Relational Programming in miniKanren

Relational Programming in miniKanren

William E. Byrd

Typeset by the author in XeLaTeX, using Tufte-Style Book from http://www.LaTeXTemplates.com.
June 23, 2013 version.

For my H211 students: Indiana University, Fall 2010 & 2011, and Team pw0ni3.

Learning with always trumps learning from.

—Woodie Flowers

Contents

Preface

This book is about writing programs that "run backwards." Such programs represent mathematical relations; we therefore call them *relational programs*, or just *relations*. Relational programming is closely related to relational databases, and can also be seen as an especially pure form of logic programming. For writing our relational programs, we use the *miniKanren* family of languages. miniKanren is designed specifically for relational programming, and makes relational programming easy.

Okay, that last part is a lie. Writing relational programs is . . . tricky, hence this book. miniKanren does make relational programming *less difficult* than it would be otherwise. What *is* true is that writing relational programs is extremely interesting, challenging, and fun. As in, "I can write a short, non-obfuscated program that will melt your brain" level of fun.

The Datalog family of languages is even more closely related to relational databases than the miniKanren language described in this book. Datalog is more expressive than SQL, but less expressive than miniKanren or Prolog. Datalog is sub-Turing complete, and cannot express infinite computations, which is generally a desireable property for a query language. More on Datalog can be found in:

S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about Datalog (and never dared to ask). *IEEE Trans. on Knowl. and Data Eng.*, 1(1): 146–166, Mar. 1989

Audience

This book is written for intermediate-to-advanced programmers, computer science students, and researchers. For this book, *intermediate* means that you are comfortable writing simple recursive procedures in a functional programming language, such as Scheme, Racket, Clojure, Lisp, ML, or Haskell. I also assume you have a reading knowledge of Scheme. No knowledge of relational programming, logic programming, or programming language theory is required.

If you want to learn about relational programming, but are new to programming, Dan Friedman, Oleg Kiselyov, and I have written a book just for you, called *The Reasoned Schemer*[1]. In that book we assume you are familiar with the material in *The Little Schemer*[2], which is a very gentle introduction to recursion and functional programming.

If you are an experienced programmer, but weak on recursion, you, too, might benefit from *The Little Schemer*. If you are comfortable with recursion, but not functional programming, good introductions include *Scheme and the Art of Programming*[3] and the classic *Structure and Interpretation of Computer Programs*[4].

> I have attempted to deliver [these lectures] in a spirit that should be recommended to all students embarking on the writing of their PhD theses: imagine that you are explaining your ideas to your former smart, but ignorant, self, at the beginning of your studies!
>
> —Richard P. Feynman
> *The Feynman Lectures on Computation*

[1] D. P. Friedman, W. E. Byrd, and O. Kiselyov. *The Reasoned Schemer*. MIT Press, Cambridge, MA, 2005

[2] D. P. Friedman and M. Felleisen. *The Little Schemer (4th ed.)*. MIT Press, Cambridge, MA, 1996

[3] G. Springer and D. P. Friedman. *Scheme and the Art of Programming*. MIT Press, Cambridge, MA, 1989

[4] H. Abelson and G. J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, 2nd edition, 1996
(full text at `http://mitpress.mit.edu/sicp/full-text/book/book.html`)

If you are an experienced functional programmer, but do not know Scheme, the beginning of *Structure and Interpretation of Computer Programs* should get you up to speed, while *The Scheme Programming Language, 4th Edition*[5] describes the language in detail.

[5] R. K. Dybvig. *The Scheme Programming Language, 4th Edition*. The MIT Press, 4th edition, 2009
(full text at http://www.scheme.com/tspl4/)

Goals

The high-level goals for this book are to make relational programming accessible to a broader audience, and to explain the state-of-the-art in the design, implementation, and use of the miniKanren language.

Specifically, this book aims to:

1. describe the current state of the miniKanren language, and its Constraint Logic Programming (CLP) extensions;

2. present a variety of relational programs that exemplify the relational style, and that introduce important relational idioms;

3. explain how to derive miniKanren relations from pure Scheme functions in a systematic manner;

4. explain in detail the canonical implementation of core miniKanren, and show how to hack the implementation to change or extend the language;

5. explain how constraint solving works in the cKanren Constraint Logic Programming framework, and show how to design and implement new constraint domains;

6. demonstrate how to debug miniKanren relations;

7. explain how miniKanren differs from related languages, including Prolog, Curry, Mercury, and Oz;

8. describe important open problems in relational programming, and limitations of miniKanren;

9. explain the rationale behind key design decisions for miniKanren;

10. use relational programming to explain important and interesting concepts from programming languages, such as interpreters, type inferencers, and Continuation-Passing Style.

An important side-effect of these goals is to provide the background needed to understand academic papers and talks on miniKanren.

Margin Notes

This book is typeset in the style of Edward Tufte's magnificent and beautiful *The Visual Display of Quantitative Information*[6]. I share Tufte's love of margin notes, and use them in this book to help solve the problem of addressing readers with widely varying knowledge of computer science and programming. To make the book accessible as possible, in the main text I assume the reader is the hypothetical *intermediate-level* programmer or student described in the *Audience* section above. In the margin notes, however, anything goes.

William E. Byrd
Salt Lake City, Utah
June 2013

This book is set using the "Tufte-Style Book" LaTeX style, freely available from http://www.LaTeXTemplates.com

[6] E. R. Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, CT, 1986

Another great lover of marginalia was David Foster Wallace (1962–2008). The *Harry Ransom Center*'s DFW collection includes heavily annotated books from Wallace's personal library: http://www.hrc.utexas.edu/press/releases/2010/dfw/books/. Wallace's love of margin notes is best demonstrated by his essay "Host", in:

D. F. Wallace. *Consider the Lobster and Other Essays*. Little, Brown and Co., 2005

Introduction

gl hf!

(Traditional greeting in the Koprulu Sector)

Conclusion

G.G.

—Sean “Day[9]” Plott

Bibliography

H. Abelson and G. J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, 2nd edition, 1996.

S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about Datalog (and never dared to ask). *IEEE Trans. on Knowl. and Data Eng.*, 1(1):146–166, Mar. 1989.

R. K. Dybvig. *The Scheme Programming Language, 4th Edition*. The MIT Press, 4th edition, 2009.

D. P. Friedman and M. Felleisen. *The Little Schemer (4th ed.)*. MIT Press, Cambridge, MA, 1996.

D. P. Friedman, W. E. Byrd, and O. Kiselyov. *The Reasoned Schemer*. MIT Press, Cambridge, MA, 2005.

G. Springer and D. P. Friedman. *Scheme and the Art of Programming*. MIT Press, Cambridge, MA, 1989.

E. R. Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, CT, 1986.

D. F. Wallace. *Consider the Lobster and Other Essays*. Little, Brown and Co., 2005.