A Relational View of Computing

A Relational View of Computing

William E. Byrd

Typeset by the author in XeLaTeX. January 6, 2014 version.

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

Learning with always trumps learning from.

—Woodie Flowers

Contents

Preface

> *I think the weakest way to solve a problem is just to solve it; that's what they teach in elementary school. In some math and science courses they often teach you it's better to change the problem. I think it's much better to change the context in which the problem is being stated. Some years ago, Marvin Minsky said, "You don't understand something until you understand it more than one way." I think that what we're going to have to learn is the notion that we have to have multiple points of view.*
>
> —Alan C. Kay[1]

This book is about changing the context in which computational problems are stated, attempting to understand these problems from the viewpoint of relational programming.

In elementary school I learned to multiply small whole numbers by memorization: I memorized the fact that 3 times 4 is 12. I also learned how to handle larger numbers by repeatedly multiplifying pairs of digits—one digit from each number—so that I could apply my mental collection of facts. I solved countless multiplication problems without the faintest notion that I was using ideas fundamental to computing: following an algorithm, lookuping up pre-computed partial results, using recursion to simplify a complex problem. *I learned to just solve the problem.*

Years later I learned that numbers can be represented in binary: the multiplication problem 3×4 can be changed into the equivalent binary multiplication problem $11_2 \times 100_2$. I also learned that multiplying a number by 4 in binary is equivalent to adding two 0's to the end of that number: 11_2

[1]From Kay's 20th anniversary Stanford Computer Forum talk in 1988, "Predicting the Future" (Kay 1989). Also at `http://www.ecotopia.com/webpress/futures.htm`.

multiplied by 4 is 1100_2, which is the binary representation of 12. On many computers, using a "shift left" instruction to add two 0's to the end of a binary number is faster than multiplying the number by 4. *I learned it is often better to change the problem.*

Even later, in college, I learned that multiplication can be described in the context of mathematical relations: the relation defining multiplication is a collection of "triples" of numbers, such as (3 4 12). These triples correspond, of course, to the multiplication facts I learned in elementary school, with one key difference: the multiplication relation includes *all* of the infinitely many triples $(X\ Y\ Z)$ for which $X \times Y = Z$. From this perspective, the multiplication problem 3×4 is represented as the triple $(3\ 4\ Z)$, where Z is a variable with an unknown value. Multiplying 3 by 4 is equivalent to finding a triple in the multiplication relation that matches $(3\ 4\ Z)$—in this case, the triple (3 4 12) matches, telling us that the solution is 12.

The relational view of multiplication is fascinating because we place variables in the first and second positions of a triple. For example, the triple $(X\ 4\ 12)$ represents both the multiplication problem $X \times 4 = 12$ and the division problem $12 \div 4 = X$. Since this triple matches (3 4 12), the solution to both problems is $X = 3$. The triple $(X\ Y\ 12)$ matches multiple elements of the multiplication relation, including (1 12 12), (3 4 12), and (6 2 12). And, of course, the triple $(X\ Y\ Z)$ matches each of the infinitely many triples in the mulplication relation.

In graduate school I learned that this relational view of multiplication has a computational interpretation. If we restrict the multiplication relation to include only a finite number of triples—for example, by setting an upper bound on the size of the numbers in each triple—then finding all triples that match $(X\ Y\ 12)$ is equivalent to performing a database query. If we relax this restriction, and allow the multiplication relation to include all of the infinitely many triples, then we enter the realm of pure logic programming.

If we go futher, and decide to write *all* of our programs as relations, we quickly find that the techniques and tools of traditional logic programming are insufficient. By fully committing to this relational perspective on computation—by taking relational programming seriously—we are confronted with countless challenges and opportunities that allow us to view classic notions of computing in new ways. *We learn it is much better to change the context in which the problem is being stated.*

Alan J. Perlis famously said, "A language that doesn't affect the way you think about programming, is not worth knowing."[2] The point of view we adopt in this book—the context in which we state every problem—is similar in spirit:

A program that isn't relational is not worth writing.

William E. Byrd
Salt Lake City, Utah
January 2014

---

[2]Epigram 19 from "Epigrams on Programming" (Perlis 1982).
Also at `http://www.cs.yale.edu/quotes.html`.

Chapter 1

Introduction

Bibliography

Alan C. Kay. Predicting the future. *Stanford Engineering*, 1(1), Autumn 1989.

Alan J. Perlis. Special feature: Epigrams on programming. *SIGPLAN Not.*, 17(9):7–13, September 1982. URL http://doi.acm.org/10.1145/947955.1083808.