
Fun with
First-Class
Datatypes
Conor McBride
University of
Strathclyde

further proletarian adventures. . .

. . . in the post-revolutionary workers' republic

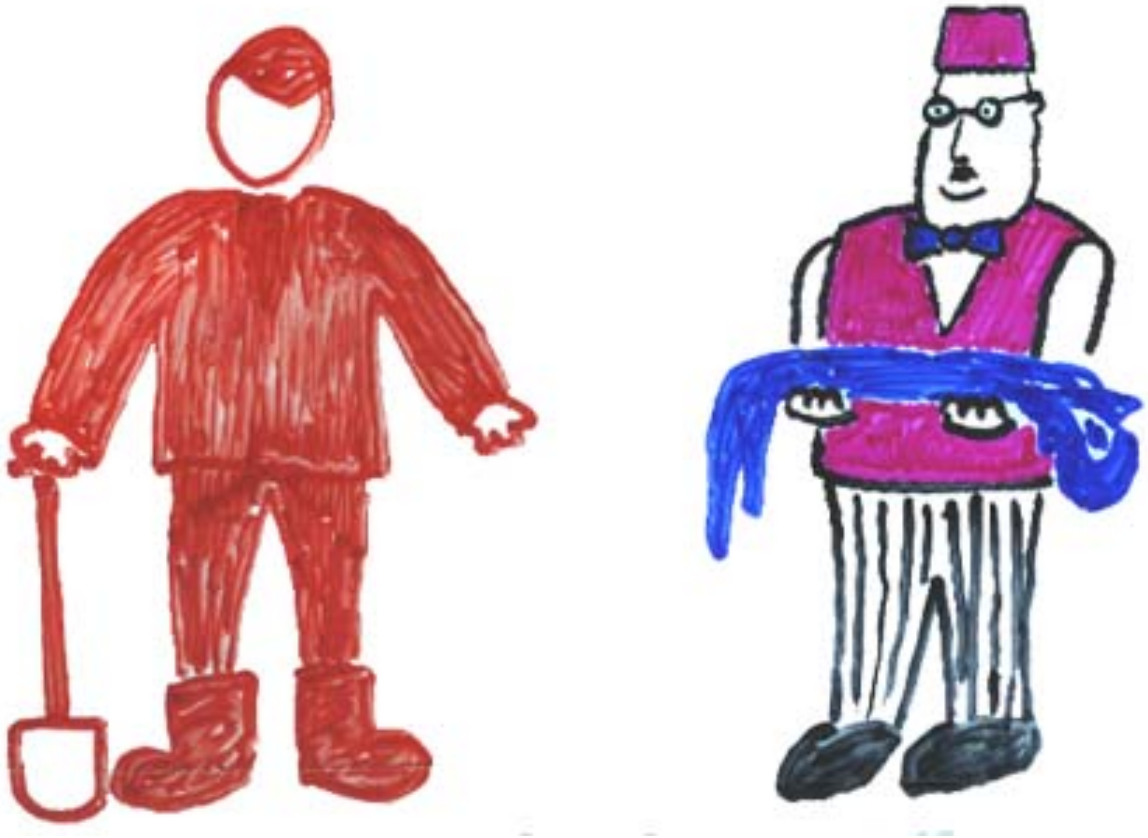
Dependent types allow
data and computation
in types.

This is terrifying,


Dependent types allow
data and computation
in types.
This is terrifying,
exhilarating, liberating,
powerful and strongly normalizing.

a term goes to a shop where he knows adventures can begin

as if by magic, the shopkeeper appears



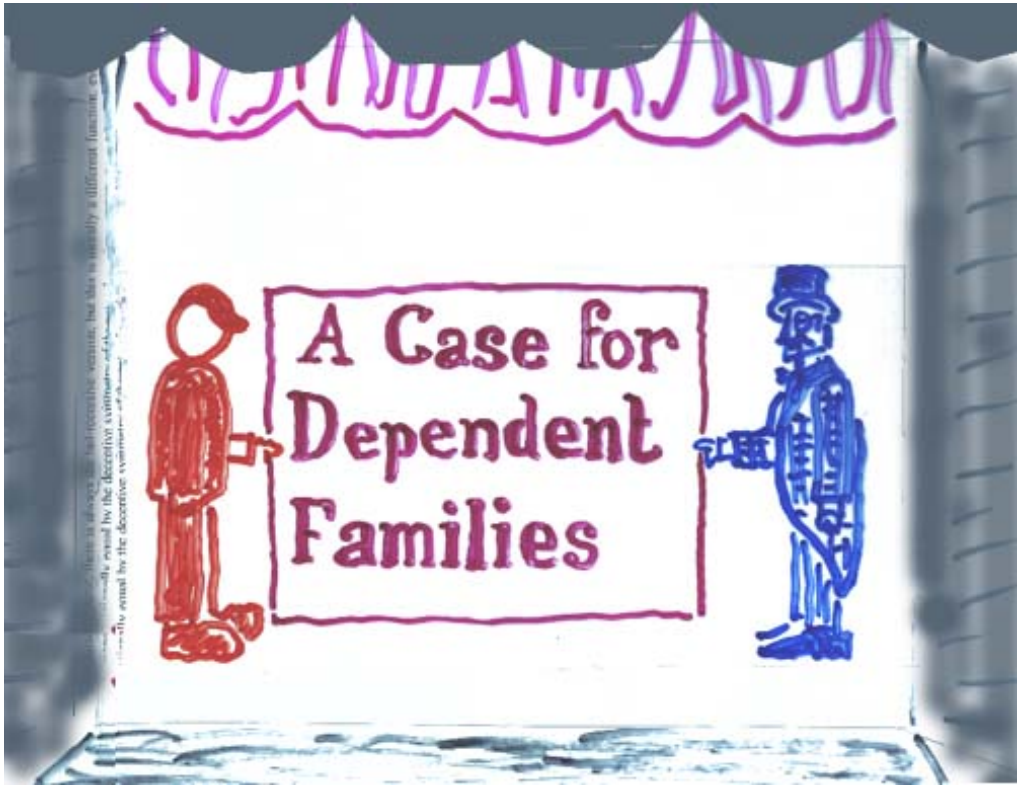
the term goes into the fitting room...

. . . and tries on the clothes; they fit perfectly

universe constructions

- we may define a datatype $U : \mathsf{Type}$ containing names of types ...
- ... and a family $T : U \to \mathsf{Type}$, Tu containing the data for the type with name u
- we can write generic programs over arbitrary Tu
- we can calculate u's without adding any 'new' notion of computation

slides from a talk I gave in Edinburgh in 2000

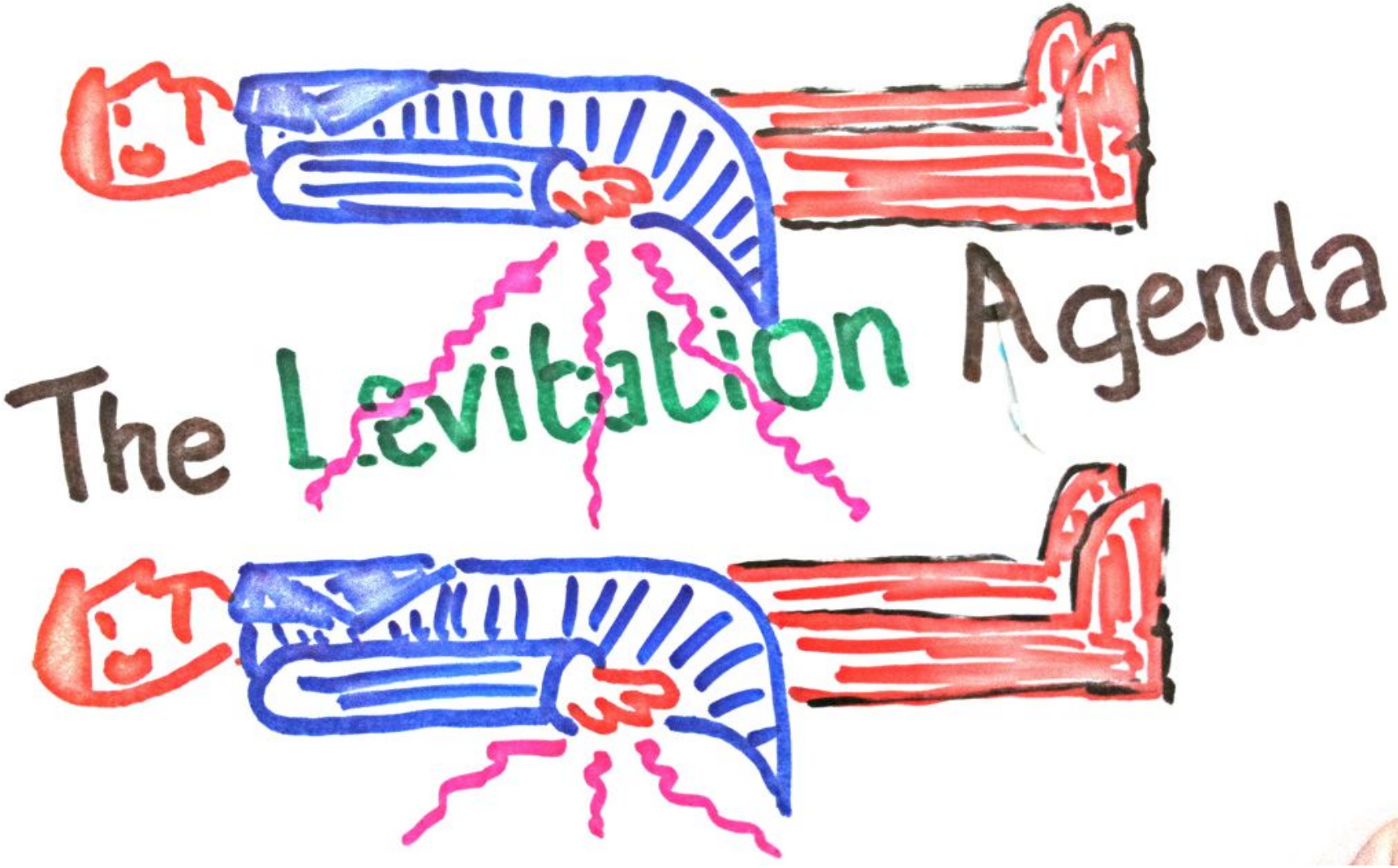


`http://strictlypositive.org/a-case`

more later. . .

Universes of datatypes:

- P Dybjer, A Setzer *A finite axiomatization of inductive-recursive denitions.* TLCA 1999
- C McBride *The derivative of a reguler type is its type of one-hole contexts* Rejected, LICS 2001
- T Altenkirch, C McBride *Generic programming within dependently typed programming* Generic Programming 2002
- M Benke, P Dybjer, P Jansson *Universes for generic programs and proofs in dependent type theory* Nord. J. Comp. 2003
- W Verbruggen, E de Vries, A Hughes *Polytypic programming in Coq* WGP 2008
- P Morris, T Altenkirch, N Ghani *A universe of strictly positive families* IJCS 2009
- J Chapman, P-E Dagand, C McBride, P Morris **The Gentle Art of Levitation** ICFP 2010


The Levitation Agenda

- construct $\mathbb{F}_n : \mathrm{Set}_n \to \mathrm{Set}_{n+1}$

  (index set: Set_n)

  $[\![\_]\!] : \mathbb{F}_n\ I \to (I \to \mathrm{Set}_n) \to (I \to \mathrm{Set}_n)$

  (endofunctor)

- abolish datatypes, except

  $\mu_n : \mathbb{F}_n\ I \to I \to \mathrm{Set}_n$

  $\langle\_\rangle : [\![F]\!]\ (\mu F) \dot{\to} \mu F$

  ('pointwise →')

- $\mu_n F$ has a recursor
- $\mathbb{F}_n$ is defined using μ_{n+1},
  i.e. coded via $\mathbb{F}_{n+1}$
- recursor for μ_{n+1} gives
  'generic' programming for μ_n types
- let's code up one layer in Agda

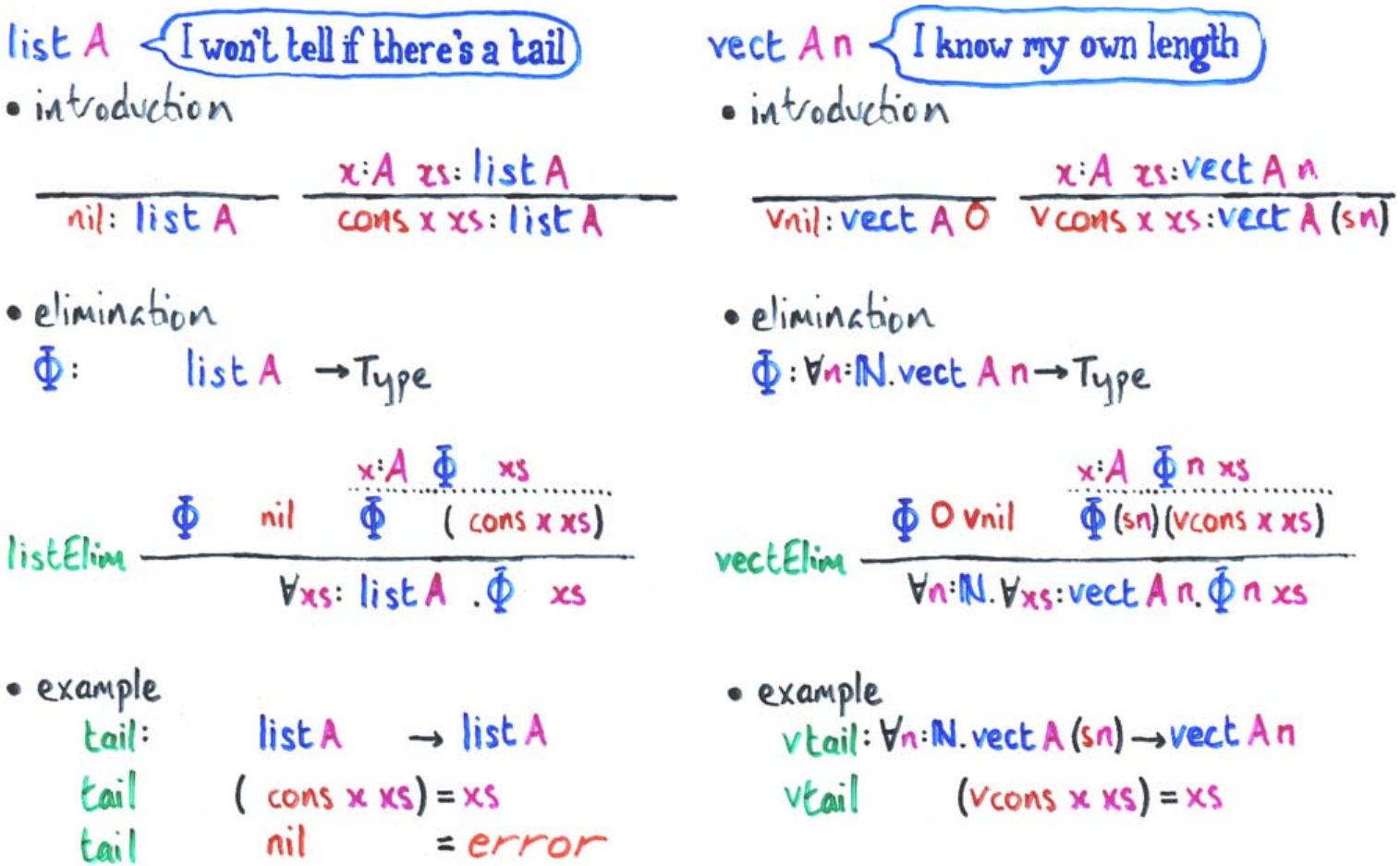
Here goes nothing.

generic programs

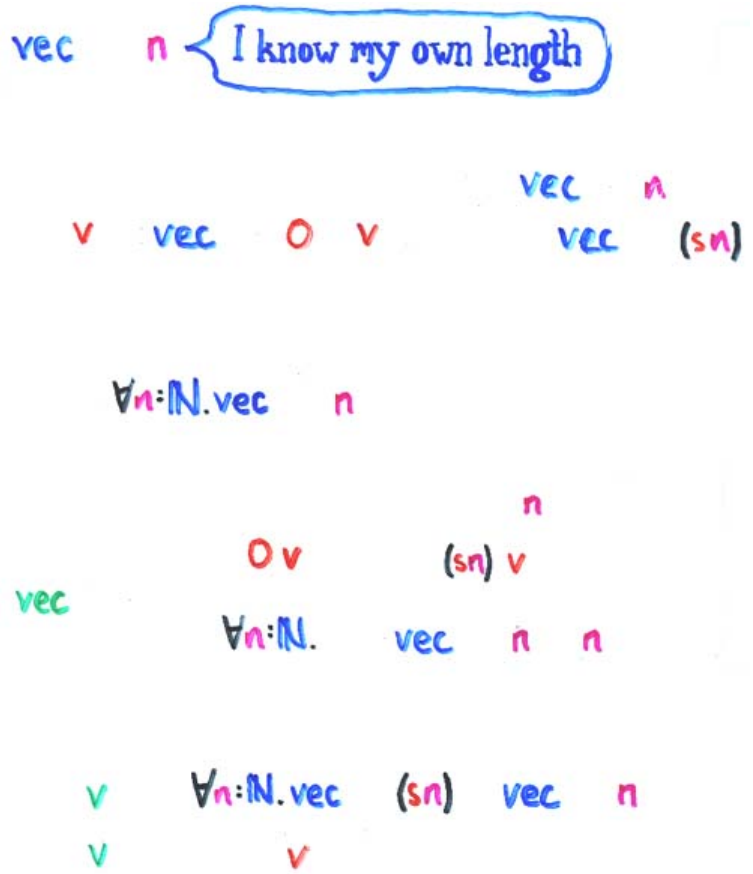
- map and fold
- substitution induces isomorphism
- differential calculus
- and ...

Here goes more nothing.

two more frames from '00



a suggestive overlay — can we formalize it?



ornaments and their algebras and their ornaments

Here goes even more nothing.

abolish data declaration;

point out the structure

which induces the function

abstract and delete