Herramientas para la exploración y formalización de modelos de programas probabilísticos

Carlos J. Gonzalía

Lab. de Investigación y Desarrollo en Ing. de Software y Sist. de Información / Dpto. de Cs. e Ing. de la Computación / Univ. Nac. del Sur
Av. Alem 1253, Bahía Blanca, Argentina / Teléfono: (54)(291)4595135
cjg@cs.uns.edu.ar

Resumen

El objetivo de nuestra investigación es la construcción de herramientas que permitan explorar las propiedades tanto formales como intuitivas de programas probabilísticos. Los programas que tenemos en mente son aquellos expresables en pGCL, una extensión del lenguaje imperativo de "comandos con guardas" de Dijkstra. El significado y conducta de tales programas probabilísticos es bastante difícil de establecer para una persona no experta en el tema, incluso a la hora de entender una simple traza completa de los mismos. Es clara entonces la necesidad de crear herramientas de software que ayuden tanto en la parte formal (demostraciones formalizadas de propiedades de programas) como intuitiva (exploración de modelos) de estos programas. La aplicación de dichas herramientas sería de utilidad tanto en el área de métodos formales como en la de lenguajes de programación modernos. En particular, nos interesa integrar todas las herramientas a desarrollar alrededor del lenguaje de programación funcional Haskell, de modo de aprovechar la riqueza de conexiones con herramientas de formalización (como ser Agda), como así otras librerías y herramientas de desarrollo para el programador de Haskell.

Palabras clave: *programas probabilísticos, herramientas de software, métodos formales, verificación de programas, programación funcional.*

Contexto

El tema a investigar y el autor del mismo son parte del proyecto de investigación "Integración de Información y Servicios en la Web", código 24/N027, de la Universidad Nacional del Sur, Bahía Blanca, Argentina, y dirigido por el Dr. Pablo Fillottrani de la misma, siendo la financiación también de la UNS. Las etapas preliminares de la investigación en el tema se originan en un proyecto previo (PICT 2009-100) que está llegando a su fin, parte de la relocalización del autor desde el exterior a la UNS, comprendido en el FONCyT y financiado por la Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT) del gobierno nacional.

Introducción

Bajo el término programación probabilística se suele indicar el uso de lenguajes de programación que contienen construcciones para expresar la elección de diferentes resultados de acuerdo a una distribución de probabilidad descripta en tal construcción. En principio el lenguaje puede pertenecer a cualquier paradigma de programación, pero nuestro interés se centrará en programas imperativos y/o funcionales que involucren tales construcciones probabilísticas.

En particular estamos interesados en el lenguaje formal pGCL [12], el cual es una extensión del clásico lenguaje imperativo con elección nodeterminística de Dijkstra [3] para incluír elección probabilística. Como se mencionó en el resumen, es en general bastante difícil para alguien interesado en aplicar pGCL o alguna de sus variantes o fragmentos el entender la conducta de un programa probabilístico, y esta dificultad es aún mayor si se desea tener una seguridad formal en cuanto a las propiedades de tal programa cuando se desea verificarlo.

El uso de herramientas de software que asistan al desarrollador o formalizador resulta claramente deseable, entonces. Por desgracia, pocas de tales herramientas existen en la actualidad, en particular herramientas que se puedan aplicar directamente a programas escritos en la notación de pGCL.

En cuanto a la verificación con asistencia de computadora de propiedades formales de programas probabilísticos, la tarea de mayor interés (y al mismo tiempo la de mayor dificultad de las planteadas en este tema) es la construcción de una librería de propiedades formales en un asistente interactivo de demostración ("logical framework"), en particular uno basado en lógica constructivista (en la cual las formalizaciones tienen contenido computacional de aplicación práctica en la programación) como Agda [1].

Por otro lado, es deseable tener herramientas que permitan también explorar en forma más informal e intuitiva los comportamientos y propiedades de los programas probabilísticos que escribimos. Esto involucra usualmente la simulación o traza de tales programas, para poder explorar las distribuciones probabilísticas sobre sus estados (finales o intermedios), que es la esencia del comportamiento de tales programas [13].

Estas herramientas que podemos caracterizar como de exploración de modelos tienen similitudes con los verificadores de modelos [9] ("model checkers" – de hecho el autor ha colaborado con éxito en trabajos anteriores creando herramientas de exploración de modelos que permitían verificar refinamientos entre programas probabilísticos [7]), pero estarían más bien orientadas a la observación incremental de un programa probabilístico en ejecución. Teniendo en cuenta eso, esperamos también posibles aplicaciones pedagógicas de nuestras futuras herramientas, además de las aplicaciones originales de asistencia al programador de lenguajes similares a pGCL.

Los lenguajes de programación funcionales se prestan especialmente bien para la construcción de herramientas que deben manipular una buena cantidad de representaciones simbólicas complejas, como es en nuestro caso. Por tal razón (y también por experiencia previa) centraremos el esfuerzo de implementación alrededor del lenguaje Haskell [11], sus librerías y sus herramientas ya existentes para problemas relacionados con los que nos interesan. En particular, Haskell es un buen puente de comunicación con el asistente de

demostración Agda, en el cual esperamos poder formalizar librerías de prueba formal que sean útiles para el tema de esta investigación.

Un aspecto de mucho interés que se destaca al usar Haskell es la posibilidad de construír uno o más DSELs ("domain-specific embedded language") [8] que expresen la programación probabilística en alguna variante de pGCL. La conexión rica y exitosa entre la programación funcional y la probabilística ya fue establecida por otros autores [6], en particular la observación central de que las distribuciones de probabilidad forman una mónada [5]. A partir de la misma, por medio de transformaciones, se obtienen otras mónadas para comportamientos adicionales [10] de dichas distribuciones. Si bien la notación resultante, aún con la abundancia de "syntactic sugar" disponible en Haskell, puede resultar bastante oscura al no especialista, recientemente han habido propuestas interesantes de usar nuevas notaciones en teoría de la probabilidad [14], y dichas notaciones tienen mucho en común con las comprensiones de listas de Haskell. Un DSEL probabilístico dentro de Haskell podría interactuar fácilmente con otras herramientas a desarrollar o preexistentes, y sobre todo podría ayudar a realizar grandes cantidades de (pre)procesamiento de representaciones simbólicas de programas y sus estados para tales herramientas externas.

Líneas de Investigación, Desarrollo e Innovación

Podemos sintetizar el tema a investigar de la siguiente manera:

- mejorar y adaptar librerías de pruebas formales para abarcar programas probabilísticos y sus propiedades de interés.
- mejorar prototipos de herramientas de software propias para ser más efectivas en la exploración de modelos de programas probabilísticos.
- implementar en forma de un DSEL dentro de un lenguaje funcional moderno alguna variante del lenguaje formal pGCL que utilice las nuevas notaciones propuestas recientemente en teoría de la probabilidad.

Resultados y Objetivos

El tema a investigar como tal recién comienza a ser abordado. Como base de inicio y resultado experimental previo, el autor ha implementado una herramienta para verificar refinamientos o encontrar contraejemplos a los mismos interactuando con una herramienta externa de resolución de problemas de satisfacibilidad [4]. Algunos experimentos internos modificando un DSEL probabilístico en Haskell creado por otros investigadores ha servido para ganar confianza en la decisión de usar el concepto de un nuevo DSEL probabilístico en Haskell como columna vertebral del esfuerzo de implementación de las futuras herramientas.

Los objetivos futuros probablemente abarquen dos años estimados adicionales de investigación:

- verificar la ausencia de problemas en el uso de mónadas que son a la vez probabilísticas, nodeterminstas, y acarrean estados complejos, en particular en la forma en que existen como parte de las

versiones actuales de Haskell y sus librerías [15] (meses 1 a 3)

- implementar un DSEL basado en dichas mónadas y usando una sintaxis similar a las comprensiones de listas para expresar conceptos probabilísticos (meses 4 a 9)
- implementar componentes de Haskell que permitan la interacción de nuestro DSEL con herramientas futuras o preexistentes (meses 10 a 12)
- implementar nuevas y mejores versiones de las herramientas prototipo para refinamientos de programas probabilísticos en pGCL (meses 13 a 17)
- adaptar(posiblemente usando [2]), mejorar (o crear, si es necesario) librerías de pruebas formales en Agda para realizar verificaciones de propiedades de interés de programas probabilísticos (meses 18 a 24)

Formación de Recursos Humanos

En este momento el tema de investigación es llevado a cabo sólo por el autor. Se espera que el dictado de asignaturas y cursos de métodos formales durante los próximos meses por parte del autor (tales asignaturas y cursos prácticamente han sido inexistentes en la institución del autor hasta la fecha) permita formar y captar estudiantes de grado y posgrado interesados en sumar su esfuerzo a la implementación de herramientas para programas probabilísticos. La riqueza del tema y sus varias facetas ya descriptas brinda abundancia de oportunidades para interesados en realizar tesis de grado o posgrado, o proyectos de implementación, basados en el tema.

Referencias

1. A. Bove, P. Dybjer, U. Norell: A Brief Overview of Agda – A Functional Language with Dependent Types. *Theorem Proving in High Order Logics* (22nd International Conference, TPHOLs 2009), Lect. Notes in Comp. Sci., Vol. 5674, pp. 73-78, Springer, 2009.

2. D. Cock: Verifying Probabilistic Correctness in Isabelle with pGCL. *Systems Software Verification Conference 2012*, EPTCS 102, pp. 167-178.

3. E. Dijkstra: Guarded commands, nondeterminacy and formal derivation of programs. Communications of the ACM, Vol. 18, No. 8, pp. 453-457.

4. B. Dutertre y L. de Moura: A Fast Linear-Arithmetic Solver for DPLL(T)*. *CAV'06,* Lecture Notes in Computer Science 4144, pp. 81-94, Springer, 2006.

5. M. Erwig y S. Kollmansberger: Probabilistic Functional Programming in Haskell. *Journal of Functional Programming*, Vol. 16, No. 1, pp. 21-34, 2006.

6. M. Giry: A Categorical Approach to Probability Theory. *Categorical Aspects of Topology and Analysis.* Lecture Notes in Mathematics, vol. 915, pp. 68-85, 1982

7. C. Gonzalía: An experimental tool for checking probabilistic program refinement. IX Workshop de Ingeniería de Software, CACIC 2012. *Anales del XVIII Congreso Argentino de Ciencias de la Computación*, pp. 927-936, 2012.

8. P. Hudak: Modular Domain Specific Languages and Tools. *Proceedings 5th International Conference on Software Reuse*, pp. 134-142, IEEE Computer Society Press, 1998.

9. M. Kwiatowska, G. Norman, D. Parker: PRISM 4.0 – Verification of Probabilistic Real-time Systems. *Computer Aided Verification* (23rd International Conference, CAV'11), Lect. Notes in Comp. Sci., Vol. 6806, pp. 585-591, Springer, 2011.

10. S. Liang, P. Hudak, M. Jones: Monad transformers and modular interpreters. Proceedings of POPL'95 (22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages), pp. 333-343, ACM Press, 1995.

11. S. Marlow (editor): *Haskell 2010 – Language Report*. Disponible en http://www.haskell.org

12. A. McIver, C. Morgan: *Abstraction, Refinement and Proof for Probabilistic Systems.* Monographs in Computer Science, Springer, 2005.

13. A. McIver, C. Morgan and C. Gonzalía: Proofs and refutations for probabilistic systems. *FM 2008: Formal Methods* (15th International Symposium on Formal Methods), Lect. Notes in Comp. Sci., Vol. 5014, pp. 100-115, Springer, 2008.

14. C. Morgan: Elementary Probability Theory in the Eindhoven Style. *Mathematics of Program Construction* (11th International Conference, MPC 2012), Lect. Notes in Comp. Sci, Vol. 7342, pp. 48-73, Springer, 2012.

15. P. Wadler: Monads for functional programming. *Advanced Functional Programming - Tutorial Text* (1st International School on Advanced Functional Programming Techniques), Lecture Notes in Computer Science, Vol. 925, pp. 24-52, Springer, 1995.