Functional Programming and Climate Impact Research

Patrik Jansson[1]

Functional Programming unit, Chalmers University of Technology

2024-03-25

Abstract

This is a course aimed at PhD students or MSc students interested in the application of functional programming, domain-specific languages, and dependent types to climate impact research. **Note**. This course is run as a seminar / reading course. Therefore, you must have the motivation and capacity to digest material with limited teacher presence.

[1] Joint work with N Botta, C Ionescu

Lecture / seminar / reading plan

The course is based on material from the following research papers:

1. Vulnerability modelling with functional programming and dependent types
2. Testing versus proving in climate impact research
3. Dependently-Typed Programming in Scientific Computing - Examples from Economic Modelling
4. Towards a Computational Theory of GSS: a Case for Domain-Specific Languages
5. Sequential decision problems, dependent types and generic solutions
6. Contributions to a computational theory of policy advice and avoidability
7. The impact of uncertainty on optimal emission policies
8. Responsibility Under Uncertainty: Which Climate Decisions Matter Most?

Good background reading is provided by these recent books:

- "Computing the Climate" by Steve M. Easterbrook and
- "Domain-Specific Languages of Mathematics" by Jansson, Ionescu, Bernardy.

Examination / admin

Examination

The course is examined through

- a sequence of graded hand-ins (the hand-ins will be further specified during the course)
- active participation in most of the weekly seminars

The course is worth 7.5 higher education credits (ECTS).

Prerequisites

- BSc degree in Computer Science and Engineering or equivalent.
- Functional programming (ideally in Haskell, but other languages are also OK)
- Formal methods (ideally using dependent types, but other methods are also OK)

Learning outcomes

- Use FP spec. / impl. / formalisation as a way of understanding new domains
- Understand a selection of topics in Climate Impact Research
- Effectively use Haskell and Agda for formalisation
- Knowledge and understanding
  - Master the terminology, concepts and theories associated with the selected area;
  - Dem. deep knowledge and understanding in the area, and insight into current R&D;
  - Dem. deep methodological knowledge in the area of the course;
- Skills and abilities
  - Demonstrate the ability to critically and systematically integrate knowledge and to analyse, assess, and deal with complex issues in the area of the course;
- Judgement and approach
  - Search for, and extract, necessary information from scientific publications in the selected area of the course, with the purpose of identifying strengths and weakness of solutions, approaches and methodologies.

Scheduling

Week	Date	Time	Event	Room
13	Mon 2024-03-25	15.15-17.00	FPClimate seminar 1 (Patrik Jansson)	EDIT 6128
13	Tue 2024-03-26	14.15-15.15	Talk by Anil Madhavapeddy	HB2
14	(Patrik away)			
15	Mon 2024-04-08	15.15-17.00	FPClimate seminar 2 (Cezar Ionescu)	EDIT 6128
16	Mon 2024-04-15	15.15-17.00	FPClimate seminar 3	EDIT 6128
17	Mon 2024-04-22	15.15-17.00	FPClimate seminar 4	EDIT 6128
18	Mon 2024-04-29	15.15-17.00	FPClimate seminar 5 (Nicola Botta)	EDIT 6128
19	Mon 2024-05-06	15.15-17.00	FPClimate seminar 6	EDIT 6128
20	Mon 2024-05-13	15.15-17.00	FPClimate seminar 7	EDIT 6128
21	Mon 2024-05-20	15.15-17.00	FPClimate seminar 8 (Nicola Botta)	EDIT 5128
22	Mon 2024-05-27	15.15-17.00	FPClimate seminar 9	EDIT 6128

Zoom: `https://chalmers.zoom.us/my/CUTpatrikja`; Time zone: CET (UTC+1) now, later CEST (UTC+2)

First few weeks

Week 13:

- FPClimate seminar 1: Introduction (Patrik Jansson)
- Talk by Prof. Anil Madhavapeddy
- Title: (Functional) Programming for the Planet

Week 14:

- solve W13 exercises
- read “Vulnerability modelling . . . ”

Week 15:

- FPClimate seminar 2: Lecture by Prof. Cezar Ionescu about Vulnerability modelling
- solve W15 exercises
- read "Testing versus proving in climate impact research"

Equations, problems and solutions

In mathematics we say that $x = 1$ and $x = -1$ are solutions of $x^2 = 1$.

What does this precisely mean?

Equations, problems and solutions

In mathematics we say that $x = 1$ and $x = -1$ are solutions of $x^2 = 1$.

What does this precisely mean?

$x = 1$, $x = -1$ and $x^2 = 1$ are all equations. But they are in certain relations to each other.

We have $x = 1 \Rightarrow x^2 = 1$ and also $x = -1 \Rightarrow x^2 = 1$

These implications are what justifies saying that $x = 1$ and $x = -1$ *solve* $x^2 = 1$.

Equations, problems and solutions, cont.

The equation $x = 1$ ($x = -1$) is different from $x^2 = 1$ also from another point of view: the first equation determines *the* value of x directly, without computations.

The equation $x^2 = 1$ specifies a problem: that of finding *values* whose square is one. We can specify the problem a little bit more explicitly:

Find $x \in \mathbb{R}$ ***s.t.*** $x^2 = 1$

This is an example of a *mathematical specification*. This problem has two solutions.

Equations, problems and solutions, cont.

The equation $x = 1$ ($x = -1$) is different from $x^2 = 1$ also from another point of view: the first equation determines *the* value of x directly, without computations.

The equation $x^2 = 1$ specifies a problem: that of finding *values* whose square is one. We can specify the problem a little bit more explicitly:

Find $x \in \mathbb{R}$ ***s.t.*** $x^2 = 1$

This is an example of a *mathematical specification*. This problem has two solutions.

We can go one step further and specify the problem of finding the square root of any number:

Given $y \in \mathbb{R}$, ***find*** $x \in \mathbb{R}$ ***s.t.*** $x^2 = y$

Equations, problems and solutions, cont.

The equation $x = 1$ ($x = -1$) is different from $x^2 = 1$ also from another point of view: the first equation determines *the* value of x directly, without computations.

The equation $x^2 = 1$ specifies a problem: that of finding *values* whose square is one. We can specify the problem a little bit more explicitly:

Find $x \in \mathbb{R}$ ***s.t.*** $x^2 = 1$

This is an example of a *mathematical specification*. This problem has two solutions.

We can go one step further and specify the problem of finding the square root of any number:

Given $y \in \mathbb{R}$, ***find*** $x \in \mathbb{R}$ ***s.t.*** $x^2 = y$

This problem is not solvable (for $y = -1$, for example). The specification is *infeasible*.

The problem here is that the requirements on y are too weak. We can obtain a *feasible* specification if we require y to be non-negative:

Given $y \in \mathbb{R}$, $0 \leqslant y$, ***find*** $x \in \mathbb{R}$ ***s.t.*** $x^2 = y$

Equations, problems and solutions, cont. 2

What about the following problem? Is it solvable?

Given $y \in \mathit{Double}$, $0 \leqslant y$, ***find*** $x \in \mathit{Double}$ ***s.t.*** $x^2 = y$

Equations, problems and solutions, cont. 2

What about the following problem? Is it solvable?

Given $y \in \mathit{Double}$, $0 \leqslant y$, ***find*** $x \in \mathit{Double}$ ***s.t.*** $x^2 = y$

No, not in general, due to finite precision. When doing real computations, we typically accept that this equation will only be satisfied up to a certain tolerance. We do not want to deal with these kind of problems here.

(Fun fact: 17.0 has a square root in *Double*; the square of $r = 4.123105625617661$ happens to be / round to exactly 17.0.)

Functions as solutions

The previous spec. does not say anything about which root shall be found for a given y, thus it is *relational*. We often want to be more precise and require the solution of a problem to be a *function*. We specify a function by giving its signature and its definition. For instance

$$\begin{aligned} &\mathit{double} : \mathbb{N} \to \mathbb{N} \\ &\mathit{double}\ n = 2 * n \end{aligned}$$

Notation: we denote function application $f(x)$ by juxtaposition $f\ x$, as in Haskell and Agda.

We can specify the problem of finding a square root function as e.g.

$$\textbf{\textit{Find}}\ \sqrt{\ } : \mathbb{R} \to \mathbb{R}\ \ \textbf{\textit{s.t.}}\ \forall\ y \in \mathbb{R},\ \ 0 \leqslant y \Rightarrow (\sqrt{y})^2 = y$$

or equivalently as

$$\textbf{\textit{Find}}\ \sqrt{\ } : \mathbb{R}_{\geq 0} \to \mathbb{R}\ \ \textbf{\textit{s.t.}}\ \forall\ y \in \mathbb{R}_{\geq 0},\ \ (\sqrt{y}) * (\sqrt{y}) = y$$

This problem has many solutions including one function that always computes the negative root and one that always computes the positive square root.

Functions as solutions, cont.

Exercise

Specify the problem of finding a function that computes both square roots.

Remark: In this subsection, we followed Morgan's introduction to programming from specification [2]. There you can also find more examples and exercises.

Domain-specific notions

Mathematical specifications can also be applied to clarify notions that are specific to a given application domain. For example:

- What does it mean for $f : X \to Y$ to be a function?
- What does it mean for a strategy to be dominant?
- What does it mean for a climate state to be avoidable?

Often, giving precise answers is not easy. Sometimes, it turns out that we want a whole *family of notions*, not just one notion. An example in the context of emission problems, for instance, is

- Emission reductions imply different costs and benefits for different countries.
- The highest global benefits are obtained if most countries reduce emissions by certain (optimal, fair, ...) country-specific amounts.
- In this situation most countries have a free-ride opportunity!

Domain-specific notion example: prisoner's dilemma

The most paradigmatic example of this situation is perhaps the two-players prisoner's dilemma

	D	C
D	(1,1)	(3,0)
C	(0,3)	(2,2)

Table: Payoff matrix

Which property makes (D, D) stable and yet undesirable strategies?

Let $S = \{D, C\}$ *and* $p_1, p_2 : S \times S \to \mathbb{R}$ *payoff functions.*
A ***strategy profile*** $(x, y) \in S \times S$ *is a* ***Nash equilibrium*** *iff*
$\forall\, x', y' \in S,\ p_1(x', y) \leqslant p_1(x, y)$ *and* $p_2(x, y') \leqslant p_2(x, y)$.

Remark: Note that the definition of Nash equilibrium depends on a binary operator $\leqslant$.

(Which properties should this binary operator reasonably have?)

Prisoner's dilemma: Exercises

	D	C
D	(1,1)	(3,0)
C	(0,3)	(2,2)

Table: Payoff matrix

Exercise

Modify the payoffs of (C, C) in Table 1 for (C, C) to become a Nash equilibrium.

Exercise

Generalize the notion of Nash equilibrium to an arbitrary number of players.

Policies and optimality

Let X denote a set of states that a decision maker can observe.
(X could be a tuple of numbers that represent aggregated measures or indicators of wealth, inequality, environmental stress, etc.)

Let Y denote the options available to the decision maker. For simplicity, we assume that she has the same options in all states $x \in X$.

Functions that associate an option to every state are called *policies*.

Let $val : X \to Y \to \mathbb{R}$ be a *value function*:
$val\ x\ y$ denotes the value of taking decision y in state x.

A policy $p : X \to Y$ is called *optimal* w.r.t to *val* if it yields controls that are better or as good as any other control for all states.

Exercise

Give a mathematical specification of the notion of optimality for policies.

Optimality, cont.

If Y is finite and non-empty, one can implement

$$\begin{aligned} \mathit{max} &: (Y \to \mathbb{R}) \to \mathbb{R} \\ \mathit{argmax} &: (Y \to \mathbb{R}) \to Y \end{aligned}$$

that fulfill

$$\forall\ f : Y \to \mathbb{R},\ \forall\ y \in Y,\ f\ y \leqslant \mathit{max}\ f$$
$$\forall\ f : Y \to \mathbb{R},\ f\ (\mathit{argmax}\ f) = \mathit{max}\ f$$

Exercise

Find a function $p : (X \to Y \to \mathbb{R}) \to (X \to Y)$ that such that $p\ \mathit{val}$ is an optimal policy w.r.t to val for arbitrary val. Prove that $p\ \mathit{val}$ is indeed optimal.

Exercise

Let *Fin n* be the set of natural numbers smaller than n:

$Fin\ 0 = \{\}$

$Fin\ 1 = \{0\}$

$Fin\ 2 = \{0, 1\}$

...

$Fin\ n = \{0 \ldots n-1\}$

Give a mathematical specification of the notion of finiteness for a set X. Begin with

*A set X is **finite** iff ...*

Exercise

Apply the specification of finiteness from Exercise 2.6 to show that the two elements set $X = \{Up, Down\}$ is finite.

Mathematical specifications and modelling: Agent example

In agent-based models of green growth (opinion formation, consumption, etc.) it is common to equip a set of agents with certain features, like being employed or unemployed:

$$
\begin{array}{lll}
\mathit{status} & : \mathit{Agent} \to \{\mathit{Employed}, \mathit{Unemployed}\} & \\
\mathit{income} & : \mathit{Agent} \to \mathbb{R}_{\geq 0} & \text{-- ... have certain incomes and} \\
\mathit{buy} & : \mathit{Agent} \to \mathit{Prob}\,\{\mathit{GreenCar}, \mathit{BrownCar}, \mathit{NoCar}\} & \text{-- ... consumption behaviours}
\end{array}
$$

Here $\mathit{Prob}\ X$ represents finite probability distributions over an arbitrary set X.
Let $\mathit{Event}\ X = X \to \mathit{Bool}$ and

$$\text{prob} : \mathit{Prob}\ X \to \mathit{Event}\ X \to [0, 1]$$

be the generic function that computes the probability of an event $e : \mathit{Event}\ X$ according to a given probability distribution. Thus, for $d \in \mathit{Prob}\ X$ and $e \in \mathit{Event}\ X$

$$\text{prob}\ d\ e$$

represents the probability of e according to d.

Agent example, cont.

We want to implement an agent-based model in which agents with higher incomes are more likely to buy green cars than agents with lower incomes.

We also would like to specify that unemployed agents are less likely to buy a brown car than employed agents.

Exercise

Express these model requirements as mathematical specifications using the model-specific functions *status*, *income*, *buy* and the generic function prob.

Equational reasoning

Equational Reasoning is the proof method encouraged by the "Algebra of Programming" community [1] for reasoning about systematic, correctness preserving program transformations. Originally this form of calculation with programs was done on a semi-formal meta-level (by semi-formal we mean: on paper, not inside an implemented type theory/proof assistant).

It comes with a distinctive style of presenting proofs with justification of every transformation step (just as one would do in school when solving equations).

A very important ingredient of this algebraic approach to program correctness is *structural induction*. Here, we will look at a simple example using this technique, presented in equational reasoning style.

We will prove a property of exponentiation.

Equational reasoning, example

The exponentiation with natural numbers fulfills the properties

(1) $\forall\ x \in \mathbb{R},\ x^0 = 1$

(2) $\forall\ x \in \mathbb{R},\ m \in \mathbb{N},\ x^{1+m} = x * x^m$

From (1), (2) and the algebraic properties of $*$ and $+$ we can show that

$\forall\ x \in \mathbb{R},\ m, n \in \mathbb{N},\ x^{m+n} = x^m * x^n$

The proof is by induction on m. We first show the base case ($m = 0$)

$\forall\ x \in \mathbb{R},\ n \in \mathbb{N},\ x^{0+n} = x^0 * x^n$

Then we prove the induction step ($m \Rightarrow 1 + m$)

$\forall\ x \in \mathbb{R},\ n \in \mathbb{N},\ x^{m+n} = x^m * x^n$

$\Rightarrow$

$\forall\ x \in \mathbb{R},\ n \in \mathbb{N},\ x^{(1+m)+n} = x^{1+m} * x^n$

Equational reasoning, example, cont.

The proofs are obtained by *equational reasoning*.
Let's start with the "difficult" (induction step) case:

$$
\begin{aligned}
x^{(1+m)+n} &= \{\text{- Associativity of } (+) \text{ -}\} \\
x^{1+(m+n)} &= \{\text{- Property (2) -}\} \\
x * x^{m+n} &= \{\text{- Induction hypothesis -}\} \\
x * (x^m * x^n) &= \{\text{- Associativity of } (*) \text{ -}\} \\
(x * x^m) * x^n &= \{\text{- Property (2) -}\} \\
x^{1+m} * x^n &
\end{aligned}
$$

Exercise

Prove the base case.

Bibliography

[1] Richard S. Bird and Oege de Moor. *Algebra of programming*. Prentice Hall International series in computer science. Prentice Hall, 1997.

[2] Carroll Morgan. *Programming from specifications*. Prentice Hall International Series in computer science. Prentice Hall, 1990.