

# Informe final publicable de proyecto

## Control de tipos para Teoría de Tipos Nominal

Código de proyecto ANII: FCE\_1\_2019\_1\_156630

16/05/2023

**SZASZ CERUTTI, Nora Adriana** (Responsable Técnico - Científico)

**COPELLO GIGIREY, Ernesto Sebastián** (Investigador)

**FERNANDEZ ROMERO, Maribel Rosario** (Investigador)

**PAGANO, Miguel** (Investigador)

**SOLSONA, José** (Investigador)

**TASISTRO SOUTO, Álvaro Daniel** (Co-Responsable Técnico-Científico)

---

UNIVERSIDAD ORT. FACULTAD DE INGENIERÍA (Institución Proponente) \\ UNIVERSIDAD ORT. FACULTAD DE INGENIERÍA

## Resumen del proyecto

El objetivo estratégico de esta línea de investigación es la formulación e implementación de un un Nominal Logical Framework basado en la teoría de sintaxis abstracta nominal con tipos dependientes.

Esto incluye el diseño e implementación del sistema formal y de un algoritmo de controlador de tipos que garantice la buena formación de expresiones de los sistemas representados.

Se desarrollaron casos de representación de sistemas lógicos, incluyendo la Aritmética de Primer Orden y Lógica de Orden Superior, con el estudio correspondiente de la adecuación de cada representación.

Se compararon estos resultados con los logrados por otros medios, tales como las Estructuras de Sistemas Lógicos (Logical Frameworks) basados en el Cálculo Lambda con tipos dependientes.

**Ingeniería y Tecnología / Ingeniería Eléctrica, Ingeniería Electrónica e Ingeniería de la Información / Ingeniería de Sistemas y Comunicaciones / Computación Teórica**

**Palabras clave: Términos nominales / Teoría Constructiva de Tipos / Control de tipos /**

## Introducción

Este proyecto discurre en el campo de la llamada Sintaxis Abstracta Nominal. Ésta es una teoría o formulación de la sintaxis formal de lenguajes cuya principal aplicación, y motivación original, es el tratamiento matemático y mecanizable de la localidad de nombres. Veamos primero con un poco de detalle en qué consiste este enfoque de la sintaxis formal de lenguajes.

En primer lugar, la noción de sintaxis abstracta es ya tradicional y fundamental en el tratamiento matemático y computacional de los lenguajes formales. Básicamente propone la representación de cada una de las diversas categorías de expresiones de cada lenguaje como un tipo de datos inductivo, i.e. de las expresiones como árboles finitos. Asociados a esta forma de representación se tiene una variedad de algoritmos de control de propiedades dependientes del contexto (por ejemplo, control de tipos), de traducción, y de resolución de ecuaciones (unificación y matching) que forman parte del repertorio estándar de las tecnologías de implementación de intérpretes y compiladores de lenguajes de programación, así como de verificadores y asistentes computarizados de demostración matemática.

En la sintaxis abstracta tradicional, los nodos internos o constructores de los árboles de expresiones son palabras u operadores que juegan diversos roles de conexión o modificación, formando expresiones complejas a partir de una o más expresiones previamente construidas. Los átomos de estas estructuras moleculares son usualmente simples nombres, que pueden ser interpretados, ya sea como constantes de significado predefinido, o como variables de significado a ser determinado por el uso ulterior de la expresión. En el tratamiento más simple y directo de estos árboles, se dice que el significado de cada expresión depende del significado de esas variables,

Ahora bien, existe un fenómeno generalizado en los lenguajes formalizados, que es el uso de nombres, de significado no predeterminado, en un cierto ámbito o alcance, fuera del cual el nombre en cuestión no es significativo. Esto significa que el alcance asociado al nombre es una expresión que depende de ese nombre, pero no la dependencia desaparece fuera del ámbito de localidad. Ejemplos de esto son los cuantificadores lógicos, tales como el cuantificador existencial  $\exists$ , varios operadores matemáticos como los de sumatoria, integración y diferenciación, y las variables locales a bloques en los lenguajes de programación.

Así, por ejemplo, se tiene expresiones tales como  $(\exists x \in R) x < x + x$ , donde el cuantificador existencial introduce el nombre  $x$  asociándole un dominio de variación y luego se aplica a una expresión que utiliza el nombre  $x$ , a saber,  $x < x + x$ , en la que el operador relacional  $<$  combina o se aplica a las expresiones  $x$  y  $x + x$ , esta última formada por el operador de suma aplicado a  $x$  y a  $x$ . Usando la terminología de arriba,  $\exists$  introduce o declara  $x$ , y le asocia automáticamente un alcance, en este caso la expresión  $x < x + x$ . Es claro que ésta depende de  $x$  (en particular, es verdadera para los valores de  $x$  positivos, pero falsa para todos los demás). Sin embargo la expresión completa  $(\exists x \in R) x < x + x$  no depende de  $x$ , ya que es verdadera. Decimos que  $x$  es local a esta última expresión completa.

Esta cualidad de localidad tiene varias implicaciones, entre las cuales destacamos:

1) El nombre  $x$  es irrelevante u opaco al exterior de la expresión.

En otras palabras, las expresiones  $(\exists x \in \mathbb{R}) x < 2x$  y  $(\exists z \in \mathbb{R}) z < 2z$  son, a todos efectos, intercambiables.

Nos gustaría, de hecho, poder decir que son la misma expresión.

2) El uso de estos nombres en forma local como en el ejemplo precedente y a la vez global (i.e. no afectada por una limitación de alcance), como en  $x < 0 \ \& \ (\exists x \in \mathbb{R}) x < x + x$  (donde usamos  $\&$  para denotar la conjunción lógica “y”), obliga a mantener una disciplina de manipulación que respete la correcta interpretación de cada aparición del nombre en cuestión, o sea a no confundir apariciones que deben interpretarse diferentemente. Éste es el caso en este ejemplo, donde la primera aparición de  $x$  no tiene nada que ver con la usada localmente en el segundo conyunto. (Como quedó dicho en 1), estas últimas apariciones de  $x$  son en realidad opacas desde el exterior). Como consecuencia, la expresión del ejemplo depende de  $x$ , siendo verdadera para los valores negativos de este nombre y falsa para todos los demás. Para verificar esta aserción, es importante recordar que el segundo conyunto es verdadero, independientemente de los valores que tome  $x$ .

A los efectos de ilustrar las consideraciones a tener en cuenta para la correcta manipulación de nombres en presencia del fenómeno de localidad, considérese la fórmula

$(\forall y)(\exists x) y < x$  que es claramente verdadera si los nombres  $x$  e  $y$  son tomados como variando en por ejemplo el conjunto de los números reales. Siendo así, es claro que también será verdadera cualquier instanciación de la fórmula que constituye el alcance del cuantificador universal, en la que se adjudique al nombre  $y$  un valor real. En términos sintácticos, esta instanciación se obtiene sustituyendo el nombre  $y$  por una expresión del valor en cuestión en la fórmula afectada por el cuantificador universal. Por ejemplo, es verdadera, como instancia o caso particular de la cuantificación universal original:  $(\exists x) \sqrt{2} < x$ .

Pero supóngase ahora que esta instanciación se desee realizar para un valor  $x$  indeterminado. Esto sucede en general cuando el nombre  $x$ , en un cierto contexto, ha sido introducido como instrumento de un razonamiento genérico sobre propiedades de números reales. Si procedemos descuidadamente, la instanciación de nuestra cuantificación universal en tal caso, daría lugar a  $(\exists x) x < x$ , que es claramente falsa, contradiciendo la validez de la cuantificación universal y la consecuente validez de cualquiera de sus instancias. El problema, naturalmente, surge de la confusión de dos usos del nombre  $x$ , uno conspicuo (el global) y el otro irrelevante (el local). Al introducir sin más el nombre global en el alcance del local, el primero ha sido capturado y pierde su carácter conspicuo (de hecho, pierde su identidad original).

El problema tiene solución recordando, una vez más, la irrelevancia del nombre local, que justifica el reemplazo de éste por cualquier otro que no esté siendo usado también en forma global en el mismo alcance. Pero este detalle de manipulación da lugar a una definición de la “simple” operación de sustitución sintáctica que ha desafiado a los lógicos por décadas por su complejidad y propensión al error.

Esta complejidad tiene su raíz en los cambios en la naturaleza de las expresiones inducidos por la introducción del fenómeno de localidad de nombres. Los simples árboles finitos originales, que admiten tratamientos clásicos y directos utilizando recursión e inducción estructurales, se han complejizado al admitirse constructores que introducen nombres locales. Como consecuencia, hemos observado, en primer lugar, que la identidad de expresiones como árboles debería ser modificada de modo de permitir igualar expresiones que difieran sólo en la elección de cada nombre local. A partir de este principio, sería asimismo deseable formular métodos de recursión e inducción que fueran aplicables a esta clase de árboles con su nueva relación de equivalencia, en una forma tan natural y directa como la inducción y recursión estructurales lo son a los árboles simples originales y, por esta vía, simplificar la manipulación de nombres en operaciones como por ejemplo la sustitución.

Una formulación matemática de esta nueva noción de árboles con localidad ha sido presentada en [GP01] y estudiada intensamente desde entonces. Algunas referencias fundamentales son [Pit03] y [Gab11]. Básicamente lo que se introduce es una estructura o modelo de base construido sobre una versión de la teoría de conjuntos en la cual es posible representar diversos lenguajes haciendo uso de una construcción primitiva de abstracción de nombre, que es la que introduce nombres locales a expresiones. Para cada uno de estos lenguajes se formula un principio de inducción que respeta la identidad a menos de elección de nombres locales y que simplifica la definición de las diversas operaciones de manipulación.

Entre los lenguajes que resulta importante representar por este método se encuentran los sistemas lógicos, como las formalizaciones de la Matemática en Lógica de Primer Orden o Lógica de Orden Superior. Estos lenguajes incluyen

expresiones para individuos (términos), proposiciones (fórmulas) y demostraciones (derivaciones en el sistema formal en cuestión). Habrá entonces diversos tipos de árboles (modificados según la teoría nominal, es decir admitiendo como naturales los nombres locales) según las categorías sintácticas detalladas arriba. Pero un detalle fundamental es que para representar adecuadamente las derivaciones formales, los árboles correspondientes deberán clasificarse según la proposición que demuestran, de modo de poder dar lugar a una representación fidedigna de los correspondientes constructores, es decir, de las reglas de inferencia del sistema lógico en cuestión. Esto implica que no podrá existir un único tipo para todas las derivaciones formales, sino una familia de ellos, indizada por las fórmulas. Esto conduce a la necesidad de una reformulación del sistema de base de modo de admitir estas familias de tipos o, como son también denominados en la jerga, de tipos dependientes.

Lo que este proyecto plantea y ha logrado en buena medida es:

- Una formulación de la teoría de sintaxis abstracta nominal con tipos dependientes.
- La implementación en máquina del sistema resultante, lo cual incluye al menos un controlador de tipos que garantice la buena formación de expresiones de los sistemas representados.
- El desarrollo de una serie de ejemplos de representación de sistemas lógicos, incluyendo la Aritmética de Primer Orden y Lógica de Orden Superior, con el estudio correspondiente de la adecuación de cada representación.
- La comparación de estos resultados con los logrados por otros medios, tales como las Estructuras de Sistemas Lógicos (Logical Frameworks) basados en el Cálculo Lambda con tipos dependientes. Un título natural entonces para esta investigación y sus resultados es el de Nominal (Logical) Framework.

### **Metodología/diseño del estudio**

Se propusieron diferentes formulaciones de las teorías nominales con tipos dependientes, estudiando sus propiedades metateóricas y su posible implementación en máquina. Se realizaron experimentos de formalización de diversos sistemas lógicos utilizando estas formulaciones: Lógica de Primer Orden, Aritmética, Cálculo Lambda puro y simplemente tipado, Lógica de orden superior, entre otros. En cada caso se evaluó el grado de adecuación de la formalización, así como la complejidad de la misma, en vistas a los algoritmos de control de tipos.

Esto dio lugar a un proceso de iteración y ajuste de las distintas formulaciones, y finalmente se arribó a una versión de una teoría nominal con tipos dependientes de complejidad moderada que posee las propiedades deseadas.

Se realizaron análisis metateóricos del sistema propuesto, que resultaron exitosos y se diseñó un algoritmo de control de tipos.

La metodología de trabajo consistió principalmente en reuniones de discusión, las cuales se realizaron en forma periódica durante todo el proyecto. Participaron de las mismas todos los involucrados en el proyecto, y tuvimos la oportunidad de realizar varias instancias de intercambio presencial entre los investigadores (Montevideo, Londres) que resultaron muy fructíferas.

### **Resultados, análisis y discusión**

Se ha completado una formulación de la teoría de Sintaxis Abstracta Nominal o Nominal Framework (en adelante NF) que puede juzgarse como satisfactoria por varias razones:

- Es de una complejidad moderada.
- Los resultados metateóricos son naturales, sencillos de enunciar, y de demostración directa.
- Se ha mostrado también razonablemente adaptable a una formulación orientada al control de tipos.
- Ha soportado la codificación de ejemplos exigentes, tales como la Aritmética de Primer Orden y la Lógica de Orden Superior.

Se ha diseñado un controlador de tipos y esbozado un método de demostración de su corrección bastante directo.

Se han desarrollado varios ejemplos interesantes de codificación de sistemas formales en el NF. Las demostraciones de adecuación de estas representaciones son directas y más simples que las desarrolladas en e.g. [HHF93] para el Logical Framework basado en el Cálculo Lambda con tipos dependientes.

Se publicó un artículo de conferencia con resultados de una línea subsidiaria de investigación desarrollada, consistente en la formalización de la teoría básica de Conjuntos Nominales.

Los resultados son, por lo tanto, muy alentadores, aunque no completan en su totalidad objetivos inicialmente planteados. Los siguientes están en vías de realización:

- Redacción de artículo para revista con la presentación del NF, su metateoría, e incluyendo los ejemplos más relevantes de codificación de sistemas lógicos, con los correspondientes resultados de adecuación. Este artículo se va a completar en el correr del primer semestre del año próximo.
- Prueba de corrección del diseño de controlador de tipos y su implementación cuidadosa.
- Redacción de un artículo para conferencia o revista con los resultados del apartado precedente.

Las razones por las que estos resultados no fueron alcanzados en el periodo de ejecución del proyecto son, básicamente, las siguientes:

- El diseño del sistema resultó ser una tarea más compleja que lo que había sido evaluado originalmente. Hubo una diversidad de versiones que parecían satisfactorias pero que, puestas a prueba bajo el rigor de la demostración de propiedades, se revelaban inadecuadas, o eran ya a priori en extremo complejas para ser tratadas por métodos razonablemente directos y convincentes.
- Existió cierta discontinuidad y dispersión en el trabajo del equipo, atribuible por sobre todas las cosas a la interrupción prolongada provocada por la pandemia. Esto se intentó subsanar mediante una prórroga que fue otorgada pero que no resultó del todo suficiente para recomponer todas las líneas de trabajo iniciadas. Con todo, las estadías de investigación, realizadas de acuerdo a lo originalmente previsto, permitieron alcanzar un nivel muy aceptable de resultados y continuar al presente con las tareas de redacción y publicación. Cabe consignar que el doctorando integrante del proyecto completará su tesis en el correr del próximo año trabajando sobre las tareas relativas al controlador de tipos. Se espera que una parte sustancial de este trabajo lo realice en una visita a Londres a llevarse a cabo en el próximo semestre.

## **Conclusiones y recomendaciones**

La principal conclusión es que los principales atributos de la Sintaxis Abstracta Nominal, es decir, la posibilidad de representar lenguajes con localidad de nombres permitiendo el razonamiento inductivo sobre ellos aún respetando la identidad de expresiones a menos de elección de los nombres locales, ha sido trasladada por nuestros resultados al nivel de los sistemas formales de una manera general, incluyendo derivaciones clasificadas en función de las fórmulas por ellas probadas. Esto se ha conseguido extendiendo la teoría original de modo de admitir familias de tipos de expresiones (tipos dependientes).

A nuestro parecer, éste es un resultado ampliamente satisfactorio pues provee de una noción de logical framework no basada en sintaxis de orden superior sino de lo que se ha dado en llamar orden 1 y medio, con posibilidad de formular y utilizar principios de inducción sobre los varios tipos y familias de expresiones respetando la identidad a menos de elección de nombres locales.

El impacto de estos resultados ha quedado mediado en el tiempo y lo podremos aquilatar una vez completado el proceso de su publicación que, por razones arriba explicadas, se ha dilatado más allá del cierre del proyecto.

También valoramos el proceso de diseño, en fructífera colaboración entre los investigadores, sobre todo gracias a las estadías que se pudieron concretar, así como la próxima finalización y defensa de tesis de nuestro doctorando que tendrá como parte fundamental apartados pendientes de resolución de este proyecto.

Nuestro trabajo también permite formular ulteriores líneas de investigación, entre las que se destacan:

- La combinación entre sistemas como el presente y los basados en el Cálculo Lambda, lo cual permitiría disponer de lenguajes de programación funcionales, con tipos dependientes y localidad de nombres —eventualidad por cierto auspiciosa.
- La extensión de nuestro NF con ecuaciones de modo de concretar un lenguaje de orden 1 y medio basado en reescritura.

## Referencias bibliográficas

- [Gab11] Murdoch J. Gabbay. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. Bulletin of Symbolic Logic, Volume 17, Number 2, June 2011.
- [GP01] Murdoch J. Gabbay, Andrew M. Pitts. A new approach to abstract syntax with variable binding. Formal Aspects of Computing, 13(35):341363, July 2001.
- [HHP93] Robert Harper, Furio Honsell, Gordon Plotkin. A Framework for Defining Logics. Journal of the ACM, Volume 40, 1, Jan. 1993.
- [Pit03] Andrew M. Pitts. Nominal Logic, a First Order Theory of Names and Binding. Information and Computation, 186(2):165193, 2003.

## Licenciamiento

Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional. (CC BY-NC-ND)