98]).
- Se ha formalizado en la extensión mencionada en el primer ítem y verificado mecánicamente:
  - *Una definición de una estructura general para algoritmos de ordenamiento por inserción* [Tas96]. En este trabajo se propone una formulación tal que los algoritmos de ordenamiento más generales son aquellos que pueden ser usados para el ordenamiento de secuencias de elementos de cualquier conjunto base, provisto que una relación total pueda ser definida sobre el mismo. La estructura de algoritmos propuesta se basa en la especificación de un tipo abstracto de datos, llamado de *estructuras de inserción*. El método general de ordenamiento por inserción es entonces escrito como un programa que depende de estas

estructuras pero sin hacer mención de la particular implementación de las mismas.

- *Una porción simple de la teoría de álgebras de Boole*[Bet97]. La noción de sistema de álgebras es formalmente representada por medio de tipos de registros, el cual a su vez puede obtenerse extendiendo tipos de álgebras previamente definidos con nuevos componentes que cumplen determinados axiomas. Así, puede expresarse naturalmente en el lenguaje el hecho de que un álgebra de Boole es un reticulado distributivo junto con dos operadores nularios, 1 y 0, y un operador de complementación. La relación de subtipado inducida por registros permite obtener directamente que toda propiedad sastifecha por cualquier reticulado es también satisfecha por cualquier álgebra de Boole. Por otro lado, el uso de familias de tipos permite definir álgebras parametrizadas en ciertos componentes, lo cual puede entenderse como la especificación de un *generic package* de ADA o la definición de un *functor* de SML. El conjunto de definiciones y pruebas implementadas puede entenderse como el básico necesario para llevar a cabo verificación formal de hardware en teoría de tipos.

4. Objetivos de la investigación

Los temas de investigación que planeamos abordar durante los próximos dos años en el marco de este proyecto se describen a continuación.

Modularización y reutilización. Desde comienzos de la década de los '80 los lenguajes de programación han sido enriquecidos con mecanismos para la definición y manejo de unidades básicas de programación cuya combinación permita dar forma a (diferentes) programas. Así, desde la noción de *módulo* introducida por N. Wirth en Modula2, pasando por los *packages* de ADA y las *structures* de SML, la comunidad de teoría de la programación ha venido trabajando en la formulación y explicación de la noción de módulo y *tipo abstracto de dato*. Ortogonalmente, la noción de *subtipo*, originalmente introducida como una relación entre *clases* en el lenguaje SIMULA, provee la posibilidad de reusar programas que se han desarrollado para un cierto tipo de datos S sobre objetos de tipo T, si es el caso de que T es un subtipo de S. De la misma forma en que T hereda los programas desarrollados para S uno puede entender que las pruebas desarrolladas para objetos de este último tipo también tienen que ser válidas para objetos de tipo T.

Con el propósito de formular una teoría formal de programación que incorpore estos mecanismos es que este proyecto se plantea profundizar el estudio, ya comenzado y presentado en los trabajos [Tas97, BT98, Bet98], de posibles extensiones a la Teoría de Tipos de Martin-Löf. Un primer objetivo del trabajo es estudiar la posibilidad de extender la relación de inclusión de tipos, la cual en su actual formulación solo concierne los tipos de registros y de funciones, con el fin de abarcar tipos de datos inductivamente definidos (Listas, Arboles, etc.). Recientemente han surgido trabajos que encaran la solución de estos problemas haciendo uso de la noción de *coerción* [Bar95, Bai97, Saï97], mecanismo

que en principio parece ser ortogonal al adoptado en los trabajos arriba mencionados. La clarificación de la compatibilidad de los distintos mecanismos es uno de los efectos laterales más importantes de este aspecto de la investigación. La metodología natural para realizar este trabajo de clarificación es desarrollar abundante experimentación y en particular el área mas fértil es la formalización constructiva de teorías algebraicas (el cual a su vez es el tópico en el que se centran los trabajos referenciados arriba en último término).

Teoría de tipos y orientación a objetos. Ha sido la generación de lenguajes orientados a objetos (o clases) (Smalltalk, C++, Java) los que más extensivamente han explotado las posibilidades que brinda contar con un mecanismo de subtipado y herencia en la actividad de construcción de programas. Por esta razón es que se entiende que una extensión natural del estudio que se plantea arriba es la de abordar el estudio y desarrollo de teorías generales de lenguajes de programación orientados a objetos en función del marco conceptual provisto por la teoría de tipos que nos concierne y su extensión con construcciones que permitan expresar apropiadamente las nociones que hemos detallado previamente.

Desarrollo de asistentes de prueba La investigación a desarrollar no se limitará a obtener un análisis puramente teórico de las áreas arriba mencionadas. En particular interesa continuar la actividad de implementación de editores de prueba, por un lado, incrementando la funcionalidad del prototipo existente, el cual en su estado actual requiere del usuario, por ejemplo, una familiaridad extrema con la teoría que implementa el sistema. Por otra parte, y en una conexión mas directa con el trabajo que se plantea en este proyecto, es de gran importancia poder incorporar al asistente los mecanismos que se justifiquen a partir de los estudios teóricos detallados arriba. La posibilidad de poder experimentar con estos nuevos mecanismos es vital en el proceso de clarificación de los mismos.

5. Metodología

El proyecto incluye actividades de investigación de dos naturalezas diferentes que llamaremos *especulativa* y *concreta*.

Las de la primera clase están caracterizadas por el hecho de que no es posible fijar a priori un plazo para la obtención de resultados. En este caso, incluyen las investigaciones relacionadas con la posibilidad de formular una teoría formal de la relación de inclusión entre tipos definidos inductivamente, así como la de utilizar la teoría constructiva de tipos extendida con registros dependientes e inclusión de tipos como base para una lógica de la programación orientada a objetos. Estas investigaciones son del tipo de las realizadas en Lógica Formal y su éxito está dado por la posibilidad de abstraer reglas formales a partir de manipulaciones e inferencias concretas que se vayan identificando. Es esencial asimismo conocer en detalle propuestas similares formuladas por otros investigadores, con el fin de evaluarlas e investigar su posible reformulación o extensión. En este sentido, adquieren vital importancia las actividades de colaboración

con otros laboratorios. Como hemos mencionado, estamos en contacto con investigadores en Suecia y Finlandia que mantienen proyectos en estos temas.

Las actividades de la segunda clase se caracterizan por tener dados sus objetivos y metodología de manera concreta, por lo cual sus resultados son más previsibles. A esta clase pertenecen: el refinamiento del prototipo de asistente de desarrollo formal y la experimentación en programación y formalización de Matemática utilizando los diferentes mecanismos de subtipado ya explicados. Más precisamente, podemos prever que, como resultado de estas actividades, se obtendrá un prototipo con funcionalidades significativas para la edición de desarrollos formales, lo cual lo convertirá en una aceptable herramienta práctica. Este prototipo estará basado en una teoría en la que, al menos, los tipos de registros dependientes y la correspondiente relación de inclusión estarán incorporadas con satisfactoria generalidad. También es posible prever que se alcanzarán conclusiones significativas sobre la comparación entre los mecanismos de subtipado explícito y aquellos de subtipado implícito o coercivo. Los métodos de trabajo en estos casos son análogos a los empleados en el desarrollo de programas sofisticados.

6. Relevancia del tema propuesto

El marco general en el que este proyecto se inscribe es el de la investigación de teorías y metodologías de desarrollo de programas de corrección certificada, esto es, de programas que estén garantizados en cuanto a no contener falla alguna. Las investigaciones propuestas pueden contribuir resultados tanto en el plano teórico como en el que llamaremos *tecnológico*, el cual consiste en este caso en el desarrollo de herramientas de programación. En el primer plano, comenzamos por observar que aún no existen resultados de experimentaciones con la integración de mecanismos de subtipado explícito e implícito, así como tampoco formulaciones de teorías que contemplen la relación de inclusión entre tipos inductivos. Tampoco existen lógicas de la programación orientada a objetos, al menos en el sentido en que la teoría constructiva de tipos lo es para la programación funcional. En el plano tecnológico, cabe destacar que los asistentes de desarrollo de derivaciones formales en la teoría constructiva de tipos que han sido construidos en Europa están siendo utilizados en aplicaciones industriales, tales como el desarrollo de protocolos para comercio electrónico [Bol97]. El componente esencial de estos asistentes es un programa que cumple la misma función que el prototipo que ya ha sido construído. A la vez, éste, que será mejorado como resultado de este proyecto, ya contiene mecanismos que agregan expresividad a la teoría que implementan los desarrollados en los laboratorios europeos.

8. Justificación de recursos

8.1. **Justificación de equipos solicitados.**

Se solicita la compra de un computador personal Intel y una impresora laser PostScript con unidad duplex.

El computador personal tiene la ventaja de ser de menor costo que una estación de trabajo Risc y brinda la posibilidad de ser utilizado como estación de trabajo sobre un sistema operativo tipo UNIX. Por otra parte nuestro grupo de investigació utiliza una gran variedad de software desarrollado para este sistema operativo. Se entiende que los requerimientos mínimos del computador personal son: procesador Pentium de 200 Mhz, 64 Mb de memoria RAM y 4 GB de disco duro.

La impresora se utilizará para imprimir los documentos relacionados con el proyecto. Las características de la impresora se basan en que actualmente la difusión de documentos científicos se hace mayoritariamente en forma electrónica, y en formato PostScript.

8.2. **Justificación de viajes y estadías.**

Dentro del área de este proyecto, los grupos más productivos y con más trayectoria se encuentran en Europa. Con algunos de ellos se han desarrollado hasta ahora intercambios y el resultado exitoso del proyecto supone necesariamente la continuidad de esta actividad. Por estas razones se solicitan: dos viajes y estadías al exterior, que se utilizarán para una estadía de trabajo en alguno de esos laboratorios y para concurrir a una conferencia internacional en temas vinculados con el área de investigación.

Por otro lado, como parte del intercambio se prevee un viaje y estadía de un investigador extranjero en el Uruguay.

8.3. **Justificación de los requerimientos del equipo de investigación.**

8.3.1. *Complemento salarial.* Se solicita una compensación salarial de acuerdo a los criterios de la Universidad de la República para el Responsable Científico del Proyecto, Alvaro Tasistro. Dicho complemento salarial es por los dos años del proyecto, en un cargo de Profesor Agregado (Grado 4), 40 hs semanales.

8.3.2. *Personal nacional a contratar.* Se solicita también una compensación salarial para una persona a contratar. El perfil de esta persona es la de un estudiante de posgrado o posgraduado con conocimientos de teoría de tipos, que esté capacitado para llevar adelante parte de la implementación del proyecto. Se espera una dedicación de 15 hrs. semanales al proyecto durante el ú ltimo año.

9. Actividades específicas y cronograma de actividades

Proponemos dividir el proyecto en las siguientes etapas:

1. (6 meses) En esta primer etapa se continuará experimentando con el prototipo ya implementado. Se comenzará en paralelo a investigar el problema del subtipado entre conjuntos inductivamente definidos y se realizará un seminario con el propósito de estudiar a fondo el estado del arte de las teorías formales de lenguajes orientados a objetos.

2. (1 mes) Producción de una nueva especificación del asistente de pruebas. Eventualmente se podrá extender el prototipo con los mecanismos resultados de la investigación desarrollada en la primer etapa.
3. (6 meses) Implementación del nuevo prototipo e investigación de la posible formulación de un cálculo de objetos. Se prevé un viaje de miembros del proyecto a uno de los laboratorios europeos asociados o la posible venida de alguno de los colaboradores en el exterior.
4. (2 meses) Evaluación y documentación del trabajo realizado en las etapas previas.
5. (4 meses) Se desarrollarán experimentos utilizando la nueva implementación del prototipo, lo cual se presume generará un ciclo de solución-mejora de éste.
6. (5 meses) Proceso final de documentación y producción de artículos científicos.

Referencias

[AC96] M. Abadi and L. Cardelli. *A Theory of Objects*. Springer, 1996.

[Bai97] A. Bailey. Lego with implicit coercions, 1997. Documentation report, available at `ftp.cs.man.ac.uk/pub/baileya/Coercions`.

[Bar95] G. Barthe. Implicit coercions in type systems. In *Selected Papers from the International Workshop TYPES '95, Torino, Italy, LNCS 1158.*, 1995.

[Bar97] B. Barras et al. The Coq Proof Assistant Reference Manual – Version V6.1. Technical Report 0203, INRIA, 1997.

[Bet97] G. Betarte. Dependent record types, subtyping and proof reutilization. In *Online Proceedings of the TYPES Workshop* Subtyping, Inheritance and Modular Development of Proofs, Durham, England, September 1997.

[Bet98] G. Betarte. *Dependent Record Types and Algebraic Structures in Type Theory.* PhD thesis, PMG, Dept. of Computing Science, University of Göteborg and Chalmers University of Technology, 1998.

[Bol97] D. Bolignano. Towards the formal verification of electronic comerce protocols. In *10th IEEE Computer Security Foundations Workshop*, 1997.

[BT98] G. Betarte and A. Tasistro. *Extension of Martin-Löf's Type Theory with Record Types and Subtyping*, pages 21–39. Oxford University Press, 1998.

[Mag95] L. Magnusson. *The Implementation of ALF - a Proof Editor based on Martin-Löf's Monomorphic Type Theory with Explicit Substitution.* PhD thesis, Programming Methodology Group, Dept. of Computing Science, University of Göteborg and Chalmers University of Technology, 1995.

[Mar87] P. Martin-Löf. Philosophical Implications of Type Theory., 1987. Lectures given at the Facoltá de Lettere e Filosofia, Universitá degli Studi di Firenze, Florence, March 15th. - May 15th. Privately circulated notes.

[ML84] P. Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.

[Pol94] R. Pollack. *The Theory of LEGO: a proof checker for the Extended Calculus of Constructions.* PhD thesis, University of Edinburgh, 1994.

[Saï97] A. Saïbi. Typing algorithm in type theory with inheritance. In *24th. Annual SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1997.

[Tas96] A. Tasistro. Abstract insertion sort in type theory. In *Proceedings of the 7th Nordic Workshop on Programming Theory, Report 86 PMG, Göteborg, Sweden.*, 1996.

[Tas97] A. Tasistro. *Substitution, record types and subtyping in type theory, with applications to the theory of programming.* PhD thesis, PMG, Dept. of Computing Science, University of Göteborg and Chalmers University of Technology, 1997.

E-mail address: `[gustun,tato]@fing.edu.uy`