# Dependently-typed Programming in Scientific Computing

Cezar Ionescu (Potsdam Institute for Climate Impact Research) and Patrik Jansson (Chalmers University of Technology)

PIK addresses crucial scientific questions in the fields of global change, climate impacts and sustainable development.

Researchers from the natural and social sciences work together to generate interdisciplinary insights and to provide society with sound information for decision making.

The main methodologies are systems and scenarios analysis, modelling, computer simulation, and data integration.

PIK addresses crucial scientific questions in the fields of global change, climate impacts and sustainable development.

Researchers from the natural and social sciences work together to generate interdisciplinary insights and to provide society with sound information for decision making.

The main methodologies are systems and scenarios analysis, modelling, computer simulation, and data integration.

PIK addresses crucial scientific questions in the fields of global change, climate impacts and sustainable development.

Researchers from the natural and social sciences work together to generate interdisciplinary insights and to provide society with sound information for decision making.

The main methodologies are systems and scenarios analysis, modelling, computer simulation, and data integration.

PIK addresses crucial scientific questions in the fields of global change, climate impacts and sustainable development.

Researchers from the natural and social sciences work together to generate interdisciplinary insights and to provide society with sound information for decision making.

The main methodologies are systems and scenarios analysis, modelling, computer simulation, and data integration.

# Computer simulation

"Simulation is a third way of doing science. Like deduction, it starts with a set of explicit assumptions. But unlike deduction, it does not prove theorems.

Instead, a simulation generates data that can be analyzed inductively. Unlike typical induction, however, the simulated data comes from a rigorously specified set of rules rather than direct measurement of the real world."

R. Axelrod Advancing the Art of Simulation in the Social Sciences, 2003

# Computer simulation

"Simulation is a third way of doing science. Like deduction, it starts with a set of explicit assumptions. But unlike deduction, it does not prove theorems.

Instead, a simulation generates data that can be analyzed inductively. Unlike typical induction, however, the simulated data comes from a rigorously specified set of rules rather than direct measurement of the real world."

R. Axelrod Advancing the Art of Simulation in the Social Sciences, 2003

# Correctness of computer simulations

The correctness of a computer simulation therefore depends on

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

# Correctness of computer simulations

The correctness of a computer simulation therefore depends on

having explicit assumptions



# Correctness of computer simulations

The correctness of a computer simulation therefore depends on

- having explicit assumptions
- having rigorous rules to generate data

# Correctness of computer simulations

The correctness of a computer simulation therefore depends on

- having explicit assumptions
- having rigorous rules to generate data
- some relationship between the two

# Correctness of computer simulations

The correctness of a computer simulation therefore depends on

- having explicit assumptions
- having rigorous rules to generate data
- some relationship between the two

Sometimes, these conditions are not met...

# The Gintis model

"We thus provide, for the first time, a general, decentralized disequilibrium adjustment mechanism that renders market equilibrium dynamically stable in a highly simplified production and exchange economy."

"Our results should be considered empirical rather than theoretical: we have created a class of economies and investigated their properties for a range of parameters."

> Herbert Gintis The Emergence of a Price System from Decentralized Bilateral Exchange, 2006

## The Gintis model, ctd.

At PIK, the interest was fueld by the Lagom project:

"The model has provided the conceptual basis for two major studies commissioned by the German ministry for the Environment, the first assessing the economic implications of German climate policy, the second designing sustainable answers to the financial crisis."

From the homepage of the Lagom project,

In 2009, Mandel and Botta proved results for a simplified model with stronger assumptions. Many features of the Gintis model resisted mathematical analysis, and reproduction of the results failed.

# The Gintis model, ctd.

Independently, Pelle Evensen and Mait Märdin investigated the model and published results in *An Extensible and Scalable Agent-Based Simulation of Barter Economics* M.Sc. Thesis, Chalmers 2009.

Both groups discovered a serious bug in the implementation:

 $\frac{\sum_{j} p_{ij} x_{ij}}{\sum_{i} p_{ij} o_{i}}$ 

was implemented as

 $\frac{\sum_{j} p_{ij} x_{ij}}{\sum_{i} p_{ii} x_{ij}}$ 

This led to less variance in the computation of prices, and consequently to fast convergence.

#### The Gintis model, ctd.

Main problem: the "explicit hypothesis" were ambiguous, and the relationship to the code unclear.

"The discrepancies between the description and the original implementation of the barter economy confirm the importance of replication."

Evensen and Märdin, 2009

"In practice, however, model re-implementation on the basis of narrative descriptions is nearly impossible. For consistent, independent model re-implementation, one needs unambiguous mathematical specifications."

Botta et. al. A functional framework for agent-based models of exchange, 2011

# Specifications in scientific computing

We need *specifications* that

- ensure that "explicit hypothesis" and the "rigorously specified set of rules" are not contradicting each other
- allow checking correctness of implementations, model re-implementation, replication of results, etc.

We found little advice on specifications in scientific computing (e.g. *Writing Scientific Software – A Guide to Good Style* (Oliveira and Stewart, 2006) doesn't address specifications).

In many cases, the mathematical descriptions of their problems and algorithms are insufficient as specifications (e.g. because of discretization, approximations, introduction of arbitrary order of operations ...).

## Constructive mathematics

The gap between mathematics and programming is too large and we need to bridge it.

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

## Constructive mathematics

The gap between mathematics and programming is too large and we need to bridge it.

"Now, it is the contention of the intuitionists (or constructivists, I shall use these terms synonymously) that the basic mathematical notions, above all the notion of function, ought to be interpreted in such a way that the cleavage between mathematics, classical mathematics, that is, and programming that we are witnessing at present disappears."

> P. Martin-Löf, Constructive Mathematics and Computer Programming, 1984

## Constructive mathematics and type theory

"[Type theory] provides a precise notation not only, like other programming languages, for the programs themselves but also for the tasks that the programs are supposed to perform.

Thus the correctness of a program written in the theory of types is proved formally at the same time as it is being synthesized."

P. Martin-Löf, Constructive Mathematics and Computer Programming, 1984

## Constructive mathematics and type theory

"[Type theory] provides a precise notation not only, like other programming languages, for the programs themselves but also for the tasks that the programs are supposed to perform.

Thus the correctness of a program written in the theory of types is proved formally at the same time as it is being synthesized."

P. Martin-Löf, Constructive Mathematics and Computer Programming, 1984

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

Test: formalize basic concepts of economics.

# Models of exchange: example

Typical example:

- ► Two agents, two goods: beer and wine.
- ► For agent 1:

u(b, w) = if w < 1 then 0 else 2 \* b + w

► For agent 2:

u(b, w) = if b < 3 then 0 else b + 2 \* w

- Agent 1 has 3 bottles of wine and 2 of beer.
- Agent 2 has 1 bottle of wine and 7 of beer.

What can we expect after the agents trade?

## Basic economics: models of exchange

The quintessential economic situation: exchange of goods.

- 1.  $N_A$  agents,  $N_G$  goods,  $X_i$  units of good j.
- 2. Agent *i* has an *endowment*  $e_i = (x_{i1}, \ldots, x_{iN_G})$ .
- 3. The list of endowments  $(e_1, \ldots, e_{N_A})$  is called an *allocation*. Agents have preferences over allocations.
- Agents are allowed to exchange their goods in order to find a better allocation (e'<sub>1</sub>,..., e'<sub>N<sub>A</sub></sub>). Only *feasible* allocations are acceptable: ∑<sup>N<sub>A</sub></sup><sub>i=1</sub> x<sub>ij</sub> = X<sub>j</sub>.

What is a good allocation?

Pareto efficiency

**Definitions of Pareto efficiency.** A feasible allocation x is a **weakly Pareto efficient** allocation if there is no feasible allocation x' such that all agents strictly prefer x' to x.

Varian, p. 323

An allocation x is weakly Pareto efficient, if there exists no feasible allocation that dominates it *strictly everywhere*.

# Formalization of Pareto efficiency

A feasible allocation  $\mathbf{x}$  is a **weakly Pareto efficient** allocation if there is no feasible allocation  $\mathbf{x'}$  such that all agents strictly prefer  $\mathbf{x'}$  to  $\mathbf{x}$ .

▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ

\\_strictlyPrefers\_to\_ : Agent  $\rightarrow$ Allocation  $\rightarrow$  Allocation  $\rightarrow$  Set

# Formalization of Pareto efficiency

A feasible allocation x is a **weakly Pareto efficient** allocation if there is no feasible allocation x' such that all agents strictly prefer x' to x.

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

Introducing prices

If goods have prices  $p_j$  then an initial allocation  $\omega$  gives each agent a *budget*:

$$B_i = \sum_{j=1}^{N_G} p_j \omega_{ij}.$$

Assuming utility functions, an agent has to solve:

maximize  $u(e_i)$  such that

$$\sum_{j=1}^{N_G} p_j x_{ij} = B_i$$

Whether the resulting allocation is feasible depends on the prices.

# Walrasian equilibrium

An allocation-price pair  $(\mathbf{x}, \mathbf{p})$  is a **Walrasian equilibrium** if (1) the allocation is feasible, and (2) each agent is making an optimal choice from its budget set. In equations:

1. 
$$\sum_{i=1}^{n} \mathbf{x}_i = \sum_{i=1}^{n} \omega_i$$

2. If  $\mathbf{x}'_i$  is preferred by agent *i* to  $\mathbf{x}_i$ , then  $\mathbf{p}\mathbf{x}'_i > \mathbf{p}\boldsymbol{\omega}_i$ .

Varian, Microeconomic Analysis, p. 325

**First welfare theorem:** Walrasian equilibria are (weakly) Pareto efficient.

# Walrasian equilibrium

An allocation-price pair  $(\mathbf{x}, \mathbf{p})$  is a **Walrasian equilibrium** if (1) the allocation is feasible, and (2) each agent is making an optimal choice from its budget set.

- 1. Feasible x
- 2. If  $\mathbf{x'}_i$  is preferred by agent *i* to  $\mathbf{x}_i$ , then  $\mathbf{px'}_i > \mathbf{p}\omega_i$ .

Varian, Microeconomic Analysis, p. 325

**First welfare theorem:** Walrasian equilibria are (weakly) Pareto efficient.

# Walrasian equilibrium

An allocation-price pair  $(\mathbf{x}, \mathbf{p})$  is a **Walrasian equilibrium** if (1) the allocation is feasible, and (2) each agent is making an optimal choice from its budget set.

- 1. Feasible x
- 2. If  $\mathbf{x'}$  is preferred by agent *i* to  $\mathbf{x}$ , then  $\mathbf{px'}_i > \mathbf{p\omega}_i$ .

Varian, Microeconomic Analysis, p. 325

**First welfare theorem:** Walrasian equilibria are (weakly) Pareto efficient.

# Walrasian equilibrium

An allocation-price pair  $(\mathbf{x}, \mathbf{p})$  is a **Walrasian equilibrium** if (1) the allocation is feasible, and (2) each agent is making an optimal choice from its budget set.

- 1. Feasible x
- If x' is preferred by agent i to x, then the endowment of i in x' has greater value (according to p) than the endowment of i in ω.

# Formalizing Walrasian equilibrium

- 1. Feasible  $\mathbf{x}$
- If x' is preferred by agent i to x, then the endowment of i in x' has greater value (according to p) than the endowment of i in ω.

▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ

# Formalizing Walrasian equilibrium

- 1. Feasible x
- If x' is preferred by agent i to x, then the endowment of i in x' has greater value (according to p) than the endowment of i in ω.

# Formalizing Walrasian equilibrium

- 1. Feasible x
- If x' is preferred by agent i to x, then the endowment of i in x' has greater value (according to p) than the endowment of i in ω.

Question: if (x, p) is a Walrasian equilibrium, are all the agents "in budget"?

#### Formalizing the first welfare theorem

 $Walras \Rightarrow Pareto : (p : Price) \rightarrow (x : Allocation) \rightarrow WalrasianEq (p, x) \rightarrow WeakPareto x$ 

Walras  $\Rightarrow$  Pareto  $p \times (fx, weq) = fx, wpe$  where ...

We need

$$\begin{array}{l} allOutOfBudget : (x : Allocation) \rightarrow (p : Price) \rightarrow \\ ((a : Agent) \rightarrow \\ value (endmt x a) p > value (endmt omega a) p) \rightarrow \\ \neg (Feasible x) \end{array}$$

# Mainstream economics

#### Refinements

- several agents
- production and consumption
- iterated exchanges
- introduce agents representing banks, governments, ....

▶ ...

Most of the models used for policy advice are based on extensions of this idea (it's a good place to start for specifications).

Good news

We tested the expressive power of type theory by formalizing different equilibria in Agda and Idris, together with the relationships betwen them.

We could write specifications for certain kinds of economic agents in Gintis-like models.

We had several sessions with Lagom modelers, and they found the specifications understandable.

# Bad news

Therefore, it appears that we can express the "explicit hypothesis" and the "rules" that drive our simulations. . .

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

# Bad news

Therefore, it appears that we can express the "explicit hypothesis" and the "rules" that drive our simulations...

but not the relationship between them.

- Economic theory is mostly non-constructive (K. Vellupilai, 2002): the divide between mathematical specification and implementations is still there.
- Most modelers are not numerical analysts: they want to use external routines.
- ► No usable library of numerical methods for constructive reals.
- (Some) modelers are willing to write formal specifications, but less willing to write formal proofs, let alone *constructive* formal proofs.

## Good news

Having specifications is better than having no specifications.

Having specifications which can be partially machine-checked is better than having specifications which cannot be machine-checked at all.

Having classical proofs of correctness is better than having no proofs of correctness.

Using type theory for specifications can also guide the efforts of the constructive mathematics community.

And so on: just because we cannot now have fully verified models should not prevent us from taking advantage of what we have!

# Some Fin functions

$$\_==\_$$
: forall  $\{n\} \rightarrow$  Fin  $n \rightarrow$  Fin  $n \rightarrow$  Bool  
zero == zero = true  
zero == suc j = false  
suc i == zero = false  
suc i == suc j = i == j

toFin :  $(n : Nat) \rightarrow Fin (suc n)$ toFin zero = zero toFin (suc n) = suc (toFin n)

# Maximizing utility over a finite set

We want

$$\begin{array}{rcl} \textit{max} & : & \{n \ : \ \textit{Nat} \} \ \rightarrow & \\ & (\textit{Fin} \ (S \ n) \ \rightarrow \ \textit{Float}) \ \rightarrow & \textit{Fin} \ (S \ n) \ \land \ \textit{Float} \end{array}$$

such that

$$\begin{array}{rl} maxSpec : \{n : Nat\} \rightarrow (u : Fin (S n) \rightarrow Float) \rightarrow \\ (i : Fin (S n)) & \rightarrow \\ so (u (fst (max u)) = f snd (max u)) \land \\ so (u i \leqslant snd (max u)) \end{array}$$

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

## Haskell-style implementation

$$\begin{array}{rl} max : \{n : Nat\} \rightarrow & \\ & (Fin (S n) \rightarrow Float) \rightarrow Fin (S n) \wedge Float \\ max \{O\} u & = fO, u fO \\ max \{S n\} u = max' u (fO, u fO) fO \end{array}$$

```
max' \{n\} u (best, bestU) c' =
  let c = suc c' in
  let uc = u c in
    if (c == toFin n)
       then
         if uc \leq bestU then (best, bestU)
            else (c, uc)
       else
         if uc \leq bestU then max' u (best, bestU) c
            else max' u (c, uc) c
```

# Agda is not Haskell

$$\begin{array}{ll} max : \{n : Nat\} \rightarrow \\ (Fin (S n) \rightarrow Float) \rightarrow Fin (S n) \wedge Float \\ max \{O\} u = fO, u fO \\ max \{S n\} u = max' u (fO, u fO) fO \end{array}$$

```
max' \{n\} u (best, bestU) c' =
  let c = suc c' in
  let uc = u c in
    if (c == toFin n)
       then
         if uc \leq bestU then (best, bestU)
            else (c, uc)
       else
         if uc \leq bestU then max' u (best, bestU) c
                                                       -- !
            else max' u(c, uc) c -- !
```

# Fins are Finicky

$$\begin{array}{rcl} max' : \{n : Nat\} \rightarrow & \\ & (Fin (S n) \rightarrow Float) \rightarrow & -- \mbox{ utility} & \\ & Fin (S n) \wedge Float \rightarrow & -- \mbox{ best-so-far} & \\ & Fin n & \rightarrow & -- \mbox{ count } / \mbox{ candidate} & \\ & Fin (S n) \wedge Float & & -- \mbox{ optimum} & \end{array}$$

$$max' \{n\} u (best, bestU) c' = let c = suc c' inlet uc = u c inif (c == toFin n)thenif uc \leq bestU then (best, bestU)else (c, uc)elseif uc \leq bestU then max' u (best, bestU) c -- !else max' u (c, uc) c -- !$$

# Trust me, I'm a professional

```
coerce' : {AB : Set} \rightarrow AB \rightarrow A \rightarrow B
coerce' refl a = a
        : \{AB : Set\} \rightarrow A \rightarrow B
coerce
         = coerce' trustMe
coerce
max' \{n\} u (best, bestU) c' =
  let c = suc c' in
  let \mu c = \mu c in
    if (c == toFin n)
       then
         if uc \leq bestU then (best, bestU)
            else (c, uc)
       else
         if uc \leq bestU then max' u (best, bestU) (coerce c)
            else max' u(c, uc) (coerce c)
```

# Programming style

How do we specify that the outputs a program  $X \rightarrow Y$  have to be in the relation R with the inputs?

Nordström et. al.:

$$f : (x : X) \rightarrow \exists (\lambda(y : Y) \rightarrow R(x, y))$$

Thompson:

$$\exists (\lambda(f : X \rightarrow Y) \rightarrow (x : X) \rightarrow R(x, f x))$$

## Optimization problems, continuous case

Current practice: use an external optimizer and assume it works.

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

## Optimization problems, continuous case

Current practice: use an external optimizer and assume it works.

maxSpec serves as a documentation of this assumption.

## Optimization problems, continuous case

Current practice: use an external optimizer and assume it works.

maxSpec serves as a documentation of this assumption.

Often, the type of the utility function is constrained to functions for which *maxSpec* is less of a lie.

Optimization problems, ctd.

E.g.: for elementary functions defined on "convenient" intervals one can show that Newton-based methods converge. The result is an interval guaranteed to contain the solution.

Optimization problems, ctd.

E.g.: for elementary functions defined on "convenient" intervals one can show that Newton-based methods converge. The result is an interval guaranteed to contain the solution.

Even then, formalizing the proof in Agda is not trivial: standard proofs are classical. Thus all we can usually show is that the resulting interval *cannot fail to contain the solution*.

Optimization problems, ctd.

E.g.: for elementary functions defined on "convenient" intervals one can show that Newton-based methods converge. The result is an interval guaranteed to contain the solution.

Even then, formalizing the proof in Agda is not trivial: standard proofs are classical. Thus all we can usually show is that the resulting interval *cannot fail to contain the solution*.

At the moment, we use external libraries for interval analysis anyway...

Doing a bit better...

*Lots of* future work:

 Specify more commonly used external routines, e.g. for interpolation.

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

Doing a bit better...

Lots of future work:

- Specify more commonly used external routines, e.g. for interpolation.
- Improve notation for dependent-types, e.g. where-clauses for type declarations.

# Doing a bit better...

Lots of future work:

- Specify more commonly used external routines, e.g. for interpolation.
- Improve notation for dependent-types, e.g. where-clauses for type declarations.

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

 Develop DSLs for specifications of economic, climate, etc. models.

# Doing a bit better...

Lots of future work:

- Specify more commonly used external routines, e.g. for interpolation.
- Improve notation for dependent-types, e.g. where-clauses for type declarations.
- Develop DSLs for specifications of economic, climate, etc. models.
- Implement interval analysis methods for validated numerics.

# Doing a bit better...

Lots of future work:

- Specify more commonly used external routines, e.g. for interpolation.
- Improve notation for dependent-types, e.g. where-clauses for type declarations.
- Develop DSLs for specifications of economic, climate, etc. models.
- Implement interval analysis methods for validated numerics.

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

Prepare for the constructive mathematics revolution, e.g. results from projects such as ForMath.

# A motto for increasingly correct scientific computing

```
"The road to wisdom? Well, it's plain
and simple to express:
Err
and err
and err
and err again,
but less
and less
and less."
```

Piet Hein (1905–1996), The Road to Wisdom, in Grooks (1966).