Post Functional

Extracting energy from the Turing Tarpet

Alan Kay (Turing Centenary)

WHATs not HOWs

We Really Don't Know How to Compute!

Gerald Jay Sussman

Massachusetts Institute of Technology

CSAIL and EECS

Organizing Functional Code for Parallel Execution

or, `foldl` and `foldr` Considered Slightly Harmful

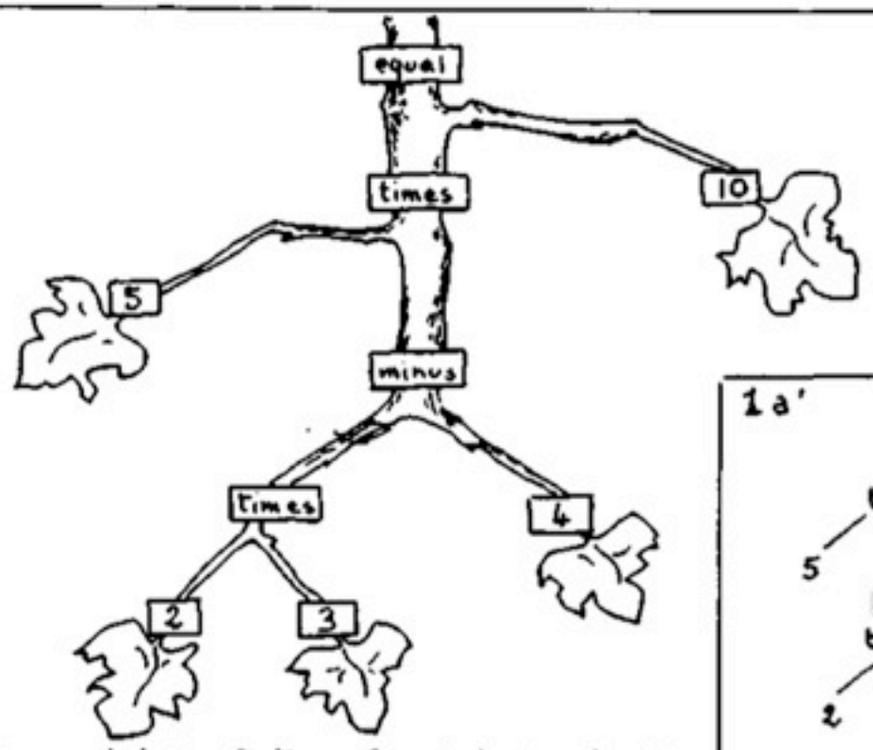
Guy Steele

Sun Fellow

Sun Microsystems Laboratories

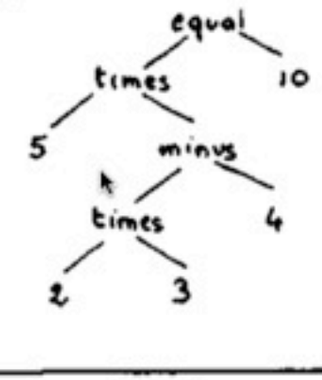
August 2009

1a

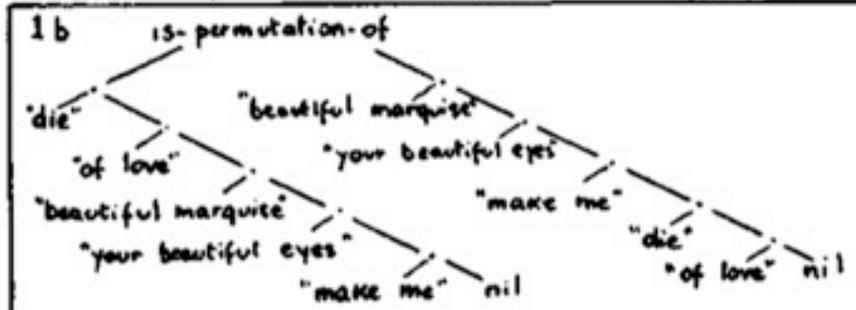


5 times the difference between 2 times 3 and 4 equals 10
that is to say: 5 x (2 x 3 - 4) = 10

1a'



1b



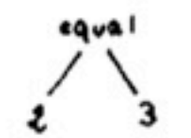
The sequence: "die", "of love", "beautiful marquise", "your beautiful eyes", "make me", is a permutation of the sequence: "beautiful marquise", "your beautiful eyes", "make me", "die", "of love".

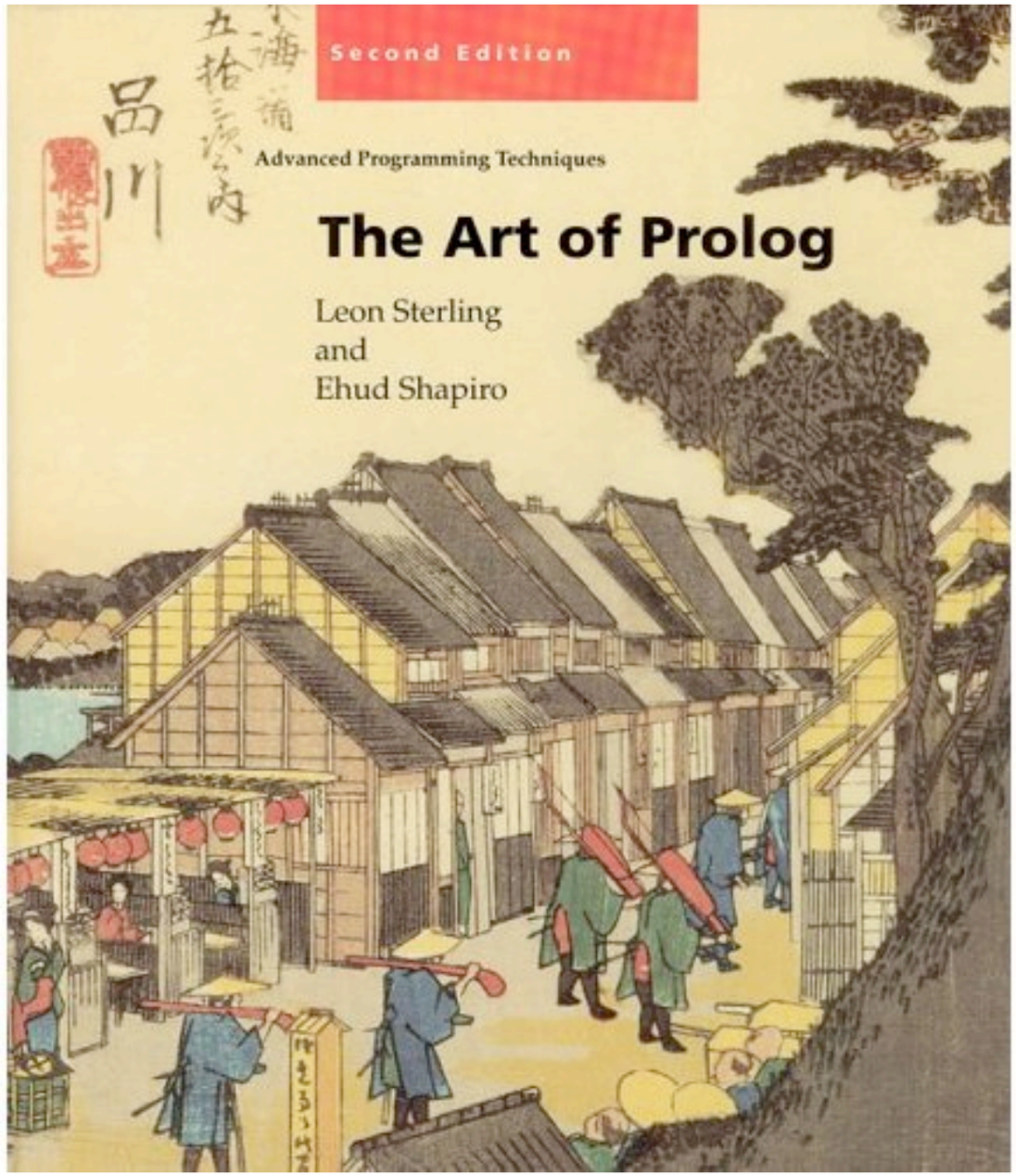
1c

The trio "radishes, sole, cake" constitutes a meal

1d

1e




Second Edition
Advanced Programming Techniques
The Art of Prolog
Leon Sterling
and
Ehud Shapiro


ERLANG

Embedding PROLOG in HASKELL

Silvija Seres Michael Spivey

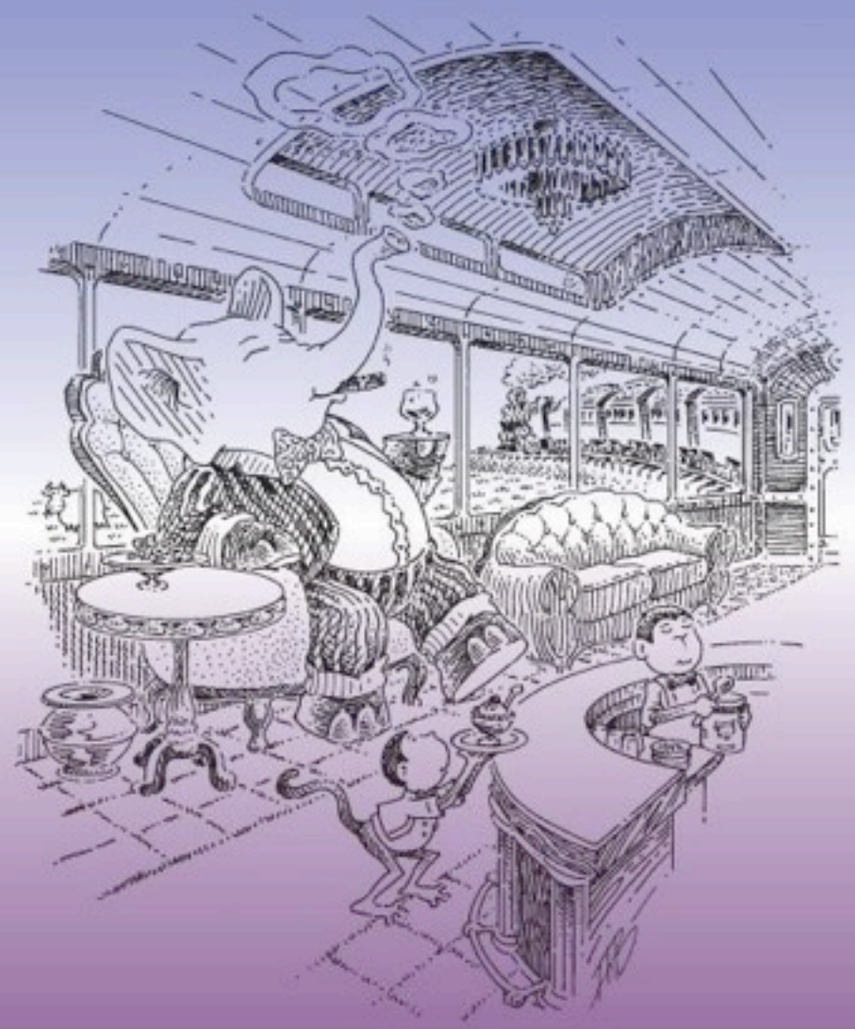
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD, U.K.

Abstract

The distinctive merit of the declarative reading of logic programs is the validity of all the laws of reasoning supplied by the predicate calculus with equality. Surprisingly many of these laws are still valid for the procedural reading; they can therefore be used safely for algebraic manipulation, program transformation and optimisation of executable logic programs.

This paper lists a number of common laws, and proves their validity for the standard (depth-first search) procedural reading of PROLOG. They also hold for alternative search strategies, e.g. breadth-first search. Our proofs of the laws are based on the standard algebra of functional programming, after the strategies have been given a rather simple implementation in Haskell.

The Reasoned Schemer



Daniel P. Friedman, William E. Byrd,
and Oleg Kiselyov

miniKanren

miniKanren

- *200* lines of Scheme

miniKanren

- *200* lines of Scheme
- Purely functional monadic design

miniKanren

- *200* lines of Scheme
- Purely functional monadic design
- Backtracking for “free”

miniKanren

- *200* lines of Scheme
- Purely functional monadic design
  - Backtracking for "free"
- Relatively efficient implementation

miniKanren

- *200* lines of Scheme
- Purely functional monadic design
  - Backtracking for "free"
- Relatively efficient implementation
- Lightweight embedding of relational programming

core.logic

core.logic

- emphasize efficiency

core.logic

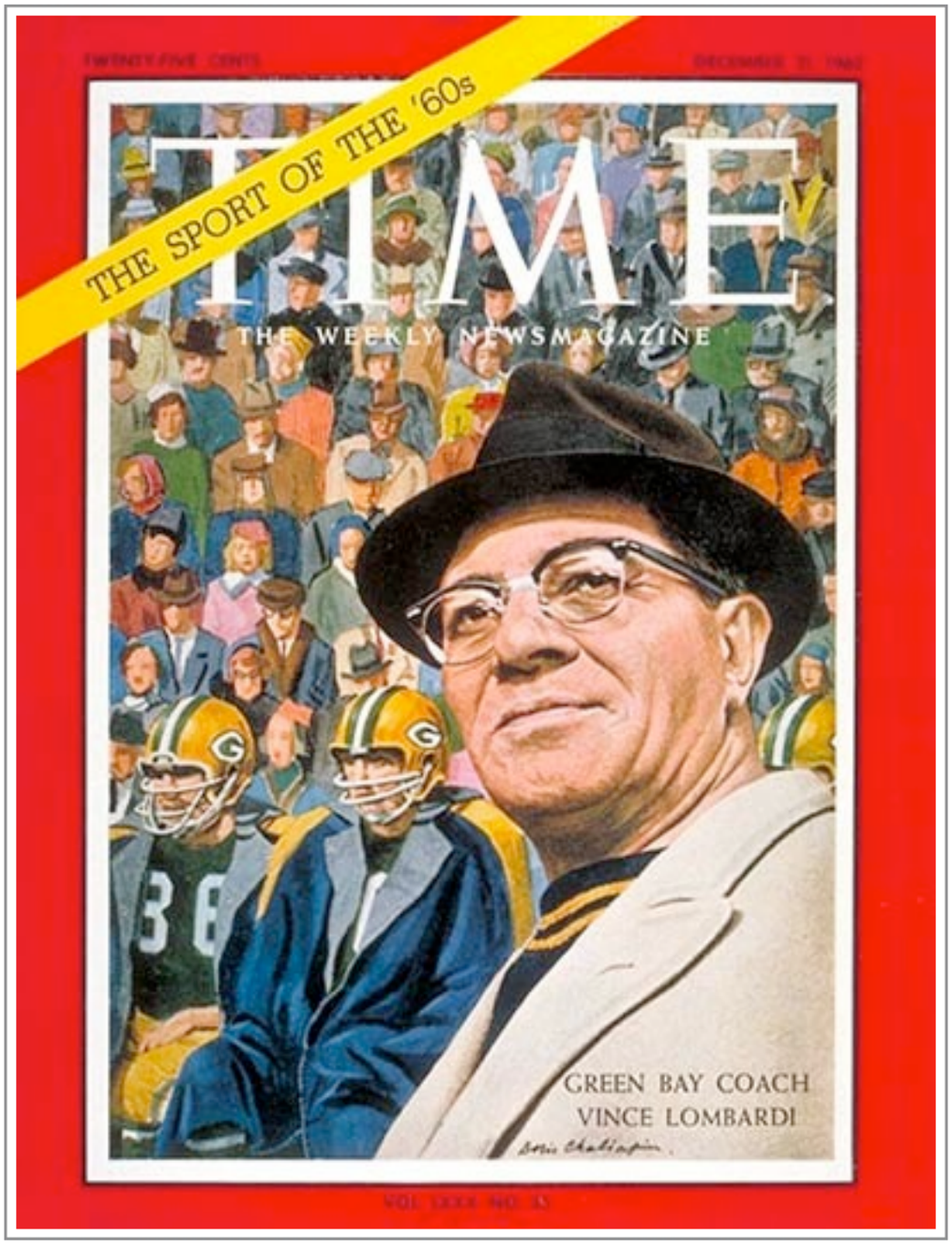
- emphasize efficiency
- emphasize polymorphism

core.logic

- emphasize efficiency
- emphasize polymorphism
- all critical parts of the system should allow open extension: unification, constraints, solvers, search strategies, etc.

Core Language

demo

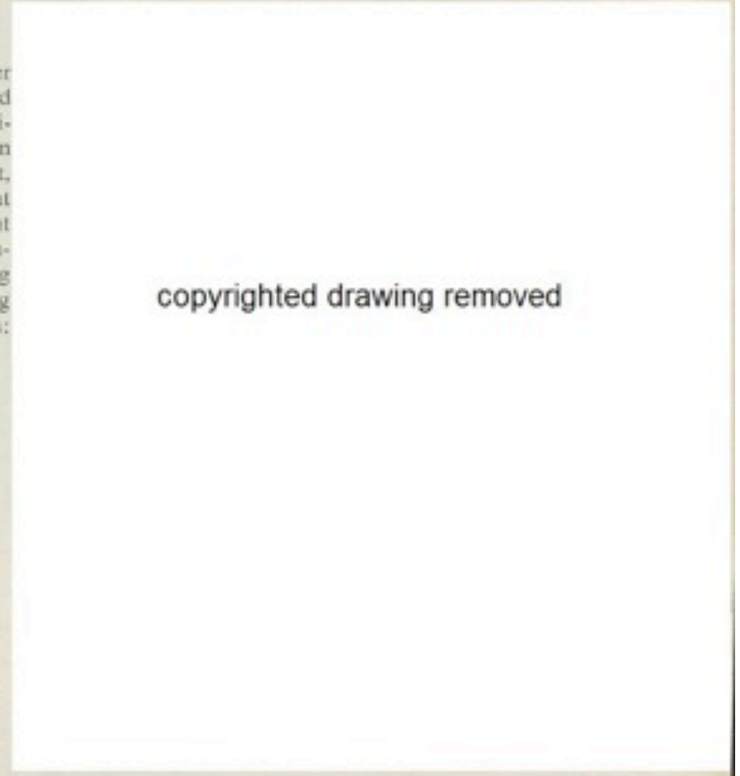


December 17, 1962

A PROBLEM FOR THE LOGICAL

Who Owns the Zebra?

In New York's posh Madison Avenue bars, stranger accosts stranger with a mimeographed sheet of paper and the question "Have you seen this?" In university dormitories, the problem is tacked to the doors. In suburban households in Westchester, Long Island and Connecticut, the ring of the telephone is likely to herald a voice that asks: "Is it the Norwegian?" The cause of the excitement is the brain-teaser reproduced on this page, with illustration provided by Steve Cook. The facts essential to solving the problem—which can indeed be solved by combining deduction, analysis and sheer persistence—are as follows:

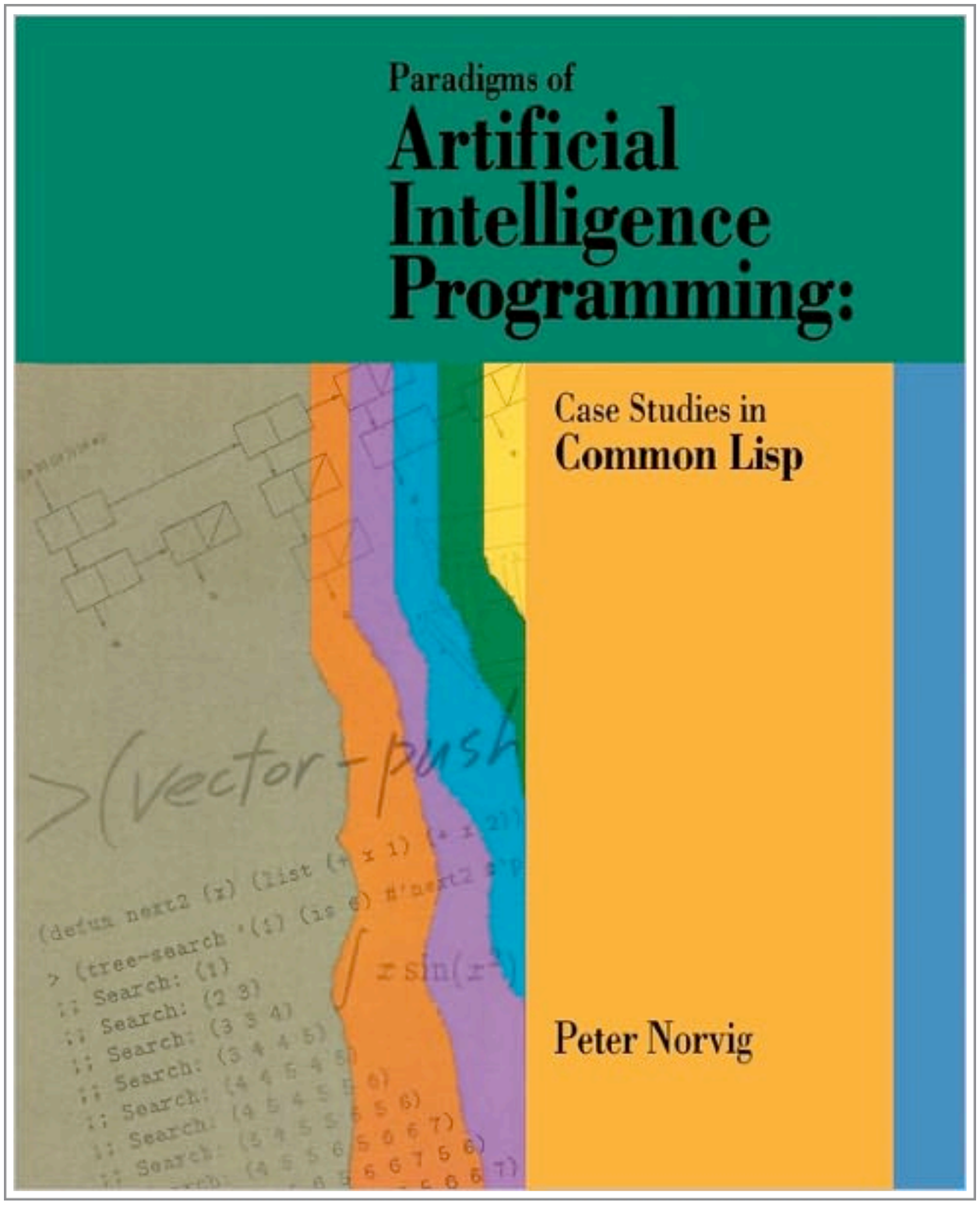


1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in the house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese smokes Parliaments.
15. The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra?

In the interest of clarity, it must be added that each of the five houses is painted a different color, and their inhabitants are of different national extractions, own different pets, drink different beverages and smoke different brands of American cigarets. One other thing: In Statement 6, *right* means *your* right.

LIFE International will be glad to receive answers from its readers and will publish one or more of those which best combine, in the editors' judgment, the proper solution with brevity and clarity in expounding the logic by which the solution was reached. No intuitive answers, please.

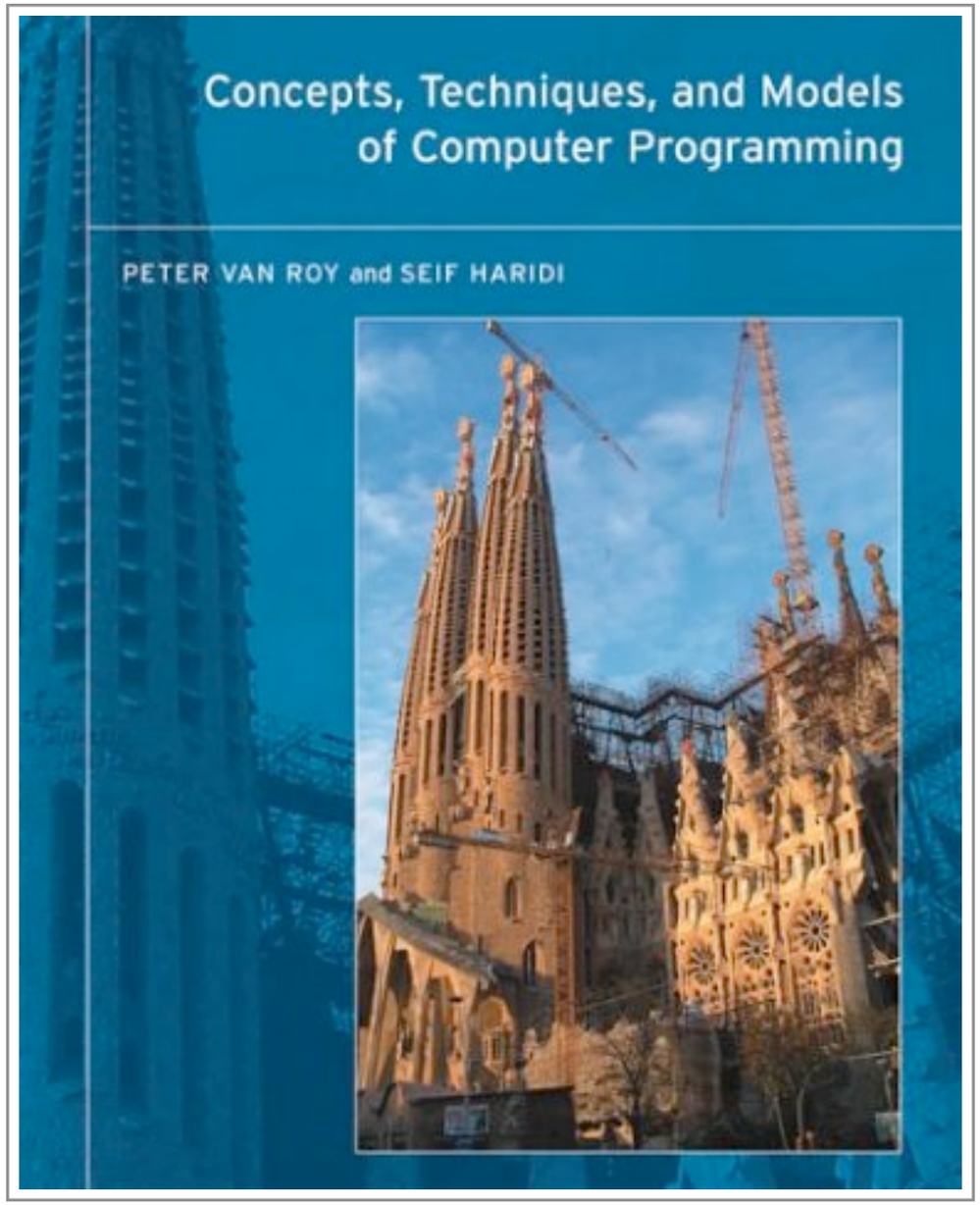

Paradigms of
Artificial Intelligence Programming:
Case Studies in Common Lisp
Peter Norvig

Peter Norvig points out,

24 billion candidate solutions

demo

Constraints!


Concepts, Techniques, and Models
of Computer Programming
PETER VAN ROY and SEIF HARIDI


4
2
5
3
6
8
9
3
4
2
9
8
2
6
7
4
5
1
3
9
4
6
4
7
2
5
6
5
3
1

demo

What's this good for?

- Complex configuration / validation
  (LonoCloud, Pallet, C2 Grammar of Graphics)

- Complex configuration / validation (LonoCloud, Pallet, C2 Grammar of Graphics)

- Static Analysis (Ekeko - Java Eclipse, Kibit - Clojure source linter)

- Complex configuration / validation (LonoCloud, Pallet, C2 Grammar of Graphics)

- Static Analysis (Ekeko - Java Eclipse, Kibit - Clojure source linter)

- Declarative GPU Programming (Indiana University)

- Complex configuration / validation (LonoCloud, Pallet, C2 Grammar of Graphics)

- Static Analysis (Ekeko - Java Eclipse, Kibit - Clojure source linter)

- Declarative GPU Programming (Indiana University)

- Supporting type inference? (Typed Clojure)

Culture

Culture

- OOP, FP, LP - all old stuff

Culture

- OOP, FP, LP - all old stuff
  - Mainstream culture around OOP

Culture

- OOP, FP, LP - all old stuff
  - Mainstream culture around OOP
  - FP gaining more and more traction

Culture

- OOP, FP, LP - all old stuff
  - Mainstream culture around OOP
  - FP gaining more and more traction
  - We're writing different (hopefully simpler) kinds of programs

Culture

- OOP, FP, LP - all old stuff
  - Mainstream culture around OOP
  - FP gaining more and more traction
    - We're writing different (hopefully simpler) kinds of programs
  - LP still at the fringes

Culture

- OOP, FP, LP - all old stuff
  - Mainstream culture around OOP
  - FP gaining more and more traction
    - We're writing different (hopefully simpler) kinds of programs
- LP still at the fringes
  - What kind of programs will we write?

Questions?