UNIVERSIDAD NACIONAL DE QUILMES

ANTEPROYECTO

Implementación de lambda cálculo para matrices de densidad con controles clásicos y probabilisiticos

---

Autor:
Alan RODAS BONJOUR

Director:
Dr. Alejandro DÍAZ-CARO
Co-Director:
Dr. Pablo E. MARTÍNEZ LÓPEZ

Licenciatura en Informática
Departmento de Ciencia y Tecnología

20 de noviembre de 2017

1. Motivación

La computación cuántica es un área de investigación prolífera de las ciencias de la computación en los últimos años. Grandes avances en este tópico podrían llevar a cambios significativos en el paradigma computacional actual, permitiendo resolver problemas que, en la práctica, no son posibles con la computación clásica.

Una forma de catalogar los lenguajes de programación para computación cuántica, es por cómo se presenta el control del programa. De esta manera, existen dos paradigmas reconocidos: control clásico con datos cuánticos, o el control y datos cuántico.

El primero, propuesto por Selinger (2004), propone realizar los cálculos cuánticos en un dispositivo acoplado a una computadora clásica, empleando el procesador cuántico como un co-procesador a demanda. Una extensión al lambda cálculo para dicho paradigma fue desarrollada por Selinger y Valiron (2006), y se transformó en la base de Quipper (Green y col., 2013), un lenguaje funcional, embebido en Haskell y escalable para la computación cuántica. La segunda alternativa intenta buscar una definición computacional de las nociones de espacio vectorial y funciones bilineales. Varios lenguajes de alto nivel con control cuántico han sido propuestos en el pasado (Altenkirch y Grattage, 2005; Ying, Yu y Feng, 2012; Ying, Yu y Feng, 2014; Bădescu y Panangaden, 2015), pero no existe ningún lambda cálculo completo para este paradigma.

Otra manera de catalogar los lenguajes de programación para computación cuántica podría ser por la forma de representar los datos cuánticos. Usando el formalismo de vectores en un espacio de Hilbert, ej. Tonder (2004), Selinger y Valiron (2006), Arrighi, Díaz-Caro y Valiron (2017) y Arrighi y Dowek (2017), o usando el formalismo de las matrices de densidad. Este último no solo permite representar estados cuánticos, sino que además permite expresar sistemas cuánticos en donde el estado no es completamente conocido, es decir, estados mixtos. Este formalismo ha sido utilizado ampliamente para expresar lenguajes de alto nivel (D'Hondt y Panangaden, 2006; Ying, 2011; Feng, Duan y Ying, 2011; Feng, Yu y Ying, 2013; Ying, Ying y Wu, 2017), incluyendo el libro *"Foundations of Quantum Programming"* (Ying, 2016) que está presentado íntegramente con matrices de densidad.

El lenguaje λ_ρ, propuesto en el trabajo de Díaz-Caro (2017) es la primer extensión al lambda cálculo para computación cuántica que utiliza dichas matrices de densidad. Díaz-Caro propone este lenguaje en el paradigma de control clásico y datos cuánticos, pero utilizando matrices de densidad en lugar de vectores para representar los datos. A posteriori, realiza una modificación de dicho lenguaje, λ_ρ°, en el que se generalizan las matrices de densidad a programas. Esta segunda aproximación, si bien no llega a convertir el lenguaje en un lenguaje con control cuántico, tiene un control que tampoco es clásico, y el autor llama *control probabilista*, o *control cuántico débil*.

En este trabajo se propone implementar intérpretes de los lenguajes λ_ρ y λ_ρ°. Dichas implementaciones permitirá evaluar la factibilidad de los formalismos de dichos lenguajes, así como realizar ajustes a los mismos en caso de ser necesario. Más aún, dichas implementaciones pueden ser utilizadas para el análisis de algoritmos cuánticos en estos lenguajes, así como extenderse a posteriori para ejecutar sobre computadoras cuánticas reales. Ciertamente, una vez obtenido los intérpretes, se puede dejar para un trabajo futuro realizar un compilador de dicho lenguaje, ya definido e implementado, a alguna de las máquinas cuánticas en la nube que existen como la máquina de IBM o la de Microsoft.

2. Objetivos

2.1. Objetivo general

El objetivo de este trabajo final de carrera es obtener una versión ejecutable de los lenguajes λ_ρ y λ_ρ°, como una manera de demostrar la factibilidad de realizar lenguajes de alto nivel a partir de estas extensiones teóricas del cálculo lambda. Si bien ya existen otros lenguajes de alto nivel que utilizan matrices de densidad, como se mencionó en la sección precedente, éste será el primero en desarrollarse a partir de un estudio de sus fundamentos teóricos en cálculo lambda.

2.2. Objetivo específico

El objetivo específico es construir los dos intérpretes mencionados, lo cual es un desafío técnico importante ya que el lenguaje tiene elementos de lógica lineal, manejo de matrices, computación probabilística y demás, que por sí mismo, cada una de esas características demanda una ingeniería de software particular, que deberá ser combinada de manera eficiente y elegante.

3. Metodología

3.1. Metodología de trabajo

El proceso de desarrollo se hará de forma iterativa e incremental, logrando durante el proceso distintos ejecutables que aumentarán en funcionalidad a medida que el proyecto avanza. Si bien el proyecto es indivual, diversas caracteristicas de las metodologías ágiles de desarrollo (usualmente pensadas para trabajo en grupo) serán utilizadas.

Particularmente se recurrirá a:

1. **Haskell** como lenguaje de desarrollo.
2. Utilización de **TDD** como proceso de desarrollo
3. Utilización de **git** como sistema de manejo de control de versiones
4. Utilización de **Metodologías ágiles** en modo individual para la gestión del proyecto

3.2. Plan de trabajo

Se propone la siguiente separación tentativa de etapas:

1. Milestone 1: Implementación de lambda cálculo simplemente tipado
   a) Implementación de un Lexer
   b) Implementación de un Parser
   c) Pruebas sobre Lexer y Parser
   d) Implementación de un verificador de tipos
   e) Pruebas sobre verificador de tipos
   f) Implementación de un intérprete
   g) Pruebas sobre intérprete
2. Milestone 2: Extensión a λ_ρ
   a) Implementación de matrices de densidad
   b) Implementación de modelos numéricos para cuántica
   c) Extensión de Lexer y Parser para λ_ρ
   d) Extensión de sistema de tipos para λ_ρ
   e) Extensión de intérprete para λ_ρ
   f) Pruebas sobre λ_ρ
3. Milestone 3: Extensión a λ_ρ°
   a) Extensión de sistema de tipos para λ_ρ°
   b) Extensión de intérprete para λ_ρ°
   c) Pruebas sobre λ_ρ°

3.3. Locación y tiempos de desarrollo

El trabajo será desarrollado en las instalaciones de la Universidad Nacional de Quilmes, y demandará un tiempo de desarrollo de entre 4 y 6 meses.

3.4. Licencia

El código será licenciado bajo la licencia una licencia libre avalada por la OSI (Open Source Initiative) y la FSF (Free Software Foundation). El código fuente será subido de forma pública a un repositorio de código online para el libre acceso de todo aquel que desee utilizar, estudiar, modificar, copiar, redistribuir, mezclar, publicar o sublicenciar el software en cuestión.

4. Justificacion

El desarrollo de este trabajo permitirá evaluar la viabilidad de λ_ρ y λ_ρ° como lenguajes, así como detectar posibles errores en su definición formal. Además, abrirá la posibilidad al uso de este cálculo lambda como herramienta para el desarrollo de algoritmos y análisis de programas cuánticos.

Por otro lado, el presente proyecto permitirá al alumno Alan Rodas Bonjour aplicar los conocimientos obtenidos durante su formación en una temática de gran interés tecnológico. Para desarrollar el mismo, el alumno deberá aplicar conocimientos de Programación Funcional, Ingeniería de Software, Parseo y Generación de Código, Características de Lenguajes, Lógica y Matemática, entre otros. Además el alumno deberá investigar y aprender nuevos conocimientos relacionados a la computación cuántica, permitiendole desarrollar nuevas herramientas y técnicas en el proceso.

Se sugiere al Dr. Alejandro Díaz-Caro (UNQ/CONICET) como director del presente trabajo, pues su principal área de investigación es la lógica y el desarrollo de lenguajes cuánticos. Además, el presente se basa en lo desarrollado en su investigación (Díaz-Caro, 2017).

Adicionalmente se sugiere al Dr. Pablo Ernesto Martínez López (UNQ) como codirector. El mismo se desempeña en investigación de lenguajes funcionales, y ha realizado numerosos desarrollos e investigaciones en Haskell, lenguaje que será utilizado para el desarrollo de este trabajo.

Cabe destacar que el presente trabajo se enmarca dentro de los objetivos de los siguiente proyectos:

- Proyecto UNQ 1370/17 *Fundamentos de lenguajes de programación cuánticos y sus consecuencias en sistemas clásicos* dirigido por el Dr. Díaz-Caro y codirigido por el Dr. Martínez López.
- Proyecto PICT 2015-1208 *Fundamentos de lenguajes de programación cuántica: hacia una lógica computacional* dirigido por el Dr. Díaz-Caro.
- Proyecto 16STIC05 *FoQCoSS: Foundations of Quantum Computation: Syntax and Semantics*, en colaboración varias instituciones francesas y brasileras, dirigido en Argentina por el Dr. Díaz-Caro.

Referencias

Altenkirch, Thorsten y Jonathan J. Grattage (2005). «A functional quantum programming language». En: págs. 249-258.

Arrighi, Pablo, Alejandro Díaz-Caro y Benoît Valiron (2017). «The vectorial λ-calculus». En: *Information and Computation* 254.1, págs. 105-139.

Arrighi, Pablo y Gilles Dowek (2017). «Lineal: A linear-algebraic lambda-calculus». En: *Logical Methods in Computer Science* 13.1:8. DOI: 10.23638/LMCS-13(1:8)2017.

Bădescu, Costin y Prakash Panangaden (2015). «Quantum Alternation: Prospects and Problems». En: *Proceedings of QPL-2015*. Ed. por Chris Heunen, Peter Selinger y Jamie Vicary. Vol. 195. Electronic Proceedings in Theoretical Computer Science, págs. 33-42.

D'Hondt, Ellie y Prakash Panangaden (2006). «Quantum weakest preconditions». En: *Mathematical Structures in Computer Science* 16 (3), págs. 429-451.

Díaz-Caro, Alejandro (2017). «A lambda calculus for density matrices with classical and probabilistic controls». En:

Feng, Yuan, Runyao Duan y Mingsheng Ying (2011). «Bisimulation for quantum processes». En: *ACM SIGPLAN Notices (POPL'11)* 46.1, págs. 523-534.

Feng, Yuan, Nengkun Yu y Mingsheng Ying (2013). «Model checking quantum Markov chains». En: *Journal of Computer and System Sciences* 79.7, págs. 1181-1198.

Green, Alexander S. y col. (2013). «Quipper: a scalable quantum programming language». En: *ACM SIGPLAN Notices (PLDI'13)* 48.6, págs. 333-342.

Selinger, Peter (2004). «Towards a quantum programming language». En: *Mathematical Structures in Computer Science* 14.4, págs. 527-586. DOI: 10.1017/S0960129504004256.

Selinger, Peter y Benoît Valiron (2006). «A lambda calculus for quantum computation with classical control». En: *Mathematical Structures in Computer Science* 16.3, págs. 527-552. DOI: 10.1017/S0960129506005238.

Tonder, André van (2004). «A Lambda Calculus for Quantum Computation». En: *SIAM Journal on Computing* 33, págs. 1109-1135. DOI: 10.1137/S0097539703432165.

Ying, Mingsheng (2011). «Floyd–hoare logic for quantum programs». En: *ACM Transactions on Programming Languages and Systems* 33.6, 19:1-19:49.

– (2016). *Foundations of Quantum Programming*. Elsevier.

Ying, Mingsheng, Shenggang Ying y Xiaodi Wu (2017). «Invariants of quantum programs: characterisations and generation». En: *ACM SIGPLAN Notices (POPL'17)* 52.1, págs. 818-832.

Ying, Mingsheng, Nengkun Yu y Yuan Feng (2012). «Defining Quantum Control Flow». En:

– (2014). «Alternation in quantum programming: from superposition of data to superposition of programs». En: