Universidade do Minho
Escola de Engenharia
Departamento de Informática

SIMPÓSIO DOUTORAL 2006

Cálculo e Transformação de Programas na Prática: Ferramentas de Apoio para uma Abordagem Genérica

Miguel Vilaça

9 de Fevereiro de 2006

Conteúdo

1 Identificação **2**

2 Contextualização **3**

3 Objectivos **5**

4 Revisão Bibliográfica **7**

5 Produção Científica **8**

6 Contribuições e Publicações **9**

7 Trabalho Futuro **11**

Capítulo 1

Identificação

Autor: Miguel Vilaça jmvilaca@di.uminho.pt

Título: Cálculo e Transformação de Programas na Prática: Ferramentas de Apoio para uma Abordagem Genérica

Orientador: Jorge Sousa Pinto, Universidade do Minho, Departamento de Informática, jsp@di.uminho.pt

Início: 1 de Dezembro de 2004

Término Previsto: 30 de Novembro de 2008

Capítulo 2

Contextualização

É um facto bem aceite que a utilização de uma linguagem de programação funcional moderna facilita as tarefas de raciocínio sobre os programas. Esta adequação deve-se à existência de caracterizações formais de todos os aspectos do comportamento destas linguagens (a sua semântica).

Quando as propriedades que se pretende provar são igualdades (extensionais, i.e. dois programas calculam os mesmos resultados para os mesmos argumentos) entre programas, um método possível é o que se baseia no *Cálculo de Programas* [1, 3]. Este método assenta sobre dois ingredientes :

- a utilização de *padrões de recursividade*, que permite escrever programas sem a utilização explícita de recursividade;
- a teoria algébrica dos tipos de dados, que permite a utilização de versões genéricas (i.e., em que os tipos de dados são abstraídos) dos ditos padrões.

Os padrões de recursividade genéricos dispõem de leis de cálculo [5] cuja utilização dispensa o recurso a métodos indutivos (com a vantagem de colocar num mesmo quadro a prova sobre programas recursivos e co-recursivos). A utilização destes padrões permite, além disso, provar propriedades sobre programas genéricos.

O cálculo de programas aparece frequentemente ligado a um estilo particular de programação *sem variáveis*, em que os programas são expressos com recurso a um conjunto de combinadores básicos (de inspiração categorial); a expressão "código sem variáveis"(*pointfree*) refere-se normalmente a programas escritos de acordo com todos os princípios enunciados.

Neste contexto, encontram-se neste momento em desenvolvimento no âmbito do projecto PURe:

- uma ferramenta para conversão de código Haskell com variáveis em código sem variáveis;
- uma biblioteca de apoio à programação sem variáveis;
- uma ferramenta de assistência à prova equacional baseada no cálculo funcional sem variáveis.

O cálculo funcional é no entanto muito mais do que uma mera ferramenta para provas de igualdade entre programas; tem também aplicação:

- na *derivação* de programas a partir de especificações;
- na *transformação* de programas (obtenção de programas optimizados a partir de programas equivalentes de concepção mais óbvia); nomeadamente [4] estuda a transformação genérica de programas baseada na aplicação da *lei de fusão* de certos padrões de recursividade, utilizando para isso um cálculo com variáveis.
- na *desflorestação* de programas (optimização por eliminação de estruturas de dados intermédias);
- no estudo de algoritmos: sua compreensão e desenho

Capítulo 3

Objectivos

Embora o cálculo propriamente dito tenha sido estudado e desenvolvido em muitos trabalhos recentes, a sua aplicação em qualquer das vertentes acima referidas não tem acompanhado estes desenvolvimentos: as ferramentas baseadas no cálculo são escassas, e muitos dos aspectos teóricos envolvidos na aplicação do cálculo devem ainda ser esclarecidos.

O presente projecto pretende continuar o desenvolvimento da *suite* de aplicações acima referida; o objectivo principal do projecto é o desenvolvimento de uma plataforma completa para o desenho e a transformação de programas baseada na visão genérica dos programas funcionais, incluindo naturalmente o tratamento dos problemas teóricos subjacentes. O trabalho centrar-se-á em duas componentes complementares:

1. Identificação e estudo aprofundado das classes de transformações que se pretende implementar, e estratégias concretas para a sua implementação com o cálculo funcional. É de esperar, por exemplo, que a introdução de acumuladores ou a utilização da técnica de *tupling* correspondam a estratégias genéricas, independentes dos tipos de dados envolvidos (dado o carácter genérico do próprio cálculo). Trata-se pois aqui de estudar estratégias de aplicação do cálculo para actividades específicas.

2. Estudo e formalização dos vários componentes da plataforma de transformação:

   - um motor básico de prova equacional baseado nas regras do cálculo funcional; deverá permitir lidar de forma simples com a carga burocrática frequentemente introduzida pela eliminação de variáveis;
   - introdução de variáveis funcionais e unificação no cálculo;
   - gestão de um conjunto de objectivos de prova e interacção com o utilizador;
   - gestão de estratégias de transformação; aplicação semi-automática destas estratégias.

Listam-se em seguida alguns tópicos adicionais que poderão ser alvo de investigação:

- cálculo com hilomorfismos;
- concepção de uma ferramenta de desenho visual de código sem variáveis;
- transformação visual de programas por interacção entre a ferramenta de desenho e a plataforma de transformação;
- possibilidade de se comutar *online* entre modos "com variáveis"e "sem variáveis", facilitando-se assim a interacção com o utilizador, bem como certas tarefas de cálculo;
- acomodação do *cálculo relacional*;

Capítulo 4

Revisão Bibliográfica

Logo no início do projecto de doutoramento surgiu a possibilidade de se versar, por um período de cerca de um ano, pela criação, desenvolvimento e implementação de uma linguagem visual para o paradigma funcional.

Após a pesquisa e leitura de material sobre o estado actual da área concluiu-se que pouco trabalho de investigação havia na área, embora fosse referido em muitos artigos e grupos científicos de discussão as vantagens que todos expectavam de uma linguagem funcional que substituísse o tradicional suporte textual por um novo suporte gráfico. Contudo, e apesar dessa receptividade pela comunidade académica, poucos são os trabalhos existentes e nenhum deles conseguiu atingir um nível suficientemente desenvolvido para ser considerado uma linguagem de programação a utilizar massivamente. Dos trabalhos existentes destacam-se [10, 6, 7].

Posteriormente as leituras incidiram sobre qualquer formalismo, notação ou ferramenta gráficos existentes noutros paradigmas, leitura esta que era acompanhada de uma análise crítica das notações e características gráficas que poderiam ser portadas para o paradigma funcional. Entre estas linguagens foram analisadas a ProGraph, Show-and-Tell, in^2, Groove, AgentSheet, BDL, VisiTile e VFPE entre outras.

Simultaneamente foram estudados alguns artigos, de vertente mais teórica, sobre Dataflow Programming e sobre Control-flow Programming. Daqui surge a descoberta que a linguagem que se pretende criar não sendo uma linguagem de dataflow terá algumas das características das linguagens de tal paradigma.

A determinada altura surge a necessidade de aprofundar conhecimentos nas áreas de reescrita de termos, transformação de grafos, Interaction Nets e reescrita de grafos para termos, tendo sido lida bibliografia da área.

Capítulo 5

Produção Científica

Paralelamente às leituras vai sendo desenvolvido um processo criativo de desenvolvimento de notações gráficas para a linguagem pretendida; algumas notações rapidamente se mostram inadequadas e desaparecem enquanto outras vão evoluindo, sendo que algumas ainda persistem actualmente.

À medida que se tem conjuntos de notações torna-se necessário avaliar a sua qualidade científica, isto é, e neste caso, a confluência desse sistema. Adquiridos conhecimentos nestas áreas começa o processo lento e delicado de verificar a confluência dos sistemas desenvolvidos e de os alterar e expandir por forma a colmatar as suas falhas; após cada alteração surge a necessidade de refazer todo o trabalho de verificação de confluência agora para o sistema recentemente modificado. E o processo repete-se e continua na actualidade.

De um trabalho conjunto com Jorge Sousa Pinto e Ian Mackie surgiu já uma versão de uma linguagem funcional restrita a suporte para alguns tipos de recursividade , apresentado no LOPSTR - International Symposium on Logic-based Program Synthesis and Transformation).

Actualmente, e após a análise de vários sistemas de transformação de grafos e de desenvolvimento de editores gráficos, estão em curso duas implementações de protótipo do sistema gráfico desenvolvido até agora (ainda que sejam conhecidas algumas lacunas no mesmo) nas tecnologias Haskell com wxHaskell e Eclipse, AGG, Tiger e GEF.

Também se ponderam já meios para a junção da notação para combinadores point-free com a notação para recursividade.

Capítulo 6

Contribuições e Publicações

Janeiro de 2004 *Simpósio doutoral do DI*
http://wiki.di.uminho.pt/wiki/bin/view/Doutoramentos/SDDI2004
Apresentação de um poster sobre transformação de programas funcionais do estilo pointwise para o estilo point-free e rudimentar aproximação a uma notação gráfica. Breve descrição do trabalho de investigação a desenvolver no doutoramento.

Abril de 2005 *Midlands Graduate School*
http://www.cs.bham.ac.uk/ pbl/mgs/
Esta escola versou fundamentalmente sobre áreas teóricas das ciências da computação, tendo adquirido conhecimentos essencialmente nas seguintes áreas:

- Teoria de Categorias
- λ-cálculo com tipos
- Semânticas denotacionais e operacionais

Junho de 2005 *Escola de verão em Program Analisys and Transformation*
http://www.diku.dk/PAT2005/
Neste evento foi possível adquirir conhecimentos sobre várias técnicas de transformação de programas nos paradigmas funcional e lógico, nomeadamente:

- Transformações de programas usando as Tree automatas e gramáticas de atributos
- Especialização de programas
- Redução parcial
- Inversão de programas
- Análise de terminação
- Semântica de reduções e transformações

Setembro de 2005 *LOPSTR International Symposium on Logic-based Synthesis and Transformation*
http://www.cs.man.ac.uk/ kung-kiu/lopstr/
Apresentação na conferência do trabalho conjunto com Jorge Sousa Pinto e Ian Mackie intitulado "Functional Programming and Program Transformation with Interaction Nets", que versa sobre a parte recursiva de uma linguagem funcional num suporte visual. Actualmente aguarda-se a decisão sobre a eventual publicação deste artigo nas Lecture Notes in Computer Science pela Springer-Verlag.

Outubro de 2005 *Workshop do projecto PURe*
Apresentação de uma versão actualizada do trabalho conjunto "Functional Programming and Program Transformation with Interaction Nets"e ainda discussão científica sobre problemas e limitações deste trabalho e possíveis soluções.

desde Dezembro de 2005 *Criação e Desenvolvimento do INblobs*
http://haskell.di.uminho.pt/jmvilaca/INblobs/
INblobs é um editor para Interaction Nets. Actualmente existe apenas um outro editor para Interaction Nets mas é excessivamente limitado em usabilidade. O estado actual deste editor permite desde já prever um enorme contributo para a comunidade científica, especialmente a relacionada com Interaction Nets. O mesmo editor pode também ser facilmente adaptado para suportar outros formalismos semelhantes.

Capítulo 7

Trabalho Futuro

A prossecução do doutoramento passa, no imediato, pela implementação da linguagem funcional para combinadores point-free e sua interpretação no mesmo ambiente usando regras globais. Numa primeira abordagem esta interpretação será orientada pelo utilizador, funcionando o sistema apenas como um verificador das transformações.

Numa componente mais teórica esta linguagem funcional para combinadores point-free será refinada com vista a tornar o sistema de transformações num sistema local e tentar-se-á ainda automatizar tal sistema. Outra área que continuará a receber atenção, pesquisa e investigação é a junção da linguagem com combinadores point-free com a linguagem para a parte recursiva que foi desenvolvida. Conseguidos e formalizados estes refinamentos novo trabalho de implementação com o objectivo de dotar o ambiente de programação com as últimas novidades teóricas serão levados a cabo.

Tanto na componente teórica como de implementação, serão efectuados esforços visando estender a expressividade da linguagem funcional visual. Idealmente a linguagem visual terá a mesma expressividade que as linguagens funcionais textuais actuais, com as vantagens e simplicidade da versão visual.

Bibliografia

[1] Richard Bird and Oege de Moor Algebra of Programming. *Prentice Hall*, 1997

[2] R Bird. The Promotion and Accumulation Strategies in Transformational Programming, *ACM Transactions on Programming Languages and Systems*, **6**(4), October 1984, 487–504.

[3] J. Gibbons. Calculating Functional Programs. In *Proceedings of ISRG/SERG Research Colloquium*. School of Computing and Mathematical Sciences, Oxford Brookes University, 1997.

[4] Oege de Moor and Ganesh Sittampalam. Generic Program Transformation In D. Swierstra and P. Henriques and J. Oliveira, editor, *Proceedings of the 3rd International Summer School on Advanced Functional Programming*, 1999, Springer-Verlag, pages 116–149.

[5] Erik Meijer and Maarten Fokkinga and Ross Paterson. Functional Programming with Bananas, Lenses, Envelopes and Barbed Wire In J. Hughes, editor, *Proceedings 5th ACM Conf. on Functional Programming Languages and Computer Architecture, FPCA'91* , 1991, Springer-Verlag, pages 124–144.

[6] K. Hanna. Interactive Visual Functional Programming. In S. P. Jones, editor, *Proc. Intnl Conf. on Functional Programming*, pages 100–112. ACM, October 2002.

[7] J. Kelso. *A Visual Programming Environment for Functional Languages*. PhD thesis, Murdoch University, 2002.

[8] Y. Lafont. Interaction Nets. In *Proceedings of the 17th ACM Symposium on Principles of Programming Languages (POPL'90)*, pages 95–108. ACM Press, Jan. 1990.

[9] M. Fernández and I. Mackie. Coinductive Techniques for Operational Equivalence of Interaction Nets. In *Proceedings of the 13th IEEE Symposium on Logic in Computer Science (LICS'98)*, pages 321–332. IEEE Computer Society Press, June 1998.

[10] H. J. Reekie. Visual Haskell: A first attempt. Research Report 94.5, Key Centre for Advanced Computing Sciences, University of Technology, Sidney, Aug. 1994.