Relational Interpreters in miniKanren
Fascicle I: Preface

William E. Byrd

An Unreasonable Schemer Production

Book 06 of Will's ShovelWare Book Pack
μEdition 1 (Version 1.0)   Sunday, January 18, 2026

For Dan.

About Will's ShovelWare Book Pack

In an experiment to see if I can train myself to finish writing a book on my own—instead of just poking at it for a few weeks before abandoning it and starting another—I have publicly committed to writing and releasing seventeen (17) books in one year, releasing one book every three weeks. I made the original announcement on Will Radio Part XXVIII[1] on September 15, 2025, with the experiment to end on September 6, 2026, when the 17th book is due.

Good or bad, short or long, my goal is to finish and release a book every 21 days, like clockwork.

The idea for this experiment came from my kiloTube Video challenge from a couple years ago. I had been wanting to make more videos, but was suffering from perfectionism—I would record 70 or 80 takes of a video before deleting it in disgust. So I publicly announced that I was going to make 1,024 (or 2^{10}) videos in a single year (one kiloTube worth), in an attempt to force myself to just make video after video, with the idea that I'd eventually learn the mechanics, and learn my own style for making videos. Although I didn't come close to making 1,024 videos that year (partly because making videos involved making noise, which caused some social issues), the practice of making an average of 3 videos a day for weeks quickly got me over my perfectionism block. I now have no trouble sitting down and making a video in one take, editing it, and uploading it in a single sitting.

My hope is that after finishing and releasing a dozen or so books, I'll start to relax out a little, and will be over the fear of releasing something less than perfect. There's a go adage, "Lose your first hundred games as quickly as possible." I'm trying to write and release my first dozen (or so) terrible books as quickly as possible. With 17 books to write, I might write a few decent ones, as well.

I am allowing myself to update books I've already released, so long as I continue releasing a new book every 3 weeks.

This is the sixth of the 17 books, due on Sunday, January 18, 2026, Anywhere on Earth (or whenever I go to sleep!). (Originally I was going to release the books on Monday nights, but Sundays work better.) The next book, book 07, will be due Sunday, February 8, 2026.

[1] `https://www.youtube.com/watch?v=d6_cB-jtxYI`

Contents

Preface

This book is the first in a series—I hope!—of chapter-sized bundles, or "fascicles," for a book on writing interpreters are relations in miniKanren. The idea is to release one fascile every three weeks for the remainder of the ShovelWare Book Pack experiment. This first fascicle includes the draft preface. I plan to release one chapter every three weeks. I should end up with approximately 11 chapters plus the preface by September 15, 2025, when the 17th ShovelWare book is due. I can then either revise the book, continue extending it, move onto another book, or take a break.

I plan to update and edit the already released fascicles as I go, releasing the latest version of the book's "prefix." Basically, I'm releasing the book one chapter at a time, old-school serial fashion.

What this Book is About

This book explores an unusual approach to writing interpreters. An *interpreter* is a computer program that can evaluate an expression, producing a value. This is a very general idea, and an amazing variety of computer programs can be considered interpreters of some form. The most expressive and powerful class of interpreters can evaluate expressions that represent arbitrary computer programs; such interpreters can even evaluate themselves! Because interpreters appear in so many areas of computation and computer science and programming, they are well worth studying and exploring, from both typical and unusual perspectives.

The unusual point of view taken in this book is that interpreters should be viewed as—and implemented as—*relations* rather than as *functions*. The typical perspective is that an interpreter is a function (or machine) that takes as an *input* an expression e (in some language) and produces as an *output* some value *v*. The notions of input and output are central to this view of an interpreter. The view taken in this book is that the distinction between input and output is unnecessary and overly restrictive. Instead, we view an interpreter as *relating* one or more expressions with one or more values.

For the most part we will write interpreters that interpret expressions in simple variants of the Scheme programming language. The interpreters themselves, however, will generally be written in miniKanren, a programming language designing for writing programs as relations. We will use the faster-miniKanren implementation of miniKanren, which is itself written in Scheme. Sometimes we'll do something trickier, such as writing a Scheme interpreter that is interpreted by a different Scheme interpreter that is written in the faster-miniKanren implement of miniKanren that is running in Scheme! This can get confusing, so we'll build up in complexity a little at a time, and draw diagrams and give examples and exercises to help keep everything straight.

The faster-miniKanren implementation of miniKanren we'll be using runs well in both Chez Scheme and in Racket. The book begins with a brief introduction to the faster-miniKanren version of the miniKanren programming language, and instructions on how to run faster-miniKanren in Chez Scheme and in Racket.

Intended Audience and Background Knowledge

This book is intended for anyone interested in exploring a *relational programming* view of interpreters and interpretation.

However, this is *not* a beginning programming book, nor is it a self-contained course in miniKanren, relational programming, Scheme, or interpreters. This book assumes that the reader has basic knowledge of the Scheme programming language, basic functional programming (including pattern matching), and basic knowledge of how to write an interpreter for Scheme, in Scheme. The book also assumes the reader understands basic relational programming in miniKanren—for example, as provided by the second edition of Friedman, Byrd, Kiselyov, and Hemann's *The Reasoned Schemer* (MIT Press, 2018), or from the various miniKanren tutorials and videos available online or in academic papers.

As much as I'd like to include all of the required background information in a single book (and I *have* tried, several times), to adequately cover this material would require presenting the equivalent of an entire series of university courses. It is my dream to eventually write such a series, but I think it makes sense to start with the most unique material first, and then work backwards.

Fortunately, there are many fine books, tutorials, videos, and courses that teach the Scheme programming language, functional programming, pattern matching, program transformations, and interpreters. A few well-known books include: *Structure and Interpretation of Computer Programs, 2nd edition* by Harold Abelson and Gerald Jay Sussman, with Julie Sussman (MIT Press, 1996); *The Scheme Programming Language, 4th edition* by R. Kent Dybvig (MIT Press, 2009); *Essentials of Programming Languages, 3rd edition* by Daniel P. Friedman and Mitchell Wand (MIT Press, 2008); *The Little Schemer, 4th edition* by Daniel P. Friedman and Matthias Felleisen (MIT Press, 1995); *Lisp in Small Pieces* by Christian Queinnec (Cambridge University Press, 2003); and *Types and Programming Languages* by Benjamin C. Pierce (MIT Press, 2002)

Pointers to miniKanren-related resources can be found at `https://minikanren.org/`. The standard introductory book on miniKanren is *The Reasoned Schemer, 2nd edition* by Daniel P. Friedman, William E. Byrd, Oleg Kiselyov, and Jason Hemann (MIT Press, 2018).

Acknowledgments

Marco Marchesin suggested releasing fascicles of a larger book instead of writing one self-contained tiny book each week, which is what I'm now trying. Thank you, Marco!

The first five ShovelWare books I've released, as bad and incomplete as they may be, helped me get in the position mentally to try this approach. So, thank you to those books, and to anyone who subjected themselves to reading them.

A proper acknowledgement of all the work on miniKanren and relational interpreters would fill many pages. As always, there is the danger of leaving people out, or mixing up contributions from years ago. I will write a proper acknowledgements section, although I know from experience it may take multiple fascicles before I dare

show the acknowledgements publicly. For now, let me thank everyone in the miniKanren community, especially my teachers, mentors, collaborators, and students.

Will Byrd
Mt. Pleasant, South Carolina
January, 2026