Propuesta de Tesis de Licenciatura
Título

?? de Febrero, 2018

Alumno: Gonzalo Ciruelos (LU: 63/14)
Director: Pablo Barenbaum

El espacio de cómputos de un programa puede tener una estructura compleja. Considere un lenguaje de programación sin efectos secundarios, como lo es por ejemplo el cálculo λ simplemente tipado. Los cómputos que se pueden realizar a partir de una tupla (A, B) son secuencias $(A, B) \to \ldots \to (A', B')$. Estas secuencias siempre pueden descomponerse en dos cómputos independientes, $A \to \ldots \to A'$ y $B \to \ldots \to B'$. La razón por la cual podemos hacer esto es que las sub-expresiones A y B no pueden interactuar la una con la otra. Dicho de forma más general, el espacio de cómputos de (A, B) puede verse como el producto de los espacios de A y B.

Por otro lado, el espacio de cómputos de la aplicación $f(A)$ no es tan fácil de caracterizar. El problema es que f puede usar o no el valor de A, y en caso de usarlo puede posiblemente hacerlo más de una vez.

La motivación de esta tesis es intentar **entender el espacio de cómputos** de programas. El objetivo de entender el espacio de cómputos del cálculo λ es comprender las propiedades de distintas *estrategias de evaluación*, como son call-by-name o call-by-value desde un punto de vista cuantitativo. Una mayor comprensión de estas estrategias desde un punto de vista cuantitativo puede sugerir optimizaciones a programas y podría permitir justificar que ciertas conversiones de programas son correctas (por ejemplo, que no convierten un programa que termina en otro que no).

Otro objetivo para entender el espacio de cómputos del cálculo λ es comprender mejor distintas nociones de equivalencias de programas [JHM69, BMPR16]. Usualmente, se pueden definir diferentes equivalencias de programas respecto a cómo un programa se evalúa en un contexto: por ejemplo, podríamos definir que dos A y B son (contextualmente) equivalentes si, para todo contexto $\mathtt{C}$, $\mathtt{C}\langle A\rangle =_\beta \mathtt{C}\langle B\rangle$. Es claro que poder descomponer el espacio de cómputos de $\mathtt{C}\langle A\rangle$ (resp. $\mathtt{C}\langle B\rangle$) en una estructura que dependa del espacio de cómputos de A (resp. B) sería conveniente para ganar intuición sobre las distintas nociones de equivalencia observacional.

Es sabido que los espacios de derivación del cálculo λ presentan ciertas regularidades [Lév78, Zil84, Lan94, Mel96, Lev15, AL13]. En su tesis de doctorado, Lévy [Lév78] probó que el espacio de derivaciones de un término es un semi-reticulado superior: todo par de derivaciones ρ, σ de un término t tienen una cota inferior mínima $\rho \sqcup \sigma$, definida como $\rho(\sigma/\rho)$, que es única módulo *permutation equivalence*.

Pero el espacio de derivación de un término no es fácil de entender en general: el Problema 2 de la lista de problemas abiertos de la RTA [DJK91] propone el asunto de investigar las propiedades de *spectra*, es decir, espacios de derivación.

En cálculo λ, el ejemplo que dimos antes se traduce en relacionar el espacio de derivación de la aplicación ts con los espacios de derivación de t y s, lo cual parece ser un problema difícil en general. De nuevo, esto se debe a que t bien puede borrar o duplicar a s, lo cual sugiere que el problema está relacionado con el de manejo de *recursos*: si supiéramos exactamente cómo t usa a s el problema se simplificaría. Por eso en esta tesis nos proponemos explorar la hipótesis de que tener una representación explícita del manejo de recursos puede proveer una mejor intuición sobre la estructura de los espacios de derivación.

Existen varios cálculos λ que abarcan el problema del manejo de recursos de manera explícita [Bou93, ER03, KL07, KR], la mayoría de los cuales están inspirados en la lógica lineal de Girard [Gir87].

Recientemente, una familia de estos formalismos, cálculos dotados con *sistemas de tipos intersección no idempotentes*, ha recibido atención [Ehr12, BL13, BKDR14, BKV17, Kes16, Via17, KRV18]. Estos sistemas de tipos permiten capturar, de manera estática, propiedades dinámicas no triviales de los términos (en particular distintos grados de *normalización*), mientras que al mismo tiempo se prestan a demostraciones simples por inducción.

Los sistemas de tipos intersección fueron originalmente propuestos por Coppo y Dezani-Ciancaglini [CD78] para estudiar la terminación del cálculo λ. Se caracterizan por la presencia de un constructor de tipos *intersección* $\tau \cap \sigma$. Los sistemas de tipos intersección *no idempotentes* se distinguen de los (clásicamente) idempotentes por el hecho de que la intersección de tipos no es idempotente, es decir que τ y $\tau \cap \tau$ no son tipos equivalentes. En cambio, la intersección se comporta como un conectivo multiplicativo en lógica lineal. Los argumentos de las funciones típicamente tendrán varios tipos dependiendo de cuántas veces y cómo son usados, lo cual da a estos sistemas una notable capacidad a la hora de rastrear cómo son utilizados los recursos de un programa.

Los sistemas de tipos intersección no idempotentes fueron originalmente formulados por Gardner [Gar94], y más tarde reintroducidos por de Carvalho [Car07].

En sistemas de tipos intersección no idempotentes típicos, la normalización de pruebas no es confluente. En esta tesis **presentaremos un sistema confluente de tipos intersección no idempotentes para el cálculo** λ. Lograremos esto escribiendo las derivaciones de tipos usando una sintaxis concisa de términos de prueba. El sistema gozará de buenas propiedades: progreso, será fuertemente normalizante, y tendrá una teoría de residuos muy regular. Estableceremos una correspondencia con el cálculo lambda mediante teoremas de simulación.

Además, los espacios de derivación de los términos del sistema propuesto tendrán una estructura muy simple. Esa simplicidad junto con los teoremas de simulación nos ayudará a estudiar un fragmento de interés de los espacios de derivación de los términos del cálculo λ con mucha facilidad.

La maquinaria de los tipos intersección no idempotentes nos permite seguir el rastro del uso de los recursos necesarios para obtener una respuesta. En particular, induce una noción de *basura*: un cómputo es basura si no contribuye a hallar una respuesta. Usando estas nociones, mostraremos que el espacio de derivaciones de un término λ puede ser factorizado usando una variante de la construcción de Grothendieck para semi-reticulados. Esto significa, en particular, que cualquier derivación del cálculo λ puede ser escrita de una única manera como un prefijo libre de basura, seguido de basura.

Palabras clave: Cálculo lambda, Tipos intersección, Espacio de derivación, Reticulado

Referencias

[AL13] Andrea Asperti and Jean-Jacques Lévy. The cost of usage in the lambda-calculus. In 28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013, pages 293–300, 2013. 1

[BKDR14] Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. In IFIP International Conference on Theoretical Computer Science, pages 341–354. Springer, 2014. 2

[BKV17] Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Non-idempotent intersection types for the lambda-calculus. Logic Journal of the IGPL, 25(4):431–464, 2017. 2

[BL13] Alexis Bernadet and Stéphane Jean Lengrand. Non-idempotent intersection types and strong normalisation. arXiv preprint arXiv:1310.1622, 2013. 2

[BMPR16] Flavien Breuvart, Giulio Manzonetto, Andrew Polonsky, and Domenico Ruoppolo. New results on morris's observational theory: The benefits of separating the inseparable. In 1st International Conference on Formal Structures for Computation and Deduction (FSCD), 2016. 1

[Bou93] Gérard Boudol. The lambda-calculus with multiplicities. In CONCUR'93, pages 1–6. Springer, 1993. 2

[Car07] Daniel de Carvalho. Sémantiques de la logique linéaire et temps de calcul. PhD thesis, Ecole Doctorale Physique et Sciences de la Matière (Marseille), 2007. 2

[CD78] Mario Coppo and Mariangiola Dezani-Ciancaglini. A new type assignment for lambda-terms. Arch. Math. Log., 19(1):139–156, 1978. 2

[DJK91] Nachum Dershowitz, Jean-Pierre Jouannaud, and Jan Willem Klop. Open problems in rewriting. In International Conference on Rewriting Techniques and Applications, pages 445–456. Springer, 1991. 2

[Ehr12] Thomas Ehrhard. Collapsing non-idempotent intersection types. In LIPIcs-Leibniz International Proceedings in Informatics, volume 16. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012. 2

[ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. Theoretical Computer Science, 309(1):1–41, 2003. 2

[Gar94] Philippa Gardner. Discovering needed reductions using type theory. In Theoretical Aspects of Computer Software, pages 555–574. Springer, 1994. 2

[Gir87] Jean-Yves Girard. Linear logic. Theoretical computer science, 50(1):1–101, 1987. 2

[JHM69] Jr James Hiram Morris. Lambda-Calculus Models of Programming Languages. PhD thesis, Massachusetts Institute of Technology, 1969. 1

[Kes16] Delia Kesner. Reasoning about call-by-need by means of types. In International Conference on Foundations of Software Science and Computation Structures, pages 424–441. Springer, 2016. 2

[KL07] Delia Kesner and Stéphane Lengrand. Resource operators for λ-calculus. Information and Computation, 205(4):419–473, 2007. 2

[KR] Delia Kesner and Fabien Renaud. The prismoid of resources. Springer. 2

[KRV18] Delia Kesner, Alejandro Ríos, and Andrés Viso. Call-by-need, neededness and all that. In 21st International Conference on Foundations of Software Science and Computation Structures (FoSSaCS), 2018. 2

[Lan94] Cosimo Laneve. Distributive evaluations of λ-calculus. Fundamenta Informaticae, 20(4):333–352, 1994. 1

[Lév78] Jean-Jacques Lévy. Réductions correctes et optimales dans le lambda-calcul. PhD thesis, Université de Paris 7, 1978. 1

[Lev15] Jean-Jacques Levy. Redexes are stable in the λ-calculus. 27:1–13, 07 2015. 1

[Mel96] Paul-André Melliès. Description Abstraite des Systèmes de Réécriture. PhD thesis, Université Paris 7, december 1996. 1

[Via17] Pierre Vial. Non-Idempotent Typing Operators, Beyond the Lambda-Calculus. PhD thesis, Université Paris 7, december 2017. 2

[Zil84] Marisa Venturini Zilli. Reduction graphs in the lambda calculus. Theor. Comput. Sci., 29:251–275, 1984. 1